

Flexible LAN-WAN Orchestration for Communication Efficient Federated Learning over Large-Scale Mobile Devices

Jinliang Yuan*, Qing Li[†], Fan Dang[‡], Xiaofang Mu^{§¶}, Hui Qi^{§¶}, Mengwei Xu[†], and Shangguang Wang[†]

*Department of Automation, Tsinghua University, China, Email: yuanjinliang@mail.tsinghua.edu.cn

[†]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China

[‡]Global Innovation Exchange (GIX), Tsinghua University, China

[§]College of Computer Science and Technology, Taiyuan Normal University, China

[¶]Shanxi Key Laboratory of Intelligent Optimization Computing and Blockchain Technology, China

Abstract—Federated learning (FL) has been widely adopted as a privacy-preserving model training paradigm. However, traditional FL protocol heavily relies on data transmission between clients and servers across the wide-area network (WAN), which is tightly constrained and unreliable, therefore causing expensive communication and slow convergence. To this end, we propose a LAN-aware FL (LanFL) protocol, which can efficiently leverage the network capacity of the local-area network (LAN). By frequent model aggregation among the devices within the same LAN, we can significantly reduce the global aggregation across WAN, thus accelerating the training process. However, due to the unique challenges introduced by LAN, it's not easy to efficiently utilize LAN resources while preserving the original dignity of FL performance. Therefore, LanFL also incorporates several critical techniques: LAN-aware hierarchical aggregation, intra-LAN device topology construction, and inter-LAN heterogeneous bandwidth coordination. Extensive real-world experiments are conducted and the experimental results show that LanFL can significantly accelerate FL training up to 6.0 \times , while preserving the model accuracy.

Index Terms—Mobile and wireless computing, network topology, local-area network

I. INTRODUCTION

Privacy protection is critical to user-centric AI applications, especially considering the ever-growing public concerns over user privacy. Federated learning (FL) [1] is a privacy-preserving machine learning (ML) paradigm that enables a large number of mobile devices to collaboratively train an ML model without uploading data to a remote server. This cross-device FL paradigm has a wide spectrum of use cases such as input method, voice assistant, and item recommendation for millions of users. It is a practical solution to strongly protect user privacy, which makes it valuable to study how to efficiently deploy FL protocols over large-scale mobile devices [2].

However, deploying cross-device FL is difficult in many aspects. From the perspective of devices, the huge consumed communication bandwidth across wide-area network (WAN) is one of the most critical challenges for most of existing FL protocols. Especially, FL deployment typically involves

thousands, or even millions of mobile devices, with limited uplink bandwidth (e.g., 24% of them < 2 Mbps [3]). As compared to distributed ML in datacenter, the cross-device FL highly relies on WAN whether through 4/5G or WiFi access, e.g., the inter-city, inter-state, or even inter-country data transmission. This is because a central aggregation server iteratively collects model updates from a large number of geographically-distributed devices and dispatches aggregated results (a new model) to them. The aggregation typically repeats for 500–10,000 rounds with model size 10-1000 MB before model convergence [4]. Such a WAN-driven design, however, leads to the following inevitable drawbacks.

(i) *WAN is known to be highly constrained and unstable* [5], which can severely slow down the convergence of FL training, e.g., 3,000 rounds and 5 days for an RNN model reported by Google [2]. In fact, network transmission has become a critical bottleneck in the FL process [6]. Two key aspects for existing work to solve the communication challenge are 1) reducing the total number of communication rounds and 2) reducing the size of the transmitted messages at each round. For the former one, [7] focused on accelerating the model convergence through designing a better federated gradient descent algorithm than *FedAvg* [1]. For the latter one, the gradients compression [8] technique was proposed to save network bandwidth at each round with tolerable accuracy loss. These work aim to reduce network transmission through algorithm re-design, lacking a scalable and communication-efficient FL system design for large-scale mobile devices. Our goal is to design such an FL framework by leveraging the hierarchical architecture of mobile communication network.

(ii) *WAN usage incurs a high monetary cost to FL practitioners*. Deploying the global aggregator of FL on public clouds is a common practice. Nowadays, major cloud service providers like AWS and Azure support charging the network cost on demand. Since FL requires very little computational and memory resources (only for weights aggregation), the network cost often dominates the total monetary cost, e.g., more than 80% as we will show in Section V. In reality, the

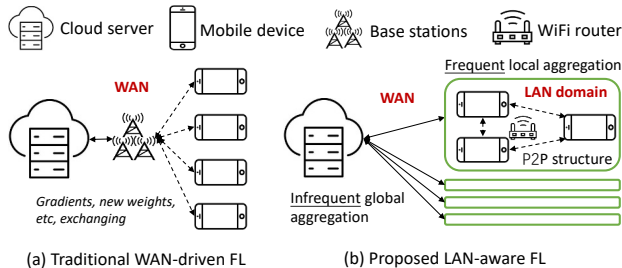


Fig. 1: A high-level comparison of traditional WAN-driven FL and our proposed LAN-aware FL.

cost is amplified by the demand to periodically update the model, e.g., to adapt to the evolving data distribution (input method) or new advanced model structures. Consequently, it becomes prohibitively costly for many small entities or individuals who want to practice on FL. The monetary cost of deploying a practical cross-device FL, as far as we know, has not been explored in prior work.

