Queuing dynamics of asynchronous Federated Learning

Louis Leconte
Lisite, Isep, Sorbonne Univ.
Math. and Algo.
Sciences Lab, Huawei Tech.

Matthieu Jonckheere
LAAS-CNRS,
Université de Toulouse,
CNRS, France

Sergey Samsonov
HSE University,
Moscow, Russia

Eric Moulines
CMAP
Ecole Polytechnique,
France

Abstract

We study asynchronous federated learning mechanisms with nodes having potentially different computational speeds. In such an environment, each node is allowed to work on models with potential delays and contribute to updates to the central server at its own pace. Existing analyses of such algorithms typically depend on intractable quantities such as the maximum node delay and do not consider the underlying queuing dynamics of the system. In this paper, we propose a non-uniform sampling scheme for the central server that allows for lower delays with better complexity, taking into account the closed Jackson network structure of the associated computational graph. Our experiments clearly show a significant improvement of our method over current state-of-the-art asynchronous algorithms on an image classification problem.

1 Introduction

Federated learning (FL) is a distributed learning paradigm that allows agents to learn a model without sharing data (Konečnỳ et al., 2015; McMahan et al., 2017). A central server (CS) coordinates the entire process. In most implementations, the CS uses synchronous operations. During each epoch, the CS communicates with a subset of clients and waits for their "local updates". The CS then uses these local updates to update the global model; McMahan et al. (2017); Wang et al. (2021). Nevertheless, different computational speeds, latencies, and/or transmission bandwidths lead to a cascade of issues such as delays and stragglers. In each epoch, CS must keep up with the pace of the slowest agent.

A solution called `FedAsync` eliminates the structured rounds of CS interaction and transitions to asynchronous optimization (Xie et al., 2019). This approach, along with subsequent works in this direction (Chen et al., 2020, 2021; Xu et al., 2021), enables asynchronous operation for the CS and agents. `FedAsync` facilitates the aggregation of agents updates through the CS, rendering the solution highly scalable. More recently, Mishchenko et al. (2022) has expanded the theoretical comprehension of purely asynchronous SGD within a homogeneous framework where all agents can access identical data; certain limitations still persist in heterogeneous scenarios.

In practical applications of asynchronous federated learning (FL), interactions between agents and the CS require the use of queues for processing (potentially) multiple jobs. The distribution of processing delays varies significantly across agents, and this variability has been shown to have a negative impact on optimization processes. In this paper, we significantly improve the analysis delineated in Koloskova et al. (2022), exploring in depth an asynchronous algorithm—`AsyncSGD`. This algorithm empowers nodes to queue tasks, initiating communication with the central server upon task completion. The subsequent analysis adheres to a virtual iterates sequence under standard non-convex assumptions. However, previous studies made overly simplistic assumptions about the dynamics of queues, choosing to represent them with an upper bound on the processing delays encountered by the CS. In contrast, our theory intricately models the queuing dynamics using a **stationary closed Jackson network**. This approach allows capturing precisely the queuing dynamics - number of buffered tasks, processing delay, etc...-, as a function of agents speed. We integrate assumptions about the service time distributions, enabling us to define the explicit stationary distribution of the number of in-service tasks.

Proceedings of the 27$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238.

Contributions.

- We identify key variables that affect the performance of the optimization procedure and depend on the queuing dynamics.
- Building on the findings of our analysis, we introduce a new algorithm called `Generalized AsyncSGD`. This algorithm exploits non-uniform agent selection and offers two notable advantages: First, it guarantees unbiased gradient updates, and second, it improves convergence bounds.
- To gain deeper insights, we delve into the limit regimes characterized by large concurrency. In these contexts, our analysis shows that heterogeneity in server speeds can be balanced by the strategic use of non-uniform sampling among agents.
- Experimental results show that our approach outperforms other asynchronous baselines on a deep learning experiment.

Related works Up to this point, the focus has been on synchronous federated learning techniques, as evidenced by notable contributions such as (Wang et al., 2020; Qu et al., 2021; Makarenko et al., 2022; Mao et al., 2022; Tyurin and Richtárik, 2022). However, synchronous methods often suffer from suboptimal resource allocation and long training times. Moreover, as the number of participating agents grows, coordinating synchronous rounds with all participants becomes an increasingly difficult task for the central server (CS).

Synchronous federated learning methods are particularly vulnerable to the challenge of stragglers, prompting the emergence of research endeavors rooted in the principles of `FedAsync` and its subsequent extensions, as elucidated by (Xie et al., 2019). The core concept revolves around updating the global model upon receiving a local model at the central server (CS). `ASO-Fed` (Chen et al., 2020) introduces memory-based mechanisms on the local client side. `AsyncFedED` (Wang et al., 2022), drawing inspiration from `FedAsync`'s instantaneous update strategy, proposes dynamic adjustments to the learning rate and the number of local epochs to mitigate staleness.

Looking at the problem from a different perspective, `QuAFL` (Zakerinia et al., 2022) introduces a concurrent algorithm that aligns closely with the `FedAvg` strategy. `QuAFL` seamlessly integrates asynchronous and compressed communication methods while ensuring convergence. In this approach, each client is allowed a maximum of K local steps and can be interrupted. To address the variability in computational speeds across nodes, `FAVAS` (as discussed by Leconte et al. (2023)) strikes a balance between the slower and faster clients. `FedBuff` (Nguyen et al., 2022) addresses asynchrony and concurrency by incorporating a buffer on the server side. Clients conduct local iterations, with the CS updating the global model solely upon completion by a predefined number of different clients.

Similarly, the work presented by Koloskova et al. (2022) revolves around Asynchronous SGD (`AsyncSGD`), offering guarantees contingent on the maximum delay. Recent developments by Fraboni et al. (2023) expand upon the ideas presented by Koloskova et al. (2022), allowing multiple clients to contribute within a single round. Liu et al. (2021) diverges from the buffer-centric approach and develops Adaptive Asynchronous Federated Learning (`AAFL`) to address speed disparities among local devices. Similar to `FedBuff`, Liu et al. (2021)'s method entails only a fraction of locally updated models contributing to the global model update. Convergence guarantees within asynchronous distributed frameworks commonly rely on an analysis contingent upon the maximum delay (Nguyen et al., 2022; Toghani and Uribe, 2022; Koloskova et al., 2022), which substantially exceeds the average delay.

Tyurin and Richtárik (2023) introduces a novel asynchronous algorithm, presenting optimal convergence guarantees under the assumption of fixed computational speeds among workers over time. Notably, Mishchenko et al. (2022) conducts an independent analysis of asynchronous stochastic gradient descent that does not rely on gradient delay. However, in the context of a heterogeneous (non-i.i.d.) setting, convergence is guaranteed up to an additive term linked to the dissimilarity limit between the gradients of local and global objective functions.

AsGrad (Islamov et al., 2023) is a recent contribution that proposes a general analysis of asynchronous FL under bounded gradient assumption, and adapt random shuffling SGD to the asynchronous case. Most of standard asynchronous baselines can be expressed in the general form proposed in Islamov et al. (2023), and strong convergence guarantees are provided. But all derivations assume delays are finite quantities.

While an impressive body of research has been dedicated to establishing theoretical tools for the performance evaluation of communication networks, including the development of intricate scheduling mechanisms and models (as exemplified in Malekpourshahraki et al. (2022); Stavrinides and Karatza (2018) and references therein), the predominant focus has revolved around performance metrics such as delays/completion times (measured in terms of physical time per node), queue lengths, and throughputs. When it comes to modeling federated learning, the application of stochastic network paradigms is significant, based on a wealth of

results in the field. However, it is important to recognize that the key metrics to be computed in FL are significantly different from traditional network metrics, as we will discuss in more detail shortly. In particular, the measurement of delay in this context must take into account server steps, which introduces a more complicated dependence on the dynamics of each queue within the network. Moreover, optimizing these novel metrics may require completely different resource allocation paradigms.

2 Problem statement

We consider the optimization problem:

$$\min_{w\in\mathbb{R}^d} \sum_{i=1}^n \mathbb{E}_{(x,y)\sim D_i^{\rm data}}[\ell_i(\mathrm{NN}(x,w),y)]\,. \quad (1)$$

Here d is the number of parameters (network weights and biases), n is the total number of clients, ℓ_i is the local loss function, $\mathrm{NN}(x,w)$ is the DNN prediction function, $D_i^{\rm data}$ is the training data distribution on node i. In FL, the distributions $D_i^{\rm data}$ are allowed to differ between clients (statistical heterogeneity). Let us denote by

$$f_i(w) := \mathbb{E}_{(x,y)\sim D_i^{\rm data}}[\ell_i(\mathrm{NN}(x,w),y)]$$

the local function optimized on node i and $f := \frac{1}{n}\sum_{i=1}^n f_i$. Each node i does not compute the true gradient of the function f_i, but has access to a *stochastic* version of the gradient, denoted by $\tilde{g}_i$.

We consider the *task* as a computation of a gradient (or possibly stochastic gradient) on one of the clients. We assume that n clients process a fixed number of tasks $\mathrm{C}\in\mathbb{N}^*$ in parallel, and the total number of CS steps is T. Once a task is completed by an agent, the corresponding update is sent to the CS, which updates the global model and then passes the updated parameters to a new agent with probabilities $(p_i)_{i=1}^n$. The selected agent might already be busy computing a previous update. When the agent is busy, the new job enters a queue that is serviced on a first-in-first-out basis (FIFO). To perform this analysis, we must establish the following definitions:

- $J_k \in [1,n]$ is the node completing a task at the k-th CS epoch (or step),
- $K_{k+1}\in[1,n]$ is the node selected at step $k\in[1,T]$,
- $X_{i,k}$ is the number of tasks in node i at step k, $i\in[1,n]$, $k\in[1,T]$.

All these random variables can be constructed as deterministic functions of the i.i.d. sequence $(R_l)_{l\in\mathbb{N}}$ which stands for the routing decisions and the sequences $(\xi_l^i)_{l\in\mathbb{N},i=1,\dots,n}$ which stand for the service times (durations) of the tasks in each node. These two i.i.d. sequences are independent.

We denote by $\mathrm{M}_{i,k}^T$ the number of CS steps between the time that a task is sent to node i and the time it is completed:

$$\mathrm{M}_{i,k}^T = \mathbb{1}_{\{i\}}(K_{k+1}) \sum_{r=k}^T \mathbb{1}_{(\sum_{l=k}^r \mathbb{1}_{J_l=i})<X_{i,k}}$$

Finally, for $k\in\{0,\dots,T\}$ we define

$$I_k^T = \sum_{l=0}^k l\cdot \mathbb{1}_{\{k-l\}}(\mathrm{M}_{K_{l+1},\ell}^T)\,,$$

which is the CS step corresponding when the task was dispatched to node J_k.

Direct analysis of the server iterate $(w_k)_{k\geq 0}$ is difficult because we do not have access to the joint distribution of $(J_k)_{k\geq0}$, $(\mathrm{M}_{i,k}^T)_{k\geq0}$ and $(I_k^T)_{k\geq0}$. Koloskova et al. (2022) assumes an upper bound on the number of gradients that are pending at step k but have not yet been applied. Koloskova et al. (2022) proposes to select new nodes with uniform probability. In `Generalized AsyncSGD` (see Algorithm 1), we add some degree of freedom by allowing the central server to select a new node K_{k+1} with (possibly non-uniform) probability $\mathbf{p}=(p_j)_{j=1}^n$.

Algorithm 1: `Generalized AsyncSGD`

Input : Number of server steps T, Number of tasks C ;

`/* At the Central Server */`

Initialize
  Initialize parameters w_0;
  Select initial set of clients $\mathcal{S}_0$, with $|\mathcal{S}_0|=\mathrm{C}$;
  Server sends w_0 to each client in $\mathcal{S}_0$;
  All clients in $\mathcal{S}_0$ compute gradients on w_0 ;
  Compute optimal $(\mathbf{p},\eta)$ by minimizing (3) ;
end
for $k=0,\dots,T$ **do**
  Server receives stochastic gradient $\tilde{g}_{J_k}(w_{I_k})$;
  Update $w_{k+1}\leftarrow w_k - \frac{\eta}{np_{J_k}}\tilde{g}_{J_k}(w_{I_k})$;
  Sample a new client K_{k+1} with prob. $p_{K_{k+1}}$;
  Send new model w_{k+1} to K_{k+1} ;
end

We denote

$$\mathrm{m}_{i,k}^T := \mathbb{E}[\mathrm{M}_{i,k}^T]\,,\ \text{and}\ \mathrm{m}_k^T := \sum_{i=1}^n \mathrm{m}_{i,k}^T/(n^2p_i^2)\,.$$

It is worth noting that $\mathrm{m}_{i,k}^T$ depends on the sampling probability p_i, but for simplicity, we prefer not to index explicitly by $\mathbf{p}:=(p_j)_{j=1}^n$. We will show in Section 4

Figure 1: Evolution of $\mathrm{m}_{i,k}^T$ w.r.t. k, for two networks of size $n = 10, 50$ initialized with full concurrency.

that, $\lim_{k\to\infty}\lim_{T\to\infty}\mathrm{m}_{i,k}^T = \mathrm{m}_i$ (these expectations become stationary) see Proposition 3. Of course, the exact analysis of the transient behavior is very complex: simple upper bounds can be computed, but these are generally not expressive and hide the influence of key parameters. In Figure 1, we simulate $n = 10$ and $n = 50$ nodes with $\mathrm{C} = n$ initial tasks. In this simulation, nodes $\{0, 1, 2, 3, 4\}$ are 10 times faster than the other nodes to compute a task. Without loss of generality, we focus on the first node ($i = 1$), and we plot the value of $\mathrm{m}_{i,k}^T$ with respect to k, for $T = 500$. The value of $\mathrm{m}_{i,k}^T$ becomes stationary when $k > 50$ and $k > 150$, for $n = 10$ and $n = 50$, respectively.

