Federated Learning with Flexible Control

Shiqiang Wang*, Jake Perazzone†, Mingyue Ji‡, Kevin S. Chan†
*IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. Email: wangshiq@us.ibm.com
†Army Research Laboratory, Adelphi, MD, USA. Email: {jake.b.perazzone.civ; kevin.s.chan.civ}@army.mil,
‡Department of ECE, University of Utah, Salt Lake City, UT, USA. Email: mingyue.ji@utah.edu

***Abstract*—Federated learning (FL) enables distributed model training from local data collected by users. In distributed systems with constrained resources and potentially high dynamics, e.g., mobile edge networks, the efficiency of FL is an important problem. Existing works have separately considered different configurations to make FL more efficient, such as infrequent transmission of model updates, client subsampling, and compression of update vectors. However, an important open problem is how to jointly apply and tune these control knobs in a single FL algorithm, to achieve the best performance by allowing a high degree of freedom in control decisions. In this paper, we address this problem and propose *FlexFL* – an FL algorithm with multiple options that can be adjusted flexibly. Our FlexFL algorithm allows both arbitrary rates of local computation at clients and arbitrary amounts of communication between clients and the server, making both the computation and communication resource consumption adjustable. We prove a convergence upper bound of this algorithm. Based on this result, we further propose a stochastic optimization formulation and algorithm to determine the control decisions that (approximately) minimize the convergence bound, while conforming to constraints related to resource consumption. The advantage of our approach is also verified using experiments.**

***Index Terms*—Compressed model update, federated learning, partial participation, stochastic optimization**

I. Introduction

Many emerging applications nowadays are driven by machine learning technologies. To train models that are used in such applications, a large training dataset is usually needed. However, it has become increasingly common that data are collected and stored by local users at their end devices or organizational servers. It is difficult to share such data with a central entity, due to privacy regulations and communication bandwidth limitation. As a result, *federated learning (FL)* has emerged as a promising technique for distributed model training from decentralized local datasets [1]–[3].

At its core, FL includes model updates at each client (e.g., user device) using its own local data and aggregation of model parameters through a server (e.g., a cloud instance). In a resource-constrained system, such as a mobile edge network, these FL operations consume both computation and communication resources. Therefore, an important research direction is *how to make the most efficient use of the limited resources to maximize the performance of FL*. Some recent works have considered this problem by tuning configuration parameters of the FL algorithm, such as the number of local updates in each FL round, participation rate of clients, and compression rate of parameters transmitted between clients and the server [4]–[20]. However, the vast majority of them only focus on adjusting a small subset of all the available control options in FL, which cannot achieve the full potential of making FL the most efficient. In particular, the automatic adaptation of both parameter compression (e.g., sparsification and quantization) and partial client participation has not been studied, to the best of our knowledge. There is also usually a tight coupling between computation and communication in existing works, which may be difficult to achieve in heterogeneous systems where the costs of different resources can vary over time.

In light of these limitations, there is an important open problem: *Is it possible to jointly apply a wide range of control options in a single FL algorithm, to support heterogeneous and time-varying costs*[1] *of multiple types of resources?* There are several challenges in answering this question. *1)* It is non-straightforward to design an FL algorithm that allows simultaneous adjustment of multiple configurations with a high degree of freedom. *2)* It is difficult to analyze and understand the influence of different control options and the interplay between them on the FL performance. *3)* It is challenging to design an efficient control algorithm to automatically determine the best configurations subject to various constraints.

In this paper, we address this problem by proposing *FlexFL*, which is an FL algorithm that includes flexible control knobs that can be adjusted based on computation and communication costs. In essence, FlexFL includes three components: *1)* partial computation at clients, *2)* compressed parameter transmission from each client to the server, and *3)* compressed parameter transmission from the server to clients. Each of these components includes its own controllable parameter to define the rate of computation (for the first component) or communication (for the second and third components).

There are several key characteristics in FlexFL. First, the amount of computation and the amount of communication are *decoupled* and can be controlled separately, allowing a high degree of freedom in control decisions to suit the current costs of different types of resources. Second, both the computation and communication rates can vary over time and they can be different for different clients and the server, which allows a high degree of *system heterogeneity* and flexible resource

This research was partly sponsored by the U.S. Army Research Laboratory under Agreement Number W911NF-16-3-0002 and the National Science Foundation (NSF) CAREER Award 2145835. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

[1]We consider the resource *cost* as a generic metric in this paper, which can be defined as related to the availability of each type of resource.

usage depending on time-varying costs. Third, FlexFL includes the special case of *multiple local computations*[2] by setting the communication rate to zero in certain rounds. Moreover, FlexFL and its analysis also allow *statistical heterogeneity*, i.e., non-i.i.d. data across clients, which is commonly observed in practical FL scenarios.

We also present a convergence analysis of our FlexFL algorithm for general non-convex objectives. The resulting convergence bound provides important insights. In particular, we reveal that the convergence error increases in the residual error and decreases in the participation (computation) rate, where the residual error captures the gap between the transmitted model parameter and the computed local parameter (at each client) or received aggregated parameter (at the server).

Finally, we formulate our control problem as *stochastic optimization* over a finite time horizon, which makes decisions on the computation and communication rates over time, to minimize the convergence error subject to time-averaged cost constraints. We propose a *distributed* and *online* algorithm to approximately solve this problem. In addition, we conduct a thorough analysis of this control algorithm and discuss its important properties and insights, based on which we explain how to balance constraint satisfaction and optimality, and also give closed-form solutions for a class of costs.

In summary, our main contributions are as follows.

1) We present an algorithm named FlexFL, which allows flexible configurations in the amount of computation at each client and the amount of communication between clients and the server. This algorithm provides a high degree of freedom in adapting the FL procedure to heterogeneous and dynamically changing resource costs.
2) We analyze the convergence error bound of FlexFL, which reveals important insights on how the residual error and participation rate affect the convergence. This result lays out the foundation for our control algorithm.
3) We propose a control algorithm that is derived from stochastic optimization, to approximately minimize the convergence error while satisfying constraints on the time-averaged resource cost. Our control algorithm makes decisions in an online and distributed manner, without requiring prior knowledge of system statistics.
4) We give an in-depth analysis of our control algorithm, revealing several insights including how to adjust the trade-off between constraint satisfaction and optimality.
5) We present experimental results on real datasets, which confirm the advantage of our proposed approach.

II. Related Works

Over the past few years, efforts have been made to make FL resource-efficient, using techniques such as computing multiple local updates between communication rounds [21]–[26], transmitting compressed (sparse or quantized) model updates [27]–[41], and allowing only a small subset of clients to participate in each FL round [42]–[45]. A large body of these works focuses on analyzing the convergence behavior of these algorithms, but the study of multiple local computations with partial client participation has been largely separate from compression. To our knowledge, there does not exist work that incorporates both partial client participation and the special case of no transmission (similar to multiple local computations) with general (possibly biased) compressors.

In addition, the above works consider fixed FL configuration parameters related to communication and computation, which can be difficult to tune. To address this problem, some recent works have considered the automatic determination of communication interval [4]–[6], rate of compression [7]–[12], client selection [13]–[20], and other aspects [46], to accommodate the dynamic availability of resources. However, many of these works require a sophisticated process of estimating parameters related to the convergence bound, while some others are mostly heuristic without convergence guarantee. Moreover, none of these works consider the joint design of partial client participation and compression at both the server and clients.

III. Federated Learning and FlexFL

A. Federated Learning Objective

We consider an FL system with N clients, where each client n has a local loss function $F_n(\mathbf{x})$ for model parameter $\mathbf{x} \in \mathsf{R}^d$. The function $F_n(\mathbf{x})$ is defined on each client n's local dataset, which represents the error (or loss) between the predicted output given by the model (with parameter vector $\mathbf{x}$) and the ground-truth output in the training dataset. The goal of FL is to minimize the global loss function $f(\mathbf{x})$, as in:

$$\min_{\mathbf{x}} f(\mathbf{x}) := \frac{1}{N} \sum_{n=1}^{N} F_n(\mathbf{x}), \tag{1}$$

where the average can be replaced by a weighted average if desired, but we consider the weighting coefficients to be part of $F_n(\mathbf{x})$ for simplicity. A characteristic of FL is that the local loss functions $\{F_n(\mathbf{x}) : \forall n\}$ are not observed directly, because the clients' raw data are not shared. Therefore, FL needs to solve (1) in a distributed manner.

B. FlexFL Algorithm

We describe our FlexFL algorithm to solve (1). Similar to other FL algorithms, intermediate model parameter updates are exchanged between clients and the server, while the raw data remain private at the clients locally. The full algorithm is given in Algorithm 1, where we consider a time-slotted system and the time slots align with the iterations[3] in FL. We explain the main procedure of this algorithm as follows.

The algorithm includes three sets of control parameters denoted by $\{q_t^n\}$, $\{\mathbf{v}_t^n\}$, and $\{\mathbf{u}_t\}$. These parameters are taken as inputs by Algorithm 1, and they can be computed by our control algorithm (Algorithm 2) described later in Section IV. We further let $\mathbf{x}_t$ denote the model parameter at the beginning of each iteration t. However, we do not transmit $\mathbf{x}_t$ directly

[2]In FlexFL, clients may perform multiple local computations (updates) with different mini-batches on the same local model parameter, and do not immediately update the parameter. This is slightly different from local updates done in the FL literature, but conceptually both approaches share similarities.

[3]We use "time slot" and "iteration" interchangeably in this paper.

