ROBUSTIFYING ASYNCHRONOUS SGD VIA SOFT THROTTLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Asynchronous SGD is a popular algorithm for distributed learning where each client's gradient update is applied on arrival. This leads to a speed-up, but also an increased vulnerability to attacks, as fast clients can dominate the total update. We introduce THROTTLE, a Byzantine-robust generalization of asynchronous SGD where the key idea is to exponentially down-weight updates from faster clients by a factor q. Both asynchronous SGD ($q = 1$) and synchronous Byzantine-robust SGD ($q \to \infty$) correspond to specific settings of THROTTLE. We provide a theoretical analysis of the convergence rate and validate the robustness to attacks both theoretically and empirically. Remarkably, our experiments show that this down-weighting mechanism can also improve performance over standard asynchronous SGD even in the non-Byzantine setting.

1 INTRODUCTION

Distributed learning trains a shared model faster by spreading the training workload across many clients. It has become essential for large-scale machine learning, and it has been widely studied in recent years as a result (Yang et al., 2019; Verbraeken et al., 2020). Typically, distributed learning consists of a succession of steps. At every step, i) the clients send their local gradients, ii) the central server performs its update once all of them are received, and iii) the updated model is sent to the clients so that the next computation step can start. One clear bottleneck is that this process can only be as fast as the *slowest* machine. It leads to considerable slowdowns in practice as clients typically compute at very different speeds. This can arise from heterogeneous computational power (Horvath et al., 2021), network latency, faulty devices (Ryabinin et al., 2021), or slow *straggler* clients in GPU clusters (Chen et al., 2016). A natural remedy is to leverage *asynchronous* SGD methods (Recht et al., 2011; Mania et al., 2017; Nguyen et al., 2022), where the central server updates the model at each feedback, and the corresponding client starts computing the next gradient immediately. These are the methods we study in this paper.

Another limitation of distributed learning is that, while integrating more clients increases the amount of data and compute, it also increases the risk of having malicious or faulty clients (Xie et al., 2020a; Baruch et al., 2019; Kairouz et al., 2021). We refer to those as Byzantine clients (Lamport et al., 2019). This is especially problematic for asynchronous methods, where one malicious client can send an overwhelming number of updates to the server, which we call a *flooding attack*. Notable works in this area typically rely either on strong assumptions that restrict flooding attacks (Dahan & Levy, 2024) or on nearly synchronous algorithms that mitigate the effects of Byzantine clients (Yang & Li, 2021; 2023). In practice, the former may offer limited robustness against Byzantine attacks that violate these assumptions, while the latter may yield smaller speedups than asynchronous SGD because it more closely resembles synchronous algorithms.

We aim to bridge this gap through a new algorithm, THROTTLE. The key idea is to exponentially decrease the stepsize when gradients are repeatedly received from the same client. This limits the influence of any individual client on the global model, thereby preventing a malicious client from dominating the overall update. We present THROTTLE in Section 3 and analyze its convergence in Section 4. In Theorem 1, we establish its Byzantine robustness and show that it recovers the convergence rate of standard asynchronous SGD up to an additive constant. Our experiments in Section 6 demonstrate robustness to a range of Byzantine attacks and show that THROTTLE is more

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

robust and faster than the existing methods evaluated. Interestingly, THROTTLE also outperforms standard asynchronous SGD even in the non-Byzantine setting.

2 PRELIMINARIES

We consider the following optimization problem:

$$f(\boldsymbol{x}) \coloneqq \mathbb{E}_{\xi\sim\mathcal{D}}[F(\boldsymbol{x};\xi)],$$

where $F : \mathbb{R}^d \times \Omega \to \mathbb{R}$ is the loss with respect to a sample $\xi \in \Omega$, and $\mathcal{D}$ is the given data distribution over the sample space Ω.

Let $n \in \mathbb{N}$ denote the total number of clients, and write $[n] := \{1, \dots, n\}$ the set of all clients. All clients have access to the loss F and sample from the same dataset $\mathcal{D}$. We assume that it admits an (unknown) partition $[n] = \mathcal{G} \sqcup \mathcal{B}$ between good and Byzantine clients. Any good client $i \in \mathcal{G}$ follows the protocol given by the algorithm when sending feedback, while a Byzantine client $i \in \mathcal{B}$ is allowed to send arbitrary updates to the server. We let the latter be omniscient, i.e., they have access to all computations made by the rest of the good clients and can send the worst updates possible. We denote the Byzantine ratio by $\delta = |\mathcal{B}|/n$, so that $|\mathcal{G}| = (1-\delta)n$. In this paper we develop an algorithm with asynchronous updates that is robust in the Byzantine setting described above.

2.1 ASYNCHRONOUS SGD

Synchronous SGD can suffer from substantial delays because each iteration is limited by the slowest client, which motivates the use of asynchronous SGD. In asynchronous SGD, the server applies each stochastic gradient the moment it arrives. The updated model is then sent to the corresponding client so that the client can start computing the next gradient.

Let j_t be the client whose gradient is applied at time t. For each client i, let $\mathrm{prev}(t,i)$ denote the most recent update before t at which client i returned a gradient, and let $\mathrm{next}(t,i)$ denote the first update at or after t at which client i returns a gradient. After returning this gradient, client i receives the updated model $\boldsymbol{x}^{(\mathrm{prev}(t,i))}$ from the server and starts computing its next gradient from this model. Therefore, when client j_t returns its next gradient at update t, that gradient is evaluated at the stale model $\boldsymbol{x}^{(\mathrm{prev}(t,j_t))}$ rather than the current model. We call $\tau_t := t - \mathrm{prev}(t, j_t)$ the delay of this gradient. The update rule of Asynchronous SGD is then written as follows:

$$\boldsymbol{x}^{(t)} = \boldsymbol{x}^{(t-1)} - \eta \nabla F(\boldsymbol{x}^{(t-\tau_t)}; \xi_{j_t}^{(t-\tau_t)}). \tag{1}$$

Thanks to this update rule, the server does not need to wait for a fresh gradient evaluated at the latest model and can instead immediately update the model using a possibly stale gradient evaluated at $\boldsymbol{x}^{(t-\tau_t)}$. Consequently, the optimization process is not bottlenecked by the slowest client, as the server can continue updating the model whenever gradients from other clients arrive.

2.2 BYZANTINE-ROBUST SYNCHRONOUS ALGORITHM

Because distributed learning involves many participating clients, some of them may be malicious or faulty. This makes robustness against such Byzantine clients an important requirement for reliable distributed learning. In this section, we briefly describe a representative Byzantine-robust method in the synchronous setting. At every round t, each client i computes a stochastic gradient $\boldsymbol{g}_i^{(t)}$ at the same point $\boldsymbol{x}^{(t)}$, and the server updates the parameter as follows:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \frac{\eta}{n}\sum_{i=1}^{n} \boldsymbol{g}_i^{(t)}.$$

Since the server simply averages the gradients received from the clients, a Byzantine client can arbitrarily manipulate the parameters of the server. Byzantine-robust methods replace this average with a *robust aggregator* (e.g., the median) (Blanchard et al., 2017; Karimireddy et al., 2021), as is standard in classical robust statistics (Huber, 2005).

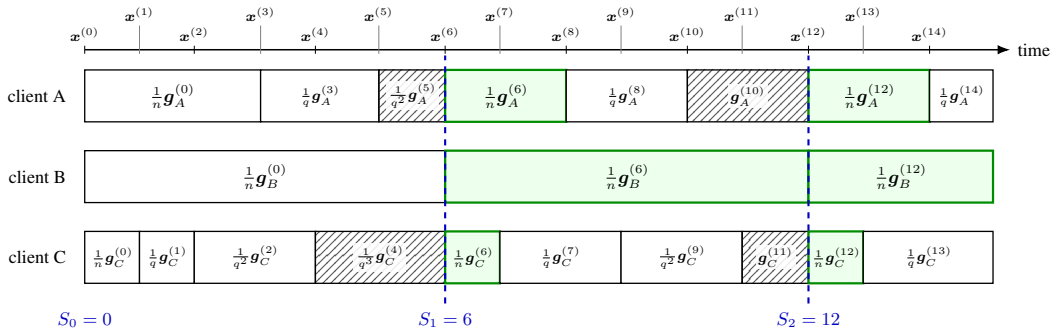


Figure 1: Illustration of THROTTLE with three clients. A block $\boldsymbol{g}_i^{(t)}$ is a gradient computation started at $\boldsymbol{x}^{(t)}$ (its left end), and its coefficient is the weight it receives on arrival, $1/n$ for a first arrival and $1/q^c$ for a repeated one. Dashed lines mark the synchronous clock S_r, at which in-flight computations are discarded (hatched blocks) and all clients restart from $\boldsymbol{x}^{(S_r)}$ (green blocks).

Definition 1 ((δ, c)-robust aggregator)**.** Suppose that $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n$ are random vectors of which a subset $\mathcal{G} \subset [n]$ if size at least $|\mathcal{G}| > (1-\delta)n$ and $\delta < 0.5$ and satisfies $\mathbb{E}\left\|\boldsymbol{g}_i - \boldsymbol{g}_j\right\|^2 \leq \rho^2,\ \forall i, j \in \mathcal{G}$. Then, the output $\hat{\boldsymbol{g}}$ of a Byzantine robust aggregator satisfies:

$$\mathbb{E}\|\hat{\boldsymbol{g}} - \overline{\boldsymbol{g}}\|^2 \leq c\delta\rho^2 \quad \text{where} \quad \hat{\boldsymbol{g}} := \mathrm{Agg}\left(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n\right) \text{ and } \overline{\boldsymbol{g}} := \frac{1}{|\mathcal{G}|}\sum_{j \in \mathcal{G}} \boldsymbol{g}_j.$$

This definition is quite common in the existing literature (Karimireddy et al., 2021; 2022; Gorbunov et al., 2023; Yang et al., 2024). Intuitively, the role of a robust aggregator Agg is to estimate the average of the non-Byzantine clients $\overline{\boldsymbol{g}}$ with an error of order $\mathcal{O}(\delta)$. Karimireddy et al. (2021) first showed that *centered clipping* satisfies this definition. Karimireddy et al. (2022) later extended this result to classical robust aggregators, namely *coordinate-wise median* (Chen et al., 2017), *Krum* (Blanchard et al., 2017), and *robust federated averaging* (RFA) (Pillutla et al., 2022), when combined with their proposed pre-aggregation technique,

2.3 BYZANTINE-ROBUST ASYNCHRONOUS ALGORITHM

Combining asynchrony with Byzantine robustness is desirable for distributed learning, as it allows the system to benefit from heterogeneous client speeds while remaining reliable in the presence of malicious or faulty clients. Several Byzantine-robust asynchronous methods have therefore been proposed (Yang & Li, 2023; Dahan & Levy, 2024). However, existing theoretical work on Byzantine-robust asynchronous SGD (Yang & Li, 2023; Dahan & Levy, 2024) tackles this issue by assuming that τ_t is bounded by a known $\tau_{\max}$ for most clients, or by directly bounding the proportion of Byzantine feedback. These assumptions circumvent the main challenge of the framework, which is precisely that a Byzantine client can make τ_t arbitrarily large through a flooding attack. We propose a Byzantine-robust asynchronous algorithm THROTTLE that removes these assumptions.

3 THROTTLE: ASYNCHRONOUS SGD WITH SOFT THROTTLING

We introduce THROTTLE, an algorithm consisting of two key components, *soft throttling* and *hard restart*, which we describe below.

3.1 SOFT THROTTLING

The key idea of our approach is to exponentially down-weight repeated updates of the same client by a factor q. More precisely, the first update has a factor $\frac{1}{n}$, the second $\frac{1}{q}$, the next $\frac{1}{q^2}$, then $\frac{1}{q^3}$, *etc*. Once every client has sent feedback, we say that the round ends. We then reinitialize all factors

to $\frac{1}{n}$ and start the next round. This way, within any given round, the impact of one client is upper bounded by $\frac{1}{n} + \sum_{i=1}^{+\infty} \frac{1}{q^i} = \frac{1}{n} + \frac{1}{q-1}$, which prevents flooding.

A smaller $q \to 1$ lets fast clients contribute more, while a larger $q \to \infty$ caps their influence more tightly and reduces it to synchronous SGD. We formally quantify this trade-off in Section 4.

Remark 1 (On the initial $\frac{1}{n}$ factor)**.** Adding an initial $1/n$ factor makes the total update with regards to the model at the start of the round $\boldsymbol{x}^{(S_r)}$ have weight 1. All subsequent updates within the round have weight at most $\frac{1}{q}$. This matches the scale of a robust aggregator's output, which simplifies the comparison in our analysis. We also observed that it leads to more stable training curves in practice.

3.2 Hard Restart

At the end of a round, the server broadcasts the current model to all clients, and discard any gradient computation in progress. This ensures that every round starts with all clients on the same model, which allows to robustly aggregate their updates. At the beginning of a round, all clients start their computation at the same point $\boldsymbol{x}^{(S_r)}$, and the added factor $\frac{1}{n}$ averages over the updates. As is standard in Byzantine-robust algorithms, we want to replace this average by a robust aggregator. The one difference from the synchronous setting is that the first arrivals from each clients do not arrive together. We therefore need an aggregator that acts on a single input and whose sum coincide with a robust aggregate over time.

Definition 2 (Local (c, δ)-robust aggregator)**.** Let Agg be a (c, δ)-robust aggregator that additionally takes a side information $\mathbf{S} \in \mathbb{S}$ as input. We call Agg *local* if there exists a map $A : \mathbb{R}^d \times \mathbb{S} \to \mathbb{R}^d$ such that, for every $\mathbf{S} \in \mathbb{S}$ and every $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$,

$$\mathrm{Agg}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n; \mathbf{S}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{A}(\boldsymbol{x}_i; \mathbf{S}).$$

Example 1 (Centered clipping)**.** Centered clipping (Karimireddy et al., 2021) clips each input's offset from an anchor $\boldsymbol{a} \in \mathbb{R}^d$ to length at most $\tau > 0$ and then averages:

$$\mathrm{Agg}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n; \boldsymbol{a}, \tau) = \boldsymbol{a} + \frac{1}{n} \sum_{i=1}^{n} \mathrm{clip}_\tau(\boldsymbol{x}_i - \boldsymbol{a}), \qquad \mathrm{clip}_\tau(\boldsymbol{v}) \coloneqq \min\Big\{1, \frac{\tau}{\|\boldsymbol{v}\|}\Big\}\boldsymbol{v}.$$

Thus, each input shifts the output by at most τ/n. It is local with side information $\mathbf{S} = (\boldsymbol{a}, \tau)$ and $\mathbb{A}(\boldsymbol{v}; \mathbf{S}) \coloneqq \boldsymbol{a} + \mathrm{clip}_\tau(\boldsymbol{v} - \boldsymbol{a})$. Following Karimireddy et al. (2021), our experiments use the previous round's first arrivals to set the anchor: $\boldsymbol{a}_r \coloneqq \Pi_G\Big(\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{g}_i^{(S_{r-1})}\Big)$, $\boldsymbol{a}_0 \coloneqq \mathbf{0}$, where Π_G is the Euclidean projection onto the ball of radius G centered at the origin.

Side information is the key to making the decomposition possible. Without side information, no (c, δ)-robust aggregator of the form $\frac{1}{n}\sum_i f(\boldsymbol{x}_i)$ exists (Proposition 2). Appendix D gives further examples and a general construction $\mathbb{A}(\boldsymbol{v}; \mathbf{S}) = \boldsymbol{a} + \psi(\boldsymbol{v} - \boldsymbol{a})$ with bounded ψ.