In our analysis, in line with the approach presented in Koloskova et al. (2022) (but the same idea was applied earlier), we introduce the *virtual iterates* μ_k as follows:

$$\begin{cases} \mu_0 = w_0, \\ \mu_1 = \mu_0 - \eta \sum_{i\in\mathcal{S}_0} \frac{1}{np_i}\tilde{g}_i(w_0), \\ \mu_{k+1} = \mu_k - \frac{\eta}{np_{K_k}}\tilde{g}_{K_k}(w_k)\,, \quad k \geq 1\,. \end{cases}$$

In fact μ_k is defined as if the selected client K_k was instantaneously contributing to the server update. Note that the gradients are computed on w_k, not on μ_k. The difference between μ_k and w_k consists of all the gradients computed (on potentially outdated w's) and not applied yet, $\mu_k - w_k = \sum_{(i,j)\in\mathcal{I}_k} \frac{-1}{np_i}\tilde{g}_i(w_j)$, with $\mathcal{I}_k = \{(i,j) \in [1,n]\times[1,k] | (X_{i,k} \neq 0)$ and $(\sum_{i=1}^n \mathrm{M}_{i,j}^T > k - j)\}$.

3 Non-convex bounds

Following the setting considered in Koloskova et al. (2022), Nguyen et al. (2022), we focus on the scenario of the optimization problem (1) with L-smooth and nonconvex objective functions f_i. Proofs are detailed in Appendix C. Our analysis is based on the following assumptions:

A1. *Uniform Lower Bound: There exists $f_* \in \mathbb{R}$ such that $f(w) \geq f_*$ for all $w \in \mathbb{R}^d$.*

A2. *Smooth Gradients: For any client i, the gradient ∇f_i is L-Lipschitz continuous for some $L > 0$, i.e. for all $w, \mu \in \mathbb{R}^d$: $\|\nabla f_i(w) - \nabla f_i(\mu)\| \leq L\|w - \mu\|$.*

A3. *Bounded Variance: For any client i, the variance of the stochastic gradients is bounded by some $\sigma^2 > 0$, i.e. for all $w \in \mathbb{R}^d$: $\mathbb{E}[\|\widetilde{g}_i(w) - \nabla f_i(w)\|^2] \leq \sigma^2$.*

A 4. *Bounded Gradient Dissimilarity: There exist constant G, such that for all $w \in \mathbb{R}^d$: $\|\nabla f_i(w) - \nabla f(w)\|^2 \leq G^2$.*

The assumption **A**3 can be generalized to the *strong growth condition* Vaswani et al. (2019):

$$\mathbb{E}[\|\widetilde{g}_i(w) - \nabla f_i(w)\|^2] \leq \sigma^2 + \rho^2 \|\nabla f_i(w)\|^2\,,$$

following Beznosikov et al. (2023). Full details are given in the appendix.

Theorem 1. *Assume **A**1 to **A**4 and let the learning rate η satisfy $\eta \leq \eta_{\max}(\mathbf{p})$, where*

$$\eta_{\max}(\mathbf{p}) =: \frac{1}{4L}\big(\mathrm{C}^{-1/2}\max_{k\leq T}\{\mathrm{m}_k^T\}^{-1/2} \wedge 2/\sum_{i=1}^n \frac{1}{n^2 p_i}\big)\,.$$

Then `Generalized AsyncSGD` *converges at rate:*

$$\sum_{k=0}^{T}\frac{\mathbb{E}[\|\nabla f(w_k)\|^2]}{8(T+1)} \leq \frac{\mathbb{E}[f(\mu_0) - f(\mu_{T+1})]}{\eta(T+1)} + \frac{\eta LB}{n}\sum_{i=1}^n \frac{1}{np_i} + \frac{\eta^2 L^2 B\,\mathrm{C}}{n}\sum_{i=1}^n \frac{\sum_{k=0}^T m_{i,k}^T}{np_i^2(T+1)}\,, \tag{2}$$

where $B = 2G^2 + \sigma^2$.

The upper bound in Theorem 1 includes three distinct terms. The first term is a standard component associated with a general nonconvex objective function; it expresses how the choice of initialization affects the convergence process.

The second and third terms depend on the statistical heterogeneity within the client distributions and the fluctuations of the minibatch gradients. If we assume uniform probabilities, the second term agrees with that of `FedAvg`: this is the bound that would be obtained with synchronous optimization. In contrast, the third term encapsulates the unique challenge posed by optimization within an asynchronous framework.

Before moving on to the main study, it is important to analyze the behavior of this bound. Note first that the upper bound is minimized by $T \to \infty$ if we set $\eta = O(T^{-1/2})$. In this setting, the third term of the upper bound, which is proportional to η^2, becomes negligible. To obtain the optimal probability value $\mathbf{p}$, one should minimize $\sum_{i=1}^n 1/p_i$ in this regime, subject to the condition $\sum_{i=1}^n p_i = 1$. This minimization is achieved when $p_i = 1/n$. Thus, with $T \to \infty$, a uniform distribution of weights turns out to be a reasonable choice.

A worked-out example For regimes other than $T \to \infty$, the bound given by Eq. (2) proves difficult to handle due to the complicated relationship between $\mathrm{m}_{i,k}^T$ and the sampling distribution $\mathbf{p}$. In the next section, we will apply queuing theory methods to shed light on these quantities. However, before diving into this detailed analysis, we will first examine the behavior of the bound using a simple example. We choose the sampling probabilities $\mathbf{p}$ and the step size η by solving the constrained optimization problem $\min_{\mathbf{p},\eta} G(\mathbf{p},\eta)$ as a function of $\eta \le \eta_{\max}(\mathbf{p})$, where

$$G(\mathbf{p},\eta) = \frac{A}{\eta(T+1)} + \frac{\eta L B}{n}\sum_{i=1}^n \frac{1}{np_i} + \frac{\eta^2 L^2 B\,\mathrm{C}}{n}\sum_{i=1}^n \frac{\sum_{k=0}^T \mathrm{m}_{i,k}^T}{np_i^2(T+1)}\,, \quad (3)$$

and where $A = \mathbb{E}[f(\mu_0) - f(\mu_{T+1})]$. To better understand this bound, let us examine a simple case. Suppose we have $n = 100$ clients that are classified as either "fast" or "slow". There are $n_f = 90$ fast clients and $n - n_f = 10$ slow clients, which are assumed to have the same characteristic (within each group). We will focus on how the proposed bound behaves based on the ratio of the processing speed of the "fast" and "slow" clients, the proportions of fast and slow clients, and the concurrency. We will also examine two situations: one in which the processing time for gradient requests is fixed, and another in which it follows an exponential distribution. By default, slow clients process a gradient in a time unit of 1 (on average in the random case), while fast clients take $\frac{1}{\mu_f} \le 1$ units on average. Let $p \in (0,1)$. We denote p the probability to select one of the fast clients, and $q = \frac{1}{n-n_f} - p\frac{n_f}{n-n_f}$ the probability of selecting one of the slow clients (we need $n_f p + (n-n_f)q = 1$). The parameters are $L = 1$, $B = 20$ (to assess the effects of gradient noise and statistical heterogeneity), $A = 100$ (to highlight the impact of initial conditions). The number of tasks is varied $\mathrm{C} = 10, 50, 100$, to assess the impact of concurrency. For the total number of CS iterations, we consider $T = 10^4$. We plot the selection probability p versus μ_f, ranging from 2 to 16 in Figure 2. Furthermore, we graphically illustrate the relative improvement of the upper bound when compared with the uniform selection problem in Figure 3. The results show that a significant improvement may be achieved (from 30% when $\mu_f = 2$ to 55% when $\mu_f = 16$). To achieve this improvement, we should decrease the probability of selecting fast clients to $p = 7.3 \cdot 10^{-3}$. The conclusion (that will be justified theoretically in the next section) is that fast agents should be selected less frequently than slow agents. Even though this result may appear to be counter intuitive, it is justified by the fact that by selecting slow customers more frequently, processing

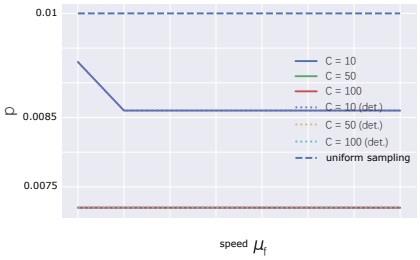


Figure 2: Optimal sampling probability p as a function of the speed for different concurrency levels. The number of nodes is fixed to $n = 100$ nodes.

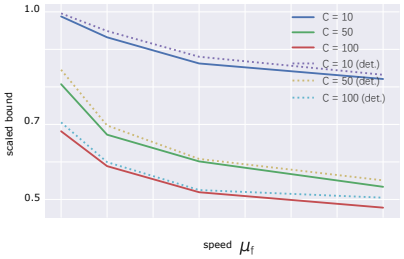


Figure 3: Relative improvements of the upper bounds as a function of the speed for different concurrency levels. The number of nodes is fixed to $n = 100$ nodes.

times are reduced. Our simulations shows that the average delay at the CS is divided by 10 and 2, for fast and slow nodes respectively. More details are given in Appendix F.

Finally, these experiments also show that the distribution of the working time required for gradient evaluation does not have a significant impact: results are very similar whether the working time is deterministic or distributed according to an exponential (and therefore random) distribution (provided that the mean are preserved).

Comparison with `FedBuff` and `AsyncSGD` We emphasize that previous analyses of both `FedBuf` and `AsyncSGD` are based on strong assumptions: the queuing process is not considered in their analysis. In practice, slow clients with delayed information contribute. Nguyen et al. (2022); Koloskova et al. (2022) propose to bound this delay uniformly by a quantity τ_{max}. We retain this notation while reporting complexity bounds in Table 1, but argue that nothing guarantees that τ_{max} is properly defined. In Figure 4 we compared the relative improvements of the upper bounds obtained with `Generalized AsyncSGD`, w.r.t. `FedBuff` and `AsyncSGD` for the scenario described in the previous paragraph. The plot illustrates the massive improvement achieved by `Generalized AsyncSGD` when optimal selection probabilities are used. It also

Table 1: Asynchronous bounds (up to numerical constants) under non-convex assumption for T server steps. C is the number of initial tasks. $A = \mathbb{E}[f(\mu_0) - f_*]$, and $B = 2G^2 + \sigma^2$. $\tau_{\max}$ is defined in Toghani and Uribe (2022) as the maximum delay. $\tau_c, \tau^i_{\text{sum}}$ are defined in Koloskova et al. (2022) as the average number of active nodes, and the sum of delays of node i, respectively.

Method	Bounds	η
`FedBuff`	$\frac{A}{\eta(T+1)} + \eta LB + \eta^2 \tau_{\max}^2 L^2 B n$	$\leq \frac{1}{L\sqrt{\tau_{\max}^3}}$
`AsyncSGD`	$\frac{A}{\eta(T+1)} + \eta LB + \eta^2 \tau_c L^2 B \sum_{i=1}^n \frac{\tau^i_{\text{sum}}}{T+1}$	$\leq \frac{1}{L\sqrt{\tau_c \tau_{\max}}}$
`Generalized AsyncSGD`	$\frac{A}{\eta(T+1)} + \eta LB \sum_{i=1}^n \frac{1}{n^2 p_i} + \eta^2 \,\mathrm{C}\, L^2 B \sum_{i=1}^n \frac{\sum_{k=0}^T \mathrm{m}^T_{i,k}}{n^2 p_i^2 (T+1)}$	$\leq \frac{1}{L\sqrt{\mathrm{C} \max_{k \leq T} \mathrm{m}^T_k}}$

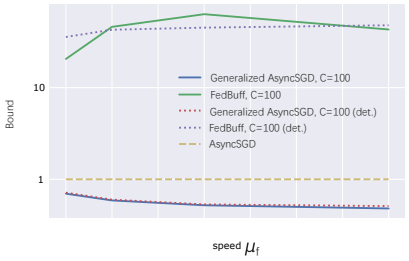


Figure 4: Relative improvement of `Generalized AsyncSGD` over `FedBuff` and `AsyncSGD` as a function of speed. The number of nodes is fixed to $n = 100$ nodes.

illustrates that the bounds previously reported in the literature do not capture the essence of the problem. In particular, this comparison holds under the condition that the work time for gradient evaluation is deterministic, such that $\tau_{\max}$ equals C times the work time of a slow client. When the working time is exponential, the maximum delay as defined in the analyses of `FedBuff` and `AsyncSGD` is infinite, and the bounds in Nguyen et al. (2022) and Koloskova et al. (2022) are then empty.

4 Closed network

The aim of this section is to obtain theoretical guarantees using precise results on closed Jackson networks (Jackson, 1954, 1957). We assume in this section that task duration follows an exponential distribution, with each user having its own mean. While it is feasible to extend these findings to deterministic durations and even almost arbitrary duration distributions, this complicates the theory. This approach not only captures the different speeds of the clients, but also allows for a precise assessment of the system dynamics. Consequently, we can accurately determine the critical values $\mathrm{m}^T_{i,k}$ in a steady state. Detailed proofs are postponed to the Appendix D.

Stationary distribution and key performance indicators Let us denote by $(D_i(t))_{i=1,\dots,n}$ the number of task departures from node i at time t, with the convention that $D_i(0) = 0$. $(T_{i,l})_{l\in\mathbb{N}}$ are the jump times associated with the counting process D_i. We further denote by $N(t)$ the number of tasks arriving at the CS, given by

$$N(t) = \textstyle\sum_{i=1}^n D_i(t)\,,$$

while $(T_l)_{l\in\mathbb{N}}$ are the jump times associated with N. Observe that the indices k used in the previous Section correspond to those jump times. Finally we define the sequences of times $(\tau_{i,l})_{l\in\mathbb{N}}$ as the arrival times to node i after time 0.

In what follows, we assume that the task duration is i.i.d. and exponentially distributed at rate μ_i, and that the routing decisions are also i.i.d. (and independent of everything else). As in the previous section, we denote by p_i the probability that the dispatcher sends a task to node i.

We denote by $X(t) = (X_1(t), \dots, X_n(t))$ the continuous-time stochastic process describing the number of tasks in each node. The unit vectors in $\mathbb{N}^n$ are denoted by $(e_i)_{i=1\dots,n}$. We have the following results for X.