Algorithm 1: FlexFL

Constants: > 0, initial (random) model parameter $\mathbf{x}_0$
Control parameters: $\{q_t^n\}$, $\{\mathbf{v}_t^n\}$, $\{\mathbf{u}_t\}$ // determined by Algorithm 2 in Section IV
Output: $\{\mathbf{x}_t\}$

1 $\mathbf{r}_0 \leftarrow \mathbf{0}$; // server residual error
2 $\mathbf{e}_0^n \leftarrow \mathbf{0}; \forall n$; // client residual error
3 **for** $t \leftarrow 0; \ldots; T-1$ **do**
4 **each client** $n \leftarrow 1; \ldots; N$ **in parallel:**
5 Sample $\mathbb{1}_t^n \sim \text{Bernoulli}(q_t^n)$; // randomized compute
6 $\mathbf{b}_t^n \leftarrow \mathbf{e}_t^n - \frac{\eta \mathbb{1}_t^n}{q_t^n} \cdot \mathbf{g}_n(\mathbf{x}_t)$; // no compute if $\mathbb{1}_t^n = 0$
7 $\mathbf{e}_{t+1}^n \leftarrow \mathbf{b}_t^n - \mathbf{v}_t^n$; // here, $\mathbf{v}_t^n$ is usually a compression of $\mathbf{b}_t^n$
8 Send $\mathbf{v}_t^n$ to the server; // transmitted local update
9 **the server:**
10 $\mathbf{a}_t \leftarrow \mathbf{r}_t + \frac{1}{N}\sum_{n=1}^N \mathbf{v}_t^n$;
11 $\mathbf{r}_{t+1} \leftarrow \mathbf{a}_t - \mathbf{u}_t$; // here, $\mathbf{u}_t$ is usually a compression of $\mathbf{a}_t$
12 Send $\mathbf{u}_t$ to all clients; // transmitted global update
13 **each client** $n \leftarrow 1; \ldots; N$ **in parallel:**
14 $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \mathbf{u}_t$; // synchronized model parameter

between clients and the server. Instead, we transmit (possibly) compressed vectors of parameter updates, as we will see next. For the purpose of description and analysis, we assume that there are T iterations in total.

1) Local Computation at Clients: In every iteration t, each client n computes a new stochastic gradient $\mathbf{g}_n(\mathbf{x}_t)$ of the local loss function $F_n(\mathbf{x}_t)$ with probability $q_t^n \in (0, 1]$ (Line 5). We use the identity $\mathbb{1}_t^n \in \{0, 1\}$ to denote the random outcome, which is equal to one if client n performs a new computation in this iteration t, and zero otherwise. If $\mathbb{1}_t^n = 1$, this stochastic gradient $\mathbf{g}_n(\mathbf{x}_t)$ is applied in Line 6 in the form of stochastic gradient descent (SGD) with a given learning rate of $\eta > 0$. We divide the learning rate by q_t^n to keep the update unbiased. If $\mathbb{1}_t^n = 0$, the last term in the right-hand side (RHS) of Line 6 is zero, and we do not make any update in this case. In practice, we *do not compute* $\mathbf{g}_n(\mathbf{x}_t)$ *if* $\mathbb{1}_t^n = 0$, which is *equivalent* to the update equation in Line 6 since the value of $\mathbf{g}_n(\mathbf{x}_t)$ has no effect on the subsequent updates if $\mathbb{1}_t^n = 0$. In this way, the probability q_t^n controls the rate of computation, where a larger q_t^n indicates that more computation is done (in expectation), consuming more computation resources, and vice versa.

2) Client-to-Server Communication: Each client n keeps a *residual error*, which is a vector that contains portions of the *changes* in the model parameter $\mathbf{x}$ that have not been transmitted from the client yet. The residual error of client n at the beginning of iteration t is denoted by $\mathbf{e}_t^n$. In Line 6, the new SGD update is accumulated on $\mathbf{e}_t^n$, giving a new temporary vector denoted by $\mathbf{b}_t^n$. Then, in Line 7, the (usually sparse or quantized) vector that is transmitted to the server (i.e., $\mathbf{v}_t^n$) is subtracted from $\mathbf{b}_t^n$, and the remaining quantity that is not transmitted is kept in $\mathbf{e}_{t+1}^n$, which is the residual error at the beginning of the next iteration $t+1$. The vector $\mathbf{v}_t^n$ is usually a compression result of $\mathbf{b}_t^n$. We will describe in Section IV how $\mathbf{v}_t^n$ is computed, with more specific examples in Section IV-E.

3) Multiple Local Computations and Decoupling: When $\mathbf{v}_t^n = \mathbf{0}$, we do not transmit in this iteration, which captures the case of multiple rounds of computation before communication happens. Noting that $\mathbf{v}_t^n = \mathbf{0}$ corresponds to no transmission by client n in iteration t, we emphasize that we can have $\mathbf{v}_t^n = \mathbf{0}$ even if $\mathbb{1}_t^n = 1$, or $\mathbf{v}_t^n \neq \mathbf{0}$ even if $\mathbb{1}_t^n = 0$. In this way, the computation and communication decisions can be *decoupled*. When some iterations have low computation cost but high communication cost, while other iterations have high computation cost but low communication cost, we may decide to compute in those iterations with low computation cost, and transmit in other iterations with low communication cost.

4) Server-to-Client Communication: After receiving the parameter updates from clients, the server averages $\{\mathbf{v}_t^n : \forall n\}$ and adds the result to its own residual error $\mathbf{r}_t$ (Line 10). If a client n does not transmit any update, the server considers $\mathbf{v}_t^n = \mathbf{0}$ for this client n and it is still included in computing the average. Then, similar to the operation at clients, a (usually sparse or quantized) vector $\mathbf{u}_t$ is transmitted to all the clients and the remaining part is kept in the residual error $\mathbf{r}_{t+1}$ for the next iteration $t+1$ (Line 11). We consider a broadcast channel from the server to the clients, hence the information sent to all the clients is the same.

Finally, each client updates its current model parameter $\mathbf{x}_t$ after receiving the update $\mathbf{u}_t$ from the server (Line 14).

C. Convergence Analysis

We analyze the convergence upper bound of Algorithm 1. First, we introduce a minimal set of assumptions that are commonly used in the literature [43].

Assumption 1. *We assume that the following hold,* $\forall n, \mathbf{x}, \mathbf{y}$.

Lipschitz gradient:

$$\|\nabla F_n(\mathbf{x}) - \nabla F_n(\mathbf{y})\| \leq L \|\mathbf{x} - \mathbf{y}\|. \tag{2}$$

Unbiased stochastic gradient with bounded variance:

$$\mathsf{E}[\mathbf{g}_n(\mathbf{x})] = \nabla F_n(\mathbf{x}) \text{ and } \mathsf{E}\left[\|\mathbf{g}_n(\mathbf{x}) - \nabla F_n(\mathbf{x})\|^2\right] \leq \sigma^2. \tag{3}$$

Bounded gradient divergence:

$$\|\nabla F_n(\mathbf{x}) - \nabla f(\mathbf{x})\|^2 \leq \epsilon^2. \tag{4}$$

The gradient divergence bound ϵ^2 captures the degree of heterogeneous (i.e., non-i.i.d.) data across clients. We now introduce our main convergence result (proof is in the appendix).

Theorem 1. *When Assumption 1 holds, if* $\frac{1}{N}\sum_{n=1}^N \frac{1}{q_t^n} \leq p$, *for all* t, *and* $\eta \leq \frac{1}{4Lp}$, *then Algorithm 1 ensures that*

$$\begin{aligned}\frac{1}{T}\sum_{t=0}^{T-1} \mathsf{E}\left[\|\nabla f(\mathbf{x}_t)\|^2\right] \leq{}& \frac{4(f(\mathbf{x}_0) - f^*)}{T} \\ &+ \frac{4L^2}{T}\sum_{t=0}^{T-1} \mathsf{E}\left[\|\mathbf{r}_t\|^2\right] + \frac{4L^2}{NT}\sum_{t=0}^{T-1}\sum_{n=1}^N \mathsf{E}\left[\|\mathbf{e}_t^n\|^2\right] \\ &+ \frac{4\eta L(\sigma^2 + \epsilon^2)}{NT}\sum_{t=0}^{T-1}\sum_{n=1}^N \mathsf{E}\left[\frac{1}{q_t^n}\right],\end{aligned} \tag{5}$$

where f^* *is the true minimum of* $f(\mathbf{x})$, *i.e.,* $f^* := \min_{\mathbf{x}} f(\mathbf{x})$.

In Theorem 1, we capture the convergence error by the time-averaged expected squared norm of the gradient. We see that the upper bound of the convergence error increases in the squared norm of residual errors $\mathbf{r}_t$ and $\mathbf{e}_t^n$ and decreases in the probability of local computation q_t^n. This observation aligns with the intuition that, in general, more communication

and computation can improve the convergence with respect to *the number of iterations*. However, doing so would also incur *higher costs* of resource usage. Therefore, we need to *strike a balance between convergence error and resource cost*, after a certain number of iterations T. In the optimization problem presented in the next section, we aim at minimizing the convergence error under pre-defined cost constraints.