3.3 Throttle

Combining these two ideas, soft throttling and hard restart, we propose Throttle, as described in Algorithm 1. At time step t, let $\boldsymbol{v}_{j_t}^{(t-\tau_t)}$ denote the gradient received from client j_t, and let η_t denote corresponding stepsize. We define

$$\boldsymbol{v}_{j_t}^{(t-\tau_t)} \coloneqq \begin{cases} \nabla F(\boldsymbol{x}^{(t-\tau_t)}; \xi_{j_t}^{(t-\tau_t)}) & \text{if } j_t \in \mathcal{G}, \\ * & \text{if } j_t \in \mathcal{B}, \end{cases} \qquad \eta_t \coloneqq \begin{cases} \frac{\eta}{n} & \text{if } c_{j_t}(t) = 0, \\ \frac{\eta}{q^{c_{j_t}(t)}} & \text{otherwise}, \end{cases} \tag{2}$$

and $c_i(t) \coloneqq \sum_{s=r(t)}^{t-1} \mathbf{1}_{\{j_s = i\}}$ is the number of gradients that client i has already delivered in the current round. Once the server receives the gradient from client, the server transforms it to $\boldsymbol{g}_{j_t}^{(t-\tau_t)}$ using either local (c, δ)-robust aggregator or gradient clipping, and updates the parameters, as shown in line 8 of Algorithm 1. Then, after the server receives the gradient from all clients, the next round starts and all counters are reinitialized to 0. We illustrate the algorithm in Figure 1 and provide a notation list in Appendix A.

Algorithm 1 THROTTLE: Asynchronous SGD with Soft Throttling

Server:

Send $\boldsymbol{x}^{(0)}$ to every client $i \in [n]$ and initialize $S_0 \coloneqq 0$ and side information $\mathbf{S}_0$, $c_i = 0,\ \forall i \in [n]$
for $t = 0, \ldots, T-1$ **do**
  Receive update $\boldsymbol{v}_{j_t}^{(t-\tau_t)}$ from some worker j_t
  if $c_{j_t} = 0$ **then** ▷ If j_t returns its update for the first time this round
    $\boldsymbol{g}_{j_t}^{(t-\tau_t)} = \mathbb{A}\left(\boldsymbol{v}_{j_t}^{(t-\tau_t)}; \mathbf{S}_{r(t)}\right)$ ▷ Apply local (c, δ)-robust aggregator
  else ▷ If server has already seen j_t in this round
    $\boldsymbol{g}_{j_t}^{(t-\tau_t)} = \mathrm{clip}_{\lambda_t}\left(\boldsymbol{v}_{j_t}^{(t-\tau_t)}\right)$ ▷ Compute clipped gradient with radius λ_t
  $\eta_t \coloneqq \begin{cases} \frac{\eta}{n} & \text{if } c_{j_t}(t) = 0, \\ \frac{\eta}{q^{c_{j_t}(t)}} & \text{otherwise,} \end{cases}$ ▷ Set the **soft throttling** stepsize (2)
  $\boldsymbol{x}^{(t)} = \boldsymbol{x}^{(t-1)} - \eta_t \boldsymbol{g}_{j_t}^{(t-\tau_t)}$ ▷ Perform the update
  $c_{j_t} \leftarrow c_{j_t} + 1$ ▷ Increment the counter
  Send $\boldsymbol{x}^{(t)}$ to worker j_t
  if $|\{i \in [n] \mid c_i \geq 1\}| = n$ **then** ▷ If all clients has shown up in the round
    $S_{r+1} = t$ and set $c_i \leftarrow 0$ for all $i \in [n]$. ▷ The next round starts
    Update the side information $\mathbf{S}_{r+1}$ ▷ Depends on the aggregator (cf Ex. 1)
    Send $\boldsymbol{x}^{(t)}$ and restart signals to all clients, $c_i \leftarrow 0$ ▷ **Hard restart**
Notify all workers to stop

Good client $i \in \mathcal{G}$**:**

repeat
  repeat
    Wait until receiving the parameter $\boldsymbol{x}^{(k)}$ for some $k \in [T]$
    Randomly sample $\xi_i^{(k)} \sim \mathcal{D}$ and then compute the stochastic gradient $\nabla F(\boldsymbol{x}^{(k)}; \xi_i^{(k)})$
    Send the stochastic gradient $\nabla F(\boldsymbol{x}^{(k)}; \xi_i^{(k)})$ to the server
  until receive server's notification to restart with $\boldsymbol{x}^{(r(t))}$
  Use restart parameter $\boldsymbol{x}^{(r(t))}$ to compute $\nabla F(\boldsymbol{x}^{(r(t))}; \xi_i^{r(t)})$ with the sample $\xi_i^{r(t)} \sim \mathcal{D}$
until receive server's notification to stop

4 THEORETICAL ANALYSIS

Here we list assumptions we use in our theoretical results.

Assumption 1 (Unbiased and uniformly bounded stochastic gradients)**.** For each $\boldsymbol{x} \in \mathbb{R}^d$, the stochastic gradients from good clients $\nabla F(\boldsymbol{x}; \xi_i)$, $i \in \mathcal{G}$, are conditionally i.i.d., unbiased, and have uniformly bounded variance:

$$\mathbb{E}[\nabla F(\boldsymbol{x}; \xi_i) \mid \boldsymbol{x}] = \nabla f(\boldsymbol{x}), \qquad \mathbb{E}\|\nabla F(\boldsymbol{x}; \xi_i) - \nabla f(\boldsymbol{x})\|^2 \leq \sigma^2.$$

Assumption 2 (L-smoothness, lower boundedness)**.** The objective $f : \mathbb{R}^d \to \mathbb{R}$ is differentiable, lower bounded by $f_\star$, and L-smooth, i.e., for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{L}{2}\|\boldsymbol{y} - \boldsymbol{x}\|^2.$$

Assumption 3 (Bounded gradient)**.** There exists $G \geq 0$ such that for each $\boldsymbol{x} \in \mathbb{R}^d$ and $\xi \in \Omega$, the stochastic gradient satisfies

$$\|\nabla F(\boldsymbol{x}; \xi)\| \leq G \quad \text{almost surely.}$$

The assumptions on smoothness and stochastic gradients are standard in the optimization literature (Bubeck, 2015; Nesterov, 2018). The bounded gradient assumption is likewise common in the analysis of asynchronous SGD (Mishchenko et al., 2022; Shi et al., 2024), as well as in the analysis of clipped gradient descent (Zhang et al., 2020a;b), which we employ in non-restart rounds.

Before we state the main theorem, we need to make an additional weak technical assumption on the local robust aggregator $\mathbb{A}$.

Assumption 4 (Bounded local contributions)**.** There exists $G' \geq 0$ such that, for every side information S used by the algorithm and every $\boldsymbol{v} \in \mathbb{R}^d$, $\|\mathbb{A}(\boldsymbol{v}; S)\| \leq G'$.

Remark 2 (Centered-clipping example)**.** Suppose that the side information is $S = (\boldsymbol{a}_r, 2G)$, where the projected anchor in Example 1 satisfies $\|\boldsymbol{a}_r\| \leq G$. For the centered-clipping map $\mathbb{A}(\boldsymbol{v}; \mathbf{S}) = \boldsymbol{a}_r + \mathrm{clip}_{2G}(\boldsymbol{v} - \boldsymbol{a}_r)$, we have $\|\mathbb{A}(\boldsymbol{v}; S)\| \leq G + 2G = 3G$ for every input, including a Byzantine input, verifying Assumption 4. Moreover, Assumption 3 implies that an honest input satisfies $\|\boldsymbol{v}\| \leq G$, and hence $\|\boldsymbol{v} - \boldsymbol{a}_r\| \leq 2G$, so centered clipping therefore leaves every honest input unchanged. Similar assumptions are used in (Malinovsky et al., 2024, Assumption 2.3).

Now, we present the main theorem of this paper. We analyze the convergence in terms of effective steps, which quantify the number of steps normalized by the decreasing stepsizes. In the synchronous setting, that is if every client submits the feedback at the same time, the number of effective steps T_{eff} is exactly the number of feedbacks (T in the synchronous setting). Moreover, it closely corresponds to the number R of rounds as $R \leq T_{\text{eff}} = R + \sum_{r=1}^{R} \sum_{i=1}^{n} \frac{1-q^{-(c_{i,r}-1)}}{q-1}$.

Theorem 1 (Non-convex convergence rate)**.** *Let $T \geq 2$ and $q > 1$. Suppose that Assumptions 1, 2, and 3, as well as Assumption 4 with $G' \leq 3G$ hold, together with the local (c, δ)-robust aggregator $\mathbb{A}$, and use*[1] *$\lambda_t = G$. Then, there exists a small enough stepsize $\eta > 0$ (detailed in Appendix C.2) such that,*

$$\mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{out}})\|^2 = \mathcal{O}\Bigg(\left(c\delta\sigma^2 + \frac{G^2|\mathcal{B}|^2}{(q-1)^2} \right) \theta_T + \sqrt{\frac{L\sigma^2\Delta}{T_{\text{eff}}} \left(1 - \theta_T + \frac{\theta_T}{|\mathcal{G}|} \right)} + \left(\frac{L\Delta}{T_{\text{eff}}} \right)^{2/3} \left[(1-\theta_T) \left((n-1)^2 G^2 + c\delta\sigma^2 + \frac{G^2|\mathcal{B}|^2}{(q-1)^2} \right) \right]^{1/3} + \frac{L\Delta}{T_{\text{eff}}} \Bigg),$$

where $\Delta \coloneqq f(\boldsymbol{x}^{(0)}) - f_*$ is the initial distance, $T_{\text{eff}} \coloneqq 1 + \sum_{t=1}^{T-1} \hat{\eta}_t/\eta$ is the number of effective time steps, $N_T \coloneqq 1 + \sum_{t=1}^{T-1} \mathbf{1}_{\{t=r(t)\}}$ is the number of rounds, $\theta_T \coloneqq \frac{N_T}{T_{\text{eff}}}$ is the restart ratio, and $\boldsymbol{x}^{(\text{out})}$ is the weighted average of $\boldsymbol{x}^{(0)}, \ldots, \boldsymbol{x}^{(T-1)}$ defined in (4).

Remark 3 (Synchronous limit $q \to \infty$)**.** As $q \to \infty$, an effective step corresponds to a round. Thus, $T_{\text{eff}} \to R$ and $\theta_T \to 1$, where R is the total number of rounds including the initial round. The convergence rate simplifies to

$$\mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{out}})\|^2 = \mathcal{O}\left(\sqrt{\frac{L\sigma^2\Delta}{|\mathcal{G}|R}} + \frac{L\Delta}{R} + c\delta\sigma^2 \right),$$

which recovers the standard rate of synchronous Byzantine-robust SGD. The non-vanishing term $\mathcal{O}(c\delta\sigma^2)$ is known to be unavoidable for any (c, δ)-robust aggregator applied only to the current stochastic gradients, and it can be removed by momentum or variance reduction (Karimireddy et al., 2021; Gorbunov et al., 2023). For finite q, however, the dominant error is the asynchrony-induced term $G^2|\mathcal{B}|^2/(q-1)^2$, which persists even with full gradients ($\sigma = 0$). Note also that $\sigma^2 \leq G^2$ in the non-clipping regime. Hence, momentum or variance reduction alone cannot remove the main source of error in the asynchronous setting, and combining momentum with stale gradients is itself nontrivial (Shi et al., 2024). We leave reducing both terms to future work.

Remark 4 (Non-Byzantine asynchronous limit $|\mathcal{B}| = 0$, $q \to 1$)**.** If there are no Byzantine clients and $q \to 1$, then we have $T_{\text{eff}} \to T$ and the convergence rate simplifies to

$$\mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{out}})\|^2 = \mathcal{O}\left(\sqrt{\frac{L\sigma^2\Delta}{T}} + \left(\frac{L\Delta}{T} \right)^{2/3} \left[(n-1)^2 G^2 \right]^{1/3} + \frac{L\Delta}{T} \right),$$

which is a similar convergence rate (up to the T^{-1} term due to normalizing rounds gradients by $1/n$) to asynchronous SGD without Byzantine clients (Mishchenko et al., 2022, Theorem 2).

[1] Under Assumption 3, honest stochastic gradients satisfy $\|\nabla F(\boldsymbol{x}; \xi)\| \leq G$ almost surely. With $\lambda_t = G$, the clipping operator therefore acts as the identity on honest gradients, which simplifies the analysis. The same assumption also gives $\sigma^2 \leq G^2$. In our experiments, we use a smaller threshold ($\lambda_t = 1$), for which clipping is typically active. Extending the analysis to this regime is left for future work.

5 Related Work

Asynchronous SGD. Asynchronous optimization has a long history dating back to the 1970s, when Baudet (1978) studied asynchronous fixed-point iterations. Subsequently, Tsitsiklis et al. (1986) established the convergence of asynchronous stochastic gradient methods in distributed settings. In the 2010s, attention turned to asynchronous variants of SGD. A notable example is Hogwild! (Recht et al., 2011), a lock-free scheme that has since been widely adopted. Classical analyses of asynchronous SGD (Agarwal & Duchi, 2012; Lian et al., 2015; Mania et al., 2017; Stich & Karimireddy, 2020) rely on a bounded delay assumption, so their convergence rates depend on the maximum delay and are often overly pessimistic. Koloskova et al. (2022) and Mishchenko et al. (2022) concurrently showed that asynchronous SGD attains convergence rates that do not depend on the maximum delay, significantly improving upon prior analyses. Since then, a growing body of work has built upon these results. For instance, Shi et al. (2024) designed a momentum scheme suited to asynchronous SGD. Islamov et al. (2024) developed a unified framework that covers many asynchronous SGD variants, including FedBuff (Nguyen et al., 2022). More broadly, asynchronous methods have been studied under data, computation, and communication heterogeneity (Tyurin & Richtárik, 2023; Tyurin et al., 2024b; Maranjyan et al., 2025), and extended to finite-sum settings (Tyurin et al., 2024a).

Byzantine-robust Learning. Classical Byzantine-robust methods in the synchronous setting replace mean aggregation with a robust aggregator, such as coordinate-wise median and trimmed mean (Yin et al., 2018), Krum (Blanchard et al., 2017), geometric median (RFA) (Pillutla et al., 2022), MDA (El-Mhamdi et al., 2020), and Bulyan (Mhamdi et al., 2018). However, due to the noise inherent in stochastic gradients, attackers can craft malicious updates that are statistically indistinguishable from honest ones and yet prevent convergence, as in the ALIE (Baruch et al., 2019) and IPM (Xie et al., 2019) attacks. Another line of work complements robust aggregation with momentum (Karimireddy et al., 2021; Mhamdi et al., 2021; Farhadkhani et al., 2022) or variance reduction (Gorbunov et al., 2023), and the optimality of such approaches has been studied in (Yin et al., 2018; Zhu et al., 2023; Shi et al., 2025). These results, however, assume synchronous updates. Incorporating momentum under asynchrony is nontrivial even in the absence of Byzantine workers, since it accumulates stale gradients (Shi et al., 2024). Byzantine robustness has further been studied under more practical conditions, including data heterogeneity (Karimireddy et al., 2022; Allouah et al., 2023a;b), heterogeneous client participation (Allouah et al., 2024; Malinovsky et al., 2024; Otsuka et al., 2026), and communication compression (Gorbunov et al., 2023; Rammal et al., 2024).