Proposition 2. *Under the above assumptions, the dynamics of $(X(t), t \geq 0)$ is that of a closed Jackson network on the complete graph with n nodes and* C *tasks. The generator of the corresponding jump Markov process is given for all $x \in \mathbb{N}^n, i \in \mathbb{N}, j \in \mathbb{N}$ by*

$$q(x, x + e_i - e_j) = p_i \mu_j \mathbb{1}(x_j > 0)\,.$$

Furthermore, defining $\theta_i = \frac{p_i}{\mu_i}$, the stationary distribution of X may be expressed as:

$$\pi_C(x_1, \dots, x_n) = H_{\mathrm{C}}^{-1} \textstyle\prod_{i=1}^n \theta_i^{x_i}\,,$$

with $H_{\mathrm{C}} = \sum_{x:\sum_i x_i = \mathrm{C}} \prod_{i=1}^n \theta_i^{x_i}$.

Building on the understanding gained in Section 2, our goal is to quantify the number of server steps that are

executed when a new task arrives at a given node (say node i) and subsequently returns to the dispatcher.

For simplicity, the analysis is performed in the stationary regime. In particular, this means that at time 0 the distribution of the number of tasks in each node follows the product measure defined in Proposition 2. We denote by $\mathbb{E}^{\mathrm{C}}$ the stationary average of the closed Jackson network when the total number of tasks is equal to C. We can now state the main result of this section:

Proposition 3. *Given the model assumptions and assuming stationarity, for all $k \in \mathbb{N}$,*

$$\lim_{T\to\infty} \mathrm{m}_{i,k}^T = \mathbb{E}^{\mathrm{C}-1}\Big[\int_0^{S_i} \sum_{j=1}^n \mu_j \mathbb{1}(X_j(s)>0)\,\mathrm{d}s\Big]\,.$$

From now on, we use the notation $\mathrm{m}_i = \lim_{T\to\infty} \mathrm{m}_{i,k}^T$. This quantity is in general difficult to simplify further. We consider in the sequel a specific regime in which we can obtain tractable expressions as the Jackson network gets close to saturation. We now describe how the queue length X_i, and m_i depend on the agents speed and selection probability $\mathbf{p}$, under this saturated stationary regime similar to the Halfin-Whitt regime in queuing theory (Halfin and Whitt, 1981).

Scaling regime We rely on scaling bounds to provide rules of thumb when certain traffic conditions are satisfied. We follow the derivation of Van Kreveld et al. (2021), which considers closed Jackson networks under a particular load regime. We assume without loss of generality that $\theta_n = \max_{i\in[1,n]}(\theta_i)$; where $\theta_i = p_i/\mu_i$ (see Proposition 2). Due to the closed nature of the network, rescaling all parameters through division by the maximum traffic load leads to a different normalization constant $\tilde{H}_C$, but otherwise has no effect on the stationary joint distribution:

$$\pi(x_1,\dots,x_{n-1}) = \tilde{H}_C^{-1}\prod_{i=1}^{n-1}{\gamma_i}^{-x_i}\,,$$

where for all $i \in [1,n]$, $\gamma_i = \theta_n/\theta_i$. We consider a scaling regime where all nodes are saturated, but at different rates. In Appendix G we also define a more complicated scenario where some queue lengths may degenerate to 0.

2 clusters under saturation We consider two clusters of nodes of size n_f and $n-n_f$, respectively. Nodes $i \in [1,n_f]$ are fast, the rest are slow. We assume that nodes from the same cluster have the same speed μ_f, μ_s, for fast and slow nodes, respectively ($\theta_f < \theta_s$). This gives the *scaled* intensity of the slow nodes $\gamma_s(\iota) = 1$, and the fast nodes $\gamma_f(\iota) := \frac{\theta_s}{\theta_f}$, where ι is the scaling parameter and the scaling regime corresponds to choosing those values as $\gamma_f(\iota) = 1 + c_f\iota^{\alpha-1}$; with $c_f > 0$ a fixed positive constant, and $\alpha \le 1$, while the total number of tasks also scales as follows:

$$\beta\iota^{1-\alpha} = \mathrm{C}+1\,.$$

Choosing $\alpha \le 1$ as in Van Kreveld et al. (2021) ensures that node loads approach 1 as $\iota\to\infty$, enabling the application of Corollary 2 from Van Kreveld et al. (2021). This yields precise results on saturated node queue lengths at high traffic loads, while queue lengths for the remaining nodes are determined by population size constraints. In this context, define X_i^ι as the stationary queue length for a scaling parameter ι and $\mathrm{m}_i(\iota)$ the corresponding value of m_i.

Proposition 4 (Corollary.2 in Van Kreveld et al. (2021))**.** *In stationary regime, as the scaling parameter $\iota\to\infty$,*

$$c_f\iota^{\alpha-1}X_i^\iota \to_{d.}\ \chi_i\,,$$

where $\chi_i = \mathbb{E}\left[E_i \,|\, \sum_{j=1}^{n_f} E_j/c_f \le \beta\right]$, $i\in[1,n_f]$, and the $(E_j)_{j\le n}$ are independent unit mean exponential distributions.

As a consequence, using uniform integrability, we can estimate the following expected value (expected stationary queue lengths of fast, and slow nodes respectively) as follows:

$$\begin{cases} \iota^{\alpha-1}\mathbb{E}[X_i^\iota] \to \frac{\Gamma(c_f\beta)}{c_f}\,, & \forall i\in[1,n_f],\\ \iota^{\alpha-1}\mathbb{E}[X_i^\iota] \to \frac{1}{n-n_f}\left(\beta - n_f\frac{1}{c_f}\Gamma(c_f\beta)\right), & \forall i\in[n_f+1,n]\,.\end{cases}$$

Denoting by $P(k,x) = 1-\sum_{i=0}^{k-1} e^{-x}\frac{x^i}{i!}$, we have:

$$\Gamma(c) = \frac{\mathbb{P}(\sum_{j=1}^{n_f+2}E_j\le c)}{\mathbb{P}(\sum_{j=1}^{n_f+1}E_j\le c)} = \frac{P(n_f+2,c)}{P(n_f+1,c)}\,.$$

We now turn to bound the key quantity $\mathrm{m}_i(\iota)$ for large ι.

Proposition 5. *Using the same assumptions as those of Proposition 4 we get that :*

$$\begin{cases} \limsup_{\iota\to\infty}\iota^{\alpha-1}\mu_f\,\mathrm{m}_i(\iota) & \le \lambda\frac{\Gamma(c_f\beta)}{c_f}\,, \quad \forall i\in[1,n_f]\,,\\ \limsup_{\iota\to\infty}\iota^{\alpha-1}\mu_s\,\mathrm{m}_i(\iota) & \le \lambda\frac{\beta-n_f\Gamma(c_f\beta)/c_f}{n-n_f}\,, \textit{otherwise,}\end{cases}$$

where $\lambda = \sum_{i=1}^n \mu_i$.

We expect these bounds to be sharp for large ι.

Numerical example Under the previous assumptions, we have $\lambda = n_f\mu_f + (n-n_f)\mu_s$. We will further assume $n_f = \frac{n}{2}$, and $p_i = \frac{1}{n}$. Under these conditions, we have that $\Gamma(c_f\beta)$ is close to 1. We can give a closed form approximations of the bounds of the expected delays:

$$\begin{cases} \mathrm{m}_i(\iota) \le \frac{n(\mu_f+\mu_s)}{2\mu_f(\mu_f/\mu_s-1)}\,, & \forall i\in[1,n_f]\,,\\ \mathrm{m}_i(\iota) \le \left(\frac{2\,\mathrm{C}}{n} - \frac{1}{\mu_f/\mu_s-1}\right)\frac{n(\mu_f+\mu_s)}{2\mu_s}\,, & \forall i\in[n_f+1,n].\end{cases}$$

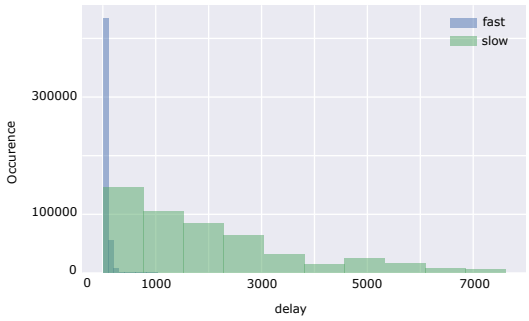


Figure 5: Histogram of fast and slow delays (in number of server steps) for a uniform sampling scheme.

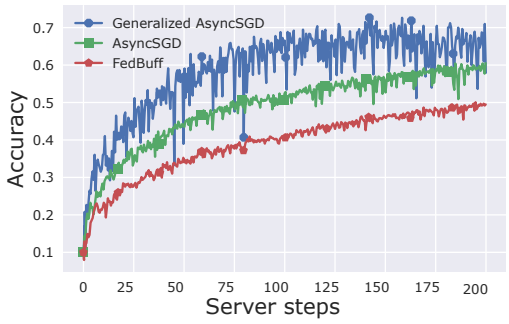


Figure 6: Accuracy on validation dataset on central server, for CIFAR-10 classification task.

All delays bounds estimations have a closed form in the 2-cluster saturated regime: they only depend on the number of tasks in the network C, on the number of nodes n, and on the intensity of nodes μ_f, μ_s. More details on the derivations, and on the following experiment are available in Appendix F. We consider a numerical simulation with $n = 10$ clients, split in two clusters of same size: fast nodes with rate $\mu_f = 1.2$, and slow nodes with rates $\mu_s = 1$. We saturate the network with $\mathrm{C} = 1000$ tasks, and we simulate up to $T = 10^6$ server steps, and plot the distribution of the delays (in number of server steps). Our numerical experiment in Figure 5 gives average delays (50 and 1950 for fast and slow nodes, respectively) and queue lengths that correspond to the theoretical expected values. It is also important to point out that the average delays are way smaller than the maximum delay experienced in the $T = 10^6$ steps. This further highlights the necessity to switch from analysis that depend on the τ_{max} quantity, to our analysis that only depends on the expected delays.

5 Deep learning experiments

We evaluate FL algos performance on a classic image classification task: CIFAR-10 (Krizhevsky et al., 2009). We consider a non-i.i.d. split of the dataset: each client takes seven classes (out of the ten possible) without replacement. This process introduces heterogeneity among the clients.

We compare different asynchronous methods in terms of CS steps. In all experiments, we track the performance of each algorithm by evaluating the server model against an unseen validation dataset.

We decide to focus on nodes with different exponential service rates as in Nguyen et al. (2022). We build `AsyncSGD` and `Generalized AsyncSGD` codes from scratch. After simulating n clients, we randomly group them into fast or slow nodes. We assume that the clients have different computational speeds, and refer the readers to Appendix H.1 for further details. We have assumed that half of the clients are slow. We compare the classic asynchronous methods `FedBuff` (Nguyen et al., 2022), and `AsyncSGD` (Koloskova et al., 2022). Details about concurrent works implementation can be found in Appendix H.2.

We use the standard data augmentations and normalizations for all methods. All methods are implemented in Pytorch, and experiments are performed on an NVIDIA Tesla-P100 GPU. Standard multiclass cross entropy loss is used for all experiments. All models are fine-tuned with $n = 100$ clients, and a batch of size 128. We have finetuned the learning rate for each method. For `FedBuff` we tried several values for the buffer size, but finally found that the default one $Z = 10$ gives the best performances.

In Figure 6, we compare the performance of a Resnet20 (He et al., 2016) with the CIFAR-10 dataset, which consists of 50000 training images and 10000 test images (in 10 classes). The total number of CS steps is set to 200. Despite the heterogeneity between client datasets, we can achieve good performance on image classification. `FedBuff` has to fill up its buffer before performing an update, slowing down the training process. `AsyncSGD` provides acceptable performance, but we can go further by sampling fast nodes slightly less than the uniform (as suggested in Section 2), and this leads to much better accuracy.

We have additionally tested `Generalized AsyncSGD` on the TinyImageNet classification task (Le and Yang, 2015), with a ResNet18. We compare `Generalized AsyncSGD` with the classic synchronous approach FedAvg (McMahan et al., 2017) and two newer asynchronous methods FedBuff (Nguyen et al., 2022) and FAVANO (Leconte et al., 2023). FAVANO (and the NN quantized QuAFL Zakerinia et al. (2022)) follows a completely different method than we do. There are no queues: Clients are triggered at the CS and either

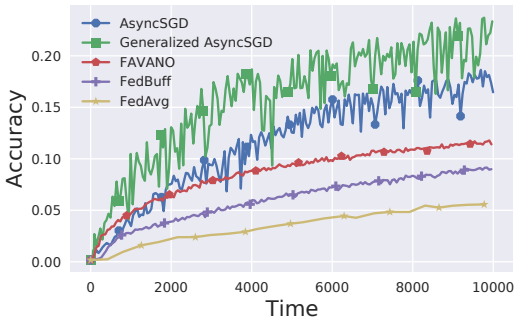


Figure 7: Test accuracy on TinyImageNet dataset with $n = 100$ total nodes.

withhold their results or are interrupted by the CS before the work is completed. In FAVANO, the clients can have a high latency. The update rate of the CS is limited by (slow) clients: The minimum time between two CS updates should be at least as long as the minimum time needed to process a gradient update. TinyImageNet has 200 classes and each class has 500 (RGB) training images, 50 validation images and 50 test images. To train ResNet18, we follow the usual practices for training NNs: we resize the input images to 64×64 and then randomly flip them horizontally during training. During testing, we center-crop them to the appropriate size. The learning rate is set to 0.001 and the total simulated time is set to 1000. Figure 7 illustrates the performance of `Generalized AsyncSGD` in this experimental setup. While the partitioning of the training dataset follows an IID strategy, TinyImageNet provides enough diversity to challenge federated learning algorithms. FedBuff is efficient when the number of stragglers is small. However, FedBuff is sensitive to the fraction of slow clients and may get stuck if the majority of clients in the buffer are more frequently the fast clients: this introduces a bias and few information from slow clients will be taken into account at the CS. FAVANO works better than FedBuff, but the CS updates should not be too small in order to allow slow clients to compute at least one local gradient step. However with AsyncSGD, no constraints are set on the time between two consecutive CS steps: it evolves freely based on the queuing processes. `Generalized AsyncSGD` presents the same advantage, and in addition, samples clients with an optimal scheme. This leads to better performance, even on the challenging TinyImageNet benchmark.