Before proceeding, we note that the decision variables $\mathbf{v}_t^n$ and $\mathbf{u}_t$ do not explicitly appear in the result in Theorem 1. For ease of presentation later, we give the following alternative upper bound that is derived from Theorem 1. Because $\mathbf{r}_0 = \mathbf{0}$ and $\mathbf{e}_0^n = \mathbf{0}$, $\forall n$, according to Algorithm 1, we can further bound the terms in (5) in the following way:

$$\begin{aligned}\textstyle\sum_{t=0}^{T-1} \mathsf{E}\left[\|\mathbf{r}_t\|^2\right] &\leq \textstyle\sum_{t=1}^{T} \mathsf{E}\left[\|\mathbf{r}_t\|^2\right] \\ &= \textstyle\sum_{t=0}^{T-1} \mathsf{E}\left[\|\mathbf{r}_t + \frac{1}{N}\sum_{n=1}^{N} \mathbf{v}_t^n - \mathbf{u}_t\|^2\right], \quad (6)\\ \textstyle\sum_{t=0}^{T-1} \mathsf{E}\left[\|\mathbf{e}_t^n\|^2\right] &\leq \textstyle\sum_{t=1}^{T} \mathsf{E}\left[\|\mathbf{e}_t^n\|^2\right] \\ &= \textstyle\sum_{t=0}^{T-1} \mathsf{E}\left[\left\|\mathbf{e}_t^n - \frac{\mathbb{I}_t^n}{q_t^n}\cdot \mathbf{g}_n(\mathbf{x}_t) - \mathbf{v}_t^n\right\|^2\right], \forall n. \quad (7)\end{aligned}$$

Corollary 1. *Under the same conditions as in Theorem 1, an alternative upper bound of* $\frac{1}{T}\sum_{t=0}^{T-1} \mathsf{E}\left[\|\nabla f(\mathbf{x}_t)\|^2\right]$ *holds by replacing the corresponding terms in* (5) *with* (6) *and* (7)*.*

IV. Control Decisions

A. Problem Formulation

The goal of our decision making problem is to determine the set of control parameters $\{q_t^n\}$, $\{\mathbf{v}_t^n\}$, $\{\mathbf{u}_t\}$ over time, to minimize the convergence error subject to resource cost constraints. Similar to existing works [5], [12], [16]–[19], we use the convergence upper bound as an approximation to the actual error, because it is generally not possible to know exactly how different configurations affect the actual error.

1) Instantaneous Costs: Let $\lambda_t^n(q_t^n)$ denote the *computation cost* in iteration t at client n. Also let $\varphi_t^n(\mathbf{v}_t^n)$ and $\psi_t(\mathbf{u}_t)$ denote the *communication cost* at client n and the server, respectively, both in iteration t. Note that the cost functions $\lambda_t^n(\cdot)$, $\varphi_t^n(\cdot)$, and $\psi_t(\cdot)$ themselves can be different for different t and n. That means, even if $q_t^n = q_{t'}^n$ for $t \neq t'$, we may have $\lambda_t^n(q_t^n) \neq \lambda_{t'}^n(q_{t'}^n)$, for instance. When there is no ambiguity, we omit the arguments q_t^n, $\mathbf{v}_t^n$, and $\mathbf{u}_t$ for simplicity, and only write λ_t^n, φ_t^n, and ψ_t which are implicitly dependent on q_t^n, $\mathbf{v}_t^n$, and $\mathbf{u}_t$, respectively.

2) Target Average Costs (Constraints): We further denote the *target time-averaged computation cost* by $\bar{\lambda}_n$ (at client n), and the *target time-averaged communication costs* by $\bar{\varphi}_n$ (at client n) and $\bar{\psi}$ (at the server). These target costs are given as inputs to our control problem. They represent how much cost each of the clients and the server would like to spend on average for the FL task. The notion of *cost* in this paper represents a generic metric. For example, it can stand for the percentage of consumed resources among all the available resources, monetary cost, energy usage, or a combination of these and other possible measures.

3) Overall Control Problem: With these definitions, we are ready to introduce our problem of minimizing the convergence upper bound given by Corollary 1 under cost constraints. Ignoring constants and common coefficients, we first define the following objective:

$$\begin{aligned}\mathcal{G} := \; & \tfrac{L}{T}\textstyle\sum_{t=0}^{T-1} \mathsf{E}\left[\left\|\mathbf{r}_t + \frac{1}{N}\sum_{n=1}^{N} \mathbf{v}_t^n - \mathbf{u}_t\right\|^2\right] \\ & + \tfrac{L}{NT}\textstyle\sum_{t=0}^{T-1}\sum_{n=1}^{N} \mathsf{E}\left[\left\|\mathbf{e}_t^n - \frac{\mathbb{I}_t^n}{q_t^n}\cdot \mathbf{g}_n(\mathbf{x}_t) - \mathbf{v}_t^n\right\|^2\right] \\ & + \tfrac{(\gamma^2+\sigma^2)}{NT}\textstyle\sum_{t=0}^{T-1}\sum_{n=1}^{N} \mathsf{E}\left[\frac{1}{q_t^n}\right]. \quad (8)\end{aligned}$$

Then, our overall optimization problem is as follows:

$$\begin{aligned}\textbf{P1:}\ \min_{\{q_t^n\},\{\mathbf{v}_t^n\},\{\mathbf{u}_t\}} \quad & \mathcal{G} && (9)\\ \text{s.t.} \quad & \tfrac{1}{T}\textstyle\sum_{t=0}^{T-1} \mathsf{E}\left[\lambda_t^n\right] \leq \bar{\lambda}_n, \ \forall n && (10)\\ & \tfrac{1}{T}\textstyle\sum_{t=0}^{T-1} \mathsf{E}\left[\varphi_t^n\right] \leq \bar{\varphi}_n, \ \forall n && (11)\\ & \tfrac{1}{T}\textstyle\sum_{t=0}^{T-1} \mathsf{E}\left[\psi_t\right] \leq \bar{\psi}. && (12)\end{aligned}$$

4) Challenges: There are several challenges in solving the problem P1 directly. *First,* there are three terms in the objective $\mathcal{G}$ defined in (8), which have different coefficients. It is generally *difficult to estimate these coefficients* as they are related to characteristics of loss functions and their stochastic gradients. *Second,* in each iteration t of Algorithm 1, there is a sequential order that first determines (according to $\mathbb{I}_t^n$) whether each client n computes an update, then transmits the update vector from clients to the server and finally from the server to clients. Considering the second term of (8), in practice, $\mathbf{g}_n(\mathbf{x}_t)$ is only computed if $\mathbb{I}_t^n = 1$, but the value of $\mathbb{I}_t^n$ is *unknown* before the value of q_t^n is determined. Thus, we cannot know $\mathbf{g}_n(\mathbf{x}_t)$ when determining q_t^n, which makes it impossible to use the exact value of the second term of (8) in the determination of q_t^n. Similarly, the value of $\frac{1}{N}\sum_{n=1}^{N} \mathbf{v}_t^n$ in the first term of (8) is *unknown* before each client n has actually computed its $\mathbf{v}_t^n$. *Third,* the overall impact of control decisions is *correlated across different iterations* through both the objective function and constraints, but we do not have prior knowledge of resource costs in practice. Therefore, we need an online algorithm that does not rely on prior knowledge.

To overcome these challenges, we first approximate P1 with three sub-problems that sequentially determine $\{q_t^n\}$, $\{\mathbf{v}_t^n\}$, and $\{\mathbf{u}_t\}$ in Section IV-B. Then, we present an online algorithm for each sub-problem in Section IV-C.

B. Approximation by Sequential Decision Making

We decompose P1 into three sub-problems as follows. In each sub-problem, one set of decision variables is determined by minimizing its corresponding term in (8). We substitute expressions inside the norms using the definitions of $\mathbf{b}_t^n$ and $\mathbf{a}_t$ in Line 6 and Line 10 of Algorithm 1, respectively.

$$\begin{aligned}\textbf{P2.1:}\ \min_{\{q_t^n\}} \quad & \tfrac{1}{NT}\textstyle\sum_t\sum_n \mathsf{E}\left[\frac{1}{q_t^n}\right] && (13)\\ \text{s.t.} \quad & \text{Constraint (10).}\end{aligned}$$

$$\begin{aligned}\textbf{P2.2:}\ \min_{\{\mathbf{v}_t^n\}} \quad & \tfrac{1}{NT}\textstyle\sum_t\sum_n \mathsf{E}\left[\|\mathbf{b}_t^n - \mathbf{v}_t^n\|^2\right] && (14)\\ \text{s.t.} \quad & \text{Constraint (11).}\end{aligned}$$

$$\textbf{P2.3:} \quad \min_{\{\mathbf{u}_t\}} \quad \frac{1}{T}\sum_t \mathsf{E}\left[\|\mathbf{a}_t - \mathbf{u}_t\|^2\right] \tag{15}$$
$$\text{s.t.} \quad \text{Constraint (12).}$$

With this decomposition, we first solve P2.1 to obtain $\{q_t^n\}$. Then, we consider $\{q_t^n\}$ as given and solve for $\{\mathbf{v}_t^n\}$ in P2.2. Finally, we consider $\{\mathbf{v}_t^n\}$ as given and solve for $\{\mathbf{u}_t\}$ in P2.3. We can regard this sequential decision-making procedure as an approximation to the original problem P1. The exact approximation error is difficult to analyze and is left for future work. However, we will see in Section V that the solution obtained by this approximation, together with the online algorithm described in Section IV-C, provides performance gain compared to baselines in experiments.

C. Online Decision Making

The problems P2.1–P2.3 are still difficult to solve directly, because both the objective functions and constraints are averaged over time, and it is difficult to predict future costs in practice. Therefore, we present an online decision making approach in the following, where the quantities q_t^n, $\mathbf{v}_t^n$, and $\mathbf{u}_t$ are determined within each iteration t without knowledge of statistics in future iterations.