Asynchronous Byzantine-robust Learning. Combining asynchrony with Byzantine robustness is challenging because asynchronous SGD applies each stochastic gradient immediately upon arrival, leaving no natural averaging step to replace with a robust aggregator. One line of work addresses this by partially *synchronizing* the updates: BASGD and BASGDm (Yang & Li, 2021; 2023) aggregate over server-side buffers, and Dahan & Levy (2024) aggregate each incoming gradient with the latest gradients of the other clients using a weighting scheme that mitigates staleness. Another line of work filters incoming gradients using additional information. Zeno++ (Xie et al., 2020b) and AFLGuard (Fang et al., 2022) rely on a trusted dataset held by the server, an assumption that may undermine the computational efficiency of distributed learning. While Kardam (Damaskinos et al., 2018) filters updates based on an empirical estimate of the smoothness constant, but may reject honest gradients along with malicious ones (Xie et al., 2020b).

6 Experiments

Our experiments show the benefits of Throttle both with and without Byzantine clients. In the Byzantine setting, it remained competitive under most standard attacks and maintains high accuracy under flooding attacks that severely degrade several existing methods. In the non-Byzantine setting, it achieved lower optimization error than minibatch SGD, asynchronous SGD, and the other evaluated baselines under the same gradient computation budget.

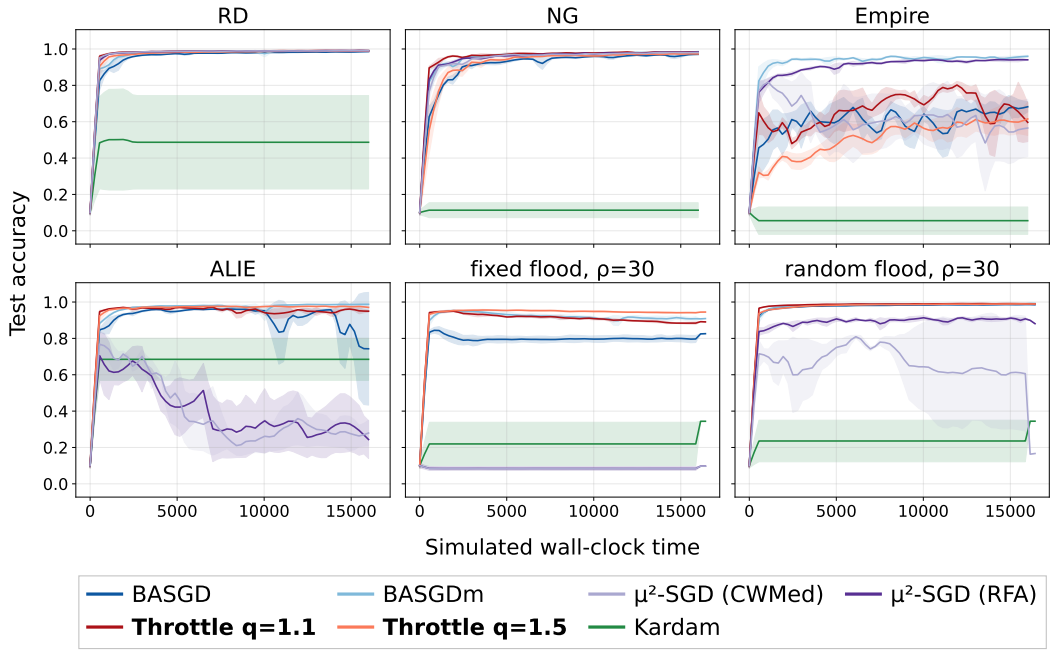


Figure 2: Test accuracy on MNIST with $n = 25$ clients, $|\mathcal{B}| = 5$ Byzantine clients. Standard attacks (RD,NG,Empire,ALIE) under the scheduler based on the codebase[3], in which Byzantine updates are at most $1/3$ of all gradients at any time. Flooding attacks under using a fixed vector/random vectors. The higher ρ, the more frequently the Byzantine clients send their updates. THROTTLE keeps a higher accuracy overall. Figure 6 also shows the results with other choices of $\rho = 1, 3, 10$.

6.1 THROTTLE IS BYZANTINE-ROBUST

We trained convolutional neural networks using our optimizer and other existing asynchronous SGD algorithms on MNIST with $n = 25$ clients and $|\mathcal{B}| = 5$ Byzantine clients. We compared THROTTLE against four standard attacks RD (Random Disturbance attack), NG (Negative Gradient attack (Yang & Li, 2023), Empire (Xie et al., 2020a), ALIE (Baruch et al., 2019) and eight flooding attacks (Figure 2 and Figure 6).[2] We compared our proposed method with Kardam (Damaskinos et al., 2018), BASGD and BASGDm (Yang & Li, 2021; 2023), and Asynchronous Robust μ^2-SGD (Dahan & Levy, 2024) based on their codebase. [3] Following Dahan & Levy (2024), we simulate asynchronous training as a sequence of updates. For the standard attacks in Figure 2 every third arrival is a Byzantine update and the arriving client is drawn with probability proportional to its index for other updates, as in Dahan & Levy (2024). In the flooding experiments, client i is an independent Poisson process with rate $r_i \propto i$ and the Byzantine rates are multiplied by ρ, giving a Byzantine fraction of updates $\rho/(2+\rho)$ ($1/3$ to 0.94 for $\rho = 1$ to 30). Appendix E gives additional details including all hyperparameter grids, attack details, and asynchrony.

THROTTLE is robust and outperforms existing algorithms in almost all type of attacks. THROTTLE outperformed BASGDm under fixed flooding even with increased ρ, and outperformed μ^2-SGD with RFA under both ALIE and flooding attacks. The exception was Empire, where THROTTLE achieved 59.6% and 61.5%, compared with 96.0% for BASGDm and 94.0% for μ^2-SGD with RFA. Since both baselines used momentum, we expect that adding momentum to THROTTLE, together with tuning the clipping radius, would narrow this gap. We leave this to future work.

[2]For ALIE and Empire, we used the latest gradients from each honest client as the information for them.
[3]https://github.com/dahan198/asynchronous-fault-tolerant-ml

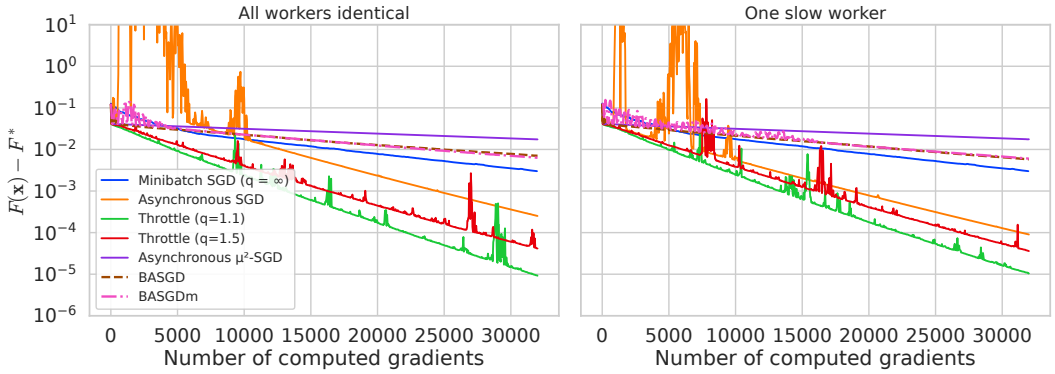


Figure 3: We ran experiments without Byzantine clients on a simple least-squares problem with random data and tuned all stepsizes for each methods under pure asynchrony using Ray[4] framework. We averaged over three seeds. *Left:* all clients compute gradients at similar speeds. *Right:* one worker sleeps for $100\times$ its compute time after every gradient computation. **THROTTLE (green) achieves faster convergence** because the *soft throttling* mechanism allows us to take slightly larger stepsize η than asynchronous SGD.

6.2 THROTTLE OUTPERFORMS STANDARD ASYNCHRONOUS SGD

Following Mishchenko et al. (2022), we study least-squares optimization with 40 asynchronous Ray [4] workers and no Byzantine clients (Figure 3). We compare THROTTLE with $q \in \{1.1, 1.5\}$ against minibatch SGD ($q = \infty$), asynchronous SGD, μ^2-SGD, BASGD, and BASGDm[5]. We tuned the stepsizes separately for each method. All methods use a budget of 32,000 computed gradients, including those discarded at hard restarts by THROTTLE. We plot the optimality gap against this count. The right panel introduces one straggler that sleeps for $100\times$ its gradient computation time after each computation. Appendix E.2 provides further details.

Effectiveness of soft throttling. With a tuned stepsize THROTTLE performs significantly better than its two limits. Among the tested values, $q = 1.1$ achieves the lowest final optimality gap in both worker configurations. Although $q = 1.5$ converges more slowly per computed gradient, it still outperforms asynchronous SGD and minibatch SGD. Asynchronous μ^2-SGD, BASGD, and BASGDm did not utilize asynchrony into faster convergence. All three existing methods had slower convergence than minibatch SGD even in the absence of Byzantine clients.

Soft throttling remains effective with a straggler. Introducing a worker that sleeps for $100\times$ its computation time has little effect on the convergence of THROTTLE per computed gradient (Figure 3, right). Even at $q = 1.5$, convergence per computed gradient is largely preserved relative to the setting without a straggler, although convergence becomes less stable.

7 CONCLUSION

We introduce soft throttling, an exponential down-weighting of updates by the speed of each client. We show that the resulting algorithm, THROTTLE, is a generalization of both synchronous Byzantine-robust SGD ($q \to \infty$), and (almost) asynchronous SGD ($q \to 1$) without Byzantine clients. The convergence rate of THROTTLE matches the known rate at each limit and holds against flooding attacks without any bound on the Byzantine update ratio. Our experiments demonstrate robustness to attacks and show that soft throttling accelerates convergence even in the absence of Byzantine clients.

[4] `https://docs.ray.io/en/latest/ray-core/examples/plot_parameter_server.html#asynchronous-parameter-server-training`

[5] For the latter three methods, we replace robust aggregation with averaging while preserving any weighting scheme used by the original method. Gradient clipping was also disabled $\lambda_t = \infty$ for THROTTLE.

AI USE STATEMENT

We used generative AI tools (large language models and LLM-based coding agents) in this work as follows.

Theory. All definitions, main theorem statements, proof strategies, and the proposed algorithm were developed by the authors. LLMs were used to check proofs for errors, correct minor mistakes, verify numerical constants, refine notation, and convert the authors' arguments into polished formal LaTeX.

Experiments. LLM-based coding agents were used to implement methods, design and refine experimental pipelines, and adapt publicly available code from prior works. LLMs also provided limited assistance with interpreting results and qualitative analysis, including cross-checking generated plots against the intended algorithm.

Writing and figures. We used generative AI tools for drafting and editing parts of the manuscript based on the authors' initial drafts, and for producing the figures as TikZ code from the authors' hand-drawn illustrations.

Exclusions. We did not use generative AI to generate synthetic datasets or to clean or reformat datasets, propose or refine hypotheses, develop the theoretical model or conceptual framework, or propose research ideas.

All LLM-assisted work has been reviewed and tested by the authors. Proofs were independently checked after LLM-assisted revision, and all LLM-assisted text was verified. We take full responsibility for the final content of this work.

REFERENCES

Alekh Agarwal and John C. Duchi. Distributed delayed stochastic optimization. In *IEEE Conference on Decision and Control*, 2012.

Youssef Allouah, Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Rafael Pinot, and John Stephan. Fixing by mixing: A recipe for optimal byzantine ML under heterogeneity. In *International Conference on Artificial Intelligence and Statistics*, 2023a.

Youssef Allouah, Rachid Guerraoui, Nirupam Gupta, Rafael Pinot, and Geovani Rizk. Robust distributed learning: Tight error bounds and breakdown point under data heterogeneity. In *Advances in Neural Information Processing Systems*, 2023b.

Youssef Allouah, Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Rafael Pinot, Geovani Rizk, and Sasha Voitovych. Byzantine-robust federated learning: Impact of client subsampling and local updates. In *International Conference on Machine Learning*, 2024.

Gilad Baruch, Moran Baruch, and Yoav Goldberg. A little is enough: Circumventing defenses for distributed learning. In *Advances in Neural Information Processing Systems*, 2019.

Gerard M Baudet. Asynchronous iterative methods for multiprocessors. In *Journal of the ACM*, 1978.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. In *Advances in Neural Information Processing Systems*, 2017.

Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and trends in Machine Learning*, 8(3-4):231–357, 2015.

Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting distributed synchronous sgd. In *International Conference on Learning Representations Workshop Track*, 2016.

Yudong Chen, Lili Su, and Jiaming Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2017.

Tehila Dahan and Kfir Y. Levy. Weight for robustness: A comprehensive approach towards optimal fault-tolerant asynchronous ML. In *Advances in Neural Information Processing Systems*, 2024.

Georgios Damaskinos, Rachid Guerraoui, Rhicheek Patra, Mahsa Taziki, et al. Asynchronous byzantine machine learning (the case of SGD). In *International Conference on Machine Learning*, 2018.

El-Mahdi El-Mhamdi, Rachid Guerraoui, Arsany Guirguis, Lê Nguyên Hoang, and Sébastien Rouault. Genuinely distributed byzantine machine learning. In *Symposium on Principles of Distributed Computing*, 2020.

Minghong Fang, Jia Liu, Neil Zhenqiang Gong, and Elizabeth S. Bentley. Aflguard: Byzantine-robust asynchronous federated learning. In *Annual Computer Security Applications Conference*, 2022.

Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Rafael Pinot, and John Stephan. Byzantine machine learning made easy by resilient averaging of momentums. In *International Conference on Machine Learning*, 2022.

Eduard Gorbunov, Samuel Horváth, Peter Richtárik, and Gauthier Gidel. Variance reduction is an antidote to byzantines: Better rates, weaker assumptions and communication compression as a cherry on the top. In *International Conference on Learning Representations*, 2023.

Samuel Horvath, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. In *Advances in Neural Information Processing Systems*, 2021.

Peter J. Huber. Robust statistics. In *Wiley Series in Probability and Statistics*, 2005.

Rustem Islamov, Mher Safaryan, and Dan Alistarh. AsGrad: A sharp unified analysis of asynchronous-SGD algorithms. In *International Conference on Artificial Intelligence and Statistics*, 2024.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. In *Foundations and Trends in Machine Learning*, 2021.

Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Learning from history for byzantine robust optimization. In *International Conference on Machine Learning*, 2021.

Sai Praneeth Karimireddy, Lie He, and Martin Jaggi. Byzantine-robust learning on heterogeneous datasets via bucketing. In *International Conference on Learning Representations*, 2022.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian U. Stich. A unified theory of decentralized SGD with changing topology and local updates. In *International Conference on Machine Learning*, 2020.

Anastasia Koloskova, Sebastian U. Stich, and Martin Jaggi. Sharper convergence guarantees for asynchronous SGD for distributed and federated learning. In *Advances in Neural Information Processing Systems*, 2022.

Anastasia Koloskova, Nikita Doikov, Sebastian U Stich, and Martin Jaggi. On convergence of incremental gradient for non-convex smooth functions. In *International Conference on Machine Learning*, 2024.

Anastasiia Koloskova, Ryan McKenna, Zachary Charles, John Rush, and H Brendan McMahan. Gradient descent with linearly correlated noise: Theory and applications to differential privacy. In *Advances in Neural Information Processing Systems*, 2023.

Leslie Lamport, Robert Shostak, and Marshall Pease. *The Byzantine generals problem*. Association for Computing Machinery (ACM) / ACM Books, 10 2019. ISBN 9781450372701. doi: 10.1145/3335772.3335936.

Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, 2015.

