Network Adaptive Federated Learning: Congestion and Lossy Compression

Parikshit Hegde
Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas, USA
hegde@utexas.edu

Gustavo de Veciana
Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas, USA
gustavo@ece.utexas.edu

Aryan Mokhtari
Electrical and Computer Engineering
The University of Texas at Austin
Austin, Texas, USA
mokhtari@austin.utexas.edu

***Abstract*—In order to achieve the dual goals of privacy and learning across distributed data, Federated Learning (FL) systems rely on frequent exchanges of large files (model updates) between a set of clients and the server. As such FL systems are exposed to, or indeed the cause of, congestion across a wide set of network resources. Lossy compression can be used to reduce the size of exchanged files and associated delays, at the cost of adding noise to model updates. By judiciously adapting clients' compression to varying network congestion, an FL application can reduce wall clock training time. To that end, we propose a Network Adaptive Compression (NAC-FL) policy, which dynamically varies the client's lossy compression choices to network congestion variations. We prove, under appropriate assumptions, that NAC-FL is asymptotically optimal in terms of directly minimizing the expected wall clock training time. Further, we show via simulation that NAC-FL achieves robust performance improvements with higher gains in settings with positively correlated delays across time.**

***Index Terms*—federated learning, rate adaptation, resilience**

I. Introduction

Communication costs and delays of sending model updates from clients to the server are a known bottleneck in training Federated Learning (FL) systems [1]–[4]. Two common techniques used to alleviate this issue are: 1) *local computations* where clients perform several local steps before communicating with the server, and 2) *(lossy) compression* where clients communicate quantized/compressed updates to the server. The eventual end goal of these approaches is to minimize the *wall clock time* for convergence of the training algorithm (hereon referred to as FL algorithm) by reducing the amount of data communicated from clients to the server.

To this end, several works have analyzed the relationship between compression, local computations and the number of rounds needed by FL algorithms to converge [5]–[12]. However, these works ignore the impact of changing network congestion, *both across clients and across time*, on the wall clock time to converge. For instance, a client may choose a high degree of compression when it sees high network congestion, while a client seeing lower congestion may *opportunistically* choose not to compress as much. In this work, we ask the following question: "Can we design a policy that adapts the amount of compression across clients and time according to changing network conditions in order to optimize the wall clock time?" To answer this question, we first characterize the impact that changing network congestion and an adaptive compression policy have on the wall clock time. Second, we propose the Network Adaptive Compression for Federated Learning (NAC-FL) policy that judiciously chooses compression levels based on network congestion to minimize the wall clock time. Crucially, NAC-FL does not rely on the prior knowledge of the distribution of network congestion. Instead, it learns to optimize its compression decisions on-the-fly based on the congestion seen by clients.

This material is based upon work of Hegde and de Veciana supported by the National Science Foundation (NSF) under grant No. 2148224 and is supported in part by funds from OUSD R&E, NIST, and industry partners as specified in the Resilient & Intelligent NextG Systems (RINGS) program and the WNCG/6G@UT industrial affiliates. The work of Mokhtari is supported in part by the NSF AI Institute for Future Edge Networks and Distributed Intelligence (AI-EDGE) via NSF grant 2112471, the Machine Learning Lab (MLL) at UT Austin, and the Wireless Networking and Communications Group (WNCG) Industrial Affiliates Program.

NAC-FL works in an opportunistic manner by adaptively choosing high or low amounts of compression across clients and across time based on low or high network congestion. It further considers two effects that compression has on the wall clock time. First, with increasing amount of compression, the FL algorithm would require more communication rounds to converge, as the server receives "noisier", and hence inaccurate, model updates. Second, with higher degrees of compression, the duration of each round would decrease as a smaller model update is communicated. Since the wall clock time is affected by both the number of rounds and the duration of each round (it is effectively the product of the two quantities), a policy for choosing compression levels should consider these jointly. Fig. 1 provides an illustrative visualization. Hence, NAC-FL aims to find the "sweet-spot" compression levels over time varying network congestion.

Contributions. We propose a general framework to study how to best adapt compression of client model updates. Assuming a stationary Markov model for the underlying network congestion state, we show that optimal policies are state dependent and characterize the expected stopping time for convergence to a predefined model accuracy.

This characterization provides the underlying insight for our proposed NAC-FL policy. To our knowledge this is the first

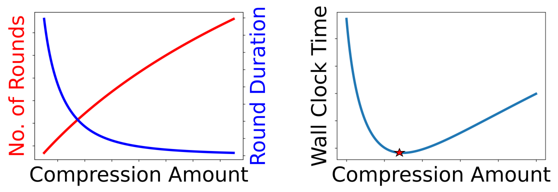

Fig. 1: Illustration of how compression level affects round duration, number of rounds and wall clock time.

policy for compression that adapts to the stochastic variations of the underlying network congestion process. Under appropriate assumptions on the FL algorithm and underlying network congestion and delays, we provide a proof of the asymptotic optimality of NAC-FL in terms of minimizing the mean time until the convergence criterion is met. To our knowledge this is the first theoretical result of this type.

Finally we demonstrate via simulation the performance gains and robustness of NAC-FL vs alternative fixed compression and/or fixed error per round policies. We explore a variety of models for network congestion, finding that in particular NAC-FL excels in the practically relevant setting where the network sees positive correlations in the network congestion accross time.

A. *Related Work*

Perhaps the most related papers to our work are [13]–[17] which explored adaptive compression schemes for FL settings. In [13]–[15] the authors propose adapting compression to network congestion. In these works, the algorithm to select compression has a per round budget, e.g., a budget on delay (or compression error) per round, and possibly heterogeneous compression levels are chosen across the clients based on the current network congestion to minimize the compression error (or delay) for the round. These works exploit the diversity of network congestion across the clients, but *not across time*. Meanwhile [16], [17] have observed that using a higher amount of compression at the start and gradually reducing compression through time may improve the wall clock time. Our proposed policy is novel in that it learns how to best exploit congestion variation across clients and across time to optimize the wall clock time.

Another line of work that aims to reduce the overall communication cost is client sampling [18]–[21], where at each round, only a subset of the clients are chosen to participate. The authors of [21] propose a client sampling and power control policy that adapts to time varying channels of clients sharing a single base station and optimizes a proxy for wall clock time. Overall we veiw lossy compression and client sampling as alternative approaches geared at addressing communication bottlenecks. A study of how to jointly adapt lossy compression and client sampling to changing network congestion is left for future work.

B. *Paper Organization*

In Section II, we introduce our system model. In Section III, we propose our NAC-FL algorithm for lossy compression and under appropriate assumptions prove it is asymptotically optimal. Section IV is devoted to exploring the method for several problem instances and in particular for various models for the underlying network congestion in terms of correlation across clients and time. In Section V, we comment on the practical aspects of estimating the file transfer delay of clients when deploying NAC-FL. Finally, in Section VI, we close the paper with some concluding remarks.

Due to space limitations, we state theoretical results without proof in this paper. See the technical report [22] for all the proofs.