1) Methodology and Challenges: Our approach is based on the Lyapunov drift-plus-penalty framework [47], but with some *notable differences*. First, while infinite T is the primary focus in [47], we allow *finite* T in this paper, both in the problem formulation (see P1 and P2 above) and in our analysis later in Section IV-D. This consideration is because, in practice, we usually train the model only for a finite number of iterations. Second, while P2.1 can depend on an underlying system state (i.e., $\omega(t)$ defined in [47]) that is independent across time t, we emphasize that the underlying states of P2.2 and P2.3 are both *time-dependent and also dependent on previous decisions* made by the control algorithm. To see this, note that the quantity $\mathbf{b}_t^n$ in P2.2 depends on the stochastic gradient computed on the model parameter $\mathbf{x}_t$, and the value of $\mathbf{x}_t$ is related to the decisions on $q^n, \mathbf{v}^n, \mathbf{u}$ made in previous iterations $\tau < t$. Similarly, $\mathbf{a}_t$ in P2.3 also depends on past decisions and other random outcomes. This dependency makes it substantially harder to analyze P2.2 and P2.3, where the standard results in [47] no longer hold.

2) Virtual Queues: We define virtual queues to capture the constraints (10)–(12). The virtual queues lengths Θ_t^n, Φ_t^n, and Ψ_t evolve according to the following recursions:[4]

$$\Theta_{t+1}^n = \max\{0, \Theta_t^n + \lambda_t^n - \bar{\lambda}_n\}, \quad \forall n, \tag{16}$$
$$\Phi_{t+1}^n = \max\{0, \Phi_t^n + \varphi_t^n - \bar{\varphi}_n\}, \quad \forall n, \tag{17}$$
$$\Psi_{t+1} = \max\{0, \Psi_t + \psi_t - \tilde{\psi}\}. \tag{18}$$

Intuitively, these virtual queues capture the accumulated violation of constraints (10)–(12). Hence, we would like to jointly minimize the objectives (13)–(15) and the virtual queue lengths. In our problem formulation (P1 and P2), the cost definitions and their constraints are separate across clients and the server, thus we can *distributedly* optimize for each entity separately. An extension to settings with coupled cost constraints is possible by sharing queue length information across clients and the server.

[4]In practice, we may set the minimum queue length to a very small positive number instead of zero, to avoid large instantaneous costs from being incurred and added to the queue (see also the objectives of problem P3).

Algorithm 2: Online Control

Constants: $V > 0$, initial queue length $W \geq 0$
Output: $\{q_t^n\}$, $\{\mathbf{v}_t^n\}$, $\{\mathbf{u}_t\}$
Run Lines 1–2 in Algorithm 1;
$\Theta_0^n = W, \forall n$; $\Phi_0^n = W, \forall n$; $\Psi_0 = W$;
for $t \leftarrow 0, \ldots, T-1$ **do**
  each client $n \leftarrow 1, \ldots, N$ **in parallel:**
    Get q_t^n from P3.1 and update Θ_t^n using (16);
    Run Lines 5–6 of Algorithm 1;
    Get $\mathbf{v}_t^n$ from P3.2 and update Φ_t^n using (17);
    Run Lines 7–8 of Algorithm 1;
  the server:
    Run Line 10 of Algorithm 1;
    Get $\mathbf{u}_t$ from P3.3 and update Ψ_t using (18);
    Run Lines 11–12 of Algorithm 1;
  Run Lines 13–14 of Algorithm 1;

3) Decision Problem for Each Iteration: Define a constant $V > 0$ that will be discussed further in Section IV-D. We have the following drift-plus-penalty minimization problems for each client n (P3.1 and P3.2) and server (P3.3) in iteration t.

$$\textbf{P3.1:} \quad \min_{q_t^n} \quad \frac{V}{q_t^n} + \Theta_t^n\left(\lambda_t^n - \bar{\lambda}_n\right). \tag{19}$$
$$\textbf{P3.2:} \quad \min_{\mathbf{v}_t^n} \quad V\|\mathbf{b}_t^n - \mathbf{v}_t^n\|^2 + \Phi_t^n\left(\varphi_t^n - \bar{\varphi}_n\right). \tag{20}$$
$$\textbf{P3.3:} \quad \min_{\mathbf{u}_t} \quad V\|\mathbf{a}_t - \mathbf{u}_t\|^2 + \Psi_t\left(\psi_t - \tilde{\psi}\right). \tag{21}$$

Note that when solving P3.1–P3.3, we consider the virtual queue lengths Θ_t^n, Φ_t^n, and Ψ_t as well as the vectors $\mathbf{b}_t^n$ and $\mathbf{a}_t$ as given variables. However, these variables are inherently random due to the random noise in stochastic gradient and probabilistic client sampling, so the objectives and constraints in P1 and P2 are expressed as expectations.

The control decisions obtained from P3.1–P3.3 and virtual queue updates (16)–(18) are combined with Algorithm 1 to provide the values of control variables. The full procedure is shown in Algorithm 2, where we may choose a non-zero initial queue length W to prevent a high degree of constraint violation in initial iterations (see (19)–(21) and Section IV-D).[5]

D. Analysis of Online Control Algorithm

We discuss the optimality and constraint satisfaction of approximately solving P2.1–P2.3 via minimizing the drift-plus-penalty objectives (19)–(21) in P3.1–P3.3, as in Algorithm 2. Our discussion shares similarities with [47]. However, there are some key differences and challenges as discussed in Section IV-C1. With a slight abuse of notation, we reuse $q_t^n, \mathbf{v}_t^n, \mathbf{u}_t, \lambda_t^n, \varphi_t^n, \psi_t$ to denote the control variables and their corresponding costs obtained from the *solutions* of P3.1–P3.3. We first make an assumption to facilitate the analysis.

Assumption 2. *We assume that* i) $q_t^n \in [1/D, 1]$, ii) $\|\mathbf{b}_t^n\|^2, \|\mathbf{a}_t\|^2 \in [0, D]$, iii) $\lambda_t^n, \varphi_t^n, \psi_t, \bar{\lambda}_n, \bar{\varphi}_n, \tilde{\psi} \in$

[5]The idea of setting initial values for (virtual) queues is called place-holder backlog in [47], but its original goal is to improve the performance-delay trade-off when $T \to \infty$. In contrast, we consider finite T in our case, and the "place-holder backlog" can guarantee arbitrarily small constraint violation.

$[0, \sqrt{2B}]$, iv) $\tilde{\lambda}_n \geq \lambda_t^n(\frac{1}{D})$ *(i.e., cost computed at* $q_t^n = 1/D$*),* v) $\varphi_t^n(\mathbf{0}) = \psi_t(\mathbf{0}) = 0$*, for some* $D > 0, B > 0$.

In this assumption, the bound on q_t^n usually holds for some $D > 0$ as long as the virtual queue length θ_t^n is bounded. The rationale behind the bounds on $\|\mathbf{b}_t^n\|^2$ and $\|\mathbf{a}_t\|^2$ is that, although the residual errors are accumulated over time, they usually will not be arbitrarily large because the parameter updates get smaller when the gradient approaches zero. Note that this is only needed for Theorems 2 and 3 below, while our algorithm can still work empirically without Assumption 2.

Theorem 2. *Under Assumption 2, solving P3.1–P3.3 for each* t *ensures the following bounds on constraint violation:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathsf{E}[\lambda_t^n] - \tilde{\lambda}_n \leq \sqrt{\frac{W^2}{T^2} + \frac{2VD+2B}{T}} - \frac{W}{T}, \quad (22)$$

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathsf{E}[\varphi_t^n] - \tilde{\varphi}_n \leq \sqrt{\frac{W^2}{T^2} + \frac{2VD+2B}{T}} - \frac{W}{T}, \quad (23)$$

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathsf{E}[\psi_t] - \tilde{\psi} \leq \sqrt{\frac{W^2}{T^2} + \frac{2VD+2B}{T}} - \frac{W}{T}. \quad (24)$$

Proof. The Lyapunov drift of virtual queue length θ_t^n is

$$\begin{aligned}\Delta(\theta_t^n) &:= \tfrac{1}{2}\left[(\theta_{t+1}^n)^2 - (\theta_t^n)^2\right] \leq \tfrac{1}{2}\left[\left(\theta_t^n + \lambda_t^n - \tilde{\lambda}_n\right)^2 - (\theta_t^n)^2\right] \\ &= \theta_t^n\left(\lambda_t^n - \tilde{\lambda}_n\right) + \frac{(\lambda_t^n - \tilde{\lambda}_n)^2}{2} \leq \theta_t^n\left(\lambda_t^n - \tilde{\lambda}_n\right) + B.\end{aligned}$$

We have $\theta_t^n\left(\lambda_t^n - \tilde{\lambda}_n\right) \leq VD$, because otherwise setting $q_t^n = \frac{1}{D}$ will give a smaller value of the objective (19) due to Assumption 2. Thus, $\Delta(\theta_t^n) \leq VD + B$. We further note that

$$\tfrac{1}{2}(\theta_T^n)^2 - \tfrac{1}{2}(\theta_0^n)^2 = \sum_{t=0}^{T-1}\Delta(\theta_t^n) \leq VDT + BT.$$