Grigory Malinovsky, Peter Richtárik, Samuel Horváth, and Eduard Gorbunov. Byzantine robustness and partial participation can be achieved at once: Just clip gradient differences. In *Advances in Neural Information Processing Systems*, 2024.

Horia Mania, Xinghao Pan, Dimitris Papailiopoulos, Benjamin Recht, Kannan Ramchandran, and Michael I Jordan. Perturbed iterate analysis for asynchronous stochastic optimization. In *SIAM Journal on Optimization*, 2017.

Arto Maranjyan, Alexander Tyurin, and Peter Richtárik. Ringmaster ASGD: The first asynchronous SGD with optimal time complexity. In *Forty-second International Conference on Machine Learning*, 2025.

El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, 2018.

El Mahdi El Mhamdi, Rachid Guerraoui, and Sébastien Rouault. Distributed momentum for byzantine-resilient stochastic gradient descent. In *International Conference on Learning Representations*, 2021.

Konstantin Mishchenko, Francis R. Bach, Mathieu Even, and Blake E. Woodworth. Asynchronous SGD beats minibatch SGD under arbitrary delays. In *Advances in Neural Information Processing Systems*, 2022.

Yurii Nesterov. *Lectures on convex optimization*. Springer, 2018.

John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. In *International conference on artificial intelligence and statistics*, 2022.

Kaoru Otsuka, Yuki Takezawa, and Makoto Yamada. Delayed momentum aggregation: Communication-efficient byzantine-robust federated learning with partial participation. In *International Conference on Machine Learning*, 2026.

Krishna Pillutla, Sham M. Kakade, and Zaïd Harchaoui. Robust aggregation for federated learning. In *IEEE Transactions on Signal Processing*, 2022.

Ahmad Rammal, Kaja Gruntkowska, Nikita Fedin, Eduard Gorbunov, and Peter Richtárik. Communication compression for byzantine robust learning: New efficient algorithms and improved rates. In *International Conference on Artificial Intelligence and Statistics*, 2024.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*, 2011.

Max Ryabinin, Eduard Gorbunov, Vsevolod Plokhotnyuk, and Gennady Pekhimenko. Moshpit SGD: Communication-efficient decentralized training on heterogeneous unreliable devices. In *Advances in Neural Information Processing Systems*, 2021.

Chang-Wei Shi, Yi-Rui Yang, and Wu-Jun Li. Ordered momentum for asynchronous SGD. In *Advances in Neural Information Processing Systems*, 2024.

Qiankun Shi, Jie Peng, Kun Yuan, Xiao Wang, and Qing Ling. Optimal complexity in byzantine-robust distributed stochastic optimization with data heterogeneity. In *Journal of Machine Learning Research*, 2025.

Sebastian U Stich and Sai Praneeth Karimireddy. The error-feedback framework: Sgd with delayed gradients. In *Journal of Machine Learning Research*, 2020.

J. Tsitsiklis, D. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. In *IEEE Transactions on Automatic Control*, 1986.

Alexander Tyurin and Peter Richtárik. Optimal time complexities of parallel stochastic optimization methods under a fixed computation model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Alexander Tyurin, Kaja Gruntkowska, and Peter Richtárik. Freya PAGE: First optimal time complexity for large-scale nonconvex finite-sum optimization with heterogeneous asynchronous computations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

Alexander Tyurin, Marta Pozzi, Ivan Ilin, and Peter Richtárik. Shadowheart SGD: Distributed asynchronous SGD with optimal time complexity under arbitrary computation and communication heterogeneity. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b.

Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S Rellermeyer. A survey on distributed machine learning. *Acm computing surveys (csur)*, 53(2): 1–33, 2020.

Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno: Distributed stochastic gradient descent with suspicion-based fault-tolerance. In *International Conference on Machine Learning*, 2019.

Cong Xie, Oluwasanmi Koyejo, and Indranil Gupta. Fall of empires: Breaking byzantine-tolerant SGD by inner product manipulation. In *Uncertainty in Artificial Intelligence*, 2020a.

Cong Xie, Sanmi Koyejo, and Indranil Gupta. Zeno++: Robust fully asynchronous SGD. In *International Conference on Machine Learning*, 2020b.

Tao Yang, Xinlei Yi, Junfeng Wu, Ye Yuan, Di Wu, Ziyang Meng, Yiguang Hong, Hong Wang, Zongli Lin, and Karl H Johansson. A survey of distributed optimization. *Annual Reviews in Control*, 47:278–305, 2019.

Yi-Rui Yang and Wu-Jun Li. Basgd: Buffered asynchronous sgd for byzantine learning. In *International conference on machine learning*, 2021.

Yi-Rui Yang and Wu-Jun Li. Buffered asynchronous SGD for byzantine learning. In *Journal of Machine Learning Research*, 2023.

Yi-Rui Yang, Chang-Wei Shi, and Wu-Jun Li. On the effect of batch size in byzantine-robust distributed learning. In *International Conference on Learning Representations*, 2024.

Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International conference on machine learning*, 2018.

Bohang Zhang, Jikai Jin, Cong Fang, and Liwei Wang. Improved analysis of clipping algorithms for non-convex optimization. *Advances in Neural Information Processing Systems*, 2020a.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie. Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*, 2020b.

Banghua Zhu, Lun Wang, Qi Pang, Shuai Wang, Jiantao Jiao, Dawn Song, and Michael I. Jordan. Byzantine-robust federated learning with optimal statistical rates. In *International Conference on Artificial Intelligence and Statistics*, 2023.

A NOTATIONS

In Algorithm 1, we write

- $\boldsymbol{x}^{(0)} \in \mathbb{R}^d$ the initial parameter,
- n the total number of clients,
- $q \in [1, \infty]$ the scaling factor,
- T the number of iterations.
- S_r the start of the r^{th} round,
- $r(t) := \max\{S_r : S_r \leq t\}$ the current round number,
- $j(t)$ the current client,
- $c_i(t) := \sum_{s=r(t)}^{t-1} \mathbf{1}_{\{j_s=i\}}$ the number of gradients that the client i has already delivered in the current round,
- $\boldsymbol{v}_{j_t}^{(t-\tau_t)} = \begin{cases} \nabla F(\boldsymbol{x}^{(t-\tau_t)}; \xi_{j_t}^{(t-\tau_t)}) & \text{if } j_t \in \mathcal{G} \\ * & \text{if } j_t \in \mathcal{B} \end{cases}$ the feedback received,
- $\eta_t := \begin{cases} \frac{\eta}{n} & \text{if } c_{j_t}(t) = 0, \\ \frac{\eta}{q^{c_{j_t}(t)}} & \text{else.} \end{cases}$ the stepsize,
- $\mathbf{S}_{r(t)}$ the current side information.

B ANALYZING ASYNCHRONOUS SGD THROUGH THE VIRTUAL ITERATE

Koloskova et al. (2022); Mishchenko et al. (2022) study a virtual iterate $\tilde{\boldsymbol{x}}^{(t)}$, where every update is performed on the current model instead of the outdated version,

$$\tilde{\boldsymbol{x}}^{(t)} := \tilde{\boldsymbol{x}}^{(t-1)} - \eta \nabla F(\boldsymbol{x}^{(t-1)}). \tag{3}$$

The sum of the updates of the virtual iterate over T steps is simply $\tilde{S} = \tilde{\boldsymbol{x}}^{(0)} + \sum_{t=1}^{T} \eta \nabla F(\boldsymbol{x}^{(t-1)})$. The key insight is that the sum of the updates of the actual iterate $S = \boldsymbol{x}^{(0)} + \sum_{t=1}^{T} \eta \nabla F(\boldsymbol{x}^{(t-\tau_t)})$ corresponds almost exactly to $\tilde{S}$. It only differs by the first factor $\boldsymbol{x}^{(0)}$ and the fact that the update $\nabla F(\boldsymbol{x}^{(0)})$ appears n times in S but only once in $\tilde{S}$, which comprises later updates instead (S can be seen as being "delayed" compared to $\tilde{S}$). Therefore, both iterates differ by at most $n-1$ stochastic gradients, regardless of T (Mishchenko et al., 2022, Lemma 1). Standard Synchronous SGD techniques can be used to analyse the convergence of the virtual iterate, and hence that of the real iterate $\boldsymbol{x}^{(t)}$.

Adapting to the Byzantine Setting. However, this virtual iterate loses its usefulness in the Byzantine setting. Indeed, it can consist of an arbitrarily large proportion of Byzantine updates, and hence be arbitrarily bad. This contrasts with standard Byzantine distributed learning settings and explains the added assumptions of previous work (Dahan & Levy, 2024). We remove this assumption by considering a modified version of the virtual iterate, which we introduce in Appendix C.

C CONVERGENCE PROOFS

Our proofs build on *perturbed iterate analysis* (Mania et al., 2017; Stich & Karimireddy, 2020). Existing analyses of asynchronous SGD (Mishchenko et al., 2022; Koloskova et al., 2022) also rely on this technique, approximating the asynchronous iterate by a *synchronized* virtual iterate. A related construction is the *virtual sequence with restart* (Koloskova et al., 2023), later adapted to incremental gradient methods (Koloskova et al., 2024). Islamov et al. (2024) combined the two ideas through a two-stage virtual sequence to analyze asynchronous SGD with shuffling.

Inspired by Islamov et al. (2024), our analysis of THROTTLE also uses a two-stage virtual sequence. The first stage is the standard approximation of the asynchronous iterate by a synchronized one

(Fig. 4). The second approximates this synchronized iterate, which still contains Byzantine updates, by a synchronized iterate free of them (Fig. 5):

$$\boldsymbol{x}^{(t)} \xrightarrow{\text{synchronization}} \boldsymbol{y}^{(t)} \xrightarrow{\text{de-Byzantinization}} \boldsymbol{z}^{(t)}$$

real iterate "synchronized" iterate de-Byzantinized iterate

Both the de-Byzantinization step and its combination with asynchronous virtual sequences with restart are new to our analysis.

In Section C.1, we show that perturbed iterate analysis also applies to standard Byzantine-robust SGD analysis (with momentum, for generality). Section C.2 then presents the full analysis of THROTTLE using the two-stage virtual sequence above.

C.1 VIRTUAL ITERATE ANALYSIS FOR SYNCHRONOUS BYZANTINE-ROBUST SGD

In this section, we present a new analysis technique for Byzantine-robust SGD. It recovers the known convergence rate and, more importantly, provides the proof strategy for our main theorem. Consider the actual iterates of Byzantine-robust SGD with momentum. We include momentum just for generality ($\alpha = 1$ reduces to SGD).

$$\boldsymbol{x}^{(t)} \coloneqq \boldsymbol{x}^{(t-1)} - \eta \operatorname{Agg}\big(\{\boldsymbol{m}_i^{(t)}\}_{i=1}^n\big), \qquad \boldsymbol{m}_i^{(t)} \coloneqq (1-\alpha)\,\boldsymbol{m}_i^{(t-1)} + \alpha \nabla F(\boldsymbol{x}^{(t-1)}; \xi_i^{(t)}).$$

We construct the virtual sequence $\{\boldsymbol{y}^{(t)}\}_{t=0}^{T-1}$ by

$$\boldsymbol{y}^{(t)} \coloneqq \boldsymbol{x}^{(t-1)} - \eta \bar{\boldsymbol{m}}^{(t)}, \qquad \boldsymbol{y}^{(0)} \coloneqq \boldsymbol{x}^{(0)},$$

that is, the point reached from $\boldsymbol{x}^{(t-1)}$ when the aggregate is replaced by the average momentum of honest clients.

Lemma 1 (Descent Lemma for Virtual Iterate Analysis for Synchronous Byzantine-robust SGDm)**.** *Suppose the stepsizes satisfy $0 < \eta_t \le 1/(8L)$. and denote $\mathbb{E}_t[\cdot] := \mathbb{E}[\cdot \mid \mathcal{F}_{t-1}]$, where $\mathcal{F}_{t-1}$ contains the full history through round $t-1$. Then, for every $t \ge 2$,*

$$\mathbb{E}_t\big[f(\boldsymbol{y}^{(t)})\big] \le f(\boldsymbol{y}^{(t-1)}) - \frac{3\eta_t}{8}\big\|\nabla f(\boldsymbol{x}^{(t-1)})\big\|^2 + \frac{5\eta_t}{4}\mathbb{E}_t\big\|\bar{\boldsymbol{m}}^{(t)} - \nabla f(\boldsymbol{x}^{(t-1)})\big\|^2 + \underbrace{\frac{3}{2\eta_t}\big\|\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}\big\|^2}_{=\frac{3\eta_{t-1}^2}{2\eta_t}\left\|\boldsymbol{m}^{(t-1)} - \bar{\boldsymbol{m}}^{(t-1)}\right\|^2}.$$

Proof. By the definition of the virtual iterate,

$$\boldsymbol{y}^{(t)} = \boldsymbol{y}^{(t-1)} - \eta_t \bar{\boldsymbol{m}}^{(t)} + \big(\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}\big).$$

Therefore, L-smoothness gives

$$\begin{aligned} f(\boldsymbol{y}^{(t)}) \le f(\boldsymbol{y}^{(t-1)}) &- \eta_t \left\langle \nabla f(\boldsymbol{y}^{(t-1)}), \bar{\boldsymbol{m}}^{(t)} \right\rangle \\ &+ \left\langle \nabla f(\boldsymbol{y}^{(t-1)}), \boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)} \right\rangle \\ &+ \frac{L\eta_t^2}{2}\left\| \bar{\boldsymbol{m}}^{(t)} - \frac{\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}}{\eta_t} \right\|^2. \end{aligned}$$

Adding and subtracting $\nabla f(\boldsymbol{x}^{(t-1)})$ inside the inner product, while retaining the virtual iterate gap, yields

$$\begin{aligned} f(\boldsymbol{y}^{(t)}) \le f(\boldsymbol{y}^{(t-1)}) &- \eta_t \left\langle \nabla f(\boldsymbol{y}^{(t-1)}), \nabla f(\boldsymbol{x}^{(t-1)}) \right\rangle \\ &- \eta_t \left\langle \nabla f(\boldsymbol{y}^{(t-1)}), \bar{\boldsymbol{m}}^{(t)} - \nabla f(\boldsymbol{x}^{(t-1)}) - \frac{\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}}{\eta_t} \right\rangle + \frac{L\eta_t^2}{2}\left\| \bar{\boldsymbol{m}}^{(t)} - \frac{\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}}{\eta_t} \right\|^2. \end{aligned}$$

For the first inner product, polarization identity and L-smoothness imply

$$\begin{aligned} -\eta_t \left\langle \nabla f(\boldsymbol{y}^{(t-1)}), \nabla f(\boldsymbol{x}^{(t-1)}) \right\rangle &= -\frac{\eta_t}{2}\big\|\nabla f(\boldsymbol{x}^{(t-1)})\big\|^2 - \frac{\eta_t}{2}\big\|\nabla f(\boldsymbol{y}^{(t-1)})\big\|^2 + \frac{\eta_t}{2}\big\|\nabla f(\boldsymbol{x}^{(t-1)}) - \nabla f(\boldsymbol{y}^{(t-1)})\big\|^2 \\ &\le -\frac{\eta_t}{2}\big\|\nabla f(\boldsymbol{x}^{(t-1)})\big\|^2 - \frac{\eta_t}{2}\big\|\nabla f(\boldsymbol{y}^{(t-1)})\big\|^2 + \frac{\eta_t L^2}{2}\big\|\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}\big\|^2. \end{aligned}$$