To address these weaknesses, we propose a LAN-aware FL protocol, namely LanFL, which fully exploits the resource of local-area network (LAN) to accelerate the FL training and reduce the monetary cost while preserving the model accuracy. We illustrate the high-level idea of our proposal in Figure 1. Within a LAN domain, the devices individually train a local model and *frequently* aggregate model updates. Across many LAN domains, the model updates will also be aggregated and exchanged through a remote cloud, yet in an *infrequent* manner. Through such a LAN-aware design, WAN traffic is much less demanded, thus the training process is accelerated. The rationales to introduce LAN-aware design are twofold: (1) Compared to WAN, LAN bandwidth resource is much more abundant, e.g., tens of Mbps as we will experimentally show in Section III-A (a nearly $10\times$ gap). This is because, during the WAN routing path, any hop can be the bandwidth bottleneck. For instance, the cloud service gateway can be a common one as it's shared by many tenants. In addition, LAN bandwidth is an unmetered resource that does not add cost to cloud services. Those advantages of LAN over WAN show the potential in accelerating FL convergence and reducing billing costs to deploy FL in the real world; (2) FL applications are commonly deployed on a large number of devices that are naturally organized into many LAN domains. For each LAN domain, such as campus and office building, there is often a substantial number of devices that are available for collaborative learning.

However, LanFL still faces the following three technique challenges: (1) *How to coordinate the collaborative training at the aspects of both devices and LAN domains?* For example, we need to determine the proper frequency for intra-LAN and inter-LAN aggregation, considering their disparate bandwidth resources; (2) *How to organize the devices into a proper topology so that the LAN bandwidth can be efficiently utilized?* A typical LAN domain comprises many interconnected access points (AP), and the devices connected to the same AP share

the AP's bandwidth capacity. An improper topology may cause the end-to-end training performance to be bottlenecked by one AP's capacity but leaving others underutilized.

To address the above challenges, LanFL incorporates several novel techniques. First, LanFL leverages the rich trade-off between training speed and model accuracy exposed by LAN domains, by separately tuning the parameters for intra-LAN and inter-LAN model aggregation. Second, LanFL adopts two widely-used topologies, i.e., parameter server [9] and Ring-AllReduce [10] for intra-LAN collaborative training. We design an adaptive topology construction algorithm, which can judiciously construct a network topology for parameter aggregation based on the device information profiled at runtime. In addition to the mentioned techniques, we also implement an end-to-end prototype of LanFL that is, as far as we know, the first end-to-end FL platform that can run on commodity mobile devices with the LAN-aware design. The major contributions are summarized as follows.

- We propose LanFL, a novel FL paradigm that can utilize the LAN bandwidth resource and thus relieve its reliance on constrained WAN, which is orthogonal to existing approaches designed from an algorithmic perspective.
- We validate our proposed LAN-aware design through theoretical analysis, and also evaluate LanFL experimentally on both simulation and real-world settings involving 30 heterogeneous mobile phones.

II. RELATED WORK

Communication optimization has been extensively explored to reduce WAN communication in FL. Some of them [6], [11] focus on reducing the total number of communication rounds. In practice, they have shown limited flexibility to adapt to communication-computation trade-offs [12]. Some other work [13] focus on reducing the transmitted data size through model compression methods, e.g., sparsification and quantization. As we experimentally show, these approaches can hardly speed up model convergence and often lead to accuracy degradation [14].

Network topology in distributed ML [9], [10], [15] considers the design of communication topology for computation nodes to efficiently share and aggregate their parameters. PS [9] and Ring-AllReduce [15] are the two main topologies in existing distributed machine learning system for the stable and efficient system performance. This is why we adopt them as the topology for our device parameter aggregation in each LAN. However, their conclusions were limited to high-speed Ethernet (10 Gbps) environments with no link bandwidth sharing, which is not realistic in wireless network.

III. PROBLEM DEFINITION AND CHALLENGES

We first follow the WAN-driven FL protocol to formulate the objective of proposed LAN-aware FL. N denotes the total number of devices scattered in WAN. The objective of traditional WAN-driven FL can be formulated as:

$$\min_{\mathbf{w}} F(\mathbf{w}) = \sum_{i=1}^N p_i \cdot f_i(\mathbf{w}), \quad (1)$$

where $f_i(\mathbf{w})$ is the local loss function of device i , $F(\mathbf{w})$ is the global loss function in the cloud, and p_i denotes the ratio of the device samples to the total samples. Existing FL protocols, e.g., *FedAvg*, mostly adopt stochastic gradient descent to perform the local optimization with E epochs and a fixed learning rate η . In particular, only n devices are randomly selected from total N devices at each global training round t . The cloud then aggregates the updates sent by each device i across WAN. The updates can be formulated as $\mathbf{w}_i(t+1) \leftarrow \mathbf{w}(t) - \eta \nabla f_i(\mathbf{w}(t))$ and then $\mathbf{w}(t+1) \leftarrow \sum_n \mathbf{w}_i(t+1)$.

A. Challenges to deploy LAN-aware FL

Given the inherent advantages of LAN over WAN in decentralized training, it's not easy to efficiently utilize LAN resources while preserving the original dignity of FL performance, e.g., converged model accuracy. The system design challenges introduced by LAN have not been explored before. These challenges go beyond the difficulty of designing a three-layered algorithm [16]–[18].

Local epoch tuning. In traditional WAN-driven FL, it is very important to tune a proper local epochs due to the complex trade-off between model accuracy and time. In comparison, our LAN domain design allows the model to be aggregated both locally (intra-LAN) and globally (inter-LANs), which increases the complexity of parameter tuning. Given the disparate behaviors of WAN and LAN bandwidth resources, it becomes more difficult to orchestrate them in a harmonious way. We will illustrate how to solve this challenge by tuning the key parameters for intra-LAN and inter-LAN collaborative learning separately in Section IV-B.