Hence, $\theta_T^n \leq \sqrt{(\theta_0^n)^2 + 2VDT + 2BT}$. From (16), we have $\theta_t^n + \lambda_t^n - \tilde{\lambda}_n \leq \theta_{t+1}^n$, thus $\lambda_t^n - \tilde{\lambda}_n \leq \theta_{t+1}^n - \theta_t^n$. This gives

$$\frac{1}{T}\sum_{t=0}^{T-1}\lambda_t^n - \tilde{\lambda}_n \leq \frac{\theta_T^n - \theta_0^n}{T} \leq \sqrt{\frac{W^2}{T^2} + \frac{2VD+2B}{T}} - \frac{W}{T},$$

where we recall that $\theta_0^n = W$. After taking expectation on both sides, we have proven (22). The results in (23) and (24) can be proven using a similar procedure. □

Theorem 3. *Under Assumption 2, solving P3.1–P3.3 for each* t *gives the following bounds related to the objectives* (13)–(15) *of P2.1–P2.3:*

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathsf{E}\left[\frac{1}{q_t^n}\right] \leq \mathsf{OPT}_{q_n} + \frac{B}{V} + \frac{W^2}{2VT}, \quad (25)$$

$$\|\mathbf{b}_t^n - \mathbf{v}_t^n\|^2 \leq \min\left\{D, \frac{\theta_t^n\sqrt{2B}}{V}\right\}, \quad (26)$$

$$\|\mathbf{a}_t - \mathbf{u}_t\|^2 \leq \min\left\{D, \frac{\theta_t^n\sqrt{2B}}{V}\right\}. \quad (27)$$

where OPT_{q_n} *denotes the optimal value of the time-averaged objective given by a possibly randomized offline algorithm that has complete statistics of all* T *iterations.*

Proof. The inequality (25) can be directly obtained from the proof of Theorem 4.8 in [47]. To prove (26), we consider an alternative choice of $\mathbf{v}_t^n$, denoted by $\mathbf{v}_t^{\prime n} = \mathbf{b}_t^n$, which makes the objective (20) equal to $\theta_t^n(\varphi_t^n(\mathbf{b}_t^n) - \tilde{\varphi}_n)$. Since $\mathbf{v}_t^n$ is the optimal solution to P3.2, we have

$$V\|\mathbf{b}_t^n - \mathbf{v}_t^n\|^2 + \theta_t^n(\varphi_t^n(\mathbf{v}_t^n) - \tilde{\varphi}_n) \leq \theta_t^n(\varphi_t^n(\mathbf{b}_t^n) - \tilde{\varphi}_n).$$

Rearranging gives

$$V\|\mathbf{b}_t^n - \mathbf{v}_t^n\|^2 \leq \theta_t^n(\varphi_t^n(\mathbf{b}_t^n) - \varphi_t^n(\mathbf{v}_t^n)).$$

Then, we note that $\varphi_t^n(\mathbf{b}_t^n) \in [0, \sqrt{2B}]$ and $\varphi_t^n(\mathbf{v}_t^n) \in [0, \sqrt{2B}]$ according to Assumption 2 and divide by V on both sides. We also have $\|\mathbf{b}_t^n - \mathbf{v}_t^n\|^2 \leq \|\mathbf{b}_t^n\|^2 \leq D$, because otherwise choosing $\mathbf{v}_t^n = \mathbf{0}$ gives a smaller value of (20), where we note that $\varphi_t^n(\mathbf{0}) = 0$ and $\|\mathbf{b}_t^n\|^2 \leq D$ according to Assumption 2 and $\theta_t^n \geq 0$. Combining the above gives (26). The result in (27) can be shown similarly. □

Insights: We first discuss some important insights provided by Theorem 2. *1)* Theorem 2 shows that the constraints (10)–(12) are satisfied as $T \to \infty$. This is a desirable property of the drift-plus-penalty algorithm in time-independent settings [47]. Here, we have shown that although our objectives $\|\mathbf{b}_t^n - \mathbf{v}_t^n\|^2$ and $\|\mathbf{a}_t - \mathbf{u}_t\|^2$ are correlated with past decisions over time (see Section IV-C1), we can still guarantee zero constraint violation when running the algorithm for a sufficiently long time. *2)* By taking the derivative with respect to W, we can further see that the RHS of (22)–(24) decreases in W. For a finite T, as W gets large, we will have $\sqrt{\frac{W^2}{T^2} + \frac{2VD+2B}{T}} - \frac{W}{T} \approx \frac{W}{T} - \frac{W}{T} = 0$. Assume that we require the RHS of (22)–(24) to be not larger than ν, for some $\nu > 0$. For any finite T, we can always find a value of W so that this requirement is satisfied. Therefore, the introduction of W in our approach *extends the constraint satisfaction* from infinite T, which is the primary focus of [47], *to finite* T. *3)* When T gets large, the dominant term in the RHS of (22)–(24) becomes $\mathcal{O}\left(\sqrt{V/T}\right)$. This shows that W controls the constraint satisfaction primarily for small T, while the effect of V becomes more prominent for large T.

Next, we discuss Theorem 3. In (25), we observe an additive optimality gap of $\mathcal{O}\left(\frac{1}{V} + \frac{W^2}{VT}\right)$. This result is similar to that in [47], because the objective $1/q_t^n$ here only depends on the decision (i.e., q_t^n) made in the current iteration t. However, because $\|\mathbf{b}_t^n - \mathbf{v}_t^n\|^2$ and $\|\mathbf{a}_t - \mathbf{u}_t\|^2$ depend on past decisions, the same result does not hold for them. Nevertheless, according to the queue-length dependent bounds in (26) and (27), the main insight that the optimality error decreases in V still holds. How to obtain a queue-independent bound for the $\|\mathbf{b}_t^n - \mathbf{v}_t^n\|^2$ and $\|\mathbf{a}_t - \mathbf{u}_t\|^2$ objectives is left for future work.

Combining the above, we have the following key insight on the parameters W and V. Increasing W or decreasing V improves constraint satisfaction but makes the objective function value less optimal, and vice versa. In addition, W primarily affects the short-term performance with finite T, while V affects the long-term performance. These observations are useful to guide the tuning of W and V, so that a desired trade-off between optimality and constraint satisfaction can be achieved.

E. Specific Costs and Compression Methods

Next, we give closed-form solutions to P3.1–P3.3 for some exemplar cost functions that have specific forms with respect to their inputs, and also discuss compression methods to obtain $\mathbf{v}_t^n$ and $\mathbf{u}_t$ from $\mathbf{b}_t^n$ and $\mathbf{a}_t$, respectively.

1) Linear Computation Cost and Solution to P3.1: Consider a linear computation cost defined as $\lambda_t^n = \alpha_t^n q_t^n$ for some $\alpha_t^n > 0$. The rationale behind this definition is that the expected amount of computation (e.g., number of CPU or GPU cycles) is usually proportional to the probability q_t^n. With this definition, we can see that the objective (19) of P3.1 is convex in q_t^n. By letting the derivative of (19) equal to zero and noting that the probability $q_t^n \in [0, 1]$, we obtain the optimal solution $\hat{q}_t^n := \arg\min_{q_t^n} \frac{V}{q_t^n} + \gamma_t^n \left(\alpha_t^n q_t^n - \bar{\lambda}_n\right)$ as:

$$\hat{q}_t^n = \min\left\{1, \sqrt{\frac{V}{\gamma_t^n \alpha_t^n}}\right\}. \tag{28}$$

2) Transmitting Compressed Update Vectors: Regardless of the exact definition of the communication cost, the vector $\mathbf{v}_t^n$ is usually derived from $\mathbf{b}_t^n$. In a widely applied compression method known as *top-k sparsification* [33]–[35], the k components of $\mathbf{b}_t^n$ with the largest magnitudes are included in $\mathbf{v}_t^n$ and transmitted to the server, while the remaining components that are not transmitted are kept in $\mathbf{e}_{t+1}^n$ for possible transmission in future iterations. When k is small, the vector $\mathbf{v}_t^n$ is usually represented as a sparse vector by index-value pairs, so that only the k selected components are transmitted. By tuning k (and the corresponding $\mathbf{v}_t^n$), we can adjust the communication cost. The vector $\mathbf{u}_t$ can be obtained from $\mathbf{a}_t$ similarly. There are other parameter compression methods such as quantization [29]–[32]. We mainly focus on top-k sparsification in subsequent discussion and experiments, while noting that the same insights also apply to other compression methods.

3) Constant-Plus-Linear Communication Cost and Solutions to P3.2–P3.3: For the communication cost, we consider a definition that includes a constant cost portion β_t^n whenever communication occurs. This constant portion captures the additional overhead caused by packet headers and any other necessary control information. In addition, there is a linear portion of the cost that is γ_t^n times the amount of information transmitted. As in top-k sparsification, $\mathbf{v}_t^n$ and $\mathbf{u}_t$ include some components of $\mathbf{b}_t^n$ and $\mathbf{a}_t^n$, respectively. Then, the number of non-zero components (floating-point numbers) in $\mathbf{v}_t^n$ or $\mathbf{u}_t$ represents the amount of communication. Based on this description, the cost φ_t^n is expressed as

$$\varphi_t^n = \begin{cases} 0, & \text{if } \|\mathbf{v}_t^n\|_0 = 0 \\ \beta_t^n + \gamma_t^n \|\mathbf{v}_t^n\|_0, & \text{if } \|\mathbf{v}_t^n\|_0 > 0 \end{cases}, \tag{29}$$

where $\beta_t^n \geq 0$, $\gamma_t^n > 0$, and $\|\cdot\|_0$ denotes the ℓ_0 norm that counts the number of non-zero elements in the vector.

