



HAL
open science

Optimizing Asynchronous Federated Learning: A Delicate Trade-Off Between Model-Parameter Staleness and Update Frequency

Abdelkrim Alahyane, Céline Comte, Matthieu Jonckheere, Éric Moulines

► **To cite this version:**

Abdelkrim Alahyane, Céline Comte, Matthieu Jonckheere, Éric Moulines. Optimizing Asynchronous Federated Learning: A Delicate Trade-Off Between Model-Parameter Staleness and Update Frequency. 2025. hal-04938472

HAL Id: hal-04938472

<https://hal.science/hal-04938472v1>

Preprint submitted on 11 Feb 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Optimizing Asynchronous Federated Learning: A Delicate Trade-Off Between Model-Parameter Staleness and Update Frequency


Abdelkrim Alahyane ✉

EMINES-UM6P, Ben Guerir, Morocco

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

Céline Comte ✉ 

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

Matthieu Jonckheere ✉ 

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

Eric Moulines ✉ 

École Polytechnique, Palaiseau, France

Abstract

Synchronous federated learning (FL) scales poorly with the number of clients due to the straggler effect. Algorithms like `FedAsync` and `GeneralizedFedAsync` address this limitation by enabling asynchronous communication between clients and the central server. In this work, we rely on stochastic modeling to better understand the impact of design choices in asynchronous FL algorithms, such as the concurrency level and routing probabilities, and we leverage this knowledge to optimize loss. We characterize in particular a fundamental trade-off for optimizing asynchronous FL: minimizing gradient estimation errors by avoiding model parameter staleness, while also speeding up the system by increasing the throughput of model updates. Our two main contributions can be summarized as follows. First, we prove a discrete variant of Little’s law to derive a closed-form expression for relative delay, a metric that quantifies staleness. This allows us to efficiently minimize the average loss per model update, which has been the gold standard in literature to date. Second, we observe that naively optimizing this metric leads us to slow down the system drastically by overemphasizing staleness at the detriment of throughput. This motivates us to introduce an alternative metric that also takes system speed into account, for which we derive a tractable upper-bound that can be minimized numerically. Extensive numerical results show that these optimizations enhance accuracy by 10% to 30%.

2012 ACM Subject Classification Computing methodologies → Machine learning; Mathematics of computing → Stochastic processes

Keywords and phrases Federated learning, relative delay, throughput, Little’s law

Funding The research presented here was supported in part by ANR EPLER, Projet E2CC, and the “Data Science & Processus Industriels” chair funded by École polytechnique, the Mohammed VI Polytechnique University, and Fondation de l’X. This research was also facilitated by the support of LabEx CIMI via the SOLACE project-team and the “Stochastic control and learning for complex networks” thematic semester.

1 Introduction

Federated learning (FL) is a distributed learning strategy that enables model training across multiple clients under the supervision of a central server (CS), without requiring data sharing [12, 18]. In synchronous FL, stochastic gradient descent (SGD) is executed in rounds; during each round, the CS sends the current model parameters to a subset of clients, waits until all of them return a new stochastic gradient estimate, and then updates the model parameters before sending them to another batch of clients. Unfortunately, the performance of this

synchronous approach is often hindered by the variability of computational speeds among clients, leading to the straggler effect [6].

Asynchronous algorithms such as **FedAsync** [25, 5, 26] aim to tackle these challenges. This framework allows clients and the CS to communicate asynchronously; in particular, the CS may update the model parameters while clients are estimating gradients (possibly based on outdated model parameters). Asynchronicity is implemented by allowing tasks (i.e., requests for gradient estimates) to be queued at the clients [11]. Intuitively, asynchronous FL has the potential to speed up the system by circumventing the straggler effect, possibly at the cost of errors due to the staleness of the model parameters used to estimate gradients.

Consequently, a significant research effort has been devoted to analyzing the performance of asynchronous FL in both homogeneous and heterogeneous data settings [5, 6, 26, 11]. However, as we will review in Section 1.1, almost all existing analyses overlook the intricate dynamics of queues and the variability in processing capabilities across clients. In contrast, [14] showed recently that the performance of asynchronous FL critically depends on these via the staleness of the model parameters used by the clients to estimate gradients. Staleness is captured by the *relative delay*, defined as the (stochastic) number of times the CS updates the model parameters while a task is held at a client (either queued or being processed). While existing analyses assume this relative delay is bounded irrespective of the system parameters, the results of [14] imply that relative delay depends heavily on these parameters and may grow arbitrarily large even in realistic scenarios.

As we will see in Section 1.2, we provide actionable means of optimizing performance in asynchronous FL. First, building on the results of [14], we derive an explicit expression for the mean relative delay and its gradient by leveraging the framework of Jackson networks [10], which in turn allows us to design a gradient-descent algorithm that optimizes performance. In a second time, we observe that minimizing the average norm of the gradient per update, as it is done traditionally in the literature, may actually be counterproductive: since it completely ignores the throughput of model updates, this leads us to slow the system down to the pace of the slowest client, thus underutilizing all other clients. Although we focus on an extension of **AsyncSGD** called **Generalized AsyncSGD**, we believe our results are relevant to other asynchronous FL algorithms.