6 Conclusion

In this study, we analyze the convergence of an Asynchronous Federated Learning mechanism in a heterogeneous environment. Through a detailed queuing dynamics analysis, we demonstrate significantly improved convergence rates for our algorithm `Generalized AsyncSGD`, eliminating dependence on the maximum delay $\tau_{\max}$ seen in previous works. Our algorithm enables non-uniform node sampling, enhancing flexibility. Empirical evaluations reveal `Generalized AsyncSGD` superior efficiency over both synchronous and asynchronous state-of-the-art methods in standard CNN training benchmarks for image classification tasks.

Acknowledgement.

Part of the work has been prepared under the auspice of the Lagrange Center for Mathematics and Computing. The work of E.M. has been partially funded by the European Union (ERC-2022-SYG-OCEAN-101071601). Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them. The work of S. Samsonov was supported by the grant for research centers in the field of AI provided by the Analytical Center for the Government of the Russian Federation (ACRF) in accordance with the agreement on the provision of subsidies (identifier of the agreement 000000D730321P5Q0002) and the agreement with HSE University No. 70-2021-00139. The work of M.J. has been conducted with the support of the Chaire DSPI at Ecole Polytechnique.

References

Baccelli, F. and Bremaud, P. (2002). *Elements of Queueing Theory: Palm Martingale Calculus and Stochastic Recurrences*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg.

Beznosikov, A., Samsonov, S., Sheshukova, M., Gasnikov, A., Naumov, A., and Moulines, E. (2023). First order methods with markovian noise: from acceleration to variational inequalities. *arXiv preprint arXiv:2305.15938*.

Chen, Y., Ning, Y., Slawski, M., and Rangwala, H. (2020). Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 15–24. IEEE.

Chen, Z., Liao, W., Hua, K., Lu, C., and Yu, W. (2021). Towards asynchronous federated learning for heterogeneous edge-powered internet of things. *Digital Communications and Networks*, 7(3):317–326.

Fraboni, Y., Vidal, R., Kameni, L., and Lorenzi, M. (2023). A general theory for federated optimization with asynchronous and heterogeneous clients updates. *Journal of Machine Learning Research*, 24(110):1–43.

Halfin, S. and Whitt, W. (1981). Heavy-traffic limits for queues with many exponential servers. *Operations research*, 29(3):567–588.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Islamov, R., Safaryan, M., and Alistarh, D. (2023). Asgrad: A sharp unified analysis of asynchronous-sgd algorithms. *arXiv preprint arXiv:2310.20452*.

Jackson, J. R. (1957). Networks of waiting lines. *Operations research*, 5(4):518–521.

Jackson, R. (1954). Queueing systems with phase type service. *Journal of the Operational Research Society*, 5(4):109–120.

Koloskova, A., Stich, S. U., and Jaggi, M. (2022). Sharper convergence guarantees for asynchronous sgd for distributed and federated learning. *Advances in Neural Information Processing Systems*, 35:17202–17215.

Konečnỳ, J., McMahan, B., and Ramage, D. (2015). Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Lannelongue, L., Grealey, J., and Inouye, M. (2021). Green algorithms: Quantifying the carbon footprint of computation. *Advanced Science*, 8(12):2100707.

Le, Y. and Yang, X. (2015). Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3.

Leconte, L., Nguyen, V. M., and Moulines, E. (2023). Favas: Federated averaging with asynchronous clients. *arXiv preprint arXiv:2305.16099*.

Liu, J., Xu, H., Wang, L., Xu, Y., Qian, C., Huang, J., and Huang, H. (2021). Adaptive asynchronous federated learning in resource-constrained edge computing. *IEEE Transactions on Mobile Computing*.

Makarenko, M., Gasanov, E., Islamov, R., Sadiev, A., and Richtarik, P. (2022). Adaptive compression for communication-efficient distributed training. *arXiv preprint arXiv:2211.00188*.

Malekpourshahraki, M., Desiniotis, C., Radi, M., and Dezfouli, B. (2022). A survey on design challenges of scheduling algorithms for wireless networks. *Int. J. Commun. Netw. Distrib. Syst.*, 28(3):219–265.

Mao, Y., Zhao, Z., Yan, G., Liu, Y., Lan, T., Song, L., and Ding, W. (2022). Communication-efficient federated learning with adaptive quantization. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–26.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR.

Mishchenko, K., Bach, F., Even, M., and Woodworth, B. (2022). Asynchronous sgd beats minibatch sgd under arbitrary delays. *arXiv preprint arXiv:2206.07638*.

Nguyen, J., Malik, K., Zhan, H., Yousefpour, A., Rabbat, M., Malek, M., and Huba, D. (2022). Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics*, pages 3581–3607. PMLR.

Qu, L., Song, S., and Tsui, C.-Y. (2021). Feddq: Communication-efficient federated learning with descending quantization. *arXiv preprint arXiv:2110.02291*.

Serfozo, R. (1999). *Introduction to Stochastic Networks*. Stochastic Modelling and Applied Probability. Springer New York.

Stavrinides, G. L. and Karatza, H. D. (2018). Scheduling techniques for complex workloads in distributed systems. In *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, ICFNDS '18, New York, NY, USA. Association for Computing Machinery.

Toghani, M. T. and Uribe, C. A. (2022). Unbounded gradients in federated leaning with

buffered asynchronous aggregation. *arXiv preprint arXiv:2210.01161*.

Tyurin, A. and Richtárik, P. (2022). Dasha: Distributed nonconvex optimization with communication compression, optimal oracle complexity, and no client synchronization. *arXiv preprint arXiv:2202.01268*.

Tyurin, A. and Richtárik, P. (2023). Optimal time complexities of parallel stochastic optimization methods under a fixed computation model. *arXiv preprint arXiv:2305.12387*.

Van Kreveld, L., Dorsman, J., and Mandjes, M. (2021). Scaling limits for closed product-form queueing networks. *Performance Evaluation*, 151:102220.

Vaswani, S., Bach, F., and Schmidt, M. (2019). Fast and faster convergence of SGD for over-parameterized models and an accelerated perceptron. In *The 22nd international conference on artificial intelligence and statistics*, pages 1195–1204. PMLR.

Wang, J., Liu, Q., Liang, H., Joshi, G., and Poor, H. V. (2020). Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623.

Wang, Q., Yang, Q., He, S., Shui, Z., and Chen, J. (2022). Asyncfeded: Asynchronous federated learning with euclidean distance based adaptive weight aggregation. *arXiv preprint arXiv:2205.13797*.

Wang, S., Zhang, C., Su, D., Wang, L., and Jiang, H. (2021). High-precision binary object detector based on a bsf-xnor convolutional layer. *IEEE Access*, 9:106169–106180.

Xie, C., Koyejo, S., and Gupta, I. (2019). Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*.

Xu, C., Qu, Y., Xiang, Y., and Gao, L. (2021). Asynchronous federated learning on heterogeneous devices: A survey. *arXiv preprint arXiv:2109.04269*.

Zakerinia, H., Talaei, S., Nadiradze, G., and Alistarh, D. (2022). Quafl: Federated averaging can be both asynchronous and communication-efficient. *arXiv preprint arXiv:2206.10032*.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] See Section 3
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] See Appendix A, and the code.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes] See Section 3
   (b) Complete proofs of all theoretical results. [Yes] See Appendix C and Appendix D
   (c) Clear explanations of any assumptions. [Yes] See Section 3

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] See supplemental material
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] See Section 5
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] See Appendix H
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes] See Section 5
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Environmental footprint

In the current context, we estimated the carbon footprint of our experiments to be about 1 kg CO2e (calculated using green-algorithms.org v2.1 Lannelongue et al. (2021)). `Generalized AsyncSGD`'s time and memory complexity is on par with concurrent methods.

B Notations and definitions

In the proof section below we will refer on the following notations. For $0 \leq k \leq T$ consider the filtration $\mathcal{F}_k$ defined as $\mathcal{F}_k = \sigma(\{w_\ell, \ell \leq k, K_m, m < k\})$. We define the *virtual iterates* μ_k as follows:

$$\begin{cases} \mu_0 = w_0, \\ \mu_1 = \mu_0 - \eta \sum_{i \in \mathcal{S}_0} \frac{1}{np_i} \tilde{g}_i(w_0), \\ \mu_{k+1} = \mu_k - \frac{\eta}{np_{K_k}} \tilde{g}_{K_k}(w_k)\,, \quad k \geq 1\,. \end{cases} \tag{4}$$

C Proofs of Section 2

We split the proof of Theorem 1 into several steps. First we bound the quantity of interest $\sum_{k=0}^T \mathbb{E}[\|\nabla f(w_k)\|^2]$ in terms of norms of difference between exact and virtual iterations $\|\mu_k - w_k\|^2$ defined in (4). More precisely, the following statement holds:

Lemma 6. *Assume **A**1 to **A**4 and let the learning rate η satisfy $\eta \leq \frac{n^2}{8L\sum_{i=1}^n \frac{1}{p_i}}$. Then for the iterates $(w_k)_{k\geq 0}$ of `Generalized AsyncSGD` it holds that*

$$\frac{\eta}{4(T+1)} \sum_{k=0}^T \mathbb{E}[\|\nabla f(w_k)\|^2] \leq \frac{f(\mu_0) - \mathbb{E}[f(\mu_{T+1})]}{T+1} + \frac{\eta L^2}{2} \frac{1}{T+1} \sum_{k=0}^T \mathbb{E}[\|\mu_k - w_k\|^2] + \eta^2 L \sum_{i=1}^n \frac{2G^2 + \sigma^2}{n^2 p_i}.$$

Proof. Using the smoothness assumption **A**2 and the definition of μ_{k+1} from (4), we obtain the following descent inequality:

$$\mathbb{E}\left[f\left(\mu_{k+1}\right) | \mathcal{F}_k\right] - f\left(\mu_k\right) \leq -\eta \mathbb{E}\left[\langle \nabla f(\mu_k), \frac{1}{np_{K_k}} \widetilde{g}_{K_k}(w_k)\rangle \,\middle|\, \mathcal{F}_k\right] + \frac{\eta^2 L}{2} \mathbb{E}\left[\|\frac{1}{np_{K_k}} \widetilde{g}_{K_k}(w_k)\|^2 \,\middle|\, \mathcal{F}_k\right].$$

With the unbiasedness property of stochastic gradients and **A**3, we get

$$\begin{aligned} \mathbb{E}\left[f\left(\mu_{k+1}\right) | \mathcal{F}_k\right] - f\left(\mu_k\right) &\leq -\eta \mathbb{E}\left[\langle \nabla f(\mu_k), \frac{1}{np_{K_k}} \nabla f_{K_k}(w_k)\rangle \,\middle|\, \mathcal{F}_k\right] + \eta^2 L \mathbb{E}\left[\|\frac{1}{np_{K_k}} \nabla f_{K_k}(w_k)\|^2 \,\middle|\, \mathcal{F}_k\right] \\ &\quad + \eta^2 L \sigma^2 \sum_{i=1}^n \frac{1}{n^2 p_i} \\ &= -\eta \langle \nabla f(\mu_k), \nabla f(w_k)\rangle + \eta^2 L \mathbb{E}\left[\|\frac{1}{np_{K_k}} \nabla f_{K_k}(w_k)\|^2 \,\middle|\, \mathcal{F}_k\right] + \eta^2 L \sigma^2 \sum_{i=1}^n \frac{1}{n^2 p_i}\,. \end{aligned}$$

In the last equality we used that $\mathbb{E}\left[\langle \nabla f(\mu_k), \frac{1}{np_{K_k}} \nabla f_{K_k}(w_k)\rangle \,\middle|\, \mathcal{F}_k\right] = \langle \nabla f(\mu_k), \nabla f(w_k)\rangle$. Now we introduce a notation

$$\Delta_k = \mathbb{E}\left[f\left(\mu_{k+1}\right) | \mathcal{F}_k\right] - f\left(\mu_k\right)\,.$$

Since $\langle a, b\rangle = 1/2(\|a\|^2 + \|b\|^2 - \|a-b\|^2)$ for any $a, b \in \mathbb{R}^d$, we get that

$$\begin{aligned} \Delta_k &\leq -\frac{\eta}{2}(\|\nabla f(\mu_k)\|^2 + \|\nabla f(w_k)\|^2 - \|\nabla f(w_k) - \nabla f(\mu_k)\|^2) + \eta^2 L \mathbb{E}\left[\|\frac{1}{np_{K_k}} \nabla f_{K_k}(w_k)\|^2 \,\middle|\, \mathcal{F}_k\right] + \eta^2 L \sigma^2 \sum_{i=1}^n \frac{1}{n^2 p_i} \\ &\leq -\frac{\eta}{2}\|\nabla f(w_k)\|^2 + \frac{\eta}{2} L^2 \|\mu_k - w_k\|^2 + \eta^2 L \mathbb{E}\left[\|\frac{1}{np_{K_k}} (\nabla f_{K_k}(w_k) - \nabla f(w_k) + \nabla f(w_k))\|^2 \,\middle|\, \mathcal{F}_k\right] + \eta^2 L \sigma^2 \sum_{i=1}^n \frac{1}{n^2 p_i} \\ &\leq -\frac{\eta}{2}\|\nabla f(w_k)\|^2 + \frac{\eta}{2} L^2 \|\mu_k - w_k\|^2 + 2\eta^2 L (\mathbb{E}\left[\|\frac{1}{np_{K_k}} (\nabla f_{K_k}(w_k) - \nabla f(w_k))\|^2 \,\middle|\, \mathcal{F}_k\right] + \|\nabla f(w_k)\|^2 \sum_{i=1}^n \frac{1}{n^2 p_i}) \\ &\quad + \eta^2 L \sigma^2 \sum_{i=1}^n \frac{1}{n^2 p_i}. \end{aligned}$$