LAN bandwidth sharing. Existing FL literature assume static or pre-assigned WAN bandwidths to each device without considering the link bandwidth sharing between multiple devices. However, in reality, the intra-LAN network throughput highly depends on the number of devices that are transmitting data simultaneously. It opens an extra trade-off between the parallel training devices and data transmission speed, which is unexplored yet critical to system performance.

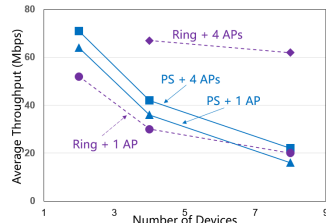


Fig. 2: A case study of P2P WiFi throughput in a campus LAN. “PS”: parameter server; “Ring”: Ring-AllReduce; AP: access point.

We compared the results with four different settings: Ring + 1 AP, Ring + 4 APs, PS + 1 AP, and PS + 4 APs. For example, Ring + 4 APs setting with 8 devices

means that every 2 devices are connected to one AP while all 4 APs are within the same LAN. The figure shows that: (1) Under PS mode, the throughput decreases almost linearly with an increasing number of devices, as the server side becomes the bottleneck of the transmission path; (2) When devices are distributed under many APs’ coverages, Ring mode significantly increases the throughput as it can efficiently utilize the available bandwidth of each AP.

The main reason behind above two findings is that the bandwidth sharing and link congestion of WiFi network become the dominant obstacle to network throughput. Therefore, building a proper network topology is critical to fully exploit the LAN bandwidth resources. However, how to obtain the topology from the enormous searching space with a tolerable computation complexity is difficult.

IV. DESIGN OF LANFL

In this section, we first illustrate our proposed LanFL’s overall system overview. We then elaborate the three key techniques, which are designed to address the three challenges aforementioned in Section III-A.

A. Overall Workflow

Algorithm 1 shows the workflow of LanFL. Overall, LanFL also adopts a C/S architecture, where a central server maintains and keeps advancing a global model \mathbf{w} , but the “client” refers to not only one device but a LAN domain comprising many devices connected through the P2P mechanism. LanFL follows *FedAvg* to aggregate the model weights updated by much local training, but such aggregation happens on both devices (intra-LAN) and cloud (inter-LAN).

We denote the total number of iterations is T . The LAN aggregation is executed after every τ_1 steps of gradient descent on device, which also means the on-device training epoch is τ_1 . The cloud aggregation is executed after every $\tau_1 \tau_2$ steps of gradient descent on device. It means that LAN aggregates τ_2 times when cloud aggregates one time.

Each device in LanFL serves as two roles: only as a training device and as both training and aggregating device. As a training device, it is responsible for updating the local model (Lines 7,8,9,15,16). As an aggregating device, it frequently aggregates models received from training devices as the communication topology. According to τ_1 , it decides whether to continue the local update or share the result across LAN for LAN aggregation (Lines 13-14). It also decides when to send the result to cloud across WAN for cloud aggregation (Lines 18-19) by the $\tau_1 \tau_2$.

Cloud is responsible for the following three tasks. First, it maintains and periodically updates a list of global information. The global model and LAN domain information for the topology are periodically update as the FL training process. Other parameters, like learning rate η , number of selected LAN m and devices n are fixed by prior knowledge. Second, the cloud adopts a random strategy (common practice to overcome the Non-IID challenge in FL) in selecting LAN domains and training devices (Lines 3,5). Through our topology algorithm (Line

Algorithm 1: Our LanFL Algorithm

```

input :  $\mathbf{w}_0, \eta, \tau_1$ : iterations of on-device training at each
         LAN aggregation,  $\tau_1 \tau_2$ : iterations of on-device
         training at each cloud aggregation,  $m$ : number of
         selected LAN,  $n$ : number of selected devices
1 initialize global model  $\mathbf{w}(t)$ 
2 for  $t \leftarrow 0$  to  $T$  do
   // LAN and devices selection on server
3   if  $t | (\tau_1 \tau_2) = 0$  then
4     randomly select  $m$  LANs with network capacity  $\{c^j\}$ 
5      $\{n^j\} \leftarrow$  obtain the number of selected devices in  $m$ 
       heterogeneous LANs, where  $n^j = \frac{c^j}{\sum_{j=1}^m c^j} \cdot n$ 
6     for each selected LAN domain  $j$  in parallel do
7       randomly select  $n^j$  devices, where  $n = \sum_{j=1}^m n^j$ 
8       distribute the global model to these devices
   // On-device training
9   for each selected device  $i$  in parallel do
10     $\mathbf{w}_i(t) \leftarrow \mathbf{w}_i(t) - \eta \nabla f_i(\mathbf{w}_i(t))$ 
11    if  $t | \tau_1 \neq 0$  then
12      update device model:  $\mathbf{w}_i(t+1) = \mathbf{w}_i(t)$ 
   // LAN-aware aggregation
13  if  $t | \tau_1 = 0$  then
14    for each selected LAN domain  $j$  in parallel do
15      obtain the topology  $G_j$ 
16      share models between devices as  $G_j$  across LAN
17      LAN aggregation:  $\mathbf{w}^j(t) = \sum_{i=1}^{n^j} p_i^j \mathbf{w}_i^j(t)$ 
18      if  $t | \tau_1 \tau_2 \neq 0$  then
19        update device model:  $\mathbf{w}_i(t+1) = \mathbf{w}^j(t)$ 
20    if  $t | \tau_1 \tau_2 = 0$  then
21      share models from devices to cloud across WAN
22      cloud aggregation:  $\mathbf{w}(t) = \sum_{j=1}^m p^j \mathbf{w}^j(t)$ 
23      update global model:  $\mathbf{w}(t+1) = \mathbf{w}(t)$ 
output: the final global model  $\mathbf{w}(T)$ 