1.1 Related work

FL has shifted decisively from synchronous [24, 20, 16, 17, 23] to asynchronous models, driven by challenges in scaling, resource allocation, and coordination. Key studies highlight the limitations of synchronous FL, such as the susceptibility to delays from stragglers and difficulties in managing communication rounds as the number of agents increases [25]. This has accelerated the adoption of asynchronous approaches, such as **FedAsync** and its variants [25, 5, 26], which incorporate memory-based updates, adaptive learning rate adjustments, and strategies to reduce staleness and handle varying computational speeds.

Nevertheless, limitations persist, especially in dealing with heterogeneous data environments [11]. Innovations such as **FedBuff** [19], with CS-side buffering, and **AsyncSGD** [11], with its delay guarantees, address some issues, but existing performance guarantees often rely on simplified assumptions about queue dynamics and processing delays, affecting FL system performance and efficiency.

Recent work like **AsGrad** [9], which extends asynchronous FL analysis under bounded-gradient assumptions, underscores the benefits of deeper insights into these dynamics. The typical finite-delay assumption reveals a major gap in the literature: existing models fail to fully capture the unpredictable and heterogeneous nature of client speeds and network

conditions, even if these are the fundamental motivation for introducing asynchronous FL in the first place. Despite progress in the understanding of asynchronous FL, a comprehensive analysis of the underlying stochastic queuing dynamics remains urgently needed. A preliminary attempt is made in [14], which introduces **Generalized AsyncSGD**—an algorithm that utilizes non-uniform client selection to address queuing dynamics and heterogeneous server speeds. However, their analysis falls short in providing explicit performance bounds, instead relying on scaling regimes to approximate the system behavior. A detailed analysis is central to the optimization of FL systems, not only in terms of traditional network performance metrics, but also for achieving effective, efficient, and fair learning outcomes with diverse and distributed clients. Our results highlight in particular that routing strategies should be adapted based on the specific bottlenecks imposed by applications, such as the number of computations rounds versus actual training time.

1.2 Contributions

Our results stem from key theoretical results allowing the explicit characterization of the impact of queue dynamics on the performance of asynchronous FL. They can be summarized as follows:

Derive tractable bounds on loss gradients Using the framework of queuing theory, we derive exact and tractable formulas for the mean relative delay and its gradient. In general, these formulas can be estimated in time $O(n^2m^2)$, where n is the number of clients and m the concurrency level, or they can be estimated through Monte Carlo simulations. Further simplifications allow us to compute them in time $O(n)$ for simple routing strategies.

Optimize performance Leveraging this result, we design an algorithm that optimizes the performance of **Generalized AsyncSGD** by minimizing the bound of [14]. Our result also allows us to gauge the bound sensitivity to crucial system parameters, such as the ratio of the concurrency level to the number of clients, and the clients' service speeds. These insights are relevant to other asynchronous FL algorithms, and they underscore that routing strategies should be adapted based on application-specific bottlenecks.

Account for clock-time performance We observe both analytically and numerically that minimizing the average norm-square of the gradient per update actually leads us to slow down the system considerably by underutilizing all clients but (the slowest) one. Roughly speaking, since this metric ignores the throughput of model updates, it is optimized by minimizing staleness, which is achieved by giving priority to the slowest client. This motivates us to introduce an alternative metric that explicitly accounts for throughput, and for which we derive a tractable upper-bound that can again be optimized efficiently.

Our findings provide both qualitative insights into the impact of queuing dynamics and efficient numerical methods to optimize performance. Our experiments on real-world datasets show that tuning the routing strategies and/or the concurrency level can improve accuracy by 10% to 30%.

1.3 Notations

$\mathbb{Z}, \mathbb{N}, \mathbb{N}_{>0}, \mathbb{R}, \mathbb{R}_{\geq 0}, \mathbb{R}_{>0}$ denote the sets of integers, non-negative integers, positive integers, real numbers, non-negative real numbers, and positive real numbers. Let $|\cdot|$ denote the ℓ_1 -norm and $\mathbb{1}[\cdot]$ the indicator function. For each $n, m \in \mathbb{N}_{>0}$, let $\mathcal{X}_{n,m} = \{x \in \mathbb{N}^n : |x| = m\}$ denote the set of n -dimensional natural-number-valued vectors with ℓ_1 -norm m . For every $n \in \mathbb{N}_{>0}$, let $\mathcal{P}_n = \{p \in \mathbb{R}^n : 0 < p_i < 1 \text{ for } i \in \{1, 2, \dots, n\} \text{ and } |p| = 1\}$.

2 Model and prior results

2.1 Asynchronous federated learning

The goal in federated learning (FL) is to optimize the average performance of a model across multiple clients under the supervision of a central server (CS): $\min_{w \in \mathbb{R}^d} f(w)$, where $f(w) = \frac{1}{n} \sum_{i=1}^n f_i(w)$, and

$$f_i(w) = \mathbb{E}_{(x,y) \sim \mathcal{D}_i} [\ell_i(\text{NN}(x, w), y)], \quad i \in \{1, 2, \dots, n\}.$$

Here, w denotes the parameters of a deep neural network, d the number of parameters (including weights and biases), $\text{NN}(x, w)$ the prediction function of the neural network, n the number of clients, ℓ_i the local loss function of client i , and \mathcal{D}_i the data distribution at client i . Each client i approximates the gradient $\nabla_w f_i(w)$ of its local loss function using a stochastic gradient denoted by $g_i(w)$. The computation of such a stochastic gradient by a client is called a *task*.