For the second inner product, Young's inequality gives

$$-\eta_t \left\langle \nabla f(\boldsymbol{y}^{(t-1)}), \bar{\boldsymbol{m}}^{(t)} - \nabla f(\boldsymbol{x}^{(t-1)}) - \frac{\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}}{\eta_t} \right\rangle$$
$$\leq \frac{\eta_t}{2}\left\|\nabla f(\boldsymbol{y}^{(t-1)})\right\|^2 + \frac{\eta_t}{2}\left\|\bar{\boldsymbol{m}}^{(t)} - \nabla f(\boldsymbol{x}^{(t-1)}) - \frac{\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}}{\eta_t}\right\|^2.$$

Combining these inequalities cancels the squared gradient norm at $\boldsymbol{y}^{(t-1)}$, leaving

$$f(\boldsymbol{y}^{(t)}) \leq f(\boldsymbol{y}^{(t-1)}) - \frac{\eta_t}{2}\left\|\nabla f(\boldsymbol{x}^{(t-1)})\right\|^2 + \frac{\eta_t L^2}{2}\left\|\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}\right\|^2$$
$$+ \frac{\eta_t}{2}\left\|\bar{\boldsymbol{m}}^{(t)} - \nabla f(\boldsymbol{x}^{(t-1)}) - \frac{\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}}{\eta_t}\right\|^2 + \frac{L\eta_t^2}{2}\left\|\bar{\boldsymbol{m}}^{(t)} - \frac{\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}}{\eta_t}\right\|^2.$$

Next, inserting $\nabla f(\boldsymbol{x}^{(t-1)})$ into the last squared norm gives

$$\left\|\bar{\boldsymbol{m}}^{(t)} - \frac{\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}}{\eta_t}\right\|^2 \leq 2\left\|\nabla f(\boldsymbol{x}^{(t-1)})\right\|^2 + 2\left\|\bar{\boldsymbol{m}}^{(t)} - \nabla f(\boldsymbol{x}^{(t-1)}) - \frac{\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}}{\eta_t}\right\|^2.$$

Moreover,

$$\left\|\bar{\boldsymbol{m}}^{(t)} - \nabla f(\boldsymbol{x}^{(t-1)}) - \frac{\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}}{\eta_t}\right\|^2 \leq 2\left\|\bar{\boldsymbol{m}}^{(t)} - \nabla f(\boldsymbol{x}^{(t-1)})\right\|^2 + \frac{2}{\eta_t^2}\left\|\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}\right\|^2.$$

Substituting both bounds and collecting terms, we obtain

$$f(\boldsymbol{y}^{(t)}) \leq f(\boldsymbol{y}^{(t-1)}) - \eta_t\left(\frac{1}{2} - L\eta_t\right)\left\|\nabla f(\boldsymbol{x}^{(t-1)})\right\|^2$$
$$+ \eta_t(1 + 2L\eta_t)\left\|\bar{\boldsymbol{m}}^{(t)} - \nabla f(\boldsymbol{x}^{(t-1)})\right\|^2$$
$$+ \left(\frac{1}{\eta_t} + 2L + \frac{\eta_t L^2}{2}\right)\left\|\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}\right\|^2.$$

Since $L\eta_t \leq 1/8$,

$$\frac{1}{2} - L\eta_t \geq \frac{3}{8}, \qquad 1 + 2L\eta_t \leq \frac{5}{4}, \qquad 1 + 2L\eta_t + \frac{L^2\eta_t^2}{2} \leq \frac{161}{128} \leq \frac{3}{2}.$$

Consequently,

$$f(\boldsymbol{y}^{(t)}) \leq f(\boldsymbol{y}^{(t-1)}) - \frac{3\eta_t}{8}\left\|\nabla f(\boldsymbol{x}^{(t-1)})\right\|^2 + \frac{5\eta_t}{4}\left\|\bar{\boldsymbol{m}}^{(t)} - \nabla f(\boldsymbol{x}^{(t-1)})\right\|^2 + \frac{3}{2\eta_t}\left\|\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)}\right\|^2.$$

Taking conditional expectations and for $t \geq 2$, the real and virtual updates imply

$$\boldsymbol{x}^{(t-1)} - \boldsymbol{y}^{(t-1)} = \left(\boldsymbol{x}^{(t-2)} - \eta_{t-1}\boldsymbol{m}^{(t-1)}\right) - \left(\boldsymbol{x}^{(t-2)} - \eta_{t-1}\bar{\boldsymbol{m}}^{(t-1)}\right)$$
$$= -\eta_{t-1}\left(\boldsymbol{m}^{(t-1)} - \bar{\boldsymbol{m}}^{(t-1)}\right).$$

Substitution proves the Lemma. □

Combining this lemma with Karimireddy et al. (2021, Lemmas 9 and 11) yields the same convergence rate up to a constant factor. This construction also provides key intuition for THROTTLE and its analysis. It corresponds to the special case of THROTTLE in which every round is a synchronization round ($q \to \infty$).

C.2 PROOF OF THE MAIN THEOREM

Let us restate the main theorem in its precise form.

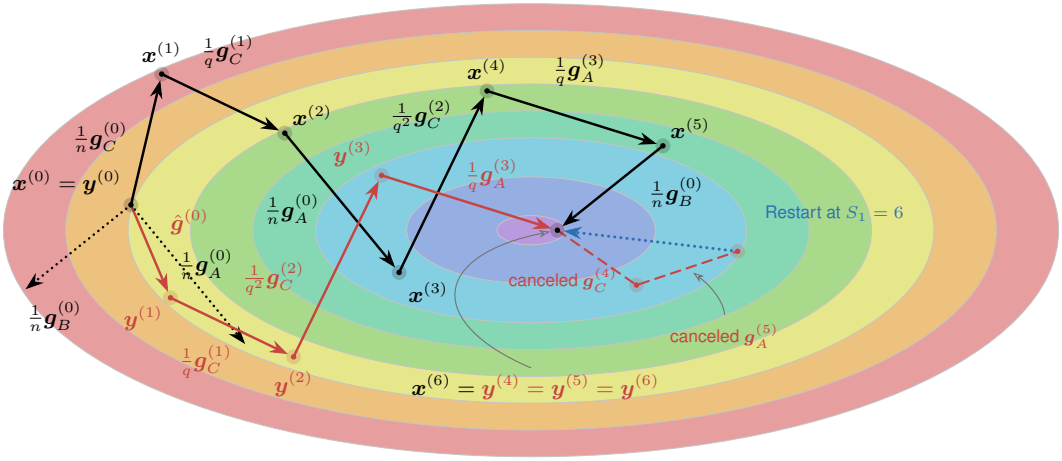


Figure 4: Actual and virtual iterates with a hard restart.

Theorem 1 (Non-convex Convergence rate)**.** Let $T \geq 2$ and $q > 1$. Suppose that Assumptions 2, 1, and 3, as well as Assumption 4 with $G' \leq 3G$ hold, together with the local (c, δ)-robust aggregator $\mathbb{A}$, and use $\lambda_t = G$. Define initial distance Δ, effective time steps T_{eff}, number of restart rounds N_T, and restart ratio θ_T as

$$\Delta \coloneqq f(\boldsymbol{x}^{(0)}) - f_*, \quad T_{\text{eff}} =: 1 + \sum_{t=1}^{T-1} \hat{\eta}_t/\eta, \quad N_T \coloneqq 1 + \sum_{t=1}^{T-1} \mathbf{1}_{\{t=r(t)\}}, \quad \theta_T \coloneqq \frac{N_T}{T_{\text{eff}}}.$$

Suppose that $\Delta > 0$, and choose the output iterate according to

$$\mathbb{P}(\boldsymbol{x}^{\text{out}} = \boldsymbol{x}^{(0)}) = \frac{1}{T_{\text{eff}}}, \quad \mathbb{P}(\boldsymbol{x}^{\text{out}} = \boldsymbol{x}^{(t)}) = \frac{\hat{\eta}_t}{\eta T_{\text{eff}}}, \qquad t \in \{1, \ldots, T-1\}. \tag{4}$$

Choose the stepsize $\eta > 0$ as

$$\eta = \min\left\{ \sqrt{\frac{\Delta}{L\sigma^2(1-\theta_T+\theta_T/|\mathcal{G}|)T_{\text{eff}}}}, \quad \left(\frac{\Delta}{L^2 T_{\text{eff}}(1-\theta_T)\left[9(n-1)^2G^2 + 4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right]} \right)^{1/3}, \frac{1}{4L} \right\}.$$

We have

$$\begin{aligned} \mathbb{E}\|\nabla f(\boldsymbol{x}^{\text{out}})\|^2 = \mathcal{O}\Bigg(& \sqrt{\frac{L\sigma^2\Delta}{T_{\text{eff}}}\left(1-\theta_T+\frac{\theta_T}{|\mathcal{G}|}\right)} \\ & + \left(\frac{L\Delta}{T_{\text{eff}}}\right)^{2/3}\left[(1-\theta_T)\left(9(n-1)^2G^2 + 4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right)\right]^{1/3} \\ & + \frac{L\Delta}{T_{\text{eff}}} + \left(4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right)\theta_T \Bigg). \end{aligned}$$

Let the real iterate be

$$\boldsymbol{x}^{(t)} = \boldsymbol{x}^{(t-1)} - \eta_t \boldsymbol{g}_{j_t}^{(t-\tau_t)}, \quad \text{where } t - \tau_t \geq r(t),$$

with the stepsize η_t in equation 2. First arrivals use the local aggregator with side information from the last synchronized round, and repeated arrivals use clipping at radius G.

Define the first virtual iterate (de-asynchronized version) as

$$\boldsymbol{y}^{(t+1)} = \begin{cases} \boldsymbol{y}^{(t)} - \hat{\eta}_t \boldsymbol{g}_{j_t}^{(t)}, & \text{if } t > r(t), \\ \boldsymbol{x}^{(t)} - \frac{\eta}{n}\sum_{i=1}^n \boldsymbol{g}_i^{(t)}, & \text{if } t = r(t), \end{cases}, \quad \text{and} \quad \boldsymbol{y}^{(1)} = \boldsymbol{x}^{(0)} - \frac{\eta}{n}\sum_{i=1}^n \boldsymbol{g}_i^{(0)}.$$

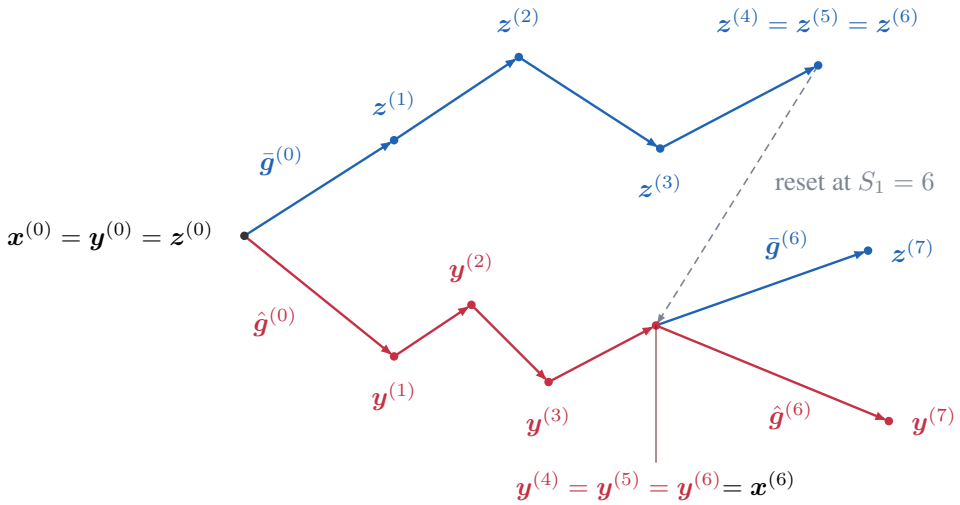


Figure 5: Virtual iterates with clipping and a hard restart.

where

$$\hat{\eta}_t := \begin{cases} \eta, & \text{if } t = r(t), \\ \eta_{\text{next}(t+1, j_t)}, & \text{if } t > r(t) \text{ and the gradient computation is not canceled by the server,} \\ 0 & t > r(t) \text{ and the gradient computation is canceled by the server.} \end{cases}$$

Furthermore, we define another virtual iterate to remove the effect of clipping and Byzantine updates:

$$z^{(t+1)} = \begin{cases} z^{(t)} - \hat{\eta}_t \nabla F(x^{(t)}; \xi_{j_t}^{(t)}), & \text{if } t > r(t), \\ y^{(t)} - \eta \cdot \frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} \nabla F(x^{(t)}; \xi_i^{(t)}), & \text{if } t = r(t), \end{cases}$$

and

$$z^{(1)} = x^{(0)} - \eta \cdot \frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} \nabla F(x^{(0)}; \xi_i^{(0)}).$$

If the job is canceled (i.e., $\hat{\eta}_t = 0$), define $y^{(t+1)} = y^{(t)}$ and $z^{(t+1)} = z^{(t)}$. By construction, we have at every synchronization time S_r, $y^{(S_r)} = x^{(S_r)}$.

The definition of $z^{(1)}$ gives one additional stochastic descent step before the virtual-iterate recursion starts.

Lemma 2 (Initial stochastic descent step)**.** *If $\eta \le 1/L$, then*

$$\mathbb{E}[f(z^{(1)}) \mid x^{(0)}] \le f(x^{(0)}) - \frac{\eta}{2} \|\nabla f(x^{(0)})\|^2 + \frac{L\eta^2\sigma^2}{2|\mathcal{G}|}.$$

Proof. Counting the initial round as restart-like, let

$$\bar{g}^{(0)} := \frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} \nabla F(x^{(0)}; \xi_i^{(0)}).$$

Conditional independence, unbiasedness, and the variance bound give

$$\mathbb{E}[\bar{g}^{(0)} \mid x^{(0)}] = \nabla f(x^{(0)}),$$
$$\mathbb{E}[\|\bar{g}^{(0)} - \nabla f(x^{(0)})\|^2 \mid x^{(0)}] \le \frac{\sigma^2}{|\mathcal{G}|}.$$

Therefore, L-smoothness and $\boldsymbol{z}^{(1)} = \boldsymbol{x}^{(0)} - \eta\overline{\boldsymbol{g}}^{(0)}$ imply

$$\begin{aligned}\mathbb{E}[f(\boldsymbol{z}^{(1)}) \mid \boldsymbol{x}^{(0)}] &\le f(\boldsymbol{x}^{(0)}) - \eta\|\nabla f(\boldsymbol{x}^{(0)})\|^2 \\ &\quad + \frac{L\eta^2}{2}\left(\|\nabla f(\boldsymbol{x}^{(0)})\|^2 + \frac{\sigma^2}{|\mathcal{G}|}\right) \\ &\le f(\boldsymbol{x}^{(0)}) - \frac{\eta}{2}\|\nabla f(\boldsymbol{x}^{(0)})\|^2 + \frac{L\eta^2\sigma^2}{2|\mathcal{G}|},\end{aligned}$$

where the last inequality uses $L\eta \le 1$. □

At a synchronization time $t = r(t)$, the aggregate used by the first virtual iterate satisfies

$$\frac{1}{n}\sum_{i=1}^{n}\mathbb{A}(\boldsymbol{v}_i^{(t)}; S) = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{g}_i^{(t)}.$$

We use a local (c, δ)-robust aggregator in the sense of the main text.