Notation. *Throughout this document, unless otherwise mentioned, quantities denoted with lowercase letters correspond to constants, and uppercase letters correspond to random variables. Bold symbols correspond to vectors, and regular symbols indicate scalars. For example, $\boldsymbol{x}$ is a constant vector, $\boldsymbol{X}$ is a random vector, x is a constant scalar, and X is a random scalar/variable. Lowercase and uppercase forms of the same letter correspond to constant and random variable notions of the same quantity. A sequence indexed by n will be denoted as $(x^n)_n$.*

II. Model Setup

In this paper, we focus on a federated architecture, where a server aims to find a model that performs well with respect to the data of a group of m clients, and in which nodes exchange updates based on their local information with only the server. More precisely, suppose the loss function associated with client j is denoted by $f_j(\boldsymbol{w})$, where $\boldsymbol{w}$ represents the weights of the model, e.g., the weights of a neural network. The goal is to find the model that minimizes the average loss across clients

$$f(\boldsymbol{w}) = \frac{1}{m}\sum_{j=1}^{m} f_j(\boldsymbol{w}).$$

The FL algorithm proceeds in rounds. Each round consists of two stages: (i) a local stage in which each client updates the most recent model received from the server via gradient-based updates based on its local data and (ii), an aggregation stage in which the server updates the global model by aggregating the local updates received from clients. We shall let $\boldsymbol{w}^n$ denote the global model at the server at round n. Further, we let τ^n denote the total number of local steps (such as gradient steps) that each client performs at round n, and let $\boldsymbol{w}_j^{\tau^n,n}$ denote the resulting local model at node j.

In this paper, we are interested in the setting where each client sends a compressed version $\tilde{\boldsymbol{g}}_{Qj}^n$ of its local model $\boldsymbol{w}_j^{\tau^n,n}$ to the server using a lossy compression algorithm (or, *compressor*) $\mathcal{Q}(\cdot,\cdot)$. The compressor accepts a vector $\boldsymbol{x}$ and a parameter $q \in [0, q_{\max}]$ indicating the amount of compression with the maximum value being $q_{\max}$, and outputs $\hat{\boldsymbol{X}} = \mathcal{Q}(\boldsymbol{x}, q)$ which is an approximation of $\boldsymbol{x}$, but has a decreased file size as compared to $\boldsymbol{x}$. $\hat{\boldsymbol{X}}$ is capitalized to highlight that the compressor $\mathcal{Q}(\cdot,\cdot)$ may use randomness in its compression. We shall denote by q_j^n the compression parameter used by client j for round n, and denote by $\boldsymbol{q}^n \triangleq (q_j^n)_{j=1}^m$ the vector of parameters used by the clients in round n. After receiving updates from all the clients, the server aggregates the compressed local models and produces the next global model $\boldsymbol{w}^{n+1}$.

Given a target tolerance $\varepsilon > 0$, the goal of FL is to generate a sequence of global models until on some round r_ε a prespecified *stopping criterion* is first met, e.g., the norm of the global loss function gradient is at most ϵ, i.e., $\|\nabla f(\boldsymbol{w}^{r_\varepsilon})\| \leq \varepsilon$. Our goal is to find an adaptive compression policy that dynamically adapts to the possibly time varying network states such that the target accuracy is achieved with a minimum overall wall clock time.

We formalize the *overall wall clock time*, denoted t_ε, required to achieve the target accuracy as follows. The duration $d(\tau^n, \boldsymbol{q}^n, \mathrm{c}^n)$ of a round n depends on:

- τ^n, the number of local computations performed by clients which we will assume to be the same across clients;
- $\boldsymbol{q}^n$, an m dimensional vector of clients' compression parameters ;
- c^n, the *network state* which models network congestion and is assumed to be an element of a finite set $\mathcal{C}$.

This allows some flexibility, e.g., the round's duration may depend on the max delay to deliver the model update from clients to server, or the sum of the delays if clients share a single resource in TDMA (Time Division Multiple Access) fashion. The total wall clock time is then given by

$$t_\varepsilon = \sum_{n=1}^{r_\varepsilon} d\left(\tau^n, \boldsymbol{q}^n, \mathrm{c}^n\right). \tag{1}$$

In our system model, the sequence of network states, $(\mathrm{c}^n)_n$, is assumed to be exogenous, i.e., not be controlled by the server or the clients nor their choices of τ^n and $\boldsymbol{q}^n$. The delays associated with the server multicasting global models to clients are assumed to be exogeneous i.e., can not be controlled by the FL server/clients and are not compressed, whence are not part of the model. Still, in this work, based on observing the network state we will devise an approach to select the clients compression parameters so as to minimize the wall clock time. As discussed in Section V, in practice observation of the network state may involve light weight in band estimation by probing delays of message bits as they are delivered in a given round.

A policy for choosing compression parameters is called a *state dependent stationary policy* if it can be expressed as a function $\boldsymbol{\pi}$ of the current network state, i.e., $\boldsymbol{q}^n = \boldsymbol{\pi}(\mathrm{c}^n)$ for all rounds $n \in \mathbb{N}$. Such a policy will be referred to simply as policy $\boldsymbol{\pi}$. Given a random sequence of network states, $(\mathrm{C}^n)_n$, let $R_\varepsilon^{\boldsymbol{\pi}}$ be the random variable denoting the minimum number of rounds needed to converge to error tolerance ε under policy $\boldsymbol{\pi}$. Then, the corresponding wall clock time, denoted by $T_\varepsilon^{\boldsymbol{\pi}}$, is expressed as,

$$T_\varepsilon^{\boldsymbol{\pi}} = \sum_{n=1}^{R_\varepsilon^{\boldsymbol{\pi}}} d\left(\tau^n, \boldsymbol{\pi}\left(\mathrm{C}^n\right), \mathrm{C}^n\right).$$

III. Network Adaptive Compression for Federated Learning (NAC-FL)

Our approach to designing a policy to adapt clients' compression parameters centers on recognizing that the expected wall clock time can be broken up into a product of the *expected number of rounds* r_ε needed to converge to an error tolerance ε and the *average duration of each round* $\hat{d}$. We start by characterizing the relationship between r_ε, $\hat{d}$, and the sequence of selected quantization parameters $(\boldsymbol{q}^n)_n$ and network states $(\mathrm{c}^n)_n$ for a given FL algorithm.

Below we state an assumption relating r_ε to $(\boldsymbol{q}^n)_n$. To that end we introduce a strictly increasing, continuous and bounded scalar function $h_\varepsilon : [0, q_{\max}] \to \mathbb{R}^+$ of compression parameter q and an associated vector function $\boldsymbol{h}_\varepsilon : [0, q_{\max}]^{\times m} \to \mathbb{R}_+^m$ of a compression vector $\boldsymbol{q}$ where $\boldsymbol{h}_{\varepsilon,j}(\boldsymbol{q}) = h_\varepsilon(q_j)$. We let $\boldsymbol{h}_\varepsilon^{-1}$ denote the inverse of this vector function.

Assumption 1. *For a given FL algorithm there exists a strictly increasing, continuous and bounded function $h_\varepsilon(q)$ and norm $\|\cdot\|$ such that given a sequence of compression parameters $(\boldsymbol{q}^n)_n$, the FL algorithm has reached the desired error tolerance ε by round r if and only if,*

$$r > \frac{1}{r} \sum_{n=1}^{r} \left\|\boldsymbol{h}_\varepsilon\left(\boldsymbol{q}^n\right)\right\|$$

for some norm.

The above assumption implies that the expected number of rounds can be written as the average of an increasing function of the sequence of selected quantization parameters. Roughly speaking, given a lossy compression policy that generates a stationary parameter sequence $(\boldsymbol{Q}^n)_n$ whose marginal distribution is the same as the random vector $\boldsymbol{Q}$, the above criterion means that the expected number of rounds to converge to the desired error tolerance is approximately $\mathbb{E}[\|\boldsymbol{h}_\varepsilon\left(\boldsymbol{Q}\right)\|]$.

This is a general condition that is motivated by convergence bounds of several FL algorithms with compression, including, [5], [8], [11]. In particular in Appendix A, we illustrate this motivation for an extension of the FedCOM algorithm [11], when q indicates the normalized-variance introduced by the compressor, the scalar function is $h_\varepsilon(q) = O(\sqrt{q+1}/\varepsilon)$ and the norm is the L_2 norm.