Generalized AsyncSGD [14] is shown in Algorithms 1 (CS) and 2 (client i). A task assigned to a busy client is queued according to the first-in-first-out (FIFO) policy. As we will see, performance depends critically on two parameters: p , the routing probability vector of tasks to clients; and m , the concurrency [11], defined as the number of tasks that are concurrently dispatched to the clients (either queued or being processed).

Let us focus on Algorithm 1 (CS perspective). The model parameter is initialized to a random vector w_0 , and the system state is initialized to a random vector $\xi = (\xi_1, \xi_2, \dots, \xi_n) \in \mathcal{X}_{n,m}$, such that ξ_i is the number of tasks initially dispatched to client i , for each $i \in \{1, 2, \dots, n\}$. After this initialization, each iteration $t \in \{0, 1, \dots, T\}$ of the **for** loop (Line 7) proceeds as follows: whenever a client C_t completes a task and reports a gradient estimate $g_{C_t}(w_{I_t})$ (Line 8; I_t will be defined shortly), the CS immediately updates the model parameter (Line 9) and sends it to the next client A_{t+1} , where $\mathbb{P}(A_{t+1} = i) = p_i$ for each $i \in \{1, 2, \dots, n\}$ (Lines 10 and 11). Recall that a client can be chosen even if it is processing a task.

Time indices are such that, for each $t \in \{1, 2, \dots, T\}$, A_t and C_t correspond to the same (model-parameter update) round, where a round is defined as the time between the assignment of a task to a client and the next task completion (leading to a model update). In Sections 4 and 5, we will see that throughput, defined as the inverse of the (clock-time) duration of a typical round, has a crucial impact on performance. For each $t \in \{0, 1, \dots, T\}$, we let $X_t = (X_{1,t}, X_{2,t}, \dots, X_{n,t}) \in \mathcal{X}_{n,m-1}$ denote the state at the end of round t , so that,

Algorithm 1 Generalized AsyncSGD (CS)

- 1: **Input:** Numbers T , n , and m of rounds, clients, and tasks; routing p ; learning rate η
 - 2: Initialize parameters w_0 randomly
 - 3: Initialize state vector $\xi \in \mathcal{X}_{n,m}$ randomly
 - 4: **for** $i = 1, 2, \dots, n$ **do**
 - 5: Send ξ_i times model parameter w_0 to client i
 - 6: **end for**
 - 7: **for** $t = 0, \dots, T$ **do**
 - 8: CS receives stochastic gradient $g_{C_t}(w_{I_t})$ from a client C_t
 - 9: Update $w_{t+1} \leftarrow w_t - \frac{\eta}{np_{C_t}} g_{C_t}(w_{I_t})$
 - 10: Sample a new client A_{t+1} according to p
 - 11: Send model parameter w_{t+1} to client A_{t+1}
 - 12: **end for**
-

Algorithm 2 Generalized AsyncSGD (Client i)

- 1: **Input:** Queue of received parameters, local dataset
 - 2: **if** Queue is not empty **then**
 - 3: Take the received parameter w from the queue using a FIFO policy
 - 4: Compute the gradient estimate $g_i(w)$
 - 5: Send the gradient to the CS
 - 6: Repeat
 - 7: **end if**
-

for each $i \in \{1, 2, \dots, n\}$, we have $X_{i,0} = \xi_i - \mathbb{1}[C_0 = i]$ and, for each $t \in \{1, 2, \dots, T\}$,

$$X_{i,t} = X_{i,t-1} + \mathbb{1}[A_t = i] - \mathbb{1}[C_t = i]. \quad (1)$$

The processing times of successive tasks at client i are independent, exponentially distributed with rate $\mu_i > 0$, for each $i \in \{1, 2, \dots, n\}$. In particular, for each $t \in \{1, 2, \dots, T\}$, we have $\mathbb{P}[C_t = i | X_{t-1}, A_t] \propto \mu_i$ for each $i \in \{1, 2, \dots, n\}$ such that $X_{i,t-1} + \mathbb{1}[A_t = i] > 0$.

Critically, the gradient estimate $g_{C_t}(w_{I_t})$ returned at the end of an iteration t is based on the model parameters w_{I_t} known to the CS at the beginning of round $I_t = \sum_{s=0}^t s \mathbb{1}[s + D_{A_s, s} = t]$ when the task was assigned to client C_t , where $D_{i,t}$ is called the *relative delay* and is defined as the number of rounds that get completed during the sojourn of a task at a client:

$$D_{i,t} = \mathbb{1}[A_t = i] R_{i,t}, \text{ where} \quad (2)$$

$$R_{i,t} = \min \left\{ r \in \mathbb{N} : \sum_{s=t}^{t+r} \mathbb{1}[C_s = i] = X_{i,t-1} + \mathbb{1}[A_t = i] \right\}. \quad (3)$$

Consistently with the literature on decentralized learning, we make the following assumptions: uniform lower bound f^* for the objective function f , L -Lipschitz continuity of the gradients ∇f_i to ensure smoothness, upper bound σ^2 on the variance of the stochastic gradients, and upper bound M^2 on the gradient dissimilarity across clients (see Assumptions A1–A4 in Section A of the supplementary materials).