Applying the bounded gradient dissimilarity assumption **A**4, we get

$$\begin{aligned}\Delta_k &\leq -\frac{\eta}{2}\|\nabla f(w_k)\|^2 + \frac{\eta}{2}L^2\|\mu_k - w_k\|^2 + 2\eta^2 L(\sum_{i=1}^n \frac{G^2}{n^2 p_i} + \|\nabla f(w_k)\|^2 \sum_{i=1}^n \frac{1}{n^2 p_i}) + \eta^2 L\sigma^2 \sum_{i=1}^n \frac{1}{n^2 p_i} \\ &\leq (-\frac{\eta}{2} + 2\eta^2 L \sum_{i=1}^n \frac{1}{n^2 p_i})\|\nabla f(w_k)\|^2 + \frac{\eta}{2}L^2\|\mu_k - w_k\|^2 + 2\eta^2 L \sum_{i=1}^n \frac{G^2}{n^2 p_i} + \eta^2 L\sigma^2 \sum_{i=1}^n \frac{1}{n^2 p_i}.\end{aligned}$$

As a consequence by taking $\eta \leq \frac{n^2}{8L\sum_{i=1}^n \frac{1}{p_i}}$ and substituting for Δ_k, we get

$$\frac{\eta}{4}\|\nabla f(w_k)\|^2 \leq f(\mu_k) - \mathbb{E}\left[f(\mu_{k+1}) \,|\, \mathcal{F}_k\right] + \frac{\eta L^2}{2}\|\mu_k - w_k\|^2 + 2\eta^2 L \sum_{i=1}^n \frac{G^2}{n^2 p_i} + \eta^2 L\sigma^2 \sum_{i=1}^n \frac{1}{n^2 p_i}.$$

Now taking sum for $k \in \{0, \dots, T\}$, we get

$$\frac{\eta}{4(T+1)} \sum_{k=0}^T \mathbb{E}[\|\nabla f(w_k)\|^2] \leq \frac{f(\mu_0) - \mathbb{E}[f(\mu_{K+1})]}{T+1} + \frac{\eta L^2}{2}\frac{1}{T+1}\sum_{k=0}^T \mathbb{E}[\|\mu_k - w_k\|^2] + \eta^2 L \sum_{i=1}^n \frac{2G^2 + \sigma^2}{n^2 p_i}.$$

□

In order to apply the result of Lemma 6, one needs to provide an upper bound on the correction term $\mathbb{E}[\|\mu_k - w_k\|^2]$. As explained in Section 2, the virtual iterates deviation from the true $\{w_k\}_{k>0}$ is made of all the gradients computed (on potentially outdated w's) and not applied yet. We can introduce the sets $\{\mathcal{I}_k\}_{k>0}$, as the sets of time and client indexes whose gradients are still on fly at time k. With $\mathcal{S}_0$ being the set of initial active workers from `Generalized AsyncSGD`, there are defined by the recursion:

$$\mathcal{I}_1 = \{(i,0) | i \in \mathcal{S}_0, i \neq J_0\}$$

$$\mathcal{I}_{k+1} = \begin{cases} \mathcal{I}_k & \text{if } I_k = k, \\ \mathcal{I}_k \setminus (J_k, I_k) \cup (K_k, k) & \text{otherwise.} \end{cases}$$

As $\|\mu_k - w_k\|$ represents the norm of gradients, it is easier to introduce the sets $\{\mathcal{G}_k\}_{k>0}$, as the set of gradients *scaled* with their respective weight $\frac{-1}{np_i}$ for each client i, that correspond to the indexes in $\{\mathcal{I}_k\}_{k>0}$:

$$\mathcal{G}_k = \{-\frac{1}{np_i}\tilde{g}_i(w_j) | (i,j) \in \mathcal{I}_k\}.$$

In the following lines, we will show that the sets $\{\mathcal{G}_k\}_{k>0}$ (and as a consequence the sets $\{\mathcal{I}_k\}_{k>0}$) have a constant cardinal: the number of running tasks in `Generalized AsyncSGD` is fixed during the whole optimization process, and only depends on the initialization.

Remark 7. *The number of running tasks is constant, but the number of active nodes is not! If there is a very slow client i, the number of active clients can be reduced to 1: all tasks are currently processed in the queue of client i.*

Lemma 8. *For the sequence $(w_k)_{k\geq 0}$ of updates produced by `Generalized AsyncSGD` and for the sequence of virtual updates $(\mu_k)_{k\geq 0}$ defined in (4), it holds that*

$$\begin{aligned}\mu_1 - w_1 &= -\eta \sum_{i\in\mathcal{S}_0} \mathbb{1}\{i \neq J_0\} \frac{1}{np_i}\tilde{g}_i(w_0)\,, \\ \mu_{k+1} - w_{k+1} &= -\eta \sum_{i\in\mathcal{S}_0} \mathbb{1}\{i \neq J_0\} \frac{1}{np_i}\tilde{g}_i(w_0) + \eta \sum_{r=1}^k (\frac{1}{np_{J_r}}\tilde{g}_{J_r}(w_{I_r}) - \frac{1}{np_{K_r}}\tilde{g}_{K_r}(w_r))\,, \quad k \geq 1\,.\end{aligned}$$

Proof. The proof follows from the definition of recurrence (4). Indeed, first we can note that

$$\begin{aligned}\mu_1 - w_1 &= (w_0 - \eta \sum_{i\in\mathcal{S}_0} \frac{1}{np_i}\tilde{g}_i(w_0)) - (w_0 - \eta\frac{1}{np_{J_0}}\tilde{g}_{J_0}(w_{I_0})) \\ &= (w_0 - \eta \sum_{i\in\mathcal{S}_0} \frac{1}{np_i}\tilde{g}_i(w_0)) - (w_0 - \eta\frac{1}{np_{J_0}}\tilde{g}_{J_0}(w_0)) \\ &= -\eta \sum_{i\in\mathcal{S}_0} \mathbb{1}\{i \neq J_0\} \frac{1}{np_i}\tilde{g}_i(w_0).\end{aligned}$$

Now for a general iteration number k we have:

$$\begin{aligned}
\mu_{k+1} - w_{k+1} &= (\mu_k - \eta \frac{1}{np_{K_k}} \tilde{g}_{K_k}(w_k)) - (w_k - \eta \frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_{I_k})) \\
&= (\mu_k - w_k) + \eta (\frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_{I_k}) - \frac{1}{np_{K_k}} \tilde{g}_{K_k}(w_k)) \\
&= (\mu_1 - w_1) + \sum\nolimits_{r=1}^{k} \eta (\frac{1}{np_{J_r}} \tilde{g}_{J_r}(w_{I_r}) - \frac{1}{np_{K_r}} \tilde{g}_{K_r}(w_r)) \\
&= -\eta \sum\nolimits_{i \in \mathcal{S}_0} \mathbb{1}\{i \neq J_0\} \frac{1}{np_i} \tilde{g}_i(w_0) + \eta \sum\nolimits_{r=1}^{k} (\frac{1}{np_{J_r}} \tilde{g}_{J_r}(w_{I_r}) - \frac{1}{np_{K_r}} \tilde{g}_{K_r}(w_r)).
\end{aligned}$$

□

Lemma 9. *The sets $\{\mathcal{G}_k\}_{k \geq 0}$ have constant cardinal and compile all the gradients in computation at step $k > 0$:*

$$\begin{cases} (i) & |\mathcal{G}_k| = |\mathcal{G}_1| = |\mathcal{S}_0| - 1, \\ (ii) & \mu_k - w_k = \eta \sum_{g \in \mathcal{G}_k} g. \end{cases}$$

Proof. **Step (i)**: We are going to proof the result by induction. Assume $|\mathcal{G}_k| = |\mathcal{S}_0| - 1$ for some k. If $I_k = k$ we can immediately conclude that $|\mathcal{G}_{k+1}| = |\mathcal{G}_k|$. Otherwise, $I_k < k$, hence there exists some $i \in [n]$ such that $-\frac{1}{np_i} \tilde{g}_i(w_{I_k}) \in \mathcal{G}_k$. In particular, client J_k is the client that finishes computation at step k, thus $-\frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_{I_k}) \in \mathcal{G}_k$. As a consequence, $|\mathcal{G}_k \backslash \{-\frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_{I_k})\}| = |\mathcal{S}_0| - 2$. Furthermore, by definition, all gradients in $\mathcal{G}_k$ are taken on models older than k. And by taking $I_k < k$, we obtain $-\frac{1}{np_{K_k}} \tilde{g}_{K_k}(w_k) \notin \mathcal{S}_k \setminus \{-\frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_{I_k})\}$. It concludes $|\mathcal{G}_{k+1}| = |\mathcal{G}_k|$.

Step (ii): We also prove it by induction. It is valid for $k = 1$. Now assume $\mu_k - w_k = \sum_{g \in \mathcal{G}_k} g$, for some $k > 1$.

$$\begin{aligned}
\mu_{k+1} - w_{k+1} &= (\mu_k - w_k) + \frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_{I_k}) - \frac{1}{np_{K_k}} \tilde{g}_{K_k}(w_k) \\
&= (\mu_k - w_k) + (\mathbb{1}_{\{0\}}(I_k) \frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_0) + \cdots + \mathbb{1}_{\{k\}}(I_k) \frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_k)) - \frac{1}{np_{K_k}} \tilde{g}_{K_k}(w_k) \\
&= (\mu_k - w_k) + \mathbb{1}_{\{0\}}(I_k) \frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_0) + \cdots + \mathbb{1}_{\{k-1\}}(I_k) \frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_{k-1}) \\
&\quad + (\mathbb{1}_{\{k\}}(I_k) \frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_k) - \frac{1}{np_{K_k}} \tilde{g}_{K_k}(w_k)).
\end{aligned}$$

By induction, we have:

$$\begin{aligned}
\mu_{k+1} - w_{k+1} &= \sum\nolimits_{g \in \mathcal{G}_k} g + \mathbb{1}_{\{0\}}(I_k) \frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_0) + \cdots + \mathbb{1}_{\{k-1\}}(I_k) \frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_{k-1}) \\
&\quad + (\mathbb{1}_{\{k\}}(I_k) \frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_k) - \frac{1}{np_{K_k}} \tilde{g}_{K_k}(w_k)).
\end{aligned}$$

If $I_k = k$ we have $J_k = K_k$: some client contributes instantaneously. This results in:

$$\begin{aligned}
\mu_{k+1} - w_{k+1} &= \sum\nolimits_{g \in \mathcal{G}_k} g + \underbrace{\mathbb{1}_{\{0\}}(I_k) \frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_0) + \cdots + \mathbb{1}_{\{k-1\}}(I_k) \frac{1}{np_{J_k}} \tilde{g}_{J_k}(w_{k-1})}_{=0} \\
&\quad + \underbrace{(\mathbb{1}_{\{k\}}(I_k) \tilde{g}_{J_k}(w_k) - \frac{1}{np_{K_k}} \tilde{g}_{K_k}(w_k))}_{=0} \\
&= \sum\nolimits_{g \in \mathcal{G}_{k+1}} g.
\end{aligned}$$

Same idea as in Step (i) allows us to conclude $\mu_{k+1} - w_{k+1} = \sum_{g \in \mathcal{G}_{k+1}} g$ when $I_k < k$. □

Lemma 10. *For the sequence $(w_k)_{k\geq 0}$ of updates produced by* `Generalized AsyncSGD` *and for the sequence of virtual updates $(\mu_k)_{k\geq 0}$ defined in (4), it holds that*

$$\begin{aligned}\frac{1}{T+1}\sum_{k=0}^{T}\mathbb{E}[\|\mu_k - w_k\|^2] \leq 2\eta^2\,\mathrm{C}\sum_{i=1}^{n}\frac{\mathrm{m}_{i,0}^T}{(n^2p_i^2)(T+1)}(2G^2+\sigma^2)\\ + 4\eta^2\,\mathrm{C}\,\frac{\mathbb{E}[\|\nabla f(w_0)\|^2]\,\mathrm{m}_0^T}{(n^2p_i^2)(T+1)}\\ + 4\eta^2\,\mathrm{C}\sum_{k=1}^{T}\frac{\mathrm{m}_k^T}{(n^2p_i^2)(T+1)}\mathbb{E}[\|\nabla f(w_k)\|^2]\\ + 2\eta^2\,\mathrm{C}\sum_{i=1}^{n}\frac{\sum_{k=1}^{K}\mathrm{m}_{i,k}^T}{(n^2p_i^2)(T+1)}(2G^2+\sigma^2).\end{aligned}$$

Proof. Using the statement of Lemma 9, we bound the expected value of the correction term as follows:

$$\mathbb{E}[\|\mu_k - w_k\|^2] = \eta^2\mathbb{E}[\|\sum_{g\in\mathcal{G}_k} g\|^2] \leq \eta^2\mathbb{E}[|\mathcal{G}_k|\sum_{g\in\mathcal{G}_k}\|g\|^2] \leq \eta^2\mathbb{E}[\mathrm{C}\sum_{g\in\mathcal{G}_k}\|g\|^2]. \tag{5}$$

As the cardinal of sets $|\mathcal{G}_k| := \mathrm{C}$ is constant among iterations, and in particular it is independent from g's, we can simplify the form above.