```

12), it constructs a proper network topology for parameter aggregation in each LAN domain. Third, the cloud iteratively updates the global model by aggregating the received models using *FedAvg* algorithm (Line 20). Noting that LanFL only needs one aggregating device in each LAN domain to transmit its aggregated model to the cloud (Line 20), which can save a lot of WAN traffic in LanFL as confirmed by the experimental results in Table III.

B. LAN Domain Design

Many prior studies have shown the importance of properly tuning the hyperparameters of *FedAvg* due to the Non-IID settings in FL. In particular, the local epoch number E exposes a rich trade-off between training time and model accuracy. On the one hand, a larger E allows for more local computation on devices and potentially reduces cloud round across WAN, which can greatly accelerate the overall convergence process. On the other hand, with Non-IID datasets distributed cross devices, a larger E may lead each device towards the optima of its local objective as opposed to the global objective, which potentially hurts convergence accuracy. What’s more, the slow WAN will exacerbate the contradiction in terms of the clock time.




	WAN 	LAN 	Datacenter 
Bandwidth (Mbps)	Low, 1 – 20	Medium, 20 – 100	High, up to 1000
Optimal local epoch	Large, 1 – 50	Medium, around 1	Small, typically 1 batch

Fig. 3: A comparison of WAN, LAN, and datacenter in consideration of their bandwidth capacity and optimal local epoch in distributed learning.

To some extent, LanFL’s LAN domain is similar to a distributed system in datacenter, where many machines are interconnected through switches. As summarized in Figure 3, regarding the network bandwidth capacity, however, LAN is somewhere between the WAN and datacenter. It indicates a new spectrum of parameter tuning different from WAN-driven FL (typically very large E) and datacenter (typically as small as one batch per aggregation).

LanFL introduces a LAN domain module, within which the models are aggregated more frequently: with fewer local epochs τ_1 (also denoted as E) but more LAN aggregations τ_2 . The module is used to aggregate those device models in each LAN domain with τ_2 LAN aggregations. In each LAN aggregation, those selected training devices update the local model with τ_1 epochs in parallel. We explain in two aspects why this design can speed up FL while ensuring accuracy. First, a smaller τ_1 can ensure the accuracy for the Non-IID datasets on devices in each LAN domain. It doesn’t increase the communication cost in LAN too much because the communication speed in LAN is much faster than that in WAN. Second, a larger τ_2 can reduce the number of aggregations in the cloud across WAN. It doesn’t significantly reduce the global model accuracy because the cloud aggregates the entire LAN domain model, not the more biased device model with more Non-IID datasets.

C. Adaptive Topology Construction

As discussed in Section III-A, the network topology about how devices within a LAN domain are organized has substantial impacts on the network throughput. For instance, when there are only a few APs, such as in a family LAN, the PS mode can achieve higher P2P communication throughput. When there are more APs and devices, such as in a campus LAN, the Ring mode allows for higher throughput. We give a high level comparison between PS and Ring mode in Figure 4. In this section, we first present an analytical model, and then further propose our topology algorithm to minimize the overall parameter transmission time with a proper network topology.

1) *Analytical Model*: Devices in a LAN are connected by some distributed routers, which can be represented as a directed graph. We denote the graph as $\mathbb{G}_1 = (\mathbb{N} \cup \mathbb{R}, \mathbb{E}_1)$, where \mathbb{N} denotes the set of mobile devices, \mathbb{R} denotes the set of routers, and \mathbb{E}_1 denotes the set of communication links. Let $d(v)$ denote the degree of each node v ($v \in \mathbb{N} \cup \mathbb{R}$). Each device $i \in \mathbb{N}$ can only access the network through a specific router $j \in \mathbb{R}$. The corresponding link is denoted as (i, j) , whose bandwidth is denoted as $b(i, j)$. The wireless bandwidth capacity of each router j is denoted as $b(j)$. Let \mathbb{N}^j denote the set of devices directly connected to router j . Devices

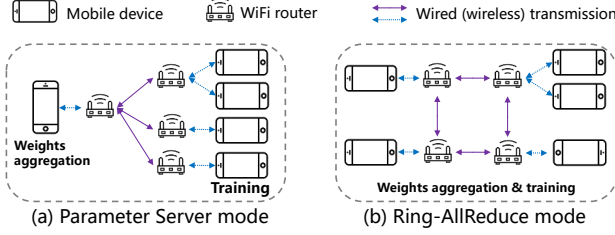


Fig. 4: A comparison of two network topologies of how devices within a LAN domain are organized: Parameter Server (PS) mode and Ring-AllReduce (Ring) mode.

connecting to the same router share the bandwidth of the router. Thus, the available bandwidth of link (i, j) is calculated:

$$b(i, j) = \frac{b(j)}{\sum_{i^* \in \mathbb{N}^j} \varphi_i(d(i^*))}, \quad (2)$$

where $i^* \in \mathbb{N}^j$ represents all devices (including device i) connected to routers j , and $\varphi_i(\cdot)$ represents the bandwidth allocation strategy for device $i \in \mathbb{N}^j$.