Assumption 2. *For any sequence of compression parameters $(\boldsymbol{q}^n)_n$ the minimum number of rounds r_ε needed to converge to an error tolerance ε is such that $r_\varepsilon = \Theta(1/poly(\varepsilon))$, where $poly(\varepsilon)$ denotes a polynomial of ε.*

Assumption 2 is a natural assumption for gradient based optimization algorithms. It requires the convergence guarantees for the FL algorithm to be such that when we require a more accurate solution, the number of required communication

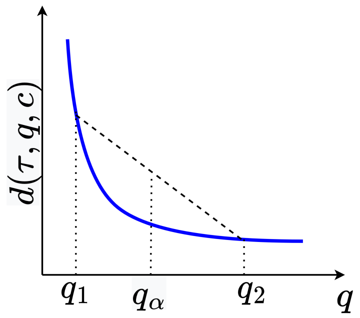

Fig. 2: Illustration of a round duration as a function of compression parameter q for a fixed local computation τ and network state c.

rounds grows. This argument indeed holds even for the settings that we do not exchange compressed signals.

We also make the following additional assumption about the round duration function.

Assumption 3. *Given a network state* c*, number of local computations* τ*, and compression parameters* $\boldsymbol{q} = \boldsymbol{h}_\varepsilon^{-1}(\boldsymbol{r})$*, the round duration* $d(\tau, \boldsymbol{q}, \mathrm{c}) = d\left(\tau, \boldsymbol{h}_\varepsilon^{-1}(\boldsymbol{r}), \mathrm{c}\right)$ *is bounded, convex in* $\boldsymbol{r}$ *and decreasing in every coordinate of* $\boldsymbol{r}$*.*

In Assumption 3, the round duration being decreasing in $\boldsymbol{r}$ is reasonable, since we expect more rounds as well as smaller file sizes with higher compression. The convexity is motivated by the notion that we use a "good compressor" as illustrated next. Consulting Fig. 2, for any two parameters q_1, q_2 and $0 < \alpha < 1$, a new time-sharing compressor $\mathcal{Q}'$ may be derived which outputs $\mathcal{Q}(\boldsymbol{x}, q_1)$ with probability α and outputs $\mathcal{Q}(\boldsymbol{x}, q_2)$ with probability $(1-\alpha)$. This compressor has expected round duration $\alpha d(\tau, q_1, \mathrm{c}) + (1-\alpha) d(\tau, q_2, \mathrm{c})$. And, in certain cases, its compression parameter is $q_\alpha = \alpha q_1 + (1-\alpha) q_2$ (such as when the stochastic quantizer parameterized by its normalized variance [5] is used). If $\mathcal{Q}$ is a "good compressor", then its round duration, $d(\tau, q_\alpha, \mathrm{c})$, should be lower compared to that of the simple time-shared compressor, $\alpha d(\tau, q_1, \mathrm{c}) + (1-\alpha) d(\tau, q_2, \mathrm{c})$ (considering $h_\varepsilon(q) \propto q$ for simplicity).

Assumption 4. *The sequence of network states* $(\mathrm{C}^n)_n$ *forms an irreducible aperiodic stationary Markov Chain on a finite state space* $\mathcal{C}$ *with invariant distribution* μ*.*

Assumption 4 is a natural assumption made to facilitate the analysis of algorithms (see e.g., [23]).

A. Expected Wall Clock Time Formulation

Given the above mentioned assumptions, we are now ready to introduce the proposed framework. We begin by showing that we need only consider state dependent stationary policies for choosing compression parameters when optimizing the overall wall clock time.

Lemma 1. *Under Assumptions 1-4 there exists a state dependent stationary policy to select compression parameters which is asymptotically optimal in terms of minimizing the wall clock time to reach a desired error tolerance of* ε *as* $\varepsilon \to 0$*.*

The proof of Lemma 1 depends on two critical observations. First, since by Assumption 2 the number of rounds needed to converge grows large as $\varepsilon \to 0$, one can expect the empirical distribution of the network states modelled by the finite state Markov Chain to concentrate around the invariant prior to the stopping time. Second, due to the convexity of the round duration function in Assumption 3, given a sequence of network states there exists a state dependent stationary policy that is near optimal and depends solely on the empirical distribution of the sequence.

Here, we will focus on the setting where ε is small, hence by Lemma 1, we only need to consider state dependent stationary policies, $\boldsymbol{q}^n = \boldsymbol{\pi}(\mathrm{c}^n)$.

Lemma 2. *Under Assumptions 1-4 and a fixed number of local computations per round* τ*, for every* $\delta > 0$*, there exists an* $\varepsilon_{th} > 0$ *such that, for all* $\varepsilon < \varepsilon_{th}$ *and any state-dependent stationary policy* $\boldsymbol{\pi}$*, the expected wall clock time is bounded as,*

$$1 - \delta \leq \frac{\mathbb{E}\left[T_\varepsilon^{\boldsymbol{\pi}}\right]}{\mathbb{E}[\|\boldsymbol{h}_\varepsilon\left(\boldsymbol{\pi}(\mathrm{C})\right)\|]\,\mathbb{E}[d\left(\tau, \boldsymbol{\pi}(\mathrm{C}), \mathrm{C}\right)]} \leq 1 + \delta, \tag{2}$$

where, C *denotes a random variable whose distributions is* μ *(see Assumption 3).*

Define,

$$\hat{t}_\varepsilon^{\boldsymbol{\pi}} \triangleq \mathbb{E}[\|\boldsymbol{h}_\varepsilon\left(\boldsymbol{\pi}(\mathrm{C})\right)\|]\,\mathbb{E}[d\left(\tau, \boldsymbol{\pi}(\mathrm{C}), \mathrm{C}\right)]. \tag{3}$$

Due to Lemma 2, for small enough ε, $\hat{t}_\varepsilon^{\boldsymbol{\pi}}$ provides an accurate approximation for $\mathbb{E}[T_\varepsilon^{\boldsymbol{\pi}}]$. Therefore, from here onwards, we shall assume implicitly that that a small ε is considered and focus on finding a policy to optimize $\hat{t}_\varepsilon^{\boldsymbol{\pi}}$.

Suppose the distribution of C is known. Then, one could compute expected wall clock time as given in (3) for any state dependent stationary policy $\boldsymbol{\pi}$. In this case, we could determine an optimal policy $\boldsymbol{\pi}^*$ by solving the optimization problem,

$$\min_{\boldsymbol{\pi} \in \mathcal{Q}_{m|\mathcal{C}|}} \quad \hat{t}_\varepsilon^{\boldsymbol{\pi}} \;=\; \mathbb{E}[\|\boldsymbol{h}_\varepsilon\left(\boldsymbol{\pi}(\mathrm{C})\right)\|]\,\mathbb{E}\left[d\left(\tau, \boldsymbol{\pi}(\mathrm{C}), \mathrm{C}\right)\right], \tag{4}$$

where $\mathcal{Q}_{m|\mathcal{C}|}$ is the set of all state-dependent stationary policies.

Alas, in practice, we often cannot directly solve the above problem, as the distribution of C is unknown. Hence, below, we propose a stochastic approximation like algorithm that achieves the optimal wall clock time of $\boldsymbol{\pi}^*$ asymptotically.