2.2 Queuing dynamics

The next result is a variant of [14, Proposition 2] and will be instrumental throughout the paper. It relates the queuing dynamics to the routing and service-rate vectors $p = (p_1, p_2, \dots, p_n)$ and $\mu = (\mu_1, \mu_2, \dots, \mu_n)$. Recall that the state space is $\mathcal{X}_{n,m-1}$ (and not $\mathcal{X}_{n,m}$) because we consider model-parameter-update times. Buzen's algorithm was introduced in [4].

► **Proposition 2.1.** *In the framework of Section 2.1, the sequence $(X_t, t \in \mathbb{N})$ defines an irreducible positive recurrent Markov chain with stationary distribution*

$$\pi_{n,m-1}(x) = \frac{1}{Z_{n,m-1}} \prod_{i=1}^n \left(\frac{p_i}{\mu_i} \right)^{x_i}, \quad x \in \mathcal{X}_{n,m-1}, \quad (4)$$

where the normalizing constant $Z_{n,m-1}$ can be computed by applying Buzen's recursive algorithm:

- $Z_{n,0} = 1$ for each $n \in \{1, 2, \dots, n\}$,
- $Z_{1,m} = \left(\frac{p_1}{\mu_1} \right)^m$ for each $m \in \{0, 1, 2, \dots, m\}$

6 Optimizing Asynchronous Federated Learning

■ $Z_{n,m} = Z_{n-1,m} + \frac{p_m}{\mu_m} Z_{n,m-1}$ for $n \in \{2, 3, \dots, n\}$ and $m \in \{1, 2, \dots, m\}$.

Proof. See Section B in the supplementary material. ◀

In the rest of the paper, we will assume that the system starts in steady state. This assumption is reasonable when the total number T of updates is sufficiently large, as the distribution of X_t converges exponentially fast with t towards the stationary distribution regardless of the initial distribution (see Theorem 13.4.14 in [2]). Concretely, we will assume that $X \triangleq X_0$ follows the stationary distribution $\pi_{n,m-1}$, and we will often drop the time index, so that for instance D_i will be a random variable distributed like $D_{i,t}$ for any $t \in \mathbb{N}_{>0}$.

3 Optimize model updates

Consistently with the literature on asynchronous FL, in this section, we assume the CS has a fixed budget T of model parameter updates, and we try to make the best of these updates. This is particularly relevant in contexts, such as cellular networks with costly data plans or satellite internet services, where data transmission costs are high.

3.1 Bound, delay, and gradient descent

[14, Theorem 1] gave the following upper bound on the ergodic mean of the norm-square of the gradient of f : there exists $\eta_{\max} > 0$ dependent on p so that, for any $\eta \in (0, \eta_{\max})$,

$$\frac{1}{T+1} \sum_{t=0}^T \mathbb{E}[\|\nabla f(w_t)\|^2] \leq 8G, \text{ where}$$

$$G = \frac{A}{\eta(T+1)} + \frac{\eta LB}{n^2} \sum_{i=1}^n \frac{1}{p_i} + \frac{\eta^2 L^2 B m}{n^2} \sum_{i=1}^n \frac{\mathbb{E}[D_i]}{p_i^2}. \quad (5)$$

The constants A , B , and L depend on the learning problem under consideration (see [14]) but not on the routing vector¹ p , and they are estimated heuristically. If needed, the dependency of G on the routing vector p will be made explicit by writing $G(p)$. We will use G as a proxy to minimize the ergodic norm-squared of the gradient of f .

The first term in G follows a classical pattern, indicating how initialization influences convergence. The next two terms depend strongly on the routing and service-rate vectors p and μ , both explicitly and implicitly via the mean relative delays $\mathbb{E}[D_i]$. In principle, the mean relative delays $\mathbb{E}[D_i]$ are complex functions of the system dynamics, as a task's relative delay may depend on an unbounded number of rounds in the future.

Theorem 3.1, our first main contribution, bypasses this difficulty by expressing the expected relative delays as functions of the mean numbers of stationary tasks, which in turn allows us to derive closed-form expressions for the mean relative delays and their gradient.

► **Theorem 3.1.** *In the framework of Section 2.1, we have*

$$\mathbb{E}[D_i] = \mathbb{E}[X_i], \quad i \in \{1, 2, \dots, n\}, \quad (6)$$

$$\frac{\partial \mathbb{E}[D_i]}{\partial p_j} = \frac{1}{p_j} \text{Cov}[X_i, X_j], \quad i, j \in \{1, 2, \dots, n\}, \quad (7)$$

¹ Starting from [14, Equation (2)], we can apply Assumption A1 to write $A = \mathbb{E}[f(w_0)] - f^*$, which does not depend on p .

where for each $i, j \in \{1, 2, \dots, n\}$,

$$\mathbb{E}[X_i] = \sum_{k=1}^{m-1} \left(\frac{p_i}{\mu_i} \right)^k \frac{Z_{n,m-1-k}}{Z_{n,m-1}}. \quad (8)$$

$$\mathbb{E}[X_i X_j] = \sum_{\substack{k,\ell=1 \\ k+\ell \leq m-1}}^{m-1} \left(\frac{p_i}{\mu_i} \right)^k \left(\frac{p_j}{\mu_j} \right)^\ell \frac{Z_{n,m-1-k-\ell}}{Z_{n,m-1}}, \quad (9)$$

and the constants $Z_{n,m}$ for $m \in \{0, 1, \dots, m-1\}$ are computed as in Proposition 2.1.