For a given number of non-zero components in $\mathbf{v}_t^n$, it is apparent that the objective (20) of P3.2 is minimized by choosing $\mathbf{v}_t^n$ to include the $\|\mathbf{v}_t^n\|_0$ largest components (in terms of magnitude) in $\mathbf{b}_t^n$. This is exactly the top-k sparsification method with $k = \|\mathbf{v}_t^n\|_0$. Now, the question is how to determine k. Let $(b_t^n)_i$ denote the i-th largest component in $\mathbf{b}_t^n$; we note that the change in the value of the objective (20) from $\|\mathbf{v}_t^n\|_0 = j-1$ to $\|\mathbf{v}_t^n\|_0 = j$ is $\gamma_t^n \gamma_t^n - V(b_t^n)_j^2$ (recall the definition of φ_t^n in (29)), for $j \geq 2$. The quantity $\gamma_t^n \gamma_t^n - V(b_t^n)_j^2$ does not decrease in j because $\{(b_t^n)_i : \forall i\}$ is sorted in descending order. Therefore, for a specific j such that $\gamma_t^n \gamma_t^n - V(b_t^n)_j^2 \geq 0$, including more than j non-zero components in $\mathbf{v}_t^n$ for transmission cannot make the objective (20) smaller. Due to the discontinuity in φ_t^n when switching from $\|\mathbf{v}_t^n\|_0 = 0$ to $\|\mathbf{v}_t^n\|_0 = 1$, we also need to check the value of the objective (20) for the case of $\varphi_t^n = 0$. Let j denote the smallest j such that $\gamma_t^n \gamma_t^n - V(b_t^n)_j^2 \geq 0$. We have the following expression for the optimal number of components to transmit:

$$k = \begin{cases} 0, \text{ if } V\|\mathbf{b}_t^n\|^2 \leq V\|\mathbf{b}_t^n - \mathbf{v}_t^n|_j\|^2 + \gamma_t^n(\beta_t^n + \gamma_t^n j) \\ j, \text{ otherwise} \end{cases}, \tag{30}$$

where $\mathbf{v}_t^n|_j \in \mathbb{R}^d$ denotes the vector that includes the j largest components in $\mathbf{b}_t^n$. Finally, the optimal $\hat{\mathbf{v}}_t^n$ is obtained by setting it to $\mathbf{v}_t^n|_k$, which includes the k largest components in $\mathbf{b}_t^n$. The solution to P3.3 has the same form after replacing the corresponding variables.

We conclude that P3.1–P3.3 with these simple but realistic cost definitions can be solved efficiently, while noting that our control algorithm works with other cost definitions too.

V. Experiments

A. Setup

1) Datasets and Models: We ran experiments of applying our approach to train models on image datasets. We consider two model and dataset combinations: *1)* a two-layer neural network with a hidden layer size of 50, trained on the Fashion-MNIST (FMNIST) dataset [48]; *2)* a convolutional neural network (CNN) with two convolutional + max-pool layers (3×3 kernel with padding, 32 filters, followed by 2×2 max-pool) and three fully-connected layers (of sizes 256, 64, 10), trained on the CIFAR-10 dataset [49]. We use ReLU activation functions (except for the last layer) and Kaiming initialization [50]. We simulate an FL system with 100 clients. Each dataset is partitioned in a non-i.i.d. manner so that each client only has data of one class (out of all the 10 classes), to simulate a challenging setup with high statistical heterogeneity.

2) Costs: The costs are defined according to the discussion in Section IV-E. For the computation cost, we assume that the linear coefficient α_t^n follows a uniform random distribution between 0 and 1. For the communication cost from clients to the server, we fix the constant portion to $\beta_t^n = 0.05$ to capture the overhead for headers, communication establishment, etc. The linear coefficient γ_t^n depends on the amount of channel usage, which is related to the channel capacity. Note that the Gaussian channel capacity is $C(\mathsf{SNR}) := \frac{1}{2}\log_2(1 + \mathsf{SNR})$ per channel use, where SNR denotes the signal-to-noise ratio (SNR). We define the linear portion of the communication cost $\gamma_t^n k$ as the number of channel use for transmitting k components with $\mathsf{SNR} = \zeta$, normalized by the number of channel use for transmitting the entire model with $d \geq k$ components and a fixed $\mathsf{SNR} = 1$. This gives $\gamma_t^n := \frac{1}{k} \cdot \frac{k}{C(\zeta)} \Big/ \frac{d}{C(1)} = \frac{1}{2dC(\zeta)}$. Here, we choose $\zeta \sim \chi_2^2$ to simulate a Rayleigh fading channel, where we note that the square of a Rayleigh-distributed channel gain follows chi-squared distribution with a degree of freedom of 2 (denoted by χ_2^2).

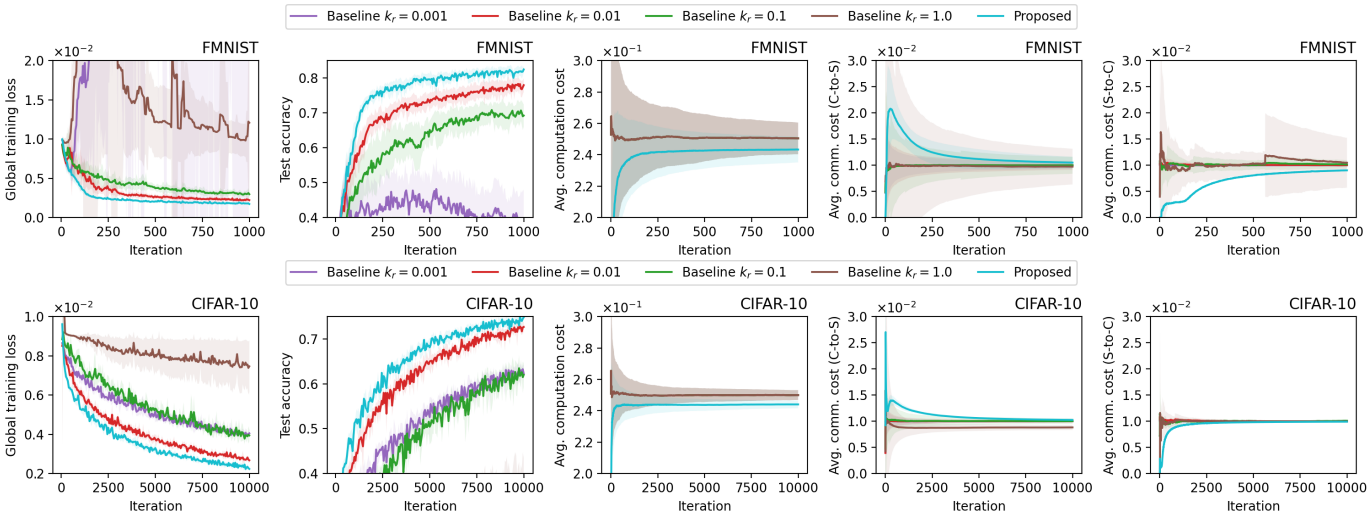

Fig. 1. FMNIST and CIFAR-10 results in comparison to baselines. The test accuracy for the baseline with $k_r := k/d = 1.0$ is below 0.4 so it is not visible in the accuracy plot. The average cost in the plots are computed up to the iteration index on the x-axis (C = client; S = server). The costs of the baseline method with different k_r largely overlap, therefore in many cases only one curve for the baseline is visible.

This definition of γ_t^n captures the random channel condition and makes the communication cost φ_t^n defined in (29) to scale only with the percentage of parameter components transmitted. The communication cost from the server to clients is defined in the same way, but it is scaled down by a factor of 5, because the downlink channel usually has higher bandwidth than the uplink channel. In general, the randomness in these cost definitions simulate random resource costs that can be time-varying and heterogeneous across clients and the server.

3) Baseline: In addition to our proposed FlexFL algorithm with online control, we also consider a baseline algorithm that either transmits k components or transmits nothing in each iteration t. When there are less than k non-zero elements in $\mathbf{v}_t^n$ or $\mathbf{u}_t$, the baseline only transmits those non-zero elements, which can be less than k. To conform to the resource constraints (10)–(12), the baseline makes a *randomized* decision of whether to transmit or not in each iteration t, so that the expected cost in each iteration is equal to the targeted average cost (either $\bar{\chi}_n$, $\bar{\varphi}_n$, or $\bar{\psi}$). The probability q_t^n is determined using a similar randomized approach by the baseline. Note, however, that when $\mathbf{v}_t^n$ or $\mathbf{u}_t$ has all zero entries, the expected cost of the baseline is also zero, which is smaller than constraint upper bounds ($\bar{\varphi}_n$ or $\bar{\psi}$). Thus, it is possible that the actual average communication cost of the baseline is slightly lower than the target (see Fig. 1). This baseline is a representative method that includes core ideas of a range of existing techniques. For example, it adapts the communication frequency based on cost constraints [4]–[6], supports partial client participation (computation) [13]–[20], and works with different sparsity values k [7]–[12]. We use this baseline instead of specific existing methods, because we are not aware of a method that captures the same set of cost constraints as in our work, and a comparison is only meaningful if the time-averaged cost constraints are aligned.

4) Other Parameters: We set the time-averaged constraints to $\bar{\chi}_n = 0.25$, $\bar{\varphi}_n = \bar{\psi} = 0.01$, to simulate an environment with limited communication resources. We also set the learning rate to $\eta = 0.1$ and the default parameters of our control algorithm $V = 0.02$ and $W = 1.0$. Each setting was run with 20 different random seeds for FMNIST and 5 different random seeds for CIFAR-10. In each plot, the curve shows the mean and the shaded area shows the standard deviation.