Let $\mathbb{G}_2 = (\mathbb{N}, \mathbb{E}_2)$ denote the direct communication among devices, where \mathbb{E}_2 denotes the direct communication path between two devices. In a LAN, routers are connected directly through a switch, which has a much higher bandwidth capacity than wireless links. Every two devices can be connected through one (when they connect to an identical router) or two routers (when they connect to different routers). We denote the communication path between device i and i' as (i, i') , where path $(i, i') \in \mathbb{E}_2$. The path's bandwidth is given as:

$$b(i, i') = \begin{cases} \min\{b(i, j), b(j, i')\} & i, i' \in \mathbb{N}^j \\ \min\{b(i, j), b(j, j'), b(j', i')\} & i \in \mathbb{N}^j, i' \in \mathbb{N}^{j'} \end{cases} \quad (3)$$

Based on Eq. 2 and Eq. 3, we seek to optimize the overall parameter transmission time among all the devices.

However, it is difficult to solve the above problem. First, the parameter transmission time is closely related to network topology and the roles that devices play in different network topologies. Second, due to the bandwidth sharing property of wireless links within an identical router, the allocated bandwidth to each link is affected by the network topology. Third, directly determining the optimal network topology and the roles of different devices can cause high computation complexity due to the enormous searching space. In this work, we explore PS and Ring-AllReduce modes to design the network topology, due to the extremely huge search space theoretically. We explain the rationales for two reasons. First, they are two most popular and well established designs in datacenter-level distributed ML. Extensive work [9], [10], [15], [19] are built atop them and achieve the state-of-the-art performance. Second, a few work have proven that these two topologies lead to the minimal network traffic [10], [15] in datacenter network. While a more suitable network topology might exist for LAN, it is not the primary concern of this work and we leave it as future work.

We notice a few advanced network topology designs other

than PS and Ring-AllReduce, for example [20], which designed an optimal communication topology that minimize the parameter transmission time in cross-silo FL. However, they do not fit into LAN scenario for the following reasons: 1) It mainly focus on the cross-silo FL with the assumption of high-speed connections and neglecting access linked delays, which is unacceptable in cross-device scenario with WiFi or 4G/5G access network. The bandwidth sharing challenge, as shown in Figure 2 and Eq. 2, makes our problem unique to [20]. 2) The determined participants and stable links makes the topology construction time of the NP-hard problem negligible, which is unacceptable in cross-device scenario with random participants and dynamic wireless network. For the cross-silo scenario, the optimal topology was generated to serve the entire training process at the beginning. However, for our LAN domain aggregation in WiFi access network, the optimal topology is dynamically generated for the current round based on the selected devices and accessed routers.

Therefore, to deal with the above challenges, we first analyze the property of two classical network topologies in distributed machine learning, and then further determine the roles of devices. To be noted, building atop PS and Ring-AllReduce does not mean our system only has two concrete topology candidates. For example, with PS topology, each client can be treated as the server so the number of topology candidates equals to the total number of clients.

2) *Our proposed algorithm:* PS and Ring are two widely used topologies in distributed machine learning for their simple and effective designs. We consider these two topologies when optimizing the overall parameter transmission time. The transmission time is bottle-necked by the link with minimum bandwidth in both PS and Ring topology [10], [21]. Let \mathbb{G} denote the graph set of the two topologies, where $\mathbb{G} \subset \mathbb{G}_2$. The minimum bandwidth b of all links in one topology $G \in \mathbb{G}$, i.e.,

$$b = \min_{(i, i') \in G} \{b_{i, i'}\}. \quad (4)$$

In PS mode, for a topology G , one device in G acts as a server that receives and aggregates parameters from other devices. We first get the minimum bandwidth b_1 in all links as Eq. 4. Then, we calculate the transmission time t :

$$t = 2 \cdot \frac{s}{b_1}, \quad (5)$$

where s denotes the transmitted data size of the global model. We first select one device from the k selected devices (where n denotes total selected devices in WAN) as the server in this LAN, resulting in k possible different choices. We then traverse all $k-1$ edges in PS topology to get the one with minimum link bandwidth based on Eq. 4. So the problem complexity is $O(k^2)$. In Ring mode, for a topology G , all devices in G are arranged in a logical ring, and each device only sends a parameter to its right neighbor and receives a parameter from its left neighbor. Meanwhile, each device immediately aggregates the new parameters until all parameters are aggregated. We can get the minimum bandwidth b_2 of all links as Eq. 4.

Dataset	Model	Model size	Client number
FEMNIST	CNN (2 CONV, 2 FC, 28×28)	25MB	3550
CelebA	CNN (4 CONV, 2 FC, 84×84)	36MB	9434

TABLE I: Datasets and models used in experiments.

Device	Qualcomm	Training Time (ms)	
		FEMNIST	CelebA
Google Pixel 4	SM8150	179	561
Redmi Note 7 pro	SDM675	588	1355
Nexus 6	APQ8084	1642	5392

TABLE II: Training time of different devices with MNN [24]

We calculate the transmission time t for Ring mode:

$$t = \frac{4(k-1)}{k} \cdot \frac{s}{b_2}. \quad (6)$$

Considering the connection characteristics of the device and router, the number of each device’s degrees in the Ring mode is 2 with one out-degree and one in-degree. So the minimum bandwidth for any Ring topology remains the same. We first select a Ring topology at random and then iterate its k edges to get the one with minimum link bandwidth. So the problem complexity is $O(k)$.