Proof. The proof of Theorem 3.1 is in Section C of the supplementary material. Here we briefly give the intuition behind (6), which is analogous to Little's law. If each task at client i pays \$1 each time a task gets completed at another client, there are two ways of collecting payments: either we receive an upfront payment of $\$R_i$ when a task is assigned to client i (yielding $\mathbb{E}[D_i]$), or we earn $\$X_i$ each time a task gets completed at another client (yielding $\mathbb{E}[X_i]$). Equations (7)–(9) follow by direct computations, after observing that the distribution (4) is an exponential family. ◀

In the remainder, we will apply (6)–(7) to compute $\mathbb{E}[D_i]$ and $\nabla_p \mathbb{E}[D_i]$. However, these equations can also be used to estimate these quantities through Monte Carlo simulations. Section E of the supplementary material gives a gradient-descent algorithm to optimize the routing vector p in view of minimizing G , which will be applied in Section 5 to run extensive numerical results.

3.2 Discussion

Theorem 3.1 allows us to gain insight into the impact on G of the system parameters.

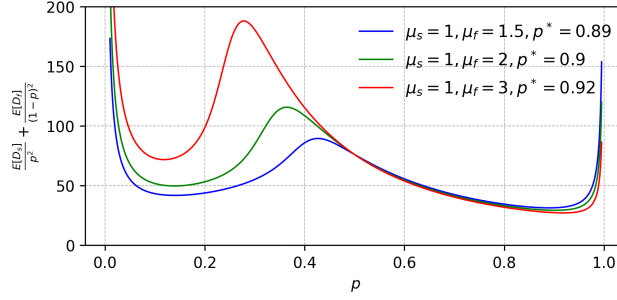
How to minimize G ?

One can verify that the second term in G is minimized by applying the uniform routing p^{uniform} , given by $p_i^{\text{uniform}} = \frac{1}{n}$ for each $i \in \{1, 2, \dots, n\}$. Figure 1 shows that minimizing the third term in G , involving the mean relative delays, is more challenging: even in a toy 2-client example, the third term is non-monotonic and is minimized by assigning almost all tasks to the slowest client.

To gain more insight into this third term, observe that Equation (6) from Theorem 3.1 yields the following simple relation between the mean relative delays:

$$\sum_{i=1}^n \mathbb{E}[D_i] = \sum_{i=1}^n \mathbb{E}[X_i] = m - 1. \quad (10)$$

This relation has several consequences, in particular: (i) the sum of the mean relative delays depends only on the numbers n and m of clients and tasks, while the vectors p and μ impact only how the relative delay is distributed across clients; (ii) decreasing the relative delay at a client necessarily comes at the cost of an increased relative delay at another client; and (iii) since both the routing probabilities p_i and relative delays $\mathbb{E}[D_i]$ have constant sums over the clients i , minimizing the third term in G requires finding a routing vector p such that a client i with a high relative delay $\mathbb{E}[D_i]$ also has a relatively large routing probability p_i .



■ **Figure 1** Third term of the bound G given in (5) vs. the routing probability to the slowest client, in a toy example with $n = 2$ clients and $m = 20$ tasks, for various speed vectors $\mu = (\mu_s, \mu_f)$.

Dependency on the number m of tasks

Another consequence of Theorem 3.1 is that the bound G is an increasing function of the number m of tasks (by combining (6) with the observation that $\mathbb{E}[X_i]$ is a non-decreasing function of m [22, Lemma 2]). In particular, keeping all other system parameters fixed, the performance is optimized when only $m = 1$ task circulates in the network! In this case, (10) implies that the third term in G is equal to zero. This is intuitive because, with a single task circulating in the network, the staleness issue is trivially eliminated, and the system works like a synchronous FL system in which the CS samples a single client at each round. This observation motivates the alternative metric we introduce in Section 4.

Simple routing strategies

Our result allows us to simplify the bound G for two noteworthy routing vectors. First, under uniform routing p^{uniform} , (10) yields

$$G(p^{\text{uniform}}) = \frac{A}{\eta(T+1)} + \eta LB + \eta^2 L^2 B m(m-1).$$

Another strategy that we can analyze explicitly is the *balanced* routing vector $p^{\propto \mu}$, such that $p_i^{\propto \mu} = \mu_i / \sum_j \mu_j$, i.e., the probability of routing a task to a client is proportional to this client’s service speed. This routing strategy is popular in queuing theory because it “balances load” across clients, in the sense that $\mathbb{E}[D_i] = \mathbb{E}[X_i] = (m-1)/n$ for each $i \in \{1, 2, \dots, n\}$ (by (4), (6), and (10)). It follows that

$$G(p^{\propto \mu}) = \frac{A}{\eta(T+1)} + \frac{\eta LB |\mu|}{n^2} \sum_{i=1}^n \frac{1}{\mu_i} + \frac{\eta^2 L^2 B m(m-1) |\mu|^2}{n^3} \sum_{i=1}^n \frac{1}{\mu_i^2}.$$

4 Optimize physical time

In this section, we aim to achieve optimal performance in terms of physical time, that is, accounting for the duration of model-parameter update rounds. In Section 4.1, we introduce a new performance metric that takes this duration into account, and we derive an upper-bound H , which serves as the counterpart to G . Section 4.2 highlights the key differences compared to Section 3.