B. Results

1) Comparing to Baseline: We define $k_r := k/d$ as the ratio of the transmitted parameter components to the total number of components. A few observations from Fig. 1 are as follows.

First, our proposed method outperforms the baseline with different k_r values in both loss and accuracy values for both datasets (and models). This shows the advantage of our method that optimizes the convergence upper bound over time, which can choose different k_r depending on instantaneous cost and virtual queue lengths, and it is more flexible and performs better than fixing k_r as in the baseline.

Second, the average costs of our proposed method get close to or are below their target values when the number of iterations (i.e., T) is large enough. This aligns with our theory in Section IV-D that has shown the constraint violation is bounded and approaches zero when T gets large. It is also interesting to see that, in our proposed approach, the computation cost and server-to-client communication cost start from the lower end below the target value, while the client-to-server communication cost becomes larger than the target value in initial iterations but reduces later. This shows that the client-to-server communication is the main bottleneck with the current choice of cost and constraint parameters.

2) Comparing Different Configurations of V and W: The trade-off between constraint satisfaction and optimality can be tuned by V and W, as discussed in Section IV-D. We verify this using experiments and their results are shown in Figs. 2–3. Due to space limitation, we only show the accuracy, computation cost, and client-to-server communication cost for

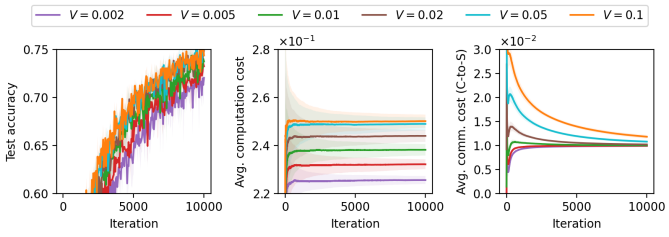

Fig. 2. Proposed method with $W = 1.0$ and different V (CIFAR-10).

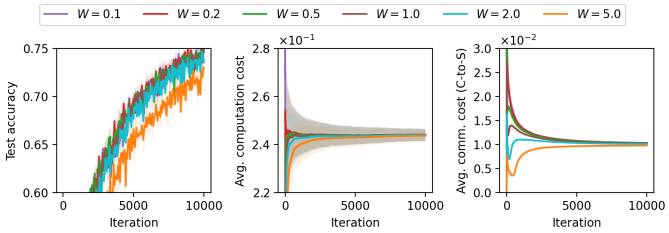

Fig. 3. Proposed method with $V = 0.02$ and different W (CIFAR-10).

CIFAR-10. The main observations remain the same for the other metrics and dataset. We can clearly see that the choice of W mainly affects the costs in initial iterations, while the choice of V has a more long-term effect. In addition, a smaller V or a larger W reduces the cost and gives a slightly lower accuracy. This aligns with our theoretical results in Section IV-D.

VI. Conclusion

In this paper, we have proposed FlexFL and its online control algorithm. FlexFL has a set of flexible control knobs to adjust the amount of computation and communication. It includes no communication as a special case and randomly decides whether to compute in each iteration according to an adjustable probability, therefore supporting multiple local computations and partial participation. By analyzing its convergence, we have provided a theoretical foundation on how the amount of computation and communication affect the model training performance. Accordingly, we have proposed a control algorithm to automatically determine the configuration parameters of FlexFL subject to time-averaged cost constraints. The control algorithm includes useful parameters V and W, which can be tuned to adjust the trade-off between constraint satisfaction and optimality. Our experiments show that coarsely chosen V and W can provide good results on two different datasets without the need of detailed tuning. If desired, V and W can be further tuned for fine-grained control.

There are direct extensions possible to our algorithm. For example, we may only optimize a subset of the configuration parameters in FlexFL, we can also choose different V and W for different types of costs and different entities. Moreover, our work provides a comprehensive methodology of optimizing multiple configuration options in FL using stochastic optimization, which can inspire future works.

Appendix: Proof of Theorem 1

We first note some preliminary inequalities that will be used throughout the proof. From Jensen's inequality, for any $\{\mathbf{z}_m \in \mathsf{R}^d : m \in \{1,2,\dots,M\}\}$, we have $\left\|\frac{1}{M}\sum_{m=1}^{M}\mathbf{z}_m\right\|^2 \leq \frac{1}{M}\sum_{m=1}^{M}\|\mathbf{z}_m\|^2$, which directly gives $\left\|\sum_{m=1}^{M}\mathbf{z}_m\right\|^2 \leq M\sum_{m=1}^{M}\|\mathbf{z}_m\|^2$. Peter-Paul inequality (also known as the generalized version of Young's inequality) gives $\langle \mathbf{z}_1, \mathbf{z}_2\rangle \leq \frac{k\|\mathbf{z}_1\|^2}{2} + \frac{k\|\mathbf{z}_2\|^2}{2}$, for any $\rho > 0$ and any $\mathbf{z}_1, \mathbf{z}_2 \in \mathsf{R}^d$. In addition, we use the notations in Algorithm 1. We also let $\mathsf{E}_t[\cdot] := \mathsf{E}[\cdot \mid \mathbf{x}_t, \mathbf{r}_t, \{\mathbf{e}_t^n\}]$. We define

$$\mathbf{x}_t := \mathbf{x}_t + \mathbf{r}_t + \frac{1}{N}\sum_{n=1}^{N}\mathbf{e}_t^n. \tag{31}$$

From Algorithm 1, we know that

$$\begin{aligned}\mathbf{x}_{t+1} &= \mathbf{x}_{t+1} + \mathbf{r}_{t+1} + \frac{1}{N}\sum_{n=1}^{N}\mathbf{e}_{t+1}^n \\ &= (\mathbf{x}_t + \mathbf{u}_t) + (\mathbf{a}_t - \mathbf{u}_t) + \frac{1}{N}\sum_{n=1}^{N}(\mathbf{b}_t^n - \mathbf{v}_t^n) \\ &= \mathbf{x}_t + \mathbf{r}_t + \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{e}_t^n - \frac{\mathbb{I}_t^n}{q_t^n}\cdot\mathbf{g}_n(\mathbf{x}_t)\right) \\ &= \mathbf{x}_t - \frac{1}{N}\sum_{n=1}^{N}\frac{\mathbb{I}_t^n}{q_t^n}\cdot\mathbf{g}_n(\mathbf{x}_t).\end{aligned}$$

From smoothness, we have

$$\begin{aligned}\mathsf{E}_t[f(\mathbf{x}_{t+1})] &\leq f(\mathbf{x}_t) - \left\langle \nabla f(\mathbf{x}_t), \mathsf{E}_t\left[\frac{1}{N}\sum_{n=1}^{N}\frac{\mathbb{I}_t^n}{q_t^n}\cdot\mathbf{g}_n(\mathbf{x}_t)\right]\right\rangle \\ &\quad + \frac{L}{2}\mathsf{E}_t\left[\left\|\frac{1}{N}\sum_{n=1}^{N}\frac{\mathbb{I}_t^n}{q_t^n}\cdot\mathbf{g}_n(\mathbf{x}_t)\right\|^2\right] \\ &\leq f(\mathbf{x}_t) - \eta\langle\nabla f(\mathbf{x}_t), \nabla f(\mathbf{x}_t)\rangle + \frac{^2L}{2N}\sum_{n=1}^{N}\mathsf{E}_t\left[\left\|\frac{\mathbb{I}_t^n}{q_t^n}\cdot\mathbf{g}_n(\mathbf{x}_t)\right\|^2\right].\end{aligned} \tag{32}$$

We consider the two terms in (32) separately. We first have

$$\begin{aligned}&-\eta\langle\nabla f(\mathbf{x}_t), \nabla f(\mathbf{x}_t)\rangle \\ &= -\eta\langle\nabla f(\mathbf{x}_t) - \nabla f(\mathbf{x}_t), \nabla f(\mathbf{x}_t)\rangle - \eta\langle\nabla f(\mathbf{x}_t), \nabla f(\mathbf{x}_t)\rangle \\ &\leq \frac{L^2}{2}\|\mathbf{x}_t - \mathbf{x}_t\|^2 + \frac{}{2}\|\nabla f(\mathbf{x}_t)\|^2 - \eta\langle\nabla f(\mathbf{x}_t), \nabla f(\mathbf{x}_t)\rangle \\ &= \frac{L^2}{2}\left\|\mathbf{r}_t + \frac{1}{N}\sum_{n=1}^{N}\mathbf{e}_t^n\right\|^2 - \frac{}{2}\|\nabla f(\mathbf{x}_t)\|^2 \\ &\leq \eta L^2\|\mathbf{r}_t\|^2 + \frac{L^2}{N}\sum_{n=1}^{N}\|\mathbf{e}_t^n\|^2 - \frac{}{2}\|\nabla f(\mathbf{x}_t)\|^2.\end{aligned} \tag{33}$$

By noting that $(\mathbb{I}_t^n)^2 = \mathbb{I}_t^n$ and $\mathsf{E}_t[\mathbb{I}_t^n] = q_t^n$, we also have

$$\begin{aligned}&\sum_{n=1}^{N}\mathsf{E}_t\left[\left\|\frac{\mathbb{I}_t^n}{q_t^n}\cdot\mathbf{g}_n(\mathbf{x}_t)\right\|^2\right] = \sum_{n=1}^{N}\mathsf{E}_t\left[\frac{\mathbb{I}_t^n}{(q_t^n)^2}\right]\cdot\mathsf{E}_t\left[\|\mathbf{g}_n(\mathbf{x}_t)\|^2\right] \\ &= \sum_{n=1}^{N}\frac{1}{q_t^n}\mathsf{E}_t\left[\|\mathbf{g}_n(\mathbf{x}_t)\|^2\right] \leq \sum_{n=1}^{N}\frac{1}{q_t^n}\left(\|\nabla F_n(\mathbf{x}_t)\|^2 + \sigma^2\right) \\ &\leq \sum_{n=1}^{N}\frac{1}{q_t^n}\left(2\|\nabla f(\mathbf{x}_t)\|^2 + 2\epsilon^2 + \sigma^2\right),\end{aligned} \tag{34}$$

where the last two inequalities are due to Assumption 1 and also the variance relation $\mathsf{E}_t\left[\|\mathbf{z}\|^2\right] = \|\mathsf{E}[\mathbf{z}]\|^2 + \mathsf{E}_t\left[\|\mathbf{z} - \mathsf{E}_t[\mathbf{z}]\|^2\right]$ for any random variable $\mathbf{z}$.