We also introduce the set U_k:

$$U_k = \{i \in \{1,\dots,n\} | X_{i,k} > 0\}.$$

Hence, from (5) and **A**3, we get

$$\begin{aligned}\mathbb{E}[\|\mu_k - w_k\|^2] &\leq \eta^2\,\mathrm{C}\,\mathbb{E}[\sum_{(i,j)\in\mathcal{I}_k\cup\{U_k\times\{0\}\}}\frac{1}{n^2p_i^2}2\|\nabla f_i(w_j)\|^2 + 2\sigma^2]\\ &\leq \eta^2\,\mathrm{C}\,\mathbb{E}[\sum_{i=1}^{n}\frac{1}{n^2p_i^2}\underbrace{\mathbb{1}_{U_k\cap\mathcal{S}_0}(i)(4G^2+4\|\nabla f(w_0)\|^2+2\sigma^2)}_{\text{gradients on initial model}}\\ &\quad+\sum_{j=1}^{k}\mathbb{1}_{\mathcal{I}_k}\big((i,j)\big)(4G^2+4\|\nabla f(w_j)\|^2+2\sigma^2)]\end{aligned}$$

Now we average over T iterations:

$$\begin{aligned}\frac{1}{T+1}\sum_{k=0}^{T}\mathbb{E}[\|\mu_k - w_k\|^2] \leq 2\eta^2\,\mathrm{C}\sum_{i=1}^{n}(\sum_{k=0}^{T}\frac{\mathbb{P}(i\in U_k\cap\mathcal{S}_0)}{T+1})(\frac{2G^2+\sigma^2}{n^2p_i^2})\\ + 4\eta^2\,\mathrm{C}\,\mathbb{E}[\|\nabla f(w_0)\|^2]\sum_{i=1}^{n}\frac{1}{n^2p_i^2}(\sum_{k=0}^{T}\frac{\mathbb{P}(i\in U_k\cap\mathcal{S}_0)}{T+1})\\ + \frac{4}{T+1}\eta^2\,\mathrm{C}\sum_{i=1}^{n}\mathbb{E}[\sum_{k=1}^{T}\frac{1}{n^2p_i^2}\sum_{j=1}^{k}(\mathbb{1}_{\mathcal{I}_k}\big((i,j)\big)\|\nabla f(w_j)\|^2)]\\ + \frac{4G^2+2\sigma^2}{T+1}\eta^2\,\mathrm{C}\sum_{i=1}^{n}\mathbb{E}[\sum_{k=1}^{T}\frac{1}{n^2p_i^2}\sum_{j=1}^{k}\mathbb{1}_{\mathcal{I}_k}((i,j))].\end{aligned}$$

We rearrange the last 2 terms:

$$\begin{aligned}\frac{1}{T+1}\sum_{k=0}^{T}\mathbb{E}[\|\mu_k - w_k\|^2] \leq 2\eta^2\,\mathrm{C}\sum_{i=1}^{n}\frac{\sum_{k=1}^{T}\mathbb{P}(i\in U_k\cap\mathcal{S}_0)}{T+1}(\frac{2G^2+\sigma^2}{n^2p_i^2})\\ + 4\eta^2\,\mathrm{C}\,\mathbb{E}[\|\nabla f(w_0)\|^2]\sum_{i=1}^{n}\frac{1}{n^2p_i^2}\frac{\sum_{k=1}^{T}\mathbb{P}(i\in u_k\cap\mathcal{S}_0)}{T+1}\\ + 4\eta^2\,\mathrm{C}\sum_{k=1}^{T}\mathbb{E}[\frac{\sum_{i=1}^{n}\frac{1}{n^2p_i^2}\sum_{r=k}^{T}\mathbb{1}_{\mathcal{I}_r}((i,k))}{T+1}\|\nabla f(w_k)\|^2]\\ + 2\eta^2\,\mathrm{C}\sum_{k=1}^{T}\mathbb{E}[\frac{\sum_{i=1}^{n}\frac{1}{n^2p_i^2}\sum_{r=k}^{T}\mathbb{1}_{\mathcal{I}_r}((i,k))}{T+1}(2G^2+\sigma^2)].\end{aligned}$$

We can simplify the bounds with the following identity:

$$\sum_{r=k}^{T} \mathbb{1}_{\mathcal{I}_r}((i,k)) = \mathbb{1}_{\{i\}}(K_{k+1}) \sum_{r=k}^{T} \mathbb{1}_{(\sum_{l=k}^{r} \mathbb{1}_{J_l=i}) < X_{i,k}} = \mathrm{M}_{i,k}^{T}, \text{for } k > 0.$$

And with a slight abuse of notation, we take: $\mathrm{M}_{i,0}^{T} = \sum_{k=1}^{T} \mathbb{1}_{U_k \cap \mathcal{S}_0}(i)$. Combining the above bounds, we obtain

$$\begin{aligned}
\frac{1}{T+1}\sum_{k=0}^{T}\mathbb{E}[\|\mu_k - w_k\|^2] \leq 2\eta^2\,\mathrm{C}\sum_{i=1}^{n}\frac{\mathbb{E}[\mathrm{M}_{i,0}^{T}]}{T+1}\Big(\frac{2G^2+\sigma^2}{n^2p_i^2}\Big) \\
+ 4\eta^2\,\mathrm{C}\,\mathbb{E}[\|\nabla f(w_0)\|^2]\sum_{i=1}^{n}\frac{1}{n^2p_i^2}\frac{\mathbb{E}[\mathrm{M}_{i,0}^{T}]}{T+1} \\
+ 4\eta^2\,\mathrm{C}\sum_{k=1}^{T}\frac{\sum_{i=1}^{n}\frac{1}{n^2p_i^2}\mathbb{E}[\mathrm{M}_{i,k}^{T}]}{T+1}\mathbb{E}[\|\nabla f(w_k)\|^2] \\
+ 2\eta^2\,\mathrm{C}\sum_{i=1}^{n}\frac{\sum_{k=1}^{T}\mathbb{E}[\mathrm{M}_{i,k}^{T}]}{T+1}\Big(\frac{2G^2+\sigma^2}{n^2p_i^2}\Big)\,,
\end{aligned}$$

and the statement follows using the definition m_k^T. □

C.1 Proof of Theorem 1

We apply the bound Lemma 6 and use Lemma 10 to control the correction term $\frac{1}{T+1}\sum_{k=0}^{T}\mathbb{E}[\|\mu_k - w_k\|^2]$. Hence, we get

$$\begin{aligned}
\frac{1}{4(T+1)}\sum_{k=0}^{T}\mathbb{E}[\|\nabla f(w_k)\|^2] &\leq \frac{\mathbb{E}[f(\mu_0)-f(\mu_{T+1})]}{\eta(T+1)} + \frac{L^2}{2(T+1)}\sum_{k=0}^{T}\mathbb{E}[\|\mu_k - w_k\|^2] + \eta L\sum_{i}^{n}\frac{2G^2+\sigma^2}{n^2p_i} \\
&\leq \frac{\mathbb{E}[f(\mu_0)-f(\mu_{T+1})]}{\eta(T+1)} + \eta L\sum_{i}^{n}\frac{2G^2+\sigma^2}{n^2p_i} \\
&\quad + L^2\eta^2\,\mathrm{C}\left(\sum_{i=1}^{n}\frac{\sum_{k=0}^{T}\mathrm{m}_{i,k}^{T}}{n^2p_i^2(T+1)}(2G^2+\sigma^2) + \frac{2\mathbb{E}[\|\nabla f(w_0)\|^2]\,\mathrm{m}_0^T}{T+1}\right) \\
&\quad + \frac{L^2\eta^2\,\mathrm{C}}{T+1}\sum_{k=1}^{T}2\,\mathrm{m}_k^T\,\mathbb{E}[\|\nabla f(w_k)\|^2]\,.
\end{aligned}$$

Hence we have:

$$\begin{aligned}
\frac{1}{T+1}\sum_{k=0}^{T}(1/4 - 2\,\mathrm{m}_k^T\,L^2\eta^2\,\mathrm{C})\mathbb{E}[\|\nabla f(w_k)\|^2] \leq \frac{\mathbb{E}[f(\mu_0)-f(\mu_{T+1})]}{\eta(T+1)} + \eta L\sum_{i}^{n}\frac{2G^2+\sigma^2}{n^2p_i} \\
+ L^2\eta^2\,\mathrm{C}\sum_{i=1}^{n}\frac{\sum_{k=0}^{T}\mathrm{m}_{i,k}^{T}}{n^2p_i^2(T+1)}(2G^2+\sigma^2)\,.
\end{aligned}$$

Now we impose the step size condition

$$\eta \leq \sqrt{\frac{1}{16L^2\,\mathrm{C}\max_{k\in\{1,\ldots,T\}}\mathrm{m}_k^T}}\,,$$

which enable us to conclude that

$$\begin{aligned}
\frac{1}{8(T+1)}\sum_{k=0}^{T}\mathbb{E}[\|\nabla f(w_k)\|^2] \leq \frac{\mathbb{E}[f(\mu_0)-f(\mu_{T+1})]}{\eta(T+1)} + \eta L\sum_{i}^{n}\frac{2G^2+\sigma^2}{n^2p_i} \\
+ L^2\eta^2\,\mathrm{C}\sum_{i=1}^{n}\frac{\sum_{k=0}^{T}\mathrm{m}_{i,k}^{T}}{n^2p_i^2(T+1)}(2G^2+\sigma^2)\,,
\end{aligned}$$

and the statement follows.

C.2 Influence of the strong growth condition

The assumption **A**3 can be generalized to the *strong growth condition* Vaswani et al. (2019):

$$\mathbb{E}[\|\widetilde{g}_i(x) - \nabla f_i(x)\|^2] \leq \sigma^2 + \rho^2 \|\nabla f_i(x)\|^2 .$$

All previous derivations are impacted by a factor ρ^2. But the proofs remain the same. In particular, the quantity Δ_k from Lemma 6 can be bounded as:

$$\Delta_k \leq (-\frac{\eta}{2} + 2\eta^2 L \sum_{i=1}^n \frac{1+\rho^2}{n^2 p_i})\|\nabla f(w_k)\|^2 + \frac{\eta}{2} L^2 \|\mu_k - w_k\|^2 + 2\eta^2 L \sum_{i=1}^n \frac{(1+\rho^2)G^2}{n^2 p_i} + \eta^2 L \sigma^2 \sum_{i=1}^n \frac{1}{n^2 p_i}.$$

This slightly change the condition on the step size: $\eta \leq \frac{n^2}{8L \sum_{i=1}^n \frac{1+\rho^2}{p_i}}$. The rest of the proof is similarly impacted.

We now impose the additional step size condition:

$$\eta \leq \sqrt{\frac{1}{(1+\rho^2) 16 L^2 \, \mathrm{C} \max_{k\in\{1,\dots,T\}} \mathrm{m}_k^T}} ,$$

which enable us to conclude that

$$\begin{aligned}\frac{1}{8(T+1)} \sum_{k=0}^T \mathbb{E}[\|\nabla f(w_k)\|^2] \leq \frac{\mathbb{E}[f(\mu_0) - f(\mu_{T+1})]}{\eta(T+1)} + \eta L \sum_i^n \frac{2(1+\rho^2)G^2 + \sigma^2}{n^2 p_i} \\ + L^2 \eta^2 \, \mathrm{C} \sum_{i=1}^n \frac{\sum_{k=0}^T \mathrm{m}_{i,k}^T}{n^2 p_i^2 (T+1)} (2(1+\rho^2)G^2 + \sigma^2) .\end{aligned}$$

D Proofs of Section 4

D.1 Proof of Proposition 2

Given the assumptions, it is straightforward to check that the dynamics are those of the Markov process with the given generator.

Then the result follows from classical results in queuing theory: the Markov process corresponds to a Jackson quasi-reversible network with an explicit stationary distribution, see for instance Theorem 1.12 in Serfozo (1999).

Before continuing, we need a fundamental property of closed Jackson network which is the arrival Theorem also called MUSTA in the literature:

Theorem 11 (Arrival Theorem, Prop 4.35 in Serfozo (1999))**.** *Suppose the system is stationary. Upon arrival to a given node (i.e., just before waiting or being served in the queue), a task sees the network according to the distribution π_{C-1}, i.e.,*

$$\mathbb{P}(X_{\tau_{i,1}^-} = x) = \pi_{C-1}(x).$$

For the link with Palm probabilities, we also refer to Example 3.3.4. in Baccelli and Bremaud (2002).

Now, define S_i the first sojourn time on node i, i.e.,

$$S_i = \inf\{t \geq \tau_{i,1} | D_i(t) = X_i(\tau_{i,1}^-) + 1\}$$

Recall that, $\mathbb{E}^C$ corresponds to the stationary average for a system with C tasks (in particular the process X_t follows π_C for all times $t \geq 0$). We denote in turn by $\mathbb{E}^{C-1}$ the Palm probability associated to the event of an arrival at a given node (say i) and corresponding informally to conditioning to $\tau_{i,1} = 0$. Using the Arrival Theorem, the distribution at time 0 at node i corresponds in this case to π_{C-1}, the dynamics under the Palm probabilities being unchanged (seeBaccelli and Bremaud (2002)).

D.2 Proof of Proposition 3

Assuming the system is stationary and using the definition of $\mathrm{m}_{i,k}^T$ we have that for any k,

$$\mathrm{m}_{i,k}^T = \mathrm{m}_i^T = \mathbb{E}^{\mathrm{C}}\Big[\Big(\sum\nolimits_n \mathbb{1}(\tau_{i,1} \le T_n \le S_i)\Big) \wedge T\Big].$$

Using the monotone convergence Theorem,

$$\lim_{T\to\infty} \mathrm{m}_{i,k}^T = m_i = \mathbb{E}^{\mathrm{C}}\Big[\sum\nolimits_n \mathbb{1}(\tau_{i,1} \le T_n \le S_i)\Big].$$

We then use the arrival Theorem (11) for closed Jackson network and using the Palm probability, we can write that

$$\mathbb{E}^{\mathrm{C}}[\textstyle\sum_n \mathbb{1}(\tau_{i,1} \le T_n \le S_i)] = \mathbb{E}^{\mathrm{C}-1}[\sum_n \mathbb{1}(0 \le T_n \le S_i)].$$

Then by using the stochastic intensity formula (see Definition 1.8.10 and Example 1.8.3. in Baccelli and Bremaud (2002)):

$$\mathbb{E}^{\mathrm{C}-1}[\textstyle\sum_n \mathbb{1}(0 \le T_n \le S_i)] = \mathbb{E}^{\mathrm{C}-1}\Big[\int_0^{S_i} \sum_{j=1}^n \mu_j \mathbb{1}(X_j(s) > 0)\, ds\Big].$$

D.3 Computation of the constant Γ

$$\Gamma(c) = \frac{\mathbb{E}[X\mathbb{1}_{X+Y\le c}]}{\mathbb{E}[\mathbb{1}_{X+Y\le c}]},$$

with X an Exp(1) and Y an Erlang(F,1) independent from each other. By integrating by parts:

$$\begin{aligned}
\Gamma &= \frac{\int_0^\infty \int_0^{c-y} x e^{-x} dx \mathbb{1}(y \le c) dP_Y(y)}{\mathbb{P}(X+Y\le c)}, \\
&= \frac{\int_0^\infty (-(c-y)e^{-c+y} + \int_x 1_{x\le c-y}\mathbb{1}(y\le c) dP_X(x) dP_Y(y)}{\mathbb{P}(X+Y\le c)}, \\
&= \frac{\int_0^\infty -(c-y)e^{-c+y}\mathbb{1}(y\le c)\frac{y^{F-1}}{(F-1)!}e^{-y}dy}{\mathbb{P}(X+Y\le c)} + 1, \\
&= \frac{e^{-c}(-c^{F+1}/F! + Fc^{F+1}/(F+1)!)}{\mathbb{P}(X+Y\le c)} + 1, \\
&= \frac{-e^{-c}c^{F+1}/(F+1)! + 1 - \sum_{k=1}^F e^{-c}c^k/(k)!}{1-\sum_{k=1}^F e^{-c}c^k/(k)!}, \\
&= \frac{\mathbb{P}(\sum_{i=1}^{F+2} E_i \le c)}{\mathbb{P}(\sum_{i=1}^{F+1} E_i \le c)},
\end{aligned}$$

D.4 Proof of Proposition 5

First note that:

$$\mathrm{m}_i(\iota) = \mathbb{E}^{\mathrm{C}-1}\Big[\int_0^{S_i} \sum\nolimits_{j=1}^n \mu_j \mathbb{1}(X_j^\iota(s) > 0)\, ds\Big] \le \lambda \mathbb{E}^{\mathrm{C}-1}[S_i].$$

Then it follows from the FIFO representation that

$$\mathbb{E}^{\mathrm{C}-1}(S_i) = \frac{1}{\mu_i}(\mathbb{E}^{\mathrm{C}-1}[X_i^\iota]) + 1)$$

which implies the claim.