Among the above two modes, we choose the topology with less time as the problem’s optimal solution. The overall complexity of the problem is $O(k^2)$.

V. PERFORMANCE EVALUATION

This section uses experiments to evaluate the guarantees of our proposed approach. We use two popular FL datasets and CNN models in the experiment. We also implement a LAN-aware FL framework to show the superior performance of LanFL, compared to three baselines.

A. Experiment Setup

Datasets and models used in our experiments are summarized in Table I. We test LanFL on 2 datasets: FEMNIST, CelebA. We split each dataset into Non-IID settings and assign them to many devices.

We implement LanFL atop FLASH [14], a framework that can simulate the real-world FL process by considering device and user behavior heterogeneity.

We use three baselines in experiments to demonstrate LanFL’s benefits are: WAN-driven *FedAvg* [1] algorithm denoted as WAN-FL, and two typical gradients compression algorithms, i.e., Structured Update [22] denoted as WAN-FL-Struct and SignSGD [23] denoted as WAN-FL-Sign, respectively. Those two algorithms are expected to reduce the communication overhead in FL.

We use four metrics to evaluate LanFL and baselines:

- **Convergence Accuracy.** We test the final global model on the combined testing set from all devices.
- **Clock time.** We define the *clock_time* as the total time spent during FL till convergence, including device training time ($train_{T_C}$) and communication time across WAN (com_{T_W}) and LAN (com_{T_L}).

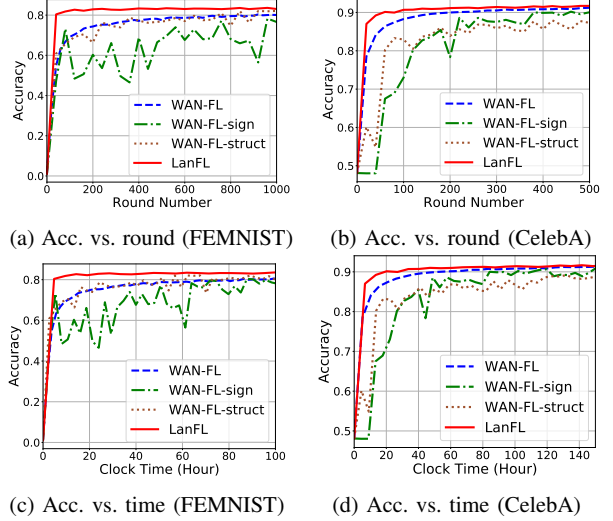


Fig. 5: The testing accuracy of LanFL and baselines across training rounds and clock time on two datasets

- **WAN traffic.** We quantify the *WAN_traffic* in terms of total traffic from cloud to devices during the FL process till model convergence.
- **Monetary cost.** We measure the *Cost* on the billing model of AWS EC2. We use a 1.2xlarge instance with 8×vCPU and 16GB memory, which is enough for model aggregation and costs \$0.204/hour. The uplink traffic is free and the downlink cost is \$0.09/GB. The cost is calculated as:

$$Cost = 0.204 \cdot clock_time + 0.09 \cdot WAN_traffic. \quad (7)$$

B. End-to-end Results

We first show LanFL’s overall results compared to baselines in Table III and Figure 5. For each device round, LanFL selects 5 LAN domains (m) with each domain involving 10 training devices (k). We set the WAN and LAN bandwidth as 2Mbps and 20Mbps, respectively.

Figure 5 shows that LanFL can significantly accelerate model convergence while preserving model accuracy. On FEMNIST, LanFL’s accuracy is 1.03% higher than WAN-FL, and it costs only 16% of clock time (i.e., 6.25× speedup). On CelebA, LanFL’s convergence accuracy is only 0.1% higher than WAN-FL, but it costs 68% of clock time (i.e., 67 hours reduction). The other two baselines reduce the convergence time but are not as significant as LanFL, and cause non-trivial accuracy drop, e.g., 2.84% for WAN-FL-Struct on CelebA.

The experimental results of Table III show that LanFL can significantly reduce WAN traffic and monetary cost. The number of cloud round required by LanFL to converge is much fewer than WAN-FL (87% on FEMNIST, 47% on CelebA) as shown in Figure 5, which brings in tremendous WAN traffic savings. Table III also shows that the WAN traffic and monetary cost of LanFL are much less than WAN-FL: On FEMNIST, LanFL reduces 99 % WAN traffic and 97%

Dataset	FL Protocols	Accuracy (%)	Round Number	WAN Traffic (GB)	Clock Time (hour)	Monetary Cost (\$)
FEMNIST	WAN-FL	81.82	1820	2221	170	234.57
	WAN-FL-Sign	80.15 (1.67↓)	1360 (75%)	74 (3%)	90 (53%)	25.02 (11%)
	WAN-FL-Struct	81.54 (0.28↓)	1820 (100%)	213 (10%)	124 (73%)	44.47 (19%)
	LanFL	82.85 (1.03↑)	240 (13%)	29 (1%)	28 (16%)	8.32 (3%)
CelebA	WAN-FL	91.56	800	1406	208	169.24
	WAN-FL-Sign	89.56 (2.00↓)	800 (100%)	44 (3%)	177 (85%)	40.16 (24%)
	WAN-FL-Struct	88.72 (2.84↓)	800 (100%)	250 (18%)	182 (88%)	59.63 (35%)
	LanFL	91.66 (0.10↑)	420 (53%)	73 (5%)	141 (68%)	35.33 (21%)

TABLE III: Summarized performance of LanFL compared to 3 baselines.