4.1 Bound, delay, and gradient descent

Theorem 4.1 below provides an upper bound on the ergodic mean of the squared norm of the gradient of f multiplied by the duration of the corresponding round. This can be interpreted as minimizing the average per-round cost, where the cost of a round is the integral of the squared norm of the gradient of f over its duration. For each $t \in \{0, 1, \dots, T\}$, τ_t denotes the duration of round t in the model of Section 2.1.

► **Theorem 4.1.** *In the framework of Section 2.1, there exist $\tilde{A} > 0$ and a random variable $\eta_{max} > 0$ so that, for any $\eta \in (0, \eta_{max})$,*

$$\mathbb{E} \left[\frac{1}{T+1} \sum_{t=0}^T \tau_t \|\nabla f(w_t)\|^2 \right] \leq 8H, \text{ where} \quad (11)$$

$$H = \frac{\tilde{A}}{\eta\lambda} + \frac{\eta LB}{\lambda n^2} \sum_{i=1}^n \frac{1}{p_i} + \frac{\eta^2 L^2 B m}{\lambda n^2} \sum_{i=1}^n \frac{\mathbb{E}[\xi_i]}{p_i^2}. \quad (12)$$

Here, B and L are as defined in (5), the random vector $\xi \in \mathcal{X}_{n,m}$ follows the distribution $\pi_{n,m}$ as defined in (4), and λ is the throughput of the Jackson network, that is, the number of rounds per unit of physical time, given as follows, with $Z_{n,m}$ and $Z_{n,m-1}$ as defined in Proposition 2.1:

$$\lambda = \sum_{i=1}^n \mu_i \mathbb{P}(\xi_i > 0) = \frac{Z_{n,m-1}}{Z_{n,m}}. \quad (13)$$

Proof. See Section D of the supplementary material. ◀

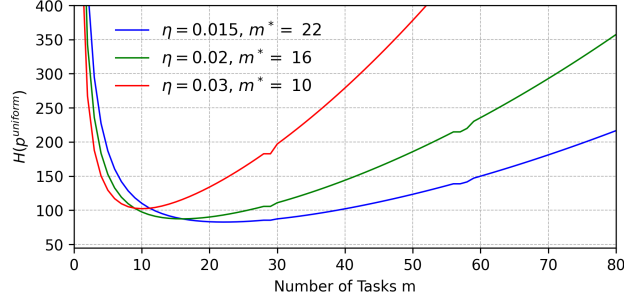
The bound H in Theorem 4.1 can be seen as the throughput-aware counterpart of the bound G in Section 3. The main differences are as follows. First, A is replaced with \tilde{A} , which is *a priori* dependent on p . We can verify that \tilde{A} can be upper-bounded by a constant independent of p when the function f is bounded, and we conjecture this is true in general; in the numerical results, we will treat it as a constant, like A . Second, the number T of updates in the first term of G , independent of p , is replaced with the throughput λ that appears in all terms of H . Lastly, the expectation $\mathbb{E}[X_i](= \mathbb{E}[D_i])$ in G , with $X \sim \pi_{n,m-1}$, is replaced with $\mathbb{E}[\xi_i]$ in H , with $\xi \sim \pi_{n,m}$. The intuition is that the relative delay (counting the number of rounds) in G is replaced with the absolute delay (counting the physical time) in H ; hence, the counterpart of Theorem 3.1 yields the state at an arbitrary time, which has state space $\mathcal{X}_{n,m}$.

4.2 Discussion

Part of the discussion in Section 3.2 can be adapted to H with minor modification. In particular, the counterpart of (10) is that $\sum_{i=1}^n \mathbb{E}[\xi_i] = m$. This equality has consequences in terms of optimization and can be used to simplify the expression of H under simple routing strategies (though the throughput may not simplify).

One fundamental difference between G and H is that H is in general *not* a non-decreasing function of the number m of tasks. Figure 2 shows the bound $H(p^{\text{uniform}})$ as a function of the number m of tasks, under different step sizes η , and it reveals the existence of an optimal number $m^* > 1$ of tasks that minimizes H . Our intuition is that, when $m < m^*$, the throughput is insufficient and clients are underutilized; conversely, when $m > m^*$, the throughput increases but so does staleness, which ultimately hinders convergence, as already

observed with G . As a side remark, observe that the optimal number m^* of tasks decreases (albeit slowly) with η . Our intuition is that, with a higher η , the impact of stale gradients becomes more significant, as each gradient carries more weight in the model-parameter updates.



■ **Figure 2** Bound $H(p^{\text{uniform}})$ as a function of the number m of tasks for different values of the step size η . The system consists of 50 clients, with speeds given by $\mu_i = e^{i/100}$ for each $i \in \{1, \dots, n\}$.

5 Numerical results

In this section, we numerically evaluate the impact of the optimizations proposed in Sections 3 and 4 on the accuracy and loss of **Generalized AsyncSGC** for several image-classification tasks. We consider mainly two data-distribution scenarios:

Homogeneous Data is distributed independently and identically across clients, and each client receives an equal number of data points from each class.