E Upper bounds simulations

E.1 Illustration of $G(\mathbf{p}, \eta)$ before optimization

We decide to simulate $n = 100$ nodes with $\mathrm{C} = 10$ initial tasks. Nodes can only be fast (sampled with $p_i = p$) or slow (sampled with $p_i = \frac{2}{n} - p$), and are evenly distributed. We estimate the values of $\mathrm{m}_{i,k}^T$ through Monte-Carlo and compute the upper bounds given in Theorem 1. All others constants are kept unitary to ease the computation. In Figure 8, we plot the value of the previously mentioned upper-bound with respect to the step size η, for

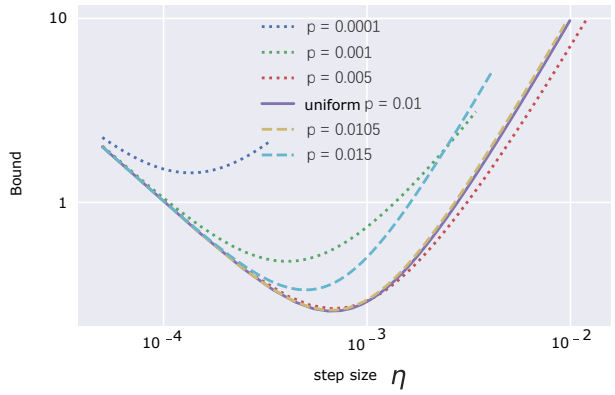


Figure 8: Variation of the non-convex upper-bound with respect to the step size η, for $K = 10^4$ server steps and different values of sampling p. The maximum step size value is different for each case and equal to $\sqrt{\frac{1}{8L^2\,\mathrm{C}\,max\,\mathrm{m}_k^T}}$.

several sampling probabilities of fast node p. When the step size considered is small, all sampling strategies are equivalent. Whereas for large value of η, sampling around the uniform one is a good strategy. Large value of p, close to the limit $\frac{2}{n}$, hinders the bound because it increases the delays for slow nodes by sampling fast nodes quite often. In Figure 2 and Figure 3, we define grids of 50 values of p around the uniform one, and for each p we compute the exact optimal step size by solving the cube roots. The optimal values of the sampling p on the grid, and of the optimal step size are further used to compute the optimal bound and compare it against the one obtained with uniform sampling.

E.2 Bounds w.r.t physical time

We want here to focus on the relative improvements of the upper bounds when time rather than CS steps is considered as fixed. Indeed, when we determine complexity in terms of number of communications, we don't take into account the time intervals between two successive arrivals at the central server. In particular, the results from Section 2 propose to sample more frequently slow nodes. But this results into an increased waiting time between two consecutive server steps. As a consequence, in this section, we choose a fixed unit of time $U = 1000$ and optimize the bounds for $T = \lambda(\mathbf{p}) \cdot U$ server steps, where $\lambda(\mathbf{p})$ is the average network speed.

We choose the sampling probabilities $\mathbf{p}$ and the step size η by solving the constrained optimization problem $\min_{\mathbf{p},\eta} G(\mathbf{p}, \eta)$ as a function of $\eta \leq \eta_{\max}(\mathbf{p})$, where

$$G(\mathbf{p}, \eta) = \frac{A}{\eta(\lambda(\mathbf{p})U + 1)} + \frac{\eta L B}{n} \sum_{i=1}^n \frac{1}{np_i} + \frac{\eta^2 L^2 B\,\mathrm{C}}{n} \sum_{i=1}^n \frac{\sum_{k=0}^{\lambda(\mathbf{p})U} \mathrm{m}_{i,k}^T}{np_i^2(\lambda(\mathbf{p})U + 1)}\,,$$

and where $A = \mathbb{E}[f(\mu_0) - f(\mu_{T+1})]$. In Figure 9 we run the same simulation as in Section 2, for a fixed amount of time U. Taking into account a fixed amount of time U instead of CS epochs T also favours our approach. The experimental results suggest to sample less fast nodes. It reduces delays (in number of steps), but increases the average time spent between two consecutive server steps. This trade-off is key for optimizing the bounds. When the concurrency is small (w.r.t. n), uniform sampling appears as the best strategy. However, by taking $p = 8.5 \cdot 10^{-3}$, for full concurrency ($\mathrm{C} = n$), the bound can be reduced by 40%.

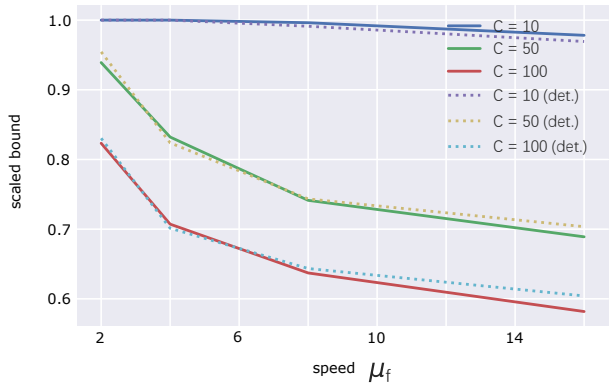


Figure 9: Relative improvements of the upper bounds as a function of the speed for different concurrency levels.

F 2 clusters under saturation

F.1 Example with 2 saturated clusters

In Section 4, we propose a study of the delays and queue lengths when the number of task goes to infinity with a rate controlled by some $\iota > 0$. In particular, we introduce the scaled intensities for slow and fast nodes as:

$$\begin{cases} \gamma_s(\iota) = \frac{\max_{i\in[1,n]}(\theta_i)}{\theta_s} = \frac{\theta_s}{\theta_s} = 1 \\ \gamma_f(\iota) = \frac{\max_{i\in[1,n]}(\theta_i)}{\theta_f} = \frac{\theta_s}{\theta_f} = 1 + \underbrace{c_f \cdot \iota^{\alpha-1}}_{\text{deviation from slow speed}} \end{cases}$$

Note $c_f > 0$ and $\alpha \leq 1$ are parameters we are free to choose to match the number of tasks in the network. In particular, the total number of tasks also scales as follows: $\beta l^{1-\alpha} = \mathrm{C}+1$. Thanks to Proposition 5, we can bound the number of server steps when a task arrive and quits some node i as:

$$\lim_{\iota\to\infty} \iota^{\alpha-1}\,\mathrm{m}_i(\iota) \leq \lim_{\iota\to\infty} \frac{\lambda}{\mu_i}(\iota^{\alpha-1}\mathbb{E}[X_i^\iota] + 1),$$

where $\lambda = n_f\mu_s + (n-n_f)\mu_s$. Hence,

$$\lim_{\iota\to\infty} \iota^{\alpha-1}\,\mathrm{m}_i(\iota) \leq \frac{n_f\mu_s + (n-n_f)\mu_s}{\mu_i}(\frac{1}{c_f}\Gamma(c_f\beta) + 1)$$

We will further assume $n_f = \frac{n}{2}$, and $p_i = \frac{1}{n}$. Under this setting, we have $\Gamma(c_F\beta) \simeq 1$ and

$$\begin{aligned} \frac{\iota^{1-\alpha}}{c_f}\Gamma(c_f\beta) &= \frac{1}{\gamma_f(\iota) - 1} \\ &= \frac{1}{\frac{\theta_s}{\theta_f} - 1} \\ &= \frac{1}{\frac{\mu_f p_s}{\mu_s p_f} - 1} \\ &= \frac{1}{\frac{\mu_f}{\mu_s} - 1}. \end{aligned}$$

For fast nodes, the delay can be simplified as:

$$\lim_{\iota\to\infty} \mathrm{m}_i(\iota) \leq \frac{n}{2}\frac{\mu_s + \mu_f}{\mu_f}\frac{1}{\frac{\mu_f}{\mu_s} - 1}.$$

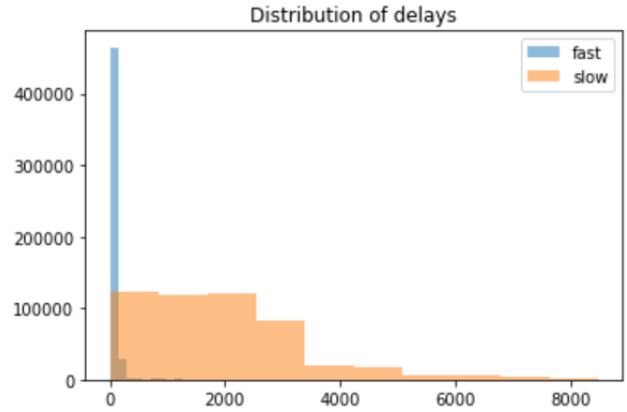
For slow nodes, this simplifies as:

$$\lim_{\iota\to\infty} \mathrm{m}_i(\iota) \leq \frac{n}{2}\frac{\mu_s + \mu_f}{\mu_s}(\frac{2}{n}\,\mathrm{C} - \frac{1}{\frac{\mu_f}{\mu_s} - 1}).$$

In the following we consider $n = 10$ clients, split in two clusters of same size: fast nodes with rate $\mu_f = 1.2$, and slow nodes with rates $\mu_s = 1$. We simulate up to $T = 10^6$ server steps, and plot the distribution of the delays (in number of server steps). We saturate the network with $\mathrm{C} = 1000$ tasks. Hence we can estimate the following:

$$\begin{cases} \lim_{\iota\to\infty} \mathrm{m}_i(\iota) \leq \frac{n}{\frac{\mu_f}{\mu_s}-1} \simeq 5n, \quad \forall i \in [1, n_f], \\ \lim_{\iota\to\infty} \mathrm{m}_i(\iota) \leq (\frac{2\mathrm{C}}{n} - \frac{1}{\frac{\mu_f}{\mu_s}-1})n \simeq 195n, \quad \forall i \in [n_f+1, n]. \end{cases}$$

All delays bounds estimations have a closed form in the 2-cluster saturated regime: they only depend on the number of tasks in the network C, on the number of nodes n, and on the intensity of nodes μ_f, μ_s.




```
avg delay fast =  59.17222612780303
avg delay slow =  1938.58848847666
avg queue fast 5.926313126313127
avg queue slow 194.07368687368688
```

Figure 10: Histogram of fast and slow delays (in number of server steps) for a uniform sampling scheme.

Our numerical experiment in Figure 5 gives an average delay of $59 \sim 5n$ for fast nodes. The average delay for slow nodes reaches the value $1938 \simeq 195n$. And the queue lengths also correspond to the expected values. It is also important to point out that the average delays are way smaller than the maximum delay experienced in the $K = 10^6$ steps. This further highlights the necessity to switch from analysis that depend on the τ_{max} quantity, to our analysis that only depends on the expected delays.