B. The NAC-FL Algorithm

Our NAC-FL approach is inspired by the Frank-Wolfe Algorithm [24]. We start by reformulating the optimization program in (4). Denote by set V_ε all possible pairs of expectations $(\hat{r}_\varepsilon, \hat{d})$,

$$V_\varepsilon = \Big\{ (\hat{r}_\varepsilon, \hat{d}) : \exists\, \boldsymbol{\pi} \in \mathcal{Q}_{m|\mathcal{C}|} \text{ s.t. } \hat{r}_\varepsilon = \mathbb{E}\left[\|\boldsymbol{h}_\varepsilon\left(\boldsymbol{\pi}(\mathrm{C})\right)\|\right], \\ \hat{d} = \mathbb{E}\left[d\left(\tau, \boldsymbol{\pi}(\mathrm{C}), \mathrm{C}\right)\right] \Big\}. \tag{5}$$

Using the set V_ε, and denoting $H(r,d) \triangleq rd$, we may write the optimization (4) characterizing the optimal policy $\boldsymbol{\pi}^*$ as

$$\min_{\hat{r}_\varepsilon, \hat{d}} \{H(\hat{r}_\varepsilon, \hat{d}) : (\hat{r}_\varepsilon, \hat{d}) \in V_\varepsilon\}. \tag{6}$$

In this case, from a point $(\hat{r}_\varepsilon^n, \hat{d}^n)$, the Frank-Wolfe update would be given as,

$$\begin{aligned}(\hat{r}_\varepsilon, \hat{d}) &= \underset{(r,d)\in V_\varepsilon}{\text{argmin}} \quad \nabla H\left(\hat{r}_\varepsilon^n, \hat{d}^n\right)^\top \begin{pmatrix} r \\ d \end{pmatrix}, \quad (7)\\ \hat{r}_\varepsilon^{n+1} &= (1-\beta)\hat{r}_\varepsilon^n + \beta \hat{r}_\varepsilon, \\ \hat{d}^{n+1} &= (1-\beta)\hat{d}^n + \beta \hat{d}.\end{aligned}$$

The gradient $\nabla H(\hat{r}_\varepsilon, \hat{d})$ is, $\nabla H(\hat{r}_\varepsilon, \hat{d}) = \begin{pmatrix} \hat{d} & \hat{r}_\varepsilon \end{pmatrix}^\top$. V_ε is a set of feasible averages of $\hat{r}_\varepsilon$ and $\hat{d}$. Therefore, at round $(n+1)$, not all the pairs $(r,d) \in V_\varepsilon$ may be achievable. Hence, NAC-FL approximates equation (7) as,

$$\boldsymbol{q}^{n+1} = \underset{\boldsymbol{q}}{\text{argmin}} \quad \hat{r}_\varepsilon^n d\left(\tau, \boldsymbol{q}, \mathrm{c}^{n+1}\right) + \hat{d}^n \left\|\boldsymbol{h}_\varepsilon(\boldsymbol{q})\right\|. \tag{8}$$

The NAC-FL policy is described in Algorithm 1. To retrieve policy derived above, the tunable parameters $(\beta_n)_n$ and α in Algorithm 1 should be set to $\beta_n = \beta$ and $\alpha = 1$.

Algorithm 1: NAC-FL

```
  Input     : Initialization: $\hat{r}_\varepsilon^{(0)}, \hat{d}^{(0)}$ ; step size
              schedule $\{\beta_n\}_{n=1}^\infty$; parameter $\alpha$.
1 for $n = 1, \ldots,$ until termination do
2     Server observes network state $\mathrm{c}^n$ ;
3     $\boldsymbol{q}^n =$
       $\underset{\boldsymbol{q}}{\text{argmin}} \quad \alpha\hat{r}_\varepsilon^{(n-1)} d(\tau, \boldsymbol{q}, \mathrm{c}^n) + \hat{d}^{(n-1)} \|\boldsymbol{h}_\varepsilon(\boldsymbol{q})\|$;
4     $\hat{r}_\varepsilon^n = (1-\beta_n)\hat{r}_\varepsilon^{(n-1)} + \beta_n \|\boldsymbol{h}_\varepsilon(\boldsymbol{q}^n)\|$ ;
5     $\hat{d}^n = (1-\beta_n)\hat{d}^{(n-1)} + \beta_n d(\tau, \boldsymbol{q}^n, \mathrm{c}^n)$;
6 end
```

Observe that since the estimates $\hat{r}_\varepsilon^n$ and $\hat{d}^n$ will initially change across rounds, NAC-FL may choose different compression parameters in two rounds for which the network was in the same state, i.e., NAC-FL is not a state-dependent stationary policy. Still, we will show NAC-FL is asymptotically near optimal.

The following assumption is required to show the asymptotic optimality of NAC-FL. A state dependent stationary policy $\boldsymbol{\pi}$ maps from a domain of finite size $|\mathcal{C}|$, to a range positive-real vectors of dimension m. Therefore, the policy may be represented by a positive-real vector, $\underline{\boldsymbol{\pi}}$, of dimension $m\,|\mathcal{C}|$. Further, a vector $\boldsymbol{r}^{\underline{\boldsymbol{\pi}}}$ may be obtained by applying $h_\varepsilon(\cdot)$ elementwise to the policy vector $\underline{\boldsymbol{\pi}}$, $\boldsymbol{r}^{\underline{\boldsymbol{\pi}}} \triangleq \boldsymbol{h}_\varepsilon(\underline{\boldsymbol{\pi}})$. This representation is used in the following assumption.

Assumption 5. *The objective function $\hat{t}_\varepsilon^{\boldsymbol{\pi}}$ of the optimization problem in* (4) *is a strictly quasiconvex function in $\boldsymbol{\pi}$ in the following sense,*

$$\boldsymbol{r}^{\underline{\boldsymbol{\pi}}\top}\left(\nabla_{\boldsymbol{r}^{\underline{\boldsymbol{\pi}}}} \hat{t}_\varepsilon^{\boldsymbol{\pi}}\right) = 0 \implies \boldsymbol{r}^{\underline{\boldsymbol{\pi}}\top}\left(\nabla^2_{\boldsymbol{r}^{\underline{\boldsymbol{\pi}}}} \hat{t}_\varepsilon^{\boldsymbol{\pi}}\right)\boldsymbol{r}^{\boldsymbol{\pi}} > 0. \tag{9}$$

Assumption 5 ensures that there is a unique state dependent stationary policy $\underline{\boldsymbol{\pi}}^*$ which optimizes (4). We have observed that the considered network model, compression model and the $\|\boldsymbol{h}_\varepsilon(\boldsymbol{q})\|$ function associated with the FedCOM algorithm indeed satisfy this assumption.

Next we shall establish an optimality property for NAC-FL. To that end we shall consider executing NAC-FL without termination with $\beta_n = \beta$ for all n and let $\left(\boldsymbol{Q}_\beta^n\right)_n$, $\hat{R}_{\varepsilon,\beta}^n$ and $\hat{D}_\beta^n$ be the corresponding sequence of compression parameters and the associated estimates.

Theorem 1. *Let $\boldsymbol{\pi}^*$ be the solution and $\hat{t}_\varepsilon^{\boldsymbol{\pi}^*}$ the minimum of the optimization problem in* (4)*. If Assumptions 1-5 hold, then there exists a positive sequence $(\beta^i)_{i=1}^\infty$ with $\beta^i \to 0$ as $i \to \infty$, such that for every $\rho > 0$, there exists a thereshold $n_{th}(\rho)$ such that,*

$$\lim_{i\to\infty} \sup_{n \geq n_{th}(\rho)/\beta^i} P\left(\left\|\begin{pmatrix}\hat{R}_{\varepsilon,\beta^i}^n - \mathbb{E}[\|\boldsymbol{h}_\varepsilon(\boldsymbol{\pi}^*(\mathrm{C}))\|] \\ \hat{D}_{\beta^i}^n - \mathbb{E}[d(\tau, \boldsymbol{\pi}^*(\mathrm{C}), \mathrm{C})]\end{pmatrix}\right\| > \rho\right) = 0,$$

Remark 1. *A sketch of the proof of this result is included in Appendix B, and the complete proof is in the technical report [22]. This result should be interpreted with some subtlety. Say the desired error-tolerance ε is very small such that the number of rounds needed to converge under any compression policy is such that $r_\varepsilon \gg n_{th}(\rho)/\beta$. Then, based on Theorem 1, one can show that NAC-FL compression choices will be near optimal after $n_{th}(\rho)/\beta$ rounds. Thereafter since r_ε is large, NAC-FL will make near optimal choices for long enough leading to a near optimal expected wall clock time.*

In applications that require a very low error-tolerance ε, one needs to run FL for a large number of rounds r_ε. Therefore, while the wall clock time obtained by using NAC-FL may be large in this setting, it is near-optimal compared to other methods of choosing compression parameters, making the asymptotic result relevant.