Lemma 3 (Synchronization aggregation error)**.** *Under Assumption 1, at every synchronization time t,*

$$\mathbb{E}\left[\left\|\frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}}\nabla F(\boldsymbol{x}^{(t)}; \xi_i^{(t)}) - \frac{1}{n}\sum_{i=1}^{n}\mathbb{A}(\boldsymbol{v}_i^{(t)}; \mathbf{S}_{r(t)})\right\|^2 \mid \mathcal{F}_t\right] \le 2c\delta\sigma^2.$$

Proof. For any $i, j \in \mathcal{G}$, conditional independence and unbiasedness give

$$\begin{aligned}\mathbb{E}[\|\nabla F(\boldsymbol{x}^{(t)}; \xi_i^{(t)}) - \nabla F(\boldsymbol{x}^{(t)}; \xi_j^{(t)})\|^2 \mid \mathcal{F}_t] &= \mathbb{E}[\|\nabla F(\boldsymbol{x}^{(t)}; \xi_i^{(t)}) - \nabla f(\boldsymbol{x}^{(t)})\|^2 \mid \mathcal{F}_t] \\ &\quad + \mathbb{E}[\|\nabla F(\boldsymbol{x}^{(t)}; \xi_j^{(t)}) - \nabla f(\boldsymbol{x}^{(t)})\|^2 \mid \mathcal{F}_t] \\ &\le 2\sigma^2.\end{aligned}$$

Thus the pairwise radius ρ^2 in Definition 1 is at most $2\sigma^2$. Applying the local (c, δ)-robust aggregator definition 1 conditionally on $\mathcal{F}_t$ proves the claim. □

Lemma 4 (Stochastic virtual-iterate relationship)**.** *For every t after the first restart,*

$$\begin{aligned}\boldsymbol{y}^{(t+1)} - \boldsymbol{z}^{(t+1)} &= \eta\left[\frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}}\nabla F(\boldsymbol{x}^{(r(t))}; \xi_i^{(r(t))}) - \frac{1}{n}\sum_{i=1}^{n}\mathbb{A}(\boldsymbol{v}_i^{(r(t))}; \mathbf{S}_{r(t)})\right] \\ &\quad + \sum_{k=r(t)+1}^{t}\hat{\eta}_k\left[\nabla F(\boldsymbol{x}^{(k)}; \xi_{j_k}^{(k)}) - \boldsymbol{g}_{j_k}^{(k)}\right].\end{aligned}$$

The second sum is zero when $t = r(t)$. Moreover,

$$\mathbb{E}\|\boldsymbol{y}^{(t+1)} - \boldsymbol{z}^{(t+1)}\|^2 \le \eta^2\left(4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right).$$

Proof. Let $r = r(t)$. At the synchronization round,

$$\boldsymbol{y}^{(r+1)} - \boldsymbol{z}^{(r+1)} = \eta\left[\frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}}\nabla F(\boldsymbol{x}^{(r)}; \xi_i^{(r)}) - \frac{1}{n}\sum_{i=1}^{n}\mathbb{A}(\boldsymbol{v}_i^{(r)}; \mathbf{S}_{r(t)})\right].$$

Every subsequent round $k > r$ satisfies

$$\boldsymbol{y}^{(k+1)} - \boldsymbol{z}^{(k+1)} = \boldsymbol{y}^{(k)} - \boldsymbol{z}^{(k)} + \hat{\eta}_k\left[\nabla F(\boldsymbol{x}^{(k)}; \xi_{j_k}^{(k)}) - \boldsymbol{g}_{j_k}^{(k)}\right].$$

Unrolling this recursion proves the first display.

For every good client, clipping is inactive by Assumption 3, and hence $\nabla F(\boldsymbol{x}^{(k)};\xi_{j_k}^{(k)}) - \boldsymbol{g}_{j_k}^{(k)} = 0$. The sum below contains only repeated-arrival contributions, whose norms are at most G by non-restart clipping at radius G. Together with Assumption 3, this gives

$$\begin{aligned}\left\| \sum_{k=r(t)+1}^{t} \frac{\hat{\eta}_k}{\eta} \left[\nabla F(\boldsymbol{x}^{(k)};\xi_{j_k}^{(k)}) - \boldsymbol{g}_{j_k}^{(k)} \right] \right\| &\le 2G \sum_{k=r(t)+1}^{t} \frac{\hat{\eta}_k}{\eta} \mathbf{1}_{\{j_k \in \mathcal{B}\}} \\ &\le \frac{2G|\mathcal{B}|}{q-1}.\end{aligned}$$

The last inequality follows by summing, for each Byzantine client, the geometric sequence $q^{-1} + q^{-2} + \cdots$ within the round. Therefore, Lemma 3 gives

$$\mathbb{E}\|\boldsymbol{y}^{(t+1)} - \boldsymbol{z}^{(t+1)}\|^2 \le \eta^2 \left(4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2} \right).$$

This proves the result. □

Lemma 5 (Real–virtual iterate difference after synchronization)**.** *Suppose Assumption 4 holds, and let r be a synchronization round. At non-restart times $t > r(t)$, under the same hard-restart indexing as in the deterministic proof,*

$$\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)} = \sum_{m \in [n]\setminus\{j_t\}} \eta_{\mathrm{next}(t,m)} \boldsymbol{g}_m^{(\mathrm{prev}(t,m))}.$$

Consequently,

$$\|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\|^2 \le 9(n-1)^2\eta^2 G^2 \quad \textit{almost surely}.$$

At restart times, $\boldsymbol{x}^{(S_r)} = \boldsymbol{y}^{(S_r)}$, so the gap is zero.

Proof. For $t > r(t)$, denote $\boldsymbol{a}_t = \boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}$. Exactly as in the deterministic proof, unrolling the two recursions after $r(t)$ gives

$$\begin{aligned}\boldsymbol{a}_t &= \frac{\eta}{n} \sum_{i=1}^{n} \boldsymbol{g}_i^{(r(t))} - \sum_{k=r(t)+1}^{t} \eta_k \boldsymbol{g}_{j_k}^{(k-\tau_k)} + \sum_{k=r(t)+1}^{t-1} \hat{\eta}_k \boldsymbol{g}_{j_k}^{(k)} \\ &= \sum_{m \in [n]\setminus\{j_t\}} \eta_{\mathrm{next}(t,m)} \boldsymbol{g}_m^{(\mathrm{prev}(t,m))}.\end{aligned}$$

Because $k - \tau_k \ge r(k)$, all but at most one pending vector per client cancel. Canceled computations have weight zero, so at most $n-1$ terms remain. First-arrival contributions have norm at most $3G$ by Assumption 4, and repeated-arrival contributions have norm at most $G \le 3G$ by clipping at radius G. Using $\eta_{\mathrm{next}(t,m)} \le \eta$ therefore gives

$$\begin{aligned}\|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\|^2 &\le (n-1) \sum_{m \in [n]\setminus\{j_t\}} \eta_{\mathrm{next}(t,m)}^2 \|\boldsymbol{g}_m^{(\mathrm{prev}(t,m))}\|^2 \\ &\le 9(n-1)^2\eta^2 G^2.\end{aligned}$$

At restart times the gap is zero by construction. This bound is pathwise and therefore also holds after taking expectation. □

Lemma 6 (Descent Lemma for Non-Restart Rounds)**.** *If $t > r(t)$ and $\hat{\eta}_t \le 1/(2L)$, then*

$$\begin{aligned}\mathbb{E}[f(\boldsymbol{z}^{(t+1)}) \mid \mathcal{F}_t] \le f(\boldsymbol{z}^{(t)}) &- \frac{\hat{\eta}_t}{4}\|\nabla f(\boldsymbol{x}^{(t)})\|^2 - \frac{\hat{\eta}_t}{2}\|\nabla f(\boldsymbol{z}^{(t)})\|^2 \\ &+ L^2\hat{\eta}_t\|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\|^2 + L^2\hat{\eta}_t\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2 \\ &+ \frac{L\sigma^2\hat{\eta}_t^2}{2}.\end{aligned}$$

Proof. By L-smoothness and conditional unbiasedness,

$$\mathbb{E}[f(\boldsymbol{z}^{(t+1)}) \mid \mathcal{F}_t] \le f(\boldsymbol{z}^{(t)}) - \hat{\eta}_t \langle \nabla f(\boldsymbol{z}^{(t)}), \nabla f(\boldsymbol{x}^{(t)}) \rangle + \frac{L\hat{\eta}_t^2}{2} \left(\|\nabla f(\boldsymbol{x}^{(t)})\|^2 + \sigma^2 \right).$$

Using

$$-\langle \boldsymbol{a}, \boldsymbol{b} \rangle = -\frac{1}{2}\|\boldsymbol{a}\|^2 - \frac{1}{2}\|\boldsymbol{b}\|^2 + \frac{1}{2}\|\boldsymbol{a} - \boldsymbol{b}\|^2,$$

we obtain

$$\mathbb{E}[f(\boldsymbol{z}^{(t+1)}) \mid \mathcal{F}_t] \le f(\boldsymbol{z}^{(t)}) - \left(\frac{\hat{\eta}_t}{2} - \frac{L\hat{\eta}_t^2}{2} \right) \|\nabla f(\boldsymbol{x}^{(t)})\|^2 - \frac{\hat{\eta}_t}{2}\|\nabla f(\boldsymbol{z}^{(t)})\|^2 + \frac{\hat{\eta}_t}{2}\|\nabla f(\boldsymbol{z}^{(t)}) - \nabla f(\boldsymbol{x}^{(t)})\|^2 + \frac{L\sigma^2\hat{\eta}_t^2}{2}.$$

Finally,

$$\frac{\hat{\eta}_t}{2}\|\nabla f(\boldsymbol{z}^{(t)}) - \nabla f(\boldsymbol{x}^{(t)})\|^2 \le L^2\hat{\eta}_t\|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\|^2 + L^2\hat{\eta}_t\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2,$$

and $\hat{\eta}_t \le 1/(2L)$ implies $\hat{\eta}_t/2 - L\hat{\eta}_t^2/2 \ge \hat{\eta}_t/4$. □

Lemma 7 (Descent Lemma for Restart Rounds)**.** *If $t = r(t)$ and $4L\eta \le 1$, then $\hat{\eta}_t = \eta$ and*

$$\mathbb{E}[f(\boldsymbol{z}^{(t+1)}) \mid \mathcal{F}_t] \le f(\boldsymbol{z}^{(t)}) - \frac{\eta}{4}\|\nabla f(\boldsymbol{x}^{(t)})\|^2 + L^2\eta\|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\|^2 + L^2\eta\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2 + \frac{1}{\eta}\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2 + \frac{L\eta^2\sigma^2}{2|\mathcal{G}|}.$$

Proof. Conditional independence gives

$$\mathbb{E}\left[\frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} \nabla F(\boldsymbol{x}^{(t)}; \xi_i^{(t)}) \;\middle|\; \mathcal{F}_t \right] = \nabla f(\boldsymbol{x}^{(t)}),$$

$$\mathbb{E}\left[\left\| \frac{1}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} \nabla F(\boldsymbol{x}^{(t)}; \xi_i^{(t)}) - \nabla f(\boldsymbol{x}^{(t)}) \right\|^2 \mid \mathcal{F}_t \right] \le \frac{\sigma^2}{|\mathcal{G}|}.$$

The restart update is

$$\boldsymbol{z}^{(t+1)} = \boldsymbol{z}^{(t)} + \boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)} - \frac{\eta}{|\mathcal{G}|} \sum_{i \in \mathcal{G}} \nabla F(\boldsymbol{x}^{(t)}; \xi_i^{(t)}).$$

By L-smoothness and conditional unbiasedness,

$$\mathbb{E}[f(\boldsymbol{z}^{(t+1)}) \mid \mathcal{F}_t] \le f(\boldsymbol{z}^{(t)}) - \eta\langle \nabla f(\boldsymbol{z}^{(t)}), \nabla f(\boldsymbol{x}^{(t)}) \rangle + \left\langle \nabla f(\boldsymbol{z}^{(t)}), \boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)} \right\rangle + L\eta^2\|\nabla f(\boldsymbol{x}^{(t)})\|^2 + L\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2 + \frac{L\eta^2\sigma^2}{2|\mathcal{G}|}.$$

The same inner-product identity as above and Young's inequality give

$$\left\langle \nabla f(\boldsymbol{z}^{(t)}), \boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)} \right\rangle \le \frac{\eta}{2}\|\nabla f(\boldsymbol{z}^{(t)})\|^2 + \frac{1}{2\eta}\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2.$$

Moreover,

$$\frac{\eta}{2}\|\nabla f(\boldsymbol{z}^{(t)}) - \nabla f(\boldsymbol{x}^{(t)})\|^2 \le L^2\eta\|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\|^2 + L^2\eta\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2.$$

Combining these estimates gives the coefficient

$$-\eta\left(\frac{1}{2} - L\eta\right)$$

in front of $\|\nabla f(\boldsymbol{x}^{(t)})\|^2$. The total coefficient in front of $\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2$ is

$$L^2\eta + L + \frac{1}{2\eta}.$$

Since $4L\eta \le 1$,

$$\frac{1}{2} - L\eta \ge \frac{1}{4}, \qquad L + \frac{1}{2\eta} \le \frac{3}{4\eta} \le \frac{1}{\eta},$$

which proves the result. □

Theorem 2 (Combined Stochastic Descent Lemma)**.** *If $\eta \le 1/(4L)$, then for all t,*

$$\begin{aligned}\mathbb{E}[f(\boldsymbol{z}^{(t+1)}) \mid \mathcal{F}_t] \le f(\boldsymbol{z}^{(t)}) &- \frac{\hat{\eta}_t}{4}\|\nabla f(\boldsymbol{x}^{(t)})\|^2 \\ &+ L^2\hat{\eta}_t\|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\|^2 + L^2\hat{\eta}_t\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2 \\ &+ \frac{L\hat{\eta}_t^2\sigma^2}{2}\left(\mathbf{1}_{\{t>r(t)\}} + \frac{\mathbf{1}_{\{t=r(t)\}}}{|\mathcal{G}|}\right) + \frac{1}{\eta}\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2\,\mathbf{1}_{\{t=r(t)\}}.\end{aligned}$$

Proof. If $t > r(t)$, apply Lemma 6 and drop the nonpositive term

$$-\frac{\hat{\eta}_t}{2}\|\nabla f(\boldsymbol{z}^{(t)})\|^2.$$

If $t = r(t)$, apply Lemma 7. These two cases prove the stated inequality. □

Lemma 8 (Summed restart perturbation bound)**.** *Let*

$$N_T \coloneqq 1 + \sum_{t=1}^{T-1}\mathbf{1}_{\{t=r(t)\}}.$$

Then

$$\begin{aligned}\frac{1}{\eta}\sum_{t=1}^{T-1}\mathbb{E}\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2\,\mathbf{1}_{\{t=r(t)\}} &\le \eta\left(4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right)(N_T - 1) \\ &\le \eta\left(4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right)N_T.\end{aligned}$$

Proof. Fix a restart time t. Applying Lemma 4 with index $t-1$ gives

$$\mathbb{E}\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2 \le \eta^2\left(4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right).$$

Dividing by η and summing over the restart times proves the claim. □

Proof of Theorem 1. Apply Lemma 2, weaken its descent coefficient from $1/2$ to $1/4$, and then sum Theorem 2 over $t = 1, \dots, T-1$. Taking total expectation, telescoping from $f(\boldsymbol{x}^{(0)})$, and using