Heterogeneous Datasets are heterogeneous across clients, both in terms of distribution and volume. For each class k , we sample a vector $p_k \sim \text{Dir}_n(0.5)$, where $p_{k,j}$ is the proportion of class- k instances allocated to client j and $\text{Dir}_n(\beta)$ the Dirichlet distribution with dimension n and concentration parameter $\beta > 0$.

Given a system as described in Section 2.1, the G -optimized (resp. H -optimized) routing vector p_G^* (resp. p_H^*) is computed by minimizing G (resp. H) using the Adam gradient-descent algorithm with standard hyperparameter values from the literature, initialized with the uniform routing vector p^{uniform} . Since G and H are non-convex, the obtained routing vector may be a local minimum. The gradients of G and H are computed in closed form, using the results of Sections 3 and 4, as detailed in Sections E and F of the supplementary material.

Given a routing vector p , we simulate the system of Section 2.1 and evaluate **Generalized AsyncSGD** on image classification tasks using the Fashion-MNIST [8], KMNIST [7], CIFAR-10, and CIFAR-100 [13] datasets. We employ the standard multi-class cross-entropy loss and evaluate performance on an unseen validation dataset. Additional details on the experiments are provided in Section G of the supplementary material.

5.1 Optimize model updates

We first focus on the performance goal of Section 3, namely, to minimize the average norm-square of the gradient of f per update round. We consider the system of Section 2.1, with $n = 20$ clients handling $m = 100$ tasks. For each $i \in \{1, \dots, 20\}$, the service speed at client i is $\mu_i = e^{i/100}$, so that the fastest client is 20% faster than the slowest. The learning

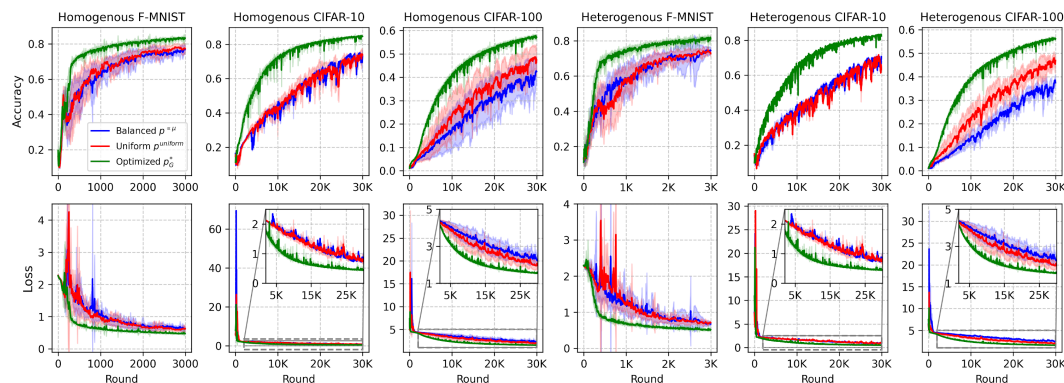
rate is $\eta = 0.01$, and $L = 1$. As discussed in Section 3.2, this high-concurrency scenario is particularly challenging when it comes to minimize gradient staleness.

Optimizing G

The optimized vector p_G^* obtained by minimizing G selects the slowest client over 40% of the time, while the routing probabilities to faster clients are all of the same magnitude but decrease as the client speed increases. This result is consistent with [14, Section 5] but counterintuitive *a priori*. Such a skewed routing vector is obtained because it significantly reduces the third term of G (see Section 3.2). Intuitively, p_G^* synchronizes the system to the pace of the slowest client, while routing tasks to faster clients in inverse proportion to their speeds, to reduce errors caused by stale gradients.

Additional numerical results (not shown here) reveal that the skewness of the optimized routing vector p_G^* towards the slowest client is accentuated when the concurrency m is large, as in the scenario we consider here. More specifically, the vector p_G^* is closer to the uniform vector p^{uniform} when the concurrency level m is small relative to the number n of clients, while the routing probability to the slowest client rises significantly as m increases. This is in line with the discussion of Section 3.2, where we observed that the relative weight of third term of G tends to increase with m .

Performance on datasets



■ **Figure 3** Performance on the validation set at the CS in the scenario of Section 5.1, with $n = 20$ clients and $m = 100$ tasks, for homogeneous and heterogeneous data. Solid lines show metrics averaged over independent simulations, and shaded areas represent the standard deviation. For Fashion-MNIST, we simulated the learning process over 3,000 rounds, repeating the simulations 10 times. The accuracy and loss on an unseen validation set were recorded every 5 rounds. For CIFAR-10 AND CIFAR-100, we applied standard normalization and data augmentation techniques, running the simulation over 30,000 rounds, repeated three times. The accuracy and loss on the unseen validation set were logged every 50 rounds.

Figure 3 shows that the optimized routing strategy p_G^* improves performance over the uniform (p^{uniform}) and balanced ($p^{\times\mu}$) routing strategies across all our experiments and throughout the learning process. In particular, while assigning a higher routing probability to the slowest client may seem to skew the model toward the data distribution associated with that client, our simulations indicate that the optimized routing strategy consistently

achieves superior and more reliable performance. The standard deviation is also markedly higher under p^{uniform} and $p^{\times\mu}$, suggesting p_G^* exhibits greater robustness. Section G.3 of the supplementary material shows that the same conclusions hold in even-more-heterogeneous data-distribution scenarios, where each client’s dataset only contains a subset of the image labels (possibly disjoint from other clients).