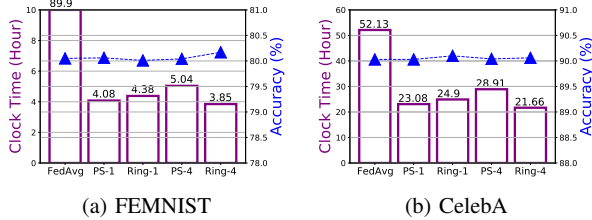


Fig. 6: The training performance of LanFL and FedAvg with different PS and Ring mode.

monetary cost to the WAN-FL; On CelebA, LanFL reduces 95% WAN traffic and 79% monetary cost to the WAN-FL. The saved cost comes from both the reduced clock time to rent hardware resources and the reduced network traffic according to Eq. 7. For comparison, the improvements of the other two gradients compression algorithms are much less effective.

C. Analysis of LAN topology

We further evaluate LanFL’s design in determining intra-LAN device topology (Section IV-C) by testing LanFL with both PS and Ring modes. We set the configurations of LanFL as in Figure 6 according to our measured results shown in Figure 2. We choose the four measured average throughputs of 8 devices in Figure 2 as the B^* of LanFL. We denote the four cases in Figure 2 as PS-1, PS-4, Ring-1, and Ring-4, and their throughputs are 22Mbps, 16Mbps, 20Mbps, and 62Mbps, respectively. Other settings are consistent with Figure 5.

The results are illustrated in Figure 6, from which we make the following key observations. First, LanFL outperforms the WAN-FL with either PS or Ring mode, e.g., $22.0\times-23.4\times$ and $2.3\times-2.4\times$ faster respectively at accuracy 80% on FEMNIST and 90% on CelebA. Second, LanFL adopts PS mode which outperforms Ring mode when the AP number is small. For example, on FEMNIST and 1 AP, LanFL takes $1.2\times$ longer time till model convergence. When the AP number is larger, however, LanFL favors Ring mode, e.g., $1.1\times$ faster than PS mode with 4 APs on FEMNIST. It shows that LanFL can judiciously pick the optimal network topology.

VI. REAL-WORLD EXPERIMENTS

A. Experiment Setup

We are not aware of any FL frameworks for real-world, cross-device deployment. Therefore, we have implemented an end-to-end cross-device FL system atop MNN [24], a mobile-oriented ML engine with both on-device training/inference

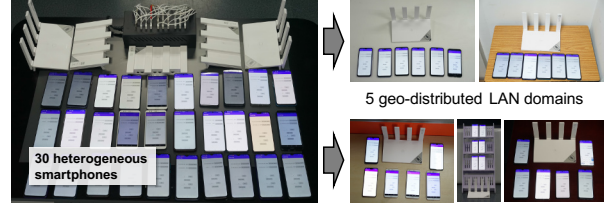


Fig. 7: The snapshots of our real-world experimental settings.

support. The system includes all modules of standard FL process, e.g., on-device learning and on-server aggregation, as well as the unique modules of LanFL, e.g., cross-device recognition and communication, on-device aggregation, network topology construction, etc. The communication module is implemented based on the WebSocket protocol, which can provide full-duplex communication channels. In total, the system comprises $\sim 3,500$ lines of code.

We use a 1.2xlarge instance3 on AWS EC2 with $8\times$ vCPU and 16GB memory as the remote server. The experiments are performed on 30 mobile phones with 12 different device models, including HUAWEI Mate 30, Redmi Note 9, Mi 11 Pro, Samsung Galaxy S21, etc. The 30 devices are separated into 5 campus LANs, within each LAN the 6 devices are connected through a wireless router.

B. Experiment Result

In this section, we study the performance of LanFL compared to FedAvg in our implemented practical end-to-end cross-device FL system. We also evaluate the on-device overhead of LanFL’s LAN aggregation, which demonstrates that our on-device aggregation design doesn’t add much extra burden to the mobile device. The 30 mobile phones are scattered in 5 different campus networks served as 5 LAN domains ($m=5$). Each campus is allocated by 6 mobile phones of different types. The 6 phones are either connected to dedicated high-speed routers (dedicated) in Figure 7 or to the campus public routers (public). We set a different numbers of classes each client may have, i.e., 2 for Non-IID-2 and 6 for Non-IID-6, with 10 classes globally. In each cloud round (when $T|\tau_1\tau_2=0$), we randomly select 10 mobile phones to participate in FL training ($n=10$). We use the WAN-FL in Section V as our baseline.

The end-to-end experimental results are illustrated in Figure 8, from which we make the following key observations. First, LanFL outperforms the WAN-FL in both Non-IID-2 and Non-IID-6 FL settings, e.g., 14% higher for accuracy and

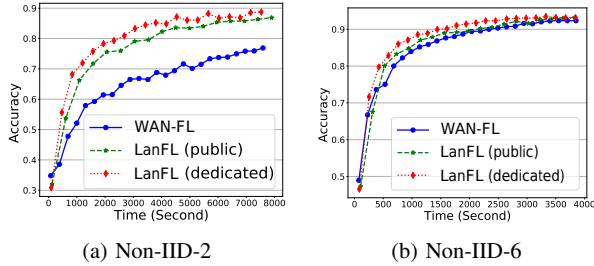


Fig. 8: The training performance of LanFL and WAN-FL with different Non-IID FL settings.