$f(\boldsymbol{z}^{(T)}) \geq f_*$ gives

$$\begin{aligned}\frac{1}{4}\left(\eta\|\nabla f(\boldsymbol{x}^{(0)})\|^2 + \sum_{t=1}^{T-1}\hat{\eta}_t\mathbb{E}\|\nabla f(\boldsymbol{x}^{(t)})\|^2\right) \leq \Delta &+ L^2\sum_{t=1}^{T-1}\hat{\eta}_t\mathbb{E}\|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\|^2 \\ &+ L^2\sum_{t=1}^{T-1}\hat{\eta}_t\mathbb{E}\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2 \\ &+ \frac{L\eta^2\sigma^2}{2|\mathcal{G}|} + \frac{L\sigma^2}{2}\sum_{\substack{t=1\\ t>r(t)}}^{T-1}\hat{\eta}_t^2 \\ &+ \frac{L\sigma^2}{2|\mathcal{G}|}\sum_{\substack{t=1\\ t=r(t)}}^{T-1}\hat{\eta}_t^2 + 4c\delta\sigma^2\eta(N_T - 1) \\ &+ \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\eta(N_T - 1).\end{aligned}$$

Lemma 5, Lemma 4, and the definition of T_{eff} imply

$$\sum_{\substack{t=1\\ t>r(t)}}^{T-1}\hat{\eta}_t = \eta(T_{\text{eff}} - N_T),$$

and, since the real–virtual gap is zero at restart times,

$$\begin{aligned}L^2\sum_{t=1}^{T-1}\hat{\eta}_t\mathbb{E}\|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\|^2 &= L^2\sum_{\substack{t=1\\ t>r(t)}}^{T-1}\hat{\eta}_t\mathbb{E}\|\boldsymbol{x}^{(t)} - \boldsymbol{y}^{(t)}\|^2 \\ &\leq 9L^2(n-1)^2G^2\eta^3(T_{\text{eff}} - N_T) \\ &= 9L^2(n-1)^2G^2\eta^3T_{\text{eff}}(1-\theta_T),\end{aligned}$$

$$L^2\sum_{\substack{t=1\\ t>r(t)}}^{T-1}\hat{\eta}_t\mathbb{E}\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2 \leq L^2\eta^3(T_{\text{eff}} - N_T)\left(4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right).$$

At restart times, the remaining part of the same sum satisfies

$$\begin{aligned}&L^2\sum_{t=1}^{T-1}\hat{\eta}_t\mathbb{E}\|\boldsymbol{y}^{(t)} - \boldsymbol{z}^{(t)}\|^2\,\mathbf{1}_{\{t=r(t)\}} \\ &\leq L^2\eta^3\left(4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right)(N_T - 1).\end{aligned}$$

Since $L^2\eta^2 \leq 1/16$, combining this restart contribution with Lemma 8 gives at most

$$\frac{17}{16}\eta\left(4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right)(N_T - 1) \leq \frac{17}{16}\eta\left(4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right)N_T.$$

Also, $\hat{\eta}_t = \eta$ at restart times, while $\hat{\eta}_t^2 \le \eta\hat{\eta}_t$ otherwise. Including the initial step, the variance weights satisfy

$$\begin{aligned}
&\frac{\eta^2}{|\mathcal{G}|} + \sum_{\substack{t=1\\ t>r(t)}}^{T-1} \hat{\eta}_t^2 + \frac{1}{|\mathcal{G}|}\sum_{\substack{t=1\\ t=r(t)}}^{T-1} \hat{\eta}_t^2 \\
&\quad= \frac{\eta^2 N_T}{|\mathcal{G}|} + \sum_{\substack{t=1\\ t>r(t)}}^{T-1} \hat{\eta}_t^2 \\
&\quad\le \eta^2\left(\frac{N_T}{|\mathcal{G}|} + T_{\mathrm{eff}} - N_T\right) \\
&\quad= \eta^2 T_{\mathrm{eff}}\left(1 - \theta_T + \frac{\theta_T}{|\mathcal{G}|}\right).
\end{aligned}$$

Collecting these estimates yields

$$\begin{aligned}
\frac{1}{4}\left(\eta\|\nabla f(\boldsymbol{x}^{(0)})\|^2 + \sum_{t=1}^{T-1}\hat{\eta}_t\mathbb{E}\|\nabla f(\boldsymbol{x}^{(t)})\|^2\right) \le \Delta &+ \frac{L\sigma^2\eta^2 T_{\mathrm{eff}}}{2}\left(1 - \theta_T + \frac{\theta_T}{|\mathcal{G}|}\right) \\
&+ 9L^2\eta^3 T_{\mathrm{eff}}(1-\theta_T)(n-1)^2G^2 \\
&+ 4L^2\eta^3 T_{\mathrm{eff}}(1-\theta_T)c\delta\sigma^2 \\
&+ 8L^2\eta^3 T_{\mathrm{eff}}(1-\theta_T)\frac{G^2|\mathcal{B}|^2}{(q-1)^2} \\
&+ \frac{17}{4}c\delta\sigma^2\eta N_T \\
&+ \frac{17}{2}\eta\frac{G^2|\mathcal{B}|^2}{(q-1)^2}N_T.
\end{aligned}$$

Dividing by $\eta T_{\mathrm{eff}}/4$ and using the output distribution gives

$$\begin{aligned}
\mathbb{E}\|\nabla f(\boldsymbol{x}^{\mathrm{out}})\|^2 \le \frac{4\Delta}{\eta T_{\mathrm{eff}}} &+ 2L\sigma^2\eta\left(1 - \theta_T + \frac{\theta_T}{|\mathcal{G}|}\right) \\
&+ 4L^2\eta^2(1-\theta_T)\left[9(n-1)^2G^2 + 4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right] \\
&+ \frac{17}{4}\left(4c\delta\sigma^2 + \frac{8G^2|\mathcal{B}|^2}{(q-1)^2}\right)\theta_T.
\end{aligned}$$

Using the stepsize tuning lemma (Lemma 17 (Koloskova et al., 2020)), we obtain the stated convergence rate. □

D Local Robust Aggregators

This appendix collects the material on local (c, δ)-robust aggregators (Definition 2) that is used in Sections 4: a general construction with examples (Appendix D.1) and the impossibility of decomposition without side information (Appendix D.2). Assumption 4 for centered clipping is verified in Remark 2.

D.1 A General Construction

Every local aggregator we use is of the following form.

Lemma 9 (Centered aggregators are decomposable)**.** *Let $\psi : \mathbb{R}^d \to \mathbb{R}^d$ be any map and take the side information to be an anchor, $S = \boldsymbol{a} \in \mathbb{R}^d$. Then*

$$\mathbb{A}(\boldsymbol{v}; \boldsymbol{a}) := \boldsymbol{a} + \psi(\boldsymbol{v} - \boldsymbol{a}), \qquad \mathrm{Agg}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n; \boldsymbol{a}) := \boldsymbol{a} + \frac{1}{n}\sum_{i=1}^n \psi(\boldsymbol{x}_i - \boldsymbol{a}),$$

satisfy $\mathrm{Agg}(\boldsymbol{x}_1, \dots, \boldsymbol{x}_n; \boldsymbol{a}) = \frac{1}{n}\sum_{i=1}^n \mathbb{A}(\boldsymbol{x}_i; \boldsymbol{a})$.

Proof. The identity is immediate from $\frac{1}{n}\sum_i(\boldsymbol{a}+\psi(\boldsymbol{x}_i-\boldsymbol{a}))=\boldsymbol{a}+\frac{1}{n}\sum_i\psi(\boldsymbol{x}_i-\boldsymbol{a})$. □

The role of ψ is to leave honest inputs (essentially) unchanged, $\psi(\boldsymbol{u})\approx\boldsymbol{u}$ when $\|\boldsymbol{u}\|$ is of the order of the honest inputs' magnitude, while bounding the effect of arbitrary inputs. Two natural choices are

$$\text{Centered Clipping (Karimireddy et al., 2021):}\quad \psi_{\mathrm{cc}}(\boldsymbol{u}):=\boldsymbol{u}\min\Big\{1,\frac{\tau}{\|\boldsymbol{u}\|}\Big\},$$

$$\text{Centered Truncated Mean (CTM, new)}\quad \psi_{\mathrm{ctm}}(\boldsymbol{u}):=\boldsymbol{u}\,\mathbb{1}_{\{\|\boldsymbol{u}\|\le R\}}.$$

Centered clipping with $S=(\boldsymbol{a},\tau)$ is Example 1. Centered clipping is (c,δ)-robust when the anchor is close to the honest mean and the radius is chosen accordingly (Karimireddy et al., 2021, Theorem IV). Here, we show that CTM is also a local (c,δ)-robust aggregator.

Proposition 1 (Centered Truncated Mean is a (c,δ)-robust aggregator)**.** *Let* $\mathrm{CTM}_{a,R}$ *be the aggregator in the preceding theorem with* ψ_{ctm}*. Let* $n\ge 2$ *and* $\frac{1}{n}\le\delta<\frac{1}{2}$*. Let* $\mathcal{G}\subseteq[n]$ *denote the set of honest inputs and* $\mathcal{B}=[n]\setminus\mathcal{G}$ *the set of Byzantine inputs, with* $\beta:=\frac{|\mathcal{B}|}{n}\le\delta$*. Fix the side information* $(\boldsymbol{a},R)$ *before observing the current inputs. Suppose that, conditional on this side information, the honest inputs are independent and satisfy*

$$\mathbb{E}\|\boldsymbol{x}_i-\boldsymbol{a}\|^2\le V^2,\quad\forall i\in\mathcal{G},$$

for some $V\ge 0$*. Choosing* $R^2=\frac{V^2}{\delta}$ *yields that*

$$\mathbb{E}\|\mathrm{CTM}_{a,R}(\boldsymbol{x}_1,\boldsymbol{x}_2,\dots,\boldsymbol{x}_n)-\bar{x}\|^2\le 10\delta V^2.$$

Consequently, if the admissible side information pair satisfies $V^2\le K\rho^2$ *for some constant* K*, then Centered Truncated Mean satisfies the definition of* (c,δ)*-robust aggregator with* $c=10K$*.*

Proof. Write

$$\psi_R(\boldsymbol{u})=\boldsymbol{u}\mathbb{1}_{\{\|\boldsymbol{u}\|\le R\}},\quad Z_i=(\boldsymbol{x}_i-\boldsymbol{a})\mathbb{1}_{\{\|\boldsymbol{x}_i-\boldsymbol{a}\|>R\}},\quad i\in\mathcal{G}.$$

By assumption, $\mathbb{E}\|Z_i\|^2\le V^2$. Also, by Markov's inequality,

$$\begin{aligned}\|\mathbb{E}Z_i\|&\le\mathbb{E}\left[\|\boldsymbol{x}_i-\boldsymbol{a}\|\mathbb{1}_{\{\|\boldsymbol{x}_i-\boldsymbol{a}\|>R\}}\right]\\&\le\frac{\mathbb{E}\|x_i-a\|^2}{R}\le\frac{V^2}{R}.\end{aligned}$$

Next, the aggregation error admits the decomposition

$$\mathrm{CTM}_{a,R}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_n)-\bar{x}=-\frac{1}{n}\sum_{i\in\mathcal{G}}Z_i+\frac{1}{n}\sum_{i\in\mathcal{B}}\psi_R(\boldsymbol{x}_i-\boldsymbol{a})-\beta(\bar{\boldsymbol{x}}-\boldsymbol{a}).$$

Because $\|\psi_R(u)\|\le R$,

$$\left\|\frac{1}{n}\sum_{i\in\mathcal{B}}\psi_R(\boldsymbol{x}_i-\boldsymbol{a})\right\|^2\le\beta R.$$

Furthermore, Jensen's inequality gives

$$\mathbb{E}\|\bar{\boldsymbol{x}}-\boldsymbol{a}\|^2\le\frac{1}{|\mathcal{G}|}\sum_{i\in\mathcal{G}}\mathbb{E}\|\boldsymbol{x}_i-\boldsymbol{a}\|^2\le V^2.$$

Therefore,

$$\mathbb{E}\left\|\frac{1}{n}\sum_{i\in\mathcal{B}}\psi_R(\boldsymbol{x}_i-\boldsymbol{a})-\beta(\bar{\boldsymbol{x}}-\boldsymbol{a})\right\|^2\le 2\beta^2(R^2+V^2).$$

Combining above result yields

$$\begin{aligned}\mathbb{E}\|\mathrm{CTM}_{a,R}(\boldsymbol{x}_1,\dots,\boldsymbol{x}_n)-\bar{\boldsymbol{x}}\|^2&\le\frac{2V^2}{n}+\frac{2V^4}{R^2}+4\delta^2(R^2+V^2)\\&\le\left(\frac{2}{n}+6\delta+4\delta^2\right)V^2\\&\le 10\delta V^2\end{aligned}$$

by $R^2=V^2/\delta$ and $\frac{1}{n}\le\delta<1/2$. The case $V=0$ is immediate with $R=0$. □

D.2 Impossibility Without Side Information

Definition 2 allows the local map $\mathbb{A}$ to depend on a side information S. The following proposition shows that this is not a technical convenience: no per-input map of the input alone can be robust.

Proposition 2 (Failure without side information)**.** *Let $n \geq 3$, $d \geq 1$, and $1/n \leq \delta < 1/2$. For any finite $c \geq 0$, there is no map $f : \mathbb{R}^d \to \mathbb{R}^d$ such that the aggregator*

$$\mathrm{Agg}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) := \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{x}_i)$$

is (c, δ)-robust in the sense of Definition 1.

Proof. Suppose such a map f exists. Let $\mathcal{G} = [n-1]$ and $\mathcal{B} = \{n\}$. This split is admissible because $|\mathcal{G}| = n - 1 \geq (1 - \delta)n$.

Fix arbitrary $\boldsymbol{a}, \boldsymbol{y} \in \mathbb{R}^d$, and take the deterministic inputs

$$\boldsymbol{x}_1 = \cdots = \boldsymbol{x}_{n-1} = \boldsymbol{a}, \qquad \boldsymbol{x}_n = \boldsymbol{y}.$$

The honest inputs are independent, their mean is $\boldsymbol{a}$, and their pairwise dispersion is zero. Definition 1, applied with $\rho = 0$, therefore gives

$$\frac{n-1}{n} f(\boldsymbol{a}) + \frac{1}{n} f(\boldsymbol{y}) = \boldsymbol{a} \qquad \text{for all } \boldsymbol{a}, \boldsymbol{y} \in \mathbb{R}^d.$$

Taking $\boldsymbol{y} = \boldsymbol{a}$ shows that $f(\boldsymbol{a}) = \boldsymbol{a}$ for every $\boldsymbol{a} \in \mathbb{R}^d$. Substituting this identity back into the display yields

$$\frac{n-1}{n} \boldsymbol{a} + \frac{1}{n} \boldsymbol{y} = \boldsymbol{a} \qquad \text{for all } \boldsymbol{a}, \boldsymbol{y} \in \mathbb{R}^d,$$

which is impossible when $\boldsymbol{y} \neq \boldsymbol{a}$. □

E Additional Experimental Details

E.1 Common settings and implementation

Two codebases were used. The least-squares experiment of Figure 3 runs *real* asynchronous workers. Every client is a Ray actor, the parameter server applies gradients in the order in which they arrive, and delays are produced by CPU contention rather than by a model. The MNIST experiment of Figure 2 uses a single-process, event-driven simulator with simulated (virtual) time based on the (Dahan & Levy, 2024) codebase. The arrival sequence of every client is generated once per (schedule, seed) and is identical for all methods, communication latency is zero, and a client's next computation starts when its previous message arrives. Kardam and μ^2-SGD were ported from the authors' public github repositories (`gdamaskinos/kardam` and `dahan198/asynchronous-fault-tolerant-ml`). BASGD, BASGDm were implemented from the papers, since no public code exists. Each seed spawns independent random streams for the data order, the arrival schedule, the initialization and the attack. Tables 1 and 3 list the complete configurations.