IV. Simulation

In this section, we present our simulation results. We begin by describing additional model details used in our simulations.

A. Additional Model Details

1) Compression Model: We shall use the *stochastic quantizer* in [5] which we will denote as $\mathcal{Q}_q(\cdot, b)$. The quantizer has a parameter $b \in \{1, \ldots, 32\}$ corresponding to the number of bits used to represent each co-ordinate, in addition to the bit used to denote signs. When input a vector $\boldsymbol{x}$, it outputs,

$$\mathcal{Q}_q(\boldsymbol{x}, b) = \|\boldsymbol{x}\|_\infty \, \mathsf{sign}(\boldsymbol{x}) \zeta(\boldsymbol{x}, b) \tag{10}$$

where $\mathsf{sign}(\boldsymbol{x})$ is the element-wise sign operator and where the function $\zeta(\boldsymbol{x}, b)$ uniformly quantizes each co-ordinate amongst

$2^b - 1$ levels between 0 and 1. That is, if $x_i/\|\boldsymbol{x}\|_\infty \in \left[\frac{l}{2^b-1}, \frac{l+1}{2^b-1}\right)$, then it is quantized as,

$$\zeta_i(\boldsymbol{x}, b) = \begin{cases} \frac{l+1}{2^b-1}, & \text{with prob. } \frac{|x_i|}{\|\boldsymbol{x}\|_\infty}(2^b-1) - l, \\ \frac{l}{2^b-1}, & \text{otherwise.} \end{cases}$$

When $\boldsymbol{x}$ is quantized to b-bits per co-ordinate, its file size is given by the function, $s(b) = \|\boldsymbol{x}\|_0 (b+1) + 32$ bits. Here, the zero-norm, $\|\boldsymbol{x}\|_0$, gives the length of the vector, the 1 indicates the bit used to denote the sign, and the 32 bits are for a floating point number denoting the norm, $\|\boldsymbol{x}\|_\infty$. Finally, if client j uses the parameter b_j, then the vector of parameters used by the clients is denoted as, $\boldsymbol{b} = (b_j)_{j=1}^m$.

2) Network Congestion Model: For purposes of evaluating the performance of various algorithms over different types of network congestion we propose the following general, albeit idealized, model. We let $\boldsymbol{C}^n$ be a m dimensional random vector denoting the *Bit Transmission Delay* (BTD) for clients during round n. We further let $\boldsymbol{C}^n = \exp(\boldsymbol{Z}^n)$ i.e., coordinate-wise exponentiation of an m dimensional first order autoregressive process given by $(\boldsymbol{Z}_i)_{i=0}^\infty$ where $\boldsymbol{Z}_0 = \boldsymbol{0}$, where

$$\boldsymbol{Z}^n = A\,\boldsymbol{Z}^{(n-1)} + \boldsymbol{E}^n, \quad \forall n \geq 1, \tag{11}$$

where A is an $m \times m$ deterministic matrix, and $\boldsymbol{E}^n \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ are i.i.d., m dimensional normal random vectors. Different correlations across time and clients may be modelled by varying A, $\boldsymbol{\mu}$ and Σ. The marginal distributions of $\boldsymbol{C}^n$ are thus log-normal but can be correlated in different ways based on the underlying autoregressive process. In particular:

Homogeneous Independent: the parameters are set to $A = 0$, $\boldsymbol{\mu} = \boldsymbol{1}$, and $\Sigma = \sigma^2 I$. This results in a process which is independent and identically distributed across clients and time.

Heterogeneous Independent: the parameters are set to $A = 0$, $\mu_i = 0$ for $i \in \{1, \ldots, 5\}$ and $\mu_i = 2$ for $i \in \{6, \ldots, 10\}$, and $\Sigma = I$. This results in a process which is independent across clients and time, with the BTD being lower for the first 5 clients compared to the rest.

Perfectly correlated: the parameters are set to A such that $A_{i,j} = \frac{a}{m}$ where $a \in (0,1)$, $\boldsymbol{\mu} = \boldsymbol{0}$, and Σ such that $\Sigma_{i,j} = \sigma^2 = 1$. This results in a process where all clients see the same positively correlated time-varying delays.

Partially correlated: the parameters are set to A such that $A_{i,j} = \frac{a}{m}$, $\boldsymbol{\mu} = \boldsymbol{0}$, and Σ such that $\Sigma_{i,i} = 1$ and $\Sigma_{i,j} = 1/2$ for $i \neq j$. This results in a process where delays are positively correlated accross clients and time.

3) Model for Round Durations: We will model the duration of a round as the maximum across clients' delays, i.e., $d(\tau, \boldsymbol{b}, \mathrm{c}) = \max_j[\theta\tau + c_j s(b_j)]$, where θ represents the compute time per local computation, and $c_j s(b_j)$ the BTD of client j times the size of the client j's file capturing the time taken to communicate its update. For simplicity we will set $\theta = 0$.

4) Compression Level Choice Policies: We compare NAC-FL to the following policies,

a) Fixed Bit: Here, a number b is fixed, and all the clients use the stochastic quantizer $\mathcal{Q}_q(\boldsymbol{x}, b)$ from (10) with the parameter b. We present results for $b \in \{1,2,3\}$, as we didn't notice a performance improvement for larger parameters in our experiments.

b) Fixed Error: This method was suggested in [13] and is parameterized by a number q. At each round n, the parameters $\boldsymbol{b}^n$ of the stochastic quantizers are such that the average normalized-variance $\bar{q}^n \triangleq 1/m \sum_j q_j^n$ is smaller than q, and the duration of the round $d(\tau, \boldsymbol{q}^n, \mathrm{c}^n)$ is minimized. We fix $q = 5.25$ in all our experiments after finding it to be performing well across different settings.

5) Machine Learning Model: We consider $m = 10$ clients. We consider the MNIST dataset [25] which may be distributed homogeneously or heterogeneously amongst the clients. Since data is heterogeneous across clients in most FL applications, we consider the heterogenous data case. That is, each client has data corresponding to 1 unique label. The MNIST dataset has 60,000 training samples, 10,000 test samples and 10 labels. The clients and the server aim to train a fully connected neural network with the architecture $(784, 250, 10)$ with the sigmoid activation for the hidden layer. The learning rate is initialized to $\eta_0 = 0.07$, and is decayed by a factor 0.9 every 10 rounds. The aggregation rate and local computations per round are fixed throughout the training to $\gamma = 1$ and $\tau = 2$ respectively. As for the parameters of the NAC-FL policy, we set $\beta_n = \frac{1}{n}$, and $\alpha = 2$.

We measure the performance of the global model using the following,

a) Training Loss: The training loss of the global model is the empirical cross entropy loss across the entire set of training samples.

b) Test Accuracy: The test accuracy is measured over all the test samples. Here, in some experiments, we run 20 simulations with different random seeds, and report the mean, 90th percentile and 10th percentile times to reach a test accuracy of 90%. The 90th and 10th percentile scores are reported to capture the variation in performance across the 20 simulations. We also report a *gain* metric, which is sample mean of the time gained to reach 90% accuracy by NAC-Fl compared to a another policy reported in percentage. For instance, let x_i, y_i be the times under NAC-FL and another policy for a random seed i, then the gain is $100 * \left(\sum_{i=1}^{20} y_i/x_i - 1\right)/20$.