1.4 \times faster for clock time, respectively. On the Non-IID-2 setting, the serious class imbalance across datasets on devices results in low accuracy of FL training. However, the results in Figure 8 can also prove that our LanFL can improve the accuracy, especially in a more serious class imbalance Non-IID setting. On the Non-IID-6 setting, the accuracy of LanFL is only 0.7% higher than WAN-FL, but it takes 71% of clock time (i.e., reducing 1009 seconds). Such an observation is consistent with the simulation results in Section V-B. Second, the convergence accuracy of LanFL dedicated and LanFL public is basically the same, but the former converges faster than the latter in both FL settings. LanFL with public routers takes 2200 seconds and 650 seconds longer than LanFL with dedicated routers on Non-IID-2 and Non-IID-6, respectively. Noting that, many other devices are also connected to the campus public routers, which leads to the bandwidth of our FL training devices becoming smaller.

VII. CONCLUSIONS

Traditional FL protocols heavily rely on WAN that often slows down the learning process and adds billing costs to developers. To address those raised limitations, this work proposes a LAN-aware FL paradigm. The main idea is to exploit the abundant and unmetered bandwidth of LAN, which leverages both LAN for frequent local aggregation and WAN for infrequent global aggregation. It also incorporates several critical techniques to tackle the unique challenges introduced by the LAN-aware design. The experimental results show that it can significantly save the consumed WAN bandwidth and reduce the monetary cost while preserving the model accuracy.

VIII. ACKNOWLEDGES

This work was supported by National Key R&D Program of China (No.2021ZD0113001) and supported in part by the NSFC under grant No. 12371381.

REFERENCES

- [1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.
- [2] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," in *Proceedings of Machine Learning and Systems*, 2019.

- [3] P. Zhou, H. Xu, L. H. Lee, P. Fang, and P. Hui, "Are you left out?: An efficient and fair federated learning for personalized profiles on wearable devices of inferior networking conditions," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies.*, vol. 6, no. 2, pp. 91:1–91:25, 2022.
- [4] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *The Journal of Found. Trends Mach. Learn.*, vol. 14, no. 1-2, pp. 1–210, 2021.
- [5] "The state of wifi vs mobile network experience as 5g arrives," https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2018-11/state_of_wifi_vs_mobile_opensignal_201811.pdf.
- [6] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems*, 2018.
- [7] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *Proceedings of International Conference on Learning Representations*, 2020, pp. 1–16.
- [8] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Proceedings of Advances in Neural Information Processing Systems*, 2018, pp. 7652–7662.
- [9] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, "Scaling distributed machine learning with the parameter server," in *Proceedings of Symposium on Operating Systems Design and Implementation*, 2014, pp. 583–598.
- [10] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.
- [11] V. Smith, S. Forte, C. Ma, M. Takáč, M. I. Jordan, and M. Jaggi, "Cocoa: A general framework for communication-efficient distributed optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 8590–8638, 2017.
- [12] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.
- [13] S. Caldas, J. Konečný, H. B. McMahan, and A. Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *arXiv preprint arXiv:1812.07210*, 2018.
- [14] C. Yang, Q. Wang, M. Xu, Z. Chen, K. Bian, Y. Liu, and X. Liu, "Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data," in *Proceedings of the Web Conference*, 2021, pp. 935–946.
- [15] Y. Jiang, Y. Zhu, C. Lan, B. Yi, Y. Cui, and C. Guo, "A unified architecture for accelerating distributed DNN training in heterogeneous GPU/CPU clusters," in *Symposium on Operating Systems Design and Implementation OSDI*, 2020, pp. 463–479.
- [16] Z. Wang, J. Liu, H. Huang, C. Qiao, and Y. Zhao, "Resource-efficient federated learning with hierarchical aggregation in edge computing," in *Proceedings of IEEE Conference on Computer Communications*, 2021.
- [17] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin, "Hierarchical federated learning across heterogeneous cellular networks," in *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 8866–8870.
- [18] J. Wu, Q. Liu, Z. Huang, Y. Ning, H. Wang, E. Chen, J. Yi, and B. Zhou, "Hierarchical personalized federated learning for user modeling," in *Proceedings of the Web Conference*, 2021.
- [19] A. Gibiansky, "Bringing HPC Techniques to Deep Learning," <https://andrew.gibiansky.com/blog/machine-learning/baidu-allreduce>.
- [20] O. Marfoq, C. Xu, G. Neglia, and R. Vidal, "Throughput-optimal topology design for cross-silo federated learning," in *Annual Conference on Neural Information Processing Systems NeurIPS*, 2020.
- [21] P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *Journal of Parallel and Distributed Computing*, vol. 69, no. 2, pp. 117–124, 2009.
- [22] J. Konečný, H. McMahan, F. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *ArXiv preprint arXiv:1610.05492*, 2016.
- [23] J. Bernstein, K. Azizzadenesheli, and A. Anandkumar, "signsgd with majority vote is communication efficient and fault tolerant," in *Proceedings of International Conference on Learning Representations*, 2019.
- [24] X. Jiang, H. Wang, Y. Chen, Z. Wu, B. Zou, Y. Yang, Z. Cui, Y. Cai, T. Yu, C. Lv, and Z. Wu, "Mnn: A universal and efficient inference engine," in *Proceedings of Machine Learning and Systems*, 2020.