F.2 Optimal sampling strategy under saturation

In the previous paragraph we kept the sampling probability p_i uniform. The previous computation gives

$$\begin{aligned} \frac{l^{1-\alpha}}{c_f}\Gamma(c_f\beta) &= \frac{1}{\gamma_f^l - 1} \\ &= \frac{1}{\frac{\theta_s}{\theta_f} - 1} \\ &= \frac{1}{\frac{\mu_f p_s}{\mu_s p_f} - 1}. \end{aligned}$$

Sticking to the previous assumptions, we want to minimize the quantity

$$G(\mathbf{p}, \eta) = \frac{A}{\eta(T+1)} + \frac{\eta L B}{n}\Big(\sum_{i=1}^{n_f} \frac{1}{np} + \sum_{i=n_f+1}^{n} \frac{1}{n(\frac{1}{n-n_f} - p\frac{n_f}{n-n_f})^{-1}}\Big) + \frac{\eta^2 L^2 B\,\mathrm{C}}{n}\Big(\sum_{i=1}^{n_f} \frac{m_f}{np^2} + \sum_{i=n_f+1}^{n} \frac{m_s}{n(\frac{1}{n-n_f} - p\frac{n_f}{n-n_f})^{-2}}\Big),$$

This is equivalent to minimizing:

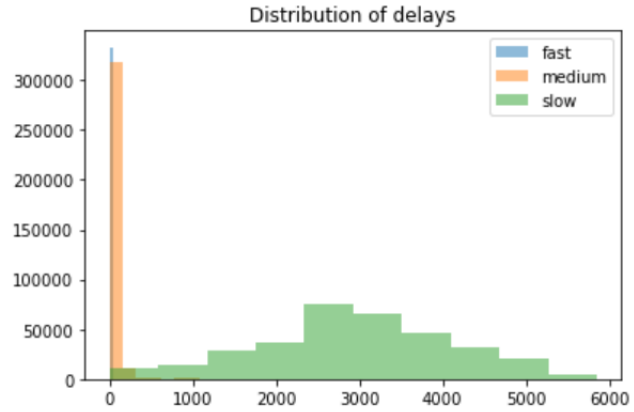
$$G(\mathbf{p},\eta) = \frac{A}{\eta(T+1)} + \frac{\eta LB}{n}\Big(\sum_{i=1}^{n_f}\frac{1}{np} + \sum_{i=n_f+1}^{n}\frac{1}{n(\frac{1}{n-n_f} - p\frac{n_f}{n-n_f})^{-1}}\Big) + \frac{\eta^2L^2B\,\mathrm{C}}{n}\Big(\sum_{i=1}^{n_f}\frac{\frac{n_f\mu_s+(n-n_f)\mu_s}{\mu_f}\frac{1}{\frac{\mu_f p_s}{\mu_s p_f}-1}}{np^2} + \sum_{i=n_f+1}^{n}\frac{\frac{n_f\mu_s+(n-n_f)\mu_s}{\mu_s}(\frac{\mathrm{C}}{n-n_f} - \frac{n_f}{n-n_f}\frac{1}{\frac{\mu_f p_s}{\mu_s p_f}-1})}{n(\frac{1}{n-n_f} - p\frac{n_f}{n-n_f})^{-2}}\Big),$$

Hence we want to find $(\mathbf{p},\eta)$ that minimize the following:

$$G(\mathbf{p},\eta) = \frac{A}{\eta(T+1)} + \frac{\eta LB}{n}\Big(\sum_{i=1}^{n_f}\frac{1}{np} + \sum_{i=n_f+1}^{n}\frac{1}{n(\frac{1}{n-n_f} - p\frac{n_f}{n-n_f})^{-1}}\Big) + \frac{\eta^2L^2B\,\mathrm{C}}{n}\Big(\sum_{i=1}^{n_f}\frac{\frac{n_f\mu_s+(n-n_f)\mu_s}{\mu_f}\frac{1}{\frac{\mu_f}{\mu_s}(\frac{1}{p(n-n_f)}-\frac{n_f}{n-n_f})-1}}{np^2} + \sum_{i=n_f+1}^{n}\frac{\frac{n_f\mu_s+(n-n_f)\mu_s}{\mu_s}(\frac{\mathrm{C}}{n-n_f} - \frac{n_f}{n-n_f}\frac{1}{\frac{\mu_f}{\mu_s}(\frac{1}{p(n-n_f)}-\frac{n_f}{n-n_f})-1})}{n(\frac{1}{n-n_f} - p\frac{n_f}{n-n_f})^{-2}}\Big),$$

The uniform sampling strategy ($p = \frac{1}{n} = 0.1$) and higher probability values, give larger average delays.But very small sampling probabilities lead to a sharp increase in the delays. It is not easy to find a closed form formula of the optimal probability value $\mathbf{p}$. Our simulations suggest an optimal value of $p = 7.5 \cdot 10^{-3}$.

In Figure 11, we have run the same simulations as in Figure 5, except that we do not sample nodes uniformly at random. Instead, we sample fast nodes with a probability $p = 7.5 \cdot 10^{-3}$, and slow nodes with a probability $\frac{2}{n} - p$. Our simulations shows the average delay is divided by 10 and 2, for fast and slow nodes respectively.




```
avg delay fast =  1.3453161915498364
avg delay medium =  55.02906515784922
avg delay slow =  2935.6114257955987
avg queue fast 0.26063426063426065
avg queue medium 6.107726107726108
avg queue slow 326.96497296497296
```

Figure 11: Histogram of fast and slow delays (in number of server steps) for an optimal sampling strategy.

G 3 clusters scaling regime under saturation

We consider three clusters of nodes of size n_f, $n_m - 1 - n_f$, and $n - 1 - n_m$, respectively. Nodes $\forall i \leq n_f$ are considered as fast, whereas nodes $\forall i > n_m$ are considered as slow (and will likely get more saturated).

The medium nodes ($\forall i \in [n_f+1, n_m]$) have an intermediate computational speed. We assume nodes from the same cluster have the same intensity μ_f, μ_m, μ_s, for fast, medium, and slow nodes respectively. For practical reason we assume now that nodes $\forall i > n_m$ are the slowest ones, and has an intensity $\theta_s > \theta_j, \forall j < n$. This assumption is not restrictive due to the close nature of the network, and allows us to simplify the problem by splitting nodes into clusters.

This results in the *scaled* intensities $\gamma_f(\iota), \gamma_m(\iota), \gamma_s(\iota)$, where $\gamma_s(\iota) = 1$, $\gamma_m(\iota) = 1 + c_m \iota^{\alpha-1}$, and $\gamma_f(\iota) = 1 + c_f \iota^{\delta-1}$; with $\alpha \leq 1$ and $\delta > 1$. The constant task constraint translates into the existence of β such that $\beta \iota^{1-\alpha} = \mathrm{C} + 1$. The particular choice of $\alpha \leq 1$ in Van Kreveld et al. (2021) allows us to obtain traffic loads of nodes that tend to 1 as $\iota \to \infty$, and we could directly apply Corollary 2 from Van Kreveld et al. (2021). But this setting is inconsistent with the practical Federated Learning framework we consider in this section: in practice most of fast clients have an empty queue. Hence, we stick to $\delta > 1$, and we can apply the results of (Corollary.3, Van Kreveld et al., 2021). This work gives a precise results on the queue length of saturated nodes, in the limit of high traffic loads. The queue length of the remaining queue are defined by the population size constraint. Note because the unnormalized queue length of fast nodes converges to a finite-mean random variable, there is no need to scale it.

Proposition 12 (Corollary.3 in Van Kreveld et al. (2021))**.** *In stationary regime, as $\iota \to \infty$, $X_i^\iota, \forall i \in [1, n_f]$ become degenerate with value 0, and*

$$c_m \iota^{\alpha-1} X_i^\iota \to_{d.} \mathbb{E}\left[E_i \,|\, \sum\nolimits_{j=n_f+1}^{n_m} \frac{E_j}{c_m} \leq \beta \right], \forall i \in [n_f+1, n_m],$$

with E_i unit mean exponential distributions.

As a consequence, using as before dominated convergence we can estimate the following expected value (expected stationary queue lengths of fast, medium, and slow nodes respectively):

$$\begin{cases} \lim_{\iota\to\infty} \mathbb{E}[X_i^\iota] = 0, \quad \forall i \in [1, n_f], \\ \lim_{\iota\to\infty} \iota^{\alpha-1} \mathbb{E}[X_i^\iota] = \frac{1}{c_m}\Gamma(c_m\beta), \quad \forall i \in [n_f+1, n_m], \\ \lim_{\iota\to\infty} \iota^{\alpha-1} \mathbb{E}[X_i^\iota] = \frac{1}{n-n_m}\left(\beta - (n_m - n_f)\frac{1}{c_m}\Gamma(c_m\beta)\right), \quad \forall i \in [n_m+1, n]. \end{cases}$$

Hence, we can estimate the number of server steps when a task arrive and quits some node i as :

$$\lim_{\iota\to\infty} \iota^{\alpha-1}\, \mathrm{m}_i(\iota) \leq \lim_{\iota\to\infty} \frac{\lambda}{\mu_i}(\iota^{\alpha-1}\mathbb{E}[X_i^\iota] + 1),$$

where $\lambda = n_f \mathbb{P}(X_f > 0)\mu_f + (n_m - n_f)\mu_m + (n - n_m)\mu_s$ (because fast nodes have *almost* empty queue X_f in the considered stationary setting).

We will further assume $n_f = \frac{n}{3}$, $n_m = \frac{2n}{3}$, and $p = \frac{1}{n}$. Under these conditions, we have $\Gamma(c_m\beta) = \Gamma(\mathrm{C}(\frac{\mu_m}{\mu_s} - 1)) \simeq 1$. For fast nodes, the delay can be simplified as:

$$\lim_{\iota\to\infty} \mathrm{m}_i(\iota) \leq \frac{\lambda}{\mu_f}.$$

For medium nodes, the delay can be simplified as:

$$\lim_{\iota\to\infty} \mathrm{m}_i(\iota) \leq \frac{\lambda}{\mu_m} \frac{1}{\frac{\mu_m}{\mu_s} - 1}.$$

For slow nodes, this simplifies as:

$$\lim_{\iota\to\infty} \mathrm{m}_i(\iota) \leq \frac{\lambda}{\mu_s}\left(\frac{3}{n}\,\mathrm{C} - \frac{1}{\frac{\mu_m}{\mu_s} - 1}\right).$$

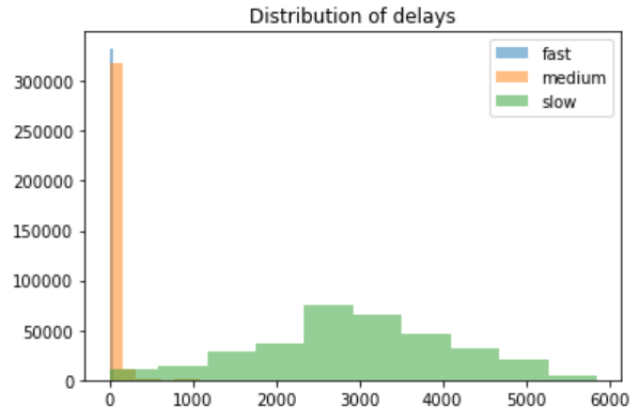
In the following we consider $n = 9$ clients, split in three clusters of same size: fast nodes with rate $\mu_f = 10$, medium nodes with rate $\mu_m = 1.2$, and slow nodes with rate $\mu_s = 1$. We simulate up to $T = 10^6$ server steps, and plot the distribution of the delays (in number of server steps). Under this setting, we have $\frac{\iota^{1-\alpha}}{c_m}\Gamma(c_m\beta) = \frac{1}{\frac{\mu_m}{\mu_s} - 1} = 5$. Hence we can estimate the following:

$$\begin{cases} \lim_{\iota\to\infty} \mathrm{m}_i(\iota) \leq \frac{n_f\mathbb{P}(X_f>0)\mu_f + (n_m - n_f)\mu_m + (n-n_m)\mu_s}{\mu_f} \simeq 3\mathbb{P}(X_f > 0), \quad \forall i \in [1, n_f], \\ \lim_{\iota\to\infty} \mathrm{m}_i(\iota) \leq \frac{n_f\mathbb{P}(X_f>0)\mu_f + (n_m - n_f)\mu_m + (n-n_m)\mu_s}{\mu_m} \frac{1}{\frac{\mu_m}{\mu_s} - 1}, \quad \forall i \in [n_f+1, n_m], \\ \lim_{\iota\to\infty} \mathrm{m}_i(\iota) \leq \frac{n_f\mathbb{P}(X_f>0)\mu_f + (n_m - n_f)\mu_m + (n-n_m)\mu_s}{\mu_s}\left(\frac{3\,\mathrm{C}}{n} - \frac{1}{\frac{\mu_m}{\mu_s} - 1}\right), \quad \forall i \in [n_m+1, n]. \end{cases}$$

Method	FedBuff	AsyncSGD	`Generalized AsyncSGD`
Accuracy on the CS test set	49.89 ± 0.77	59.09 ± 1.97	66.61 ± 3.26

Table 2: Performance average (mean $\pm$ std) over 10 random seeds for the CIFAR-10 task.

The simulation gives $\lambda \simeq 9$, and we recover the theoretical delays: the average delay for fast nodes is close to 1,




```
avg delay fast =  1.3453161915498364
avg delay medium =  55.02906515784922
avg delay slow =  2935.6114257955987
avg queue fast 0.26063426063426065
avg queue medium 6.107726107726108
avg queue slow 326.96497296497296
```

Figure 12: We assign C = 1000 tasks to a network of $n = 9$ nodes split in 3 clusters.

the average delay for medium node is $55 \simeq 5\frac{\lambda}{\mu_m}$, and the average delay for slow nodes is about $2935 \simeq 325\frac{\lambda}{\mu_s}$.

H Deep learning experiments details

H.1 Simulation

We based our simulations mainly on the code developed by Nguyen et al. (2022): we assume a server and n clients, each of which initially has a unique split of the training dataset. To adequately capture the time spent on the server side for computations and orchestration of centralized learning, two quantities are implemented: the server waiting time (the time the server waits between two consecutive calls) and the server interaction time (the time the server takes to send and receive the required data). In all experiments, they are set to 4 and 3, respectively. When a client i receives a new task, we take a new sample from an exponential distribution (with mean $\frac{1}{\mu_i}$, where μ_i is the rate of node i), and stack the gradient computation on top of the client queue.

H.2 Implementation

In Section 5 we have simulated experiments and run the code for the concurrent approaches `AsyncSGD` and `FedBuff`. We also propose an implementation of `FedAvg` in the supplemental material. `FedAvg` is a standard synchronous method. At the beginning of each round, the central node s selects clients uniformly at random and broadcast its current model. Each of these clients take the central server value and then performs exactly K local steps, and then sends the resulting model progress back to the server. The server then computes the average of the s received models and updates its model. In this synchronous structure, the server must wait in each round for the slowest client to complete its update.

`AsyncSGD` is an asynchronous method that initially randomly selects C clients. Then, a server step is done when a new task is completed and sent back to the server. The server uniformly selects a new client and send a new task. While `AsyncSGD` was tested on a simple task in Koloskova et al. (2022), we have developed a deep learning

version of the algorithm (see supplemental material) based on a list of dictionaries (to simulate a network of waiting queues).

For each global step, in `FedBuff`, the runtime is the sum of the server interaction time and the time spent feeding the buffer of size Z. The waiting time for feeding the buffer depends on the respective local runtimes of the slow and fast clients, as well as on the ratio between slow and fast clients: in the code, we reset a counter at the beginning of each global step and read the runtime when the Z^{th} local update arrives. In `AsyncSGD` and `Generalized AsyncSGD`, the runtime is defined by the closed Jackson network properties.