B. Simulation Results

1) Homogeneous Independent BTD: We simulated over $\sigma^2 \in \{1,2,3\}$ in order to study the change in performance over increasing variance. We observe that in all the cases, NAC-FL and the Fixed Error policy have very similar performance across all the considered statistics. This is because the Fixed Error policy was designed to operate well in the i.i.d., network delay case. However, both NAC-FL and Fixed Error policy perform better than all the Fixed Bit policies according to all the statistics across all the considered parameters. Moreover, we observed that the gap in the performance to Fixed Bit policies increased with increasing variance. For instance, the gain of the best Fixed Bit policy increased from 145% to 250% when the variance was increased from 1 to 3, while the gain of the worst fixed bit policy increased from 314% to 881%. This is as expected because both NAC-FL and Fixed Error policy adapt to the heterogenous delay of clients at any given time. Surprisingly, NAC-FL lagged behind Fixed Error policy in some metrics, but it performed better in terms of the gain metric in all the 3 cases, with the gain over Fixed Error policy ranging from 1% to 8%.

σ^2		1 bit	2 bits	3 bits	Fixed Error	NAC-FL
1	Mean	6.31	3.82	4.15	**1.58**	1.60
	90th	6.95	4.72	5.00	**1.86**	2.05
	10th	5.63	3.20	3.38	1.20	**1.14**
	Gain	314%	145%	168%	3%	-
2	Mean	54.8	32.5	34.9	12.5	**12.2**
	90th	70.6	44.7	43.1	**19.0**	20.8
	10th	42.5	19.2	21.0	6.26	**5.82**
	Gain	522%	216%	240%	8%	-
3	Mean	799	430	458	**165**	168
	90th	1430	752	665	**318**	320
	10th	418	157	148	**46.2**	57.9
	Gain	881%	270%	250%	1%	-

TABLE I: Performance comparison of policies with homogeneous independent BTD in terms of the mean, 90th percentile and 10th percentile times to reach 90% test accuracy under the different policies, and their average sample-path gain compared to NAC-FL. All the numbers represented are in 10^7 seconds.

2) Heterogeneous Independent BTD: We considered this case since the first 5 clients would have consistently worse delay, NAC-FL and the Fixed Error policy would consistently compress the updates of those clients heavily. Since the data distribution is heterogeneous, it may be possible heavy compression of updates from specific clients throughout the training may hurt the performance. On the other hand, the Fixed Bit policies use the same amount of compression across all clients equally irrespective of their delays. Still, we observed that NAC-FL and the Fixed Error policy perform better than the Fixed Bit policies as can be seen in Table II. In fact, performance in terms of the gain metric is very comparable to the i.i.d., network delay case with $\sigma^2 = 1$ in Table I.

	1 bit	2 bits	3 bits	Fixed Error	NAC-FL
Mean	9.49	5.85	6.46	2.49	**2.48**
90th	11.5	7.16	8.09	**3.48**	3.54
10th	8.30	4.37	4.98	1.74	**1.54**
Gain	319%	146%	173%	4%	-

TABLE II: Performance comparison of policies with heterogenous independent BTD. The numbers shown are the mean, 90th percentile and 10th percentile times to reach 90% test accuracy under the different policies, and their average sample-path gain compared to NAC-FL. All the numbers represented are in 10^8 seconds.

3) Perfectly Correlated BTD: We will demonstrate that NAC-FL performs better than Fixed Error and Fixed Bit policies under increasing correlated delay across time since they are not designed to optimize the wall clock time under this case.

To study the variation of network delay across rounds, consider the marginal auto-regressive process of 1 client which may be represented by the following scalar autoregressive process,

$$Z^n = a' Z^{(n-1)} + E^n, \tag{12}$$

where $E^n \sim \mathcal{N}(0,1)$. We define metric called *asymptotic variance*, denoted σ^2_∞, which is designed to capture the variance, and long and short term correlations of a random process, $\sigma^2_\infty \triangleq \lim_{n\to\infty} \frac{\mathbb{E}\left[\left(Z^{(1)}+\cdots+Z^n\right)^2\right]}{n}$. For the autoregressive process in (12), it may be computed to be, $\sigma^2_\infty = 1/(1-a')^2$.

Table III shows the performance of the different policies under varying asymptotic variance of the marginals. We observe that in addition to beating the baseline fixed bit policies on all the metrics, the NAC-FL performs better than the Fixed Error policy in most metrics as well. Considering the gain metric, we observe gain of 13% over the Fixed Error policy for low asymptotic variance of $\sigma^2_\infty = 1.56$, and is as large as 27% for higher asymptotic variance of $\sigma^2_\infty = 4$. Notably, in terms of the 10th percentile time to reach 90% accuracy, the Fixed Error policy required 40%, 23% and 32% more time compared to NAC-FL in the σ^2_∞=1.56, 4 and 16 cases respectively.

σ^2_∞		1 bit	2 bits	3 bits	Fixed Error	NAC-FL
1.56	Mean	5.14	3.04	3.47	2.21	**2.11**
	90th	5.94	3.65	4.43	**2.66**	3.32
	10th	3.88	2.38	2.18	1.43	**1.02**
	Gain	191%	58%	75%	13%	-
4	Mean	5.82	3.49	4.03	2.47	**2.23**
	90th	7.43	4.77	6.28	**3.94**	4.00
	10th	3.88	2.22	1.98	1.21	**0.981**
	Gain	252%	82%	107%	27%	
16	Mean	8.42	5.19	6.15	3.75	**3.36**
	90th	12.8	10.3	13.4	7.94	**7.2**
	10th	4.34	1.40	1.67	1.15	**0.87**
	Gain	316%	72%	98%	21%	-

TABLE III: Performance comparison of policies with perfectly correlated BTD in terms of the mean, 90th percentile and 10th percentile times to reach 90% test accuracy under the different policies, and their average sample-path gain compared to NAC-FL. All the numbers represented are in 10^7 seconds.

4) Partially Correlated BTD: In Table IV, we show results for the partially correlated BTD case with asymptotic variance $\sigma^2_\infty = 4$. We consider this case to demonstrate that NAC-FL is effective with positive (but, not 100%) correlation across clients as well. Indeed, we observe NAC-FL performing better compared to all the other policies across all the considered metrics, with a gain of 10% over the Fixed Error policy, and 129% over the best fixed bit policy. Notably, in terms of the 10th percentile and 90th percentile metrics, NAC-FL outperformed Fixed Error policy by 30% and 15% respectively.

Figure 3 contains sample path plots of Training Loss and Accuracy vs Wall Clock Time for the independent homogeneous ($\sigma^2 = 2$), heterogeneous and perfectly correlated ($\sigma^2_\infty = 4$) BTD cases. Both accuracy and loss plots for NAC-FL and Fixed Error are overlapping in the independent

	1 bit	2 bits	3 bits	Fixed Error	NAC-FL
Mean	13.6	8.33	9.51	4.22	**3.83**
90th	15.9	10.5	13.9	6.24	**5.46**
10th	9.51	5.47	5.80	2.64	**2.02**
Gain	307%	129%	159%	10%	-

TABLE IV: Performance comparison of policies with partially correlated BTD in terms of the mean, 90th percentile and 10th percentile times to reach 90% test accuracy under the different policies, and their average sample-path gain compared to NAC-FL. All the numbers represented are in 10^7 seconds.

homogeneous and heterogeneous BTD cases, as expected. However, in the perfectly correlated BTD case, NAC-FL dominates the performance of Fixed Error policy.