E.2 Least-squares experiment (Figure 3)

Problem. We minimize $F(x) = \frac{1}{2n}\|Ax - b\|^2$ with $n = 10{,}000$ samples and $d = 400$. The entries of A are i.i.d. uniform on $[0, 1)$ scaled by $1/\sqrt{d}$. A stochastic gradient is the gradient of F on a minibatch of 256 samples drawn without replacement; all workers hold the same data (i.i.d., no partitioning). Reported losses are $F(x) - F^*$ evaluated on all n samples every 40 accepted gradients.

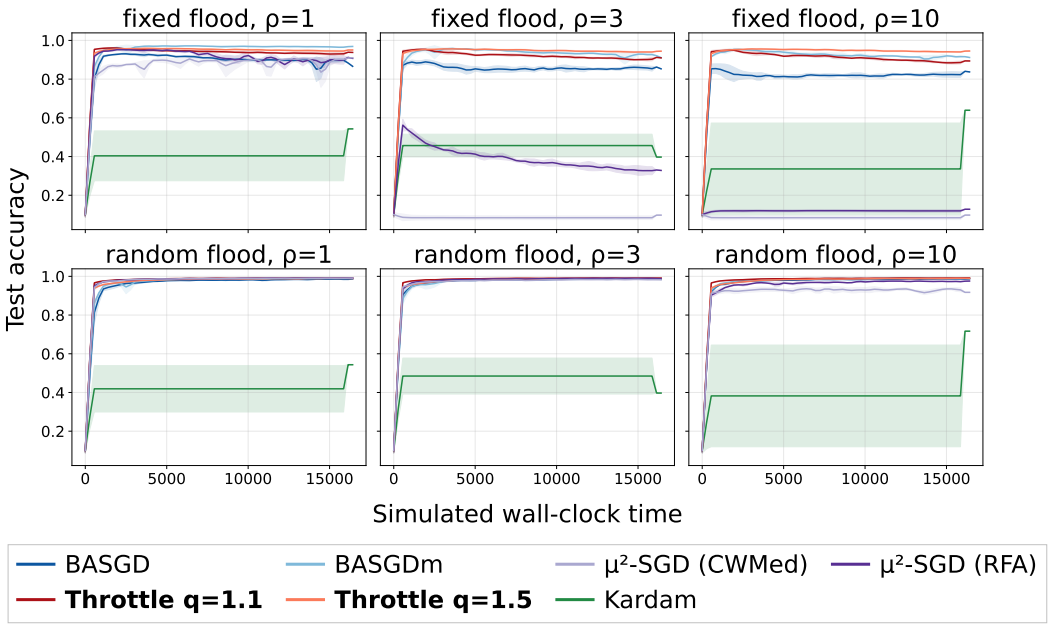


Figure 6: Additional MNIST flooding-attack results complementing Figure 2, with $n = 25$ clients, including $|\mathcal{B}| = 5$ Byzantine clients. The top and bottom rows show fixed-vector and random-vector flooding, respectively. Columns correspond to $\rho \in \{1, 3, 10\}$, extending the $\rho = 30$ results in Figure 2. Here, ρ multiplies the Byzantine clients' update rates, yielding an expected Byzantine update fraction of $\rho/(2+\rho)$. All other experimental settings are the same as in Figure 2. THROTTLE maintains high test accuracy across the three flooding rates.

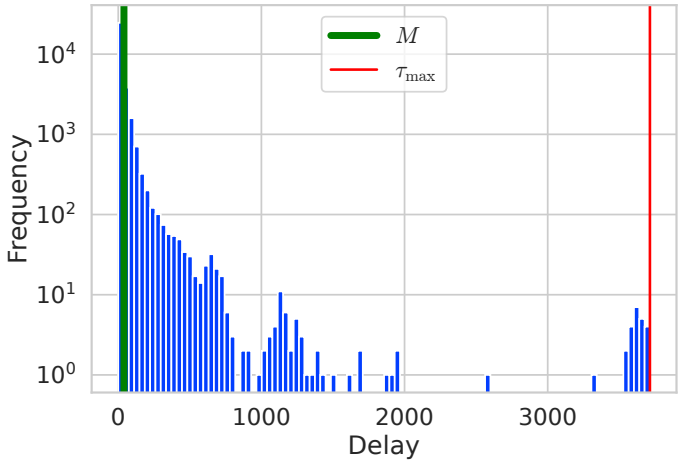


Figure 7: Empirical distribution of gradient delays in the least-squares experiment with $M = 40$ asynchronous workers on 20 CPU cores based on (Mishchenko et al., 2022). Delay is the number of server updates between a worker's dispatch and the arrival of its gradient. frequencies are shown on a logarithmic scale. The green and red vertical lines average delay M and the maximum observed delay $\tau_{\max}$, respectively, illustrating that delays can greatly exceed the number of workers.

Asynchronous execution. $M = 40$ workers run as Ray actors on a 20-core CPU machine (two workers per core), so arrival times are genuine wall-clock timings. The server applies each gradient on arrival and immediately re-dispatches the same worker with the updated model, i.e. there are always 40 computing gradients. Deley distribution (the number of server updates between a worker's

Table 1: Least-squares configuration used in Figure 3.

Objective	$F(x) = \frac{1}{2n}\|Ax - b\|^2$, $n = 10{,}000$, $d = 400$, condition number $\approx 1.9 \times 10^3$
Stochastic gradient	minibatch of 256 samples without replacement. all clients have shared i.i.d. data
Clients	$M = 40$ Ray actors on 20 CPU cores.
Straggler (right panel)	worker 39 sleeps $100\times$ its compute time after each gradient (1.7% of arrivals instead of 2.5%)
Budget	32,000 computed gradients for every method.
Methods	Minibatch SGD and asynchronous SGD are based on (Mishchenko et al., 2022), THROTTLE $q \in \{1.1, 1.5\}$, μ^2-SGD ($\beta = 0.25, \gamma = 0.1$), BASGD / BASGDm ($B = 10$, mean, $\mu = 0.9$)
THROTTLE	no clipping ($\lambda = \tau = \infty$, no Byzantine clients) hard restart when all M clients have arrived; discarded gradients included in the budget
Step-size tuning	grid search over $\eta = 2^{k/3}$, $k \in [-30, 18] \cap \mathbb{Z}$ (Table 2)
Seeds	$\{42, 43, 44\}$

Table 2: Learning rates used in Fig. 3. Grid $\eta = 2^{k/3}$ (displayed learning rates are rounded); criterion: mean of $F(x) - F^*$ over the checkpoints in the last 10% of the 32,000-computed gradient budget, averaged over seeds $\{42, 43, 44\}$; a grid point is valid only if all three repeats converge. 'tail mean' is that criterion at the selected η.

Method	η used	tail mean $\pm$ std
Minibatch SGD ($q = \infty$)	0.1984, $k = -7$	$3.46 \times 10^{-3} \pm 3.4 \times 10^{-5}$
Asynchronous SGD ($q = 1$)	0.5, $k = -3$	$1.37 \times 10^{-4} \pm 2.0 \times 10^{-5}$
Throttle ($q = 1.1$)	1.587, $k = 2$	$7.14 \times 10^{-6} \pm 7.4 \times 10^{-6}$
Throttle ($q = 1.5$)	3.175, $k = 5$	$2.55 \times 10^{-6} \pm 1.1 \times 10^{-6}$
Asynchronous μ^2-SGD	0.125, $k = -9$	$1.82 \times 10^{-2} \pm 1.4 \times 10^{-5}$
BASGD ($B = 10$)	5.04, $k = 7$	$8.89 \times 10^{-3} \pm 2.4 \times 10^{-3}$
BASGDm ($B = 10$, $\mu = 0.9$)	5.04, $k = 7$	$6.99 \times 10^{-3} \pm 3.0 \times 10^{-4}$

dispatch and its arrival) is naturally heavy-tailed. See Fig. 7 for our experiment environments' case, similarly computed as in Mishchenko et al. (2022).

In the right panel of Figure 3 one worker sleeps for 100 times its own compute time after every gradient. Figure 3 shows the seed 42 run of every method.

E.3 MNIST EXPERIMENT (FIGURE 2)

Table 3: MNIST configuration used in Figure 2.

Dataset	MNIST (60,000 train / 10,000 test), i.i.d. clients each drawing minibatches from its own reshuffling of the full training set
Model	Conv-Conv-FC-BatchNorm-FC (66,230 parameters), cross-entropy loss
Clients	$n = 20$ (15 honest, 5 Byzantine. $\delta = 0.25$)
Arrivals (standard attacks)	periodic sequence as in the Dahan & Levy (2024)'s code every third arrival is Byzantine ($1/3$ of updates). The arriving client is drawn within its group with probability $\propto$ its index
Arrivals (flooding)	Independent Poisson processes: honest client i has rate $r_i \propto i$ (honest rates sum to 1); the Byzantine group has total rate $\rho/2$, split $\propto$ index, $\rho \in \{1, 3, 10, 30\}$, i.e. an expected Byzantine share of updates $\rho/(2+\rho) = 0.33,\ 0.60,\ 0.83,\ 0.94$
Batch size	16 per client
Training horizon	16,000 honest updates (800 sync. iterations, 4.3 epochs); test every 500
Attacks	RD ($\sigma = 0.2$), NG ($-10\times$), Empire ($\epsilon = 6$), ALIE ($z_{\max} = 0.253$), fixed-vector flood, random-vector flood (norm 10)
Methods	Kardam ($f = 5$); BASGD / BASGDm ($B = 11$, trimmed mean, $\Delta = 20$, $\mu = 0.9$); μ^2-SGD (weighted CWMed / RFA; $\beta = 0.25$, $\gamma = 0.1$); THROTTLE ($q \in \{1.1, 1.5\}$, $\tau = 100$, $\lambda = 1$)
η tuning	grid $\{1, 0.1, 0.01\}$ without attack (Table 4), clipping radius $\lambda = 1$
Seeds	$\{1, 2, 3\}$

Table 4: Learning-rate selection on MNIST without attack: final test accuracy (%, mean ± std over seeds $\{1, 2, 3\}$) for $\eta \in \{0.01, 0.1, 1\}$. Throttle results are shown for $\lambda = 1$. Bold entries indicate the selected values for the baseline methods; ties are broken towards the smaller η.

Method	$\eta = 0.01$	$\eta = 0.1$	$\eta = 1$
Kardam	41.5 ± 14.7	$\mathbf{70.3 \pm 10.5}$	68.5 ± 26.4
BASGD	94.4 ± 0.4	98.3 ± 0.2	$\mathbf{98.7 \pm 0.1}$
BASGDm	94.7 ± 0.5	98.6 ± 0.1	$\mathbf{99.1 \pm 0.1}$
μ^2-SGD (CWMed)	99.1 ± 0.1	$\mathbf{99.2 \pm 0.2}$	10.6 ± 0.9
μ^2-SGD (RFA)	99.1 ± 0.1	$\mathbf{99.2 \pm 0.1}$	11.0 ± 0.6
Throttle $q = 1.1$, $\lambda = 1$	98.8 ± 0.1	99.2 ± 0.1	99.2 ± 0.1
Throttle $q = 1.5$, $\lambda = 1$	97.9 ± 0.1	99.1 ± 0.1	99.0 ± 0.1

E.4 Computing environment

Both experiments ran on one workstation (Table 5). The least-squares runs use NumPy on the CPU only (40 Ray actors on 20 cores). The MNIST study (585 runs) took 5.3 hours of wall-clock time, about 74 process-hours. The least-squares step-size sweeps took about 10 hours and the final 42 runs about 40 minutes, executed strictly one Ray instance at a time.

Table 5: Runtime hardware and software.

CPU	20 logical cores (aarch64)
GPU	NVIDIA GB10, CUDA 13.0 (MNIST, μ^2-SGD only)
Python / NumPy	3.12.3 / 2.4.4
PyTorch	2.12.0+cu130 (MNIST)
Ray	2.58.0 (least squares)

Data and model. We use the standard MNIST dataset (60,000 training and 10,000 test images). The model is the convolutional network of the Dahan & Levy (2024)'s codebase: $\texttt{Conv}(1{\to}20, 5{\times}5)$–ReLU–MaxPool$(2)$–$\texttt{Conv}(20{\to}50, 5{\times}5)$–ReLU–MaxPool$(2)$–$\texttt{FC}(800{\to}50)$–BatchNorm–ReLU–$\texttt{FC}(50{\to}10)$ with 66,230 parameters and the cross-entropy loss. Every client draws minibatches of 16 from its own independent reshuffling of the full training set (i.i.d. clients). Byzantine clients that run the honest protocol (RD, NG) use separate data, so they never consume the honest data budget. Test accuracy on the full test set is evaluated every 500 honest arrivals.

Simulator and arrival schedules. There are $n = 20$ clients, of which the last 5 are Byzantine ($\delta = 0.25$). A run ends after $16{,}000$ honest updates, i.e. 800 synchronous iterations of 20 clients or 4.3 epochs. Two arrival schedules are used. *Periodic* (standard-attack panels in Figure 2) reproduces the Dahan & Levy (2024)'s code: arrivals are numbered $1, 2, \ldots$, every third arrival is Byzantine (the code rounds its parameter $\lambda_{\text{byz}} = 0.4$ to the period $\lceil 1/0.4 \rceil = 3$, so the Byzantine share of arrivals is $1/3$, not 0.4).

Within each group (honest and Byzantine), the sending client is drawn with probability proportional to its index plus one. Therefore, honest client rates differ by a factor of 15 from Byzantine rates. For flooding attacks, every client is an independent Poisson process; honest rates are the same proportional weights normalized to sum to one. The Byzantine group's total rate is $\rho \cdot \frac{p_0}{1-p_0} = \rho/2$ with $p_0 = 1/3$. The expected Byzantine share of all arrivals is $\rho p_0/(1{-}p_0{+}\rho p_0) = 0.33, 0.60, 0.83, 0.94$ for $\rho = 1, 3, 10, 30$ (measured: $0.337, 0.604, 0.835, 0.938$) The total number of arrivals grows from $24{,}000$ to $260{,}000$ while the honest budget stays fixed.

Attacks. Let g be the update an honest client would send and L the set of honest clients whose latest update has been delivered.

- RD (random disturbance, Yang & Li, 2023) sends $g + \xi$, $\xi \sim \mathcal{N}(0, \|0.2\, g\|^2 I)$.
- NG (negative gradient, Yang & Li, 2023) sends $-10\, g$.

- Empire (Xie et al., 2020a) sends $-6\,\bar{g}$, where $\bar{g}$ is the mean of the honest updates' most recently delivered to the server.
- ALIE (Baruch et al., 2019): following Yang & Li (2023), a Byzantine client sends $\tilde{g}$ with $\tilde{g}_j = \mathrm{mean}_j - z^{\max}\,\mathrm{std}_j$, where mean_j and std_j are the mean and standard deviation of the j-th coordinate over the honest updates' most recently delivered to the server, and $z^{\max} = \Phi^{-1}\big(\frac{m-\lfloor m/2+1\rfloor}{m-r}\big)$ with m clients, r of them Byzantine, and Φ^{-1} the standard normal quantile function.

Empire and ALIE used omniscient property of Byzantine clients in the sense that they read the honest updates that have actually been delivered to the server. All Byzantine updates are generated at send time from the latest available information. The two *flooding* attacks require no gradient computation: the fixed-vector flood sends the same random direction of norm 10 in every message, the random-vector flood a fresh standard normal direction rescaled to norm 10, both at ρ times the honest group's rate as described above.