Let $Q_t := \frac{1}{N}\sum_{n=1}^{N}\frac{1}{q_t^n}$. Note that we assume $Q_t \leq p$ and $\eta \leq \frac{1}{4Lp}$. Hence, $\eta \leq \frac{1}{4Lp} \leq \frac{1}{4LQ_t}$ and $-\frac{}{2} + \eta^2 LQ_t \leq -\frac{}{4}$. Plugging (33) and (34) back into (32), we obtain

$$\begin{aligned}\mathsf{E}_t[f(\mathbf{x}_{t+1})] &\leq f(\mathbf{x}_t) + \eta L^2\|\mathbf{r}_t\|^2 + \frac{L^2}{N}\sum_{n=1}^{N}\|\mathbf{e}_t^n\|^2 \\ &\quad - \frac{}{2}\|\nabla f(\mathbf{x}_t)\|^2 + \eta^2 LQ_t\|\nabla f(\mathbf{x}_t)\|^2 + \eta^2 LQ_t(\epsilon^2 + \sigma^2) \\ &\leq f(\mathbf{x}_t) + \eta L^2\|\mathbf{r}_t\|^2 + \frac{L^2}{N}\sum_{n=1}^{N}\|\mathbf{e}_t^n\|^2 \\ &\quad - \frac{}{4}\|\nabla f(\mathbf{x}_t)\|^2 + \eta^2 LQ_t(\epsilon^2 + \sigma^2).\end{aligned}$$

Taking total expectation and rearranging, we obtain

$$\begin{aligned}\mathsf{E}\left[\|\nabla f(\mathbf{x}_t)\|^2\right] &\leq \frac{4(\mathbb{E}[f(\mathbf{x}_t)] \quad \mathbb{E}[f(\mathbf{x}_{t+1})])}{} \\ &\quad + 4L^2\mathsf{E}\left[\|\mathbf{r}_t\|^2\right] + \frac{4L^2}{N}\sum_{n=1}^{N}\mathsf{E}\left[\|\mathbf{e}_t^n\|^2\right] + 4\eta L(\epsilon^2 + \sigma^2)\mathsf{E}[Q_t].\end{aligned}$$

Averaging over all t, we obtain the final result. □

REFERENCES

[1] P. Kairouz, H. B. McMahan *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[2] 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.

[3] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, p. 12, 2019.

[4] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching LAN speeds," in *USENIX NSDI*, 2017, pp. 629–647.

[5] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[6] J. Wang and G. Joshi, "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD," in *Proceedings of Machine Learning and Systems*, A. Talwalkar, V. Smith, and M. Zaharia, Eds., vol. 1, 2019, pp. 212–229.

[7] P. Han, S. Wang, and K. K. Leung, "Adaptive gradient sparsification for efficient federated learning: An online learning approach," in *IEEE ICDCS*, 2020, pp. 300–310.

[8] S. Li, Q. Qi, J. Wang, H. Sun, Y. Li, and F. R. Yu, "Ggs: General gradient sparsification for federated learning in edge computing," in *IEEE ICC*, 2020.

[9] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, "To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices," in *IEEE INFOCOM*, 2021.

[10] A. M. Abdelmoniem and M. Canini, "Dc2: Delay-aware compression control for distributed machine learning," in *IEEE INFOCOM*, 2021.

[11] H. Xu, C.-Y. Ho, A. M. Abdelmoniem, A. Dutta, E. H. Bergou, K. Karatsenidis, M. Canini, and P. Kalnis, "Grace: A compressed communication framework for distributed machine learning," in *IEEE ICDCS*, 2021, pp. 561–572.

[12] L. Cui, X. Su, Y. Zhou, and J. Liu, "Optimal rate adaption in federated learning with compressed communications," in *IEEE INFOCOM*, 2022, pp. 1459–1468.

[13] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *IEEE ICC*, 2019.

[14] L. Wang, W. Wang, and B. Li, "CMFL: Mitigating communication overhead for federated learning," in *IEEE ICDCS*, 2019.

[15] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM*, 2020, pp. 1698–1707.

[16] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint device scheduling and resource allocation for latency constrained wireless federated learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 453–467, 2020.

[17] J. Perazzone, S. Wang, M. Ji, and K. S. Chan, "Communication-efficient device scheduling for federated learning using stochastic optimization," in *IEEE INFOCOM*, 2022, pp. 1449–1458.

[18] B. Luo, X. Li, S. Wang, J. Huang, and L. Tassiulas, "Cost-effective federated learning design," in *IEEE INFOCOM*, 2021.

[19] B. Luo, W. Xiao, S. Wang, J. Huang, and L. Tassiulas, "Tackling system and statistical heterogeneity for federated learning with adaptive client sampling," in *IEEE INFOCOM*, 2022, pp. 1739–1748.

[20] H. Wu and P. Wang, "Node selection toward faster convergence for federated learning on non-iid data," *IEEE Transactions on Network Science and Engineering*, 2022.

[21] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017, pp. 1273–1282.

[22] H. Yu, S. Yang, and S. Zhu, "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *AAAI Conference on Artificial Intelligence*, 2019, pp. 5693–5700.

[23] E. Gorbunov, F. Hanzely, and P. Richtarik, "Local SGD: Unified theory and new efficient methods," in *AISTATS*, 2021, pp. 3556–3564.

[24] F. Haddadpour, M. M. Kamani, M. Mahdavi, and V. Cadambe, "Local SGD with periodic averaging: Tighter analysis and adaptive synchronization," in *NeurIPS*, 2019.

[25] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't use large mini-batches, use local SGD," in *ICLR*, 2020.

[26] S. U. Stich, "Local SGD converges fast and communicates little," in *ICLR*, 2019.

[27] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations," in *NeurIPS*, 2019.

[28] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," in *AISTATS*, 2021, pp. 2350–2358.

[29] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," *NeurIPS*, vol. 30, 2017.

[30] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar, "signSGD: Compressed optimisation for non-convex problems," in *ICML*, 2018, pp. 560–569.

[31] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "Uveqfed: Universal vector quantization for federated learning," *IEEE Transactions on Signal Processing*, vol. 69, pp. 500–514, 2020.

[32] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *AISTATS*, 2020, pp. 2021–2031.

[33] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *NeurIPS*, 2018.

[34] F. Sattler, S. Wiedemann, K. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, Nov. 2019.

[35] A. Albasyoni, M. Safaryan, L. Condat, and P. Richtárik, "Optimal gradient compression for distributed and federated learning," *arXiv preprint arXiv:2010.03246*, 2020.

[36] E. Gorbunov, K. P. Burlachenko, Z. Li, and P. Richtárik, "Marina: Faster non-convex distributed learning with compression," in *ICML*, 2021, pp. 3788–3798.

[37] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *NeurIPS*, 2018.

[38] D. Alistarh, T. Hoefler, M. Johansson *et al.*, "The convergence of sparsified gradient methods," in *NeurIPS*, 2018, pp. 5977–5987.

[39] S. U. Stich and S. P. Karimireddy, "The error-feedback framework: Better rates for SGD with delayed gradients and compressed communication," *Journal of Machine Learning Research*, vol. 21, no. 237, pp. 1–36, 2020.

[40] S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi, "Error feedback fixes SignSGD and other gradient compression schemes," in *ICML*, 2019, pp. 3252–3261.

[41] H. Tang, C. Yu, X. Lian, T. Zhang, and J. Liu, "Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression," in *ICML*, 2019, pp. 6155–6165.

[42] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi, "Clustered sampling: Low-variance and improved representativity for clients selection in federated learning," in *ICML*, Jul. 2021, pp. 3407–3416.

[43] H. Yang, M. Fang, and J. Liu, "Achieving linear speedup with partial worker participation in non-IID federated learning," in *ICLR*, 2021.

[44] Y. J. Cho, J. Wang, and G. Joshi, "Towards understanding biased client selection in federated learning," in *AISTATS*, 2022, pp. 10 351–10 375.

[45] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *ICLR*, 2020.

[46] Z. Zhou, S. Yang, L. Pu, and S. Yu, "CEFL: Online admission control, data scheduling, and accuracy tuning for cost-efficient federated learning across edge nodes," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9341–9356, 2020.

[47] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[48] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[49] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *IEEE ICCV*, 2015.