In summary, we observe that NAC-FL's performance is robust under a range of network models considered. NAC-FL vastly outperformed the baseline Fixed Bit policies in all the network models. The performance of NAC-FL was observed to be similar to that of Fixed Error policy in the independent BTD setting, albeit, it outperformed Fixed Error policy in terms of the gain metric under all the network models. Notably, the gap between NAC-FL and Fixed Error policy was observed to be noticeably high in the perfectly and paritally correlated BTD settings, where NAC-FL was able to adapt to positive correlations of BTD across time, whereas Fixed Error could not.

V. NAC-FL in Practice

In this section we briefly comment on some practical aspects underlying estimating model update delays. This involves estimating the network's current average BTD to each client. A simple approach to doing so is to observe that for the stochastic quantizer described in Section IV-A1, clients always send the vector of signs of their updates, no matter what are the bits per coordinate that will be chosen. So, as the clients send their signs, the server may probe the delay characteristics to estimate the BTD of clients without having to request vacuous (non update related) bits to do so. It may then use these estimates to perform the optimization in (8) for the round.

VI. Conclusion

Due to their distributed character FL algorithms are exposed to congestion across a potentially large number of network resources, whence one might say they are exposed to network congestion and variability at scale. Building adaptive algorithms that minimize the impact of time varying congestion across clients presents a significant challenge, particularly when the aim is to directly optimize the expected wall clock time. NAC-FL exemplifies a new class of robust algorithms to optimally adapt clients' lossy compression. This paper further provides the technical roadmap to formalizing and showing asymptotic optimality for such algorithms.

Appendix A
Federated Learning with Adaptive Compression (FLAC)

In this section, we consider a variant of the FedCOM algorithm [11], called FedCOM-V. FedCOM is based on fixing a quantization parameter throughout the run of the FL algorithm. On the other hand, FedCOM-V allows for an arbitrary sequence of quantization parameters $(\boldsymbol{q}^n)_n$, in order to account for adaptive compression policies such as NAC-FL (see Algorithm 2).

```
Algorithm 2: FedCOM-V
  Input  : number of local computations schedule $(\tau_n)_{n=1}^{\infty}$, local learning rate schedule
           $(\eta_n)_{n=1}^{\infty}$, adaptively chosen global learning rate schedule $(\gamma_n)_{n=1}^{\infty}$, adaptively
           chosen number of rounds $r$, initial global model $\boldsymbol{w}^1$.
  Result: $\boldsymbol{w}^{r+1}$: Final model
1 for $n = 1, \ldots, r$ do
2     for each client $j \in [m]$ do
3         Set $\boldsymbol{w}_j^{1,n} = \boldsymbol{w}^n$ ;
4         for $a = 1, \ldots, \tau_n$ do
5             Sample a minibatch $\mathcal{Z}_j^{a,n}$ and compute
              $\tilde{\boldsymbol{g}}_j^{a,n} \triangleq \nabla f(\boldsymbol{w}_j^{a,n}; \mathcal{Z}_j^{a,n})$ ;
6             $\boldsymbol{w}_j^{a+1,n} = \boldsymbol{w}_j^{a,n} - \eta_n \tilde{g}_j^{a,n}$;
7         end
8         Device sends
          $\tilde{\boldsymbol{g}}_{Qj}^n = \mathcal{Q}((\boldsymbol{w}^n - \boldsymbol{w}_j^{\tau_n+1,n})/\eta_n, q_j^n)$ back to
          the server;
9     end
10    Server computes, $\tilde{\boldsymbol{g}}_Q^n = \frac{1}{m}\sum_{j=1}^m \tilde{\boldsymbol{g}}_{Qj}^n$ ;
11    Server computes $\boldsymbol{w}^{n+1} = \boldsymbol{w}^n - \eta_n\gamma_n\tilde{\boldsymbol{g}}_Q^n$ and
      broadcasts to all devices;
12 end
```

We make the following standard assumptions to analyze the convergence of FedCOM-V.

Assumption 6 (Smoothness and Lower Boundedness)**.** *The objective function $f(\cdot)$ is differentiable and L-smooth. That is, $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$, for every $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. Moreover, the optimal value of f is lower bounded, $f^* = \min_{\boldsymbol{w}} f(\boldsymbol{w}) > -\infty$.*

Assumption 7 (Bounded Variance)**.** *For all clients j and rounds n and local step a, an independent mini-batch $\mathcal{Z}_j^{a,n}$ is sampled, and an unbiased stochastic gradient $\tilde{\boldsymbol{g}}_j^{a,n} = \nabla f(\boldsymbol{w}; \mathcal{Z}_j^{a,n})$ is computed. Also, the variance is bounded by a constant σ^2, $\mathbb{E}\left[\left\|\tilde{\boldsymbol{g}}_j^{a,n} - \nabla f\left(\boldsymbol{w}_j^{a,n}\right)\right\|^2\right] \leq \sigma^2$.*

Assumption 8 (Compression Model)**.** *The output of the compressor $\mathcal{Q}(\boldsymbol{x}, q)$ is an unbiased estimator of $\boldsymbol{x}$, i.e., $\mathbb{E}[\mathcal{Q}(\boldsymbol{x}, q)|\boldsymbol{x}] = \boldsymbol{x}$, and, its variance is bounded as, $\mathbb{E}[\|\mathcal{Q}(\boldsymbol{x}, q) - \boldsymbol{x}\|^2 |\boldsymbol{x}] \leq q\|\boldsymbol{x}\|^2$.*

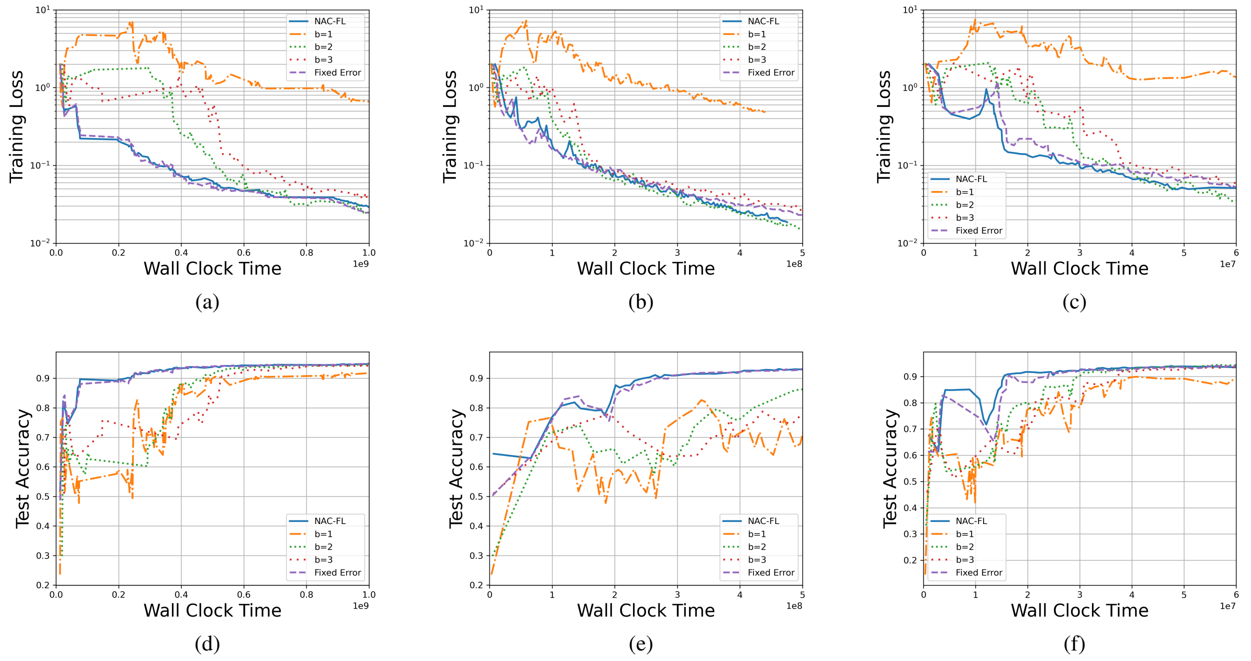

Fig. 3: Plots of Training Loss and Test Accuracy vs Wall Clock time on different network models. Figures (a) and (d) correspond to homogeneous independent BTD case ($\sigma^2 = 2$), Figures (b) and (e) correspond to the heterogeneous independent BTD case, and Figures (c) and (f) correspond to the perfectly correlated BTD case ($\sigma^2_\infty = 4$).

We denote the *average normalized-variance* used at round n by $\bar{q}^n = \frac{1}{m}\sum_{j=1}^{m} q_j^n$. The following Theorem states the relationship between $(\boldsymbol{q}^n)$, ε and r_ε.

Theorem 2. *Let Algorithm 2 be run with a sequence of compressors such that the average normalized-variance at round n is $\bar{Q}^n$. Further, assume that the sequence $\left(\bar{Q}^n\right)$ forms a stationary process with the stationary distribution represented by a random variable Q. To obtain $\mathbb{E}[\|\nabla f(\boldsymbol{w})\|^2] \leq \varepsilon$, we can choose,*

$$r_\varepsilon = O\left(\log(1/\varepsilon)\frac{\mathbb{E}\left[\sqrt{Q+1}\right]}{\varepsilon}\right), \quad \tau_n = O\left(n\right).$$

The upper bound on r_ε in Theorem 2 provides a justification for Assumption 1 with $h_\varepsilon(q) = O(\sqrt{q+1})$. Here, τ_n is a function of n, but for the purposes of NAC-FL we may use the average of τ_1 to τ_{r_ε} in the expression of the duration function. One may obtain a similar expression for other popular FL algorithms [5], [8].

Appendix B
Proof Sketch of Theorem 1

We will use the following proposition without proof.

Proposition 1: Under Assumption 5, the update in (7) has a unique fixed point $(\hat{r}_\varepsilon, \hat{d}) \in V_\varepsilon$ such that,

$$(\hat{r}_\varepsilon, \hat{d}) = \underset{(r,d)\in V_\varepsilon}{\operatorname{argmin}} \quad \nabla H\left(\hat{r}_\varepsilon, \hat{d}\right)^\top \begin{pmatrix} r \\ d \end{pmatrix}.$$

Define, $\boldsymbol{x}^{(n)} = (\hat{r}_\varepsilon^{(n)} \hat{d}^{(n)})^\top$, and let $\boldsymbol{x}(s) = \boldsymbol{x}^{(\beta n)}$ for $\beta \to 0$, which is often called the fluid limit. It can be shown [23] that the estimates of NAC-FL follow,

$$\boldsymbol{v}(s) = \underset{\boldsymbol{v}\in V_\varepsilon}{\operatorname{argmin}} \quad \nabla H\left(\boldsymbol{x}(s)\right)^\top \boldsymbol{v}, \quad \dot{\boldsymbol{x}}(s) = \boldsymbol{v}(s) - \boldsymbol{x}(s).$$

Denote, $G(\boldsymbol{x}) = \min_{\boldsymbol{v}\in V_\varepsilon} \nabla H(\boldsymbol{x})^\top(\boldsymbol{v} - \boldsymbol{x})$. Since, ∇H is a continuous function of $\boldsymbol{x}$, $G(\boldsymbol{x})$ may be shown to be a continuous function of $\boldsymbol{x}$ as well.

Due to Prop. 1, there exists a unique $\boldsymbol{x}^* \in V_\varepsilon$ such that $G(\boldsymbol{x}^*) = 0$. Moreover, due to strict quasiconvexity (Assumption 5), $\boldsymbol{x}^*$ is the minimizer of H. For all other $\boldsymbol{x} \in V_\varepsilon$, we show,

Claim 1: for all $\omega > 0$, there exists a $\delta > 0$ such that if $\|\boldsymbol{x} - \boldsymbol{x}^*\| > \omega$, then $G(\boldsymbol{x}) < -\delta$.

We prove this claim using contradiction. Suppose there exists an $\omega > 0$ such that for all $\delta > 0$, there exists an $\boldsymbol{x}$ with $\|\boldsymbol{x} - \boldsymbol{x}^*\| \geq \omega$ and $G(\boldsymbol{x}) \geq -\delta$. Since, $G(\boldsymbol{x})$ is a continuous function, taking limit $\delta \to 0$, we obtain an $\boldsymbol{x}$ with $\|\boldsymbol{x} - \boldsymbol{x}^*\| > \omega$ and $G(\boldsymbol{x}) = 0$. This contradicts Prop. 1.

Now given Claim 1, we prove the main result again by contradiction. Consider some $\omega > 0$. Let $\delta > 0$ be its associated constant according to Claim 1. Then, as a contradiction, assume that $\|\boldsymbol{x}(s) - \boldsymbol{x}^*\| > \omega$ for all $s \in \mathbb{R}_+$. Since, $dH(\boldsymbol{x}(s)) = G(\boldsymbol{x}(s)) < -\delta$, we may show in this case that $H(\boldsymbol{x}(s)) \to -\infty$ as $s \to \infty$. This is a contradiction H is a positive function. Since this is true for every ω, and $H()$ is continuous, we may prove that $\boldsymbol{x}(s) \to \boldsymbol{x}^*$ as $s \to \infty$.

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 *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[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] 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," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[4] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[5] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," *Advances in neural information processing systems*, vol. 30, 2017.

[6] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[7] S. U. Stich, "Local SGD converges fast and communicates little," *arXiv preprint arXiv:1805.09767*, 2018.

[8] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[9] S. Ghadimi and G. Lan, "Stochastic first-and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.

[10] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2021–2031.

[11] F. Haddadpour, M. M. Kamani, A. Mokhtari, and M. Mahdavi, "Federated learning with compression: Unified analysis and sharp guarantees," *arXiv preprint arXiv:2007.01154*, 2020.

[12] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[13] P. S. Bouzinis, P. D. Diamantoulakis, and G. K. Karagiannidis, "Wireless quantized federated learning: A joint computation and communication design," *arXiv preprint arXiv:2203.05878*, 2022.

[14] X. Zhang, X. Zhu, J. Wang, H. Yan, H. Chen, and W. Bao, "Federated learning with adaptive communication compression under dynamic bandwidth and unreliable networks," *Information Sciences*, vol. 540, pp. 242–262, 2020.

[15] H. Sun, X. Ma, and R. Q. Hu, "Adaptive federated learning with gradient compression in uplink NOMA," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16 325–16 329, 2020.

[16] D. Jhunjhunwala, A. Gadhikar, G. Joshi, and Y. C. Eldar, "Adaptive quantization of model updates for communication-efficient federated learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3110–3114.

[17] R. Hönig, Y. Zhao, and R. Mullins, "DAdaQuant: Doubly-adaptive quantization for communication-efficient federated learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 8852–8866.

[18] W. Chen, S. Horvath, and P. Richtarik, "Optimal client sampling for federated learning," *arXiv preprint arXiv:2010.13723*, 2020.

[19] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 269–283, 2020.

[20] M. Ribero and H. Vikalo, "Communication-efficient federated learning via optimal client sampling," *arXiv preprint arXiv:2007.15197*, 2020.

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

[22] P. Hegde, G. de Veciana, and A. Mokhtari, "Network Adaptive Federated Learning: Congestion and Lossy Compression," *arXiv e-prints*, p. arXiv:2301.04430, Jan. 2023.

[23] A. L. Stolyar, "On the asymptotic optimality of the gradient scheduling algorithm for multiuser throughput allocation," *Operations research*, vol. 53, no. 1, pp. 12–25, 2005.

[24] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.