Anticipating over Section 5.2, we note that the relative performance of the optimized routing p_G^* versus the uniform and balanced strategies would be reversed if in Figure 3 we used physical time on the x-axis: as observed in Sections 3.2 and 4.2, since G disregards throughput, it reduces gradient staleness at the cost of a lower throughput. Concretely, in Figure 3, the average (physical) time to complete the 3,000 update rounds was 7 to 8 times larger under p_G^* compared to p^{uniform} and $p^{\times\mu}$. We believe such an increase in computing time may be prohibitive even when physical time is not the primary concern.

5.2 Optimize physical time

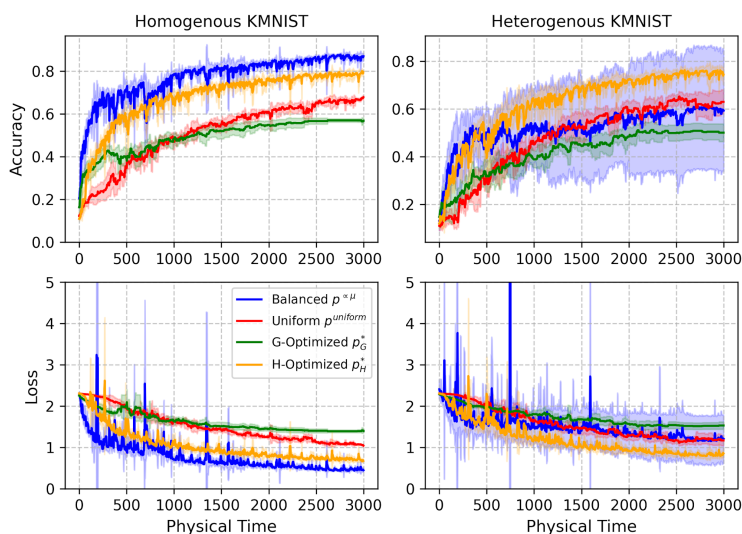
We analyze a network of $n = 30$ clients with concurrency level $m = 30$. Clients are organized into three clusters, each containing 10 clients. The slowest cluster has an average service time of 100 time units, the medium cluster of 10 time units, and the fastest cluster of 1 time unit. This low-concurrency high-speed-heterogeneity scenario is particularly challenging when it comes to optimizing throughput. We set the parameters as follows: $\tilde{A} = 15$, $L = 1$, $\sigma = 3$, $M = 10$, and $\eta = 0.01$. More details appear in Section G of the supplementary material.

Optimizing H

The H-optimized routing probabilities are (per client) $p_{H, \text{slow}}^* = 0.0068$, $p_{H, \text{medium}}^* = 0.0449$, and $p_{H, \text{fast}}^* = 0.0487$. Contrary to Section 5.1, faster clients receive a larger fraction of tasks, but $p_{H, \text{medium}}^*$ and $p_{H, \text{fast}}^*$ are still of the same order, so that p_H^* seems to a tradeoff between minimizing staleness (like p_G^*) and maximizing throughput (like $p^{\times\mu}$ —a common heuristic to maximize throughput). Concretely, within the physical time frame of 3,000 units plotted in Figure 4, $p^{\times\mu}$ completes 17,000 update rounds, which sets it apart from p_H^* (3,200 rounds), p^{uniform} (690 rounds), and p_G^* (145 rounds).

Performance on datasets

Figure 4 compares the accuracy and loss of **Generalized AsyncSGD** under these four routing strategies. First focusing on the average performance, we observe that in the homogeneous scenario $p^{\times\mu}$ performs better than p_H^* , which in turn outperforms p_G^* and p^{uniform} , while in the heterogeneous scenario p_H^* outperforms all strategies. Furthermore, the balanced strategy $p^{\times\mu}$ exhibits sharp spikes in the loss trajectory in all scenarios, and a particularly large standard deviation in the heterogeneous scenario. This contrasts with the low and stable standard deviation exhibited by both p_G^* and p_H^* . All in all, these numerical results confirm that when physical time is a performance criterion, the H -optimized strategy provides a suitable trade-off between minimizing staleness (overweighted by p_G^*) and maximizing throughput (overweighted by $p^{\times\mu}$), even though the bound H serves only as a proxy for actual performance. Additional plots are provided in Appendix G.3 for other image-classification tasks.



■ **Figure 4** Performance with respect to physical time on the validation set at the CS in the scenario of Section 5.2, with $n = 30$ clients and $m = 30$ tasks, for homogeneous and heterogeneous KMNIST datasets. We simulated the process over 3,000 physical time units, recording accuracy and loss on a validation set every 5 rounds. Each experiment was repeated independently 10 times.

6 Conclusion

We provide novel insight into the impact of queuing dynamics on the performance of asynchronous FL. This allows us not only to derive methods for optimizing performance, but also to identify and overcome fundamental limitations of existing performance objectives. Our numerical results on real datasets show that such queuing effects can have very significant impact on performance, and that the optimizations we propose can increase accuracy by 10% to 30%. We believe that these insights are relevant for **Generalized AsyncSGC**, our focus here, but also to other algorithms like **FedBuff**. Specifically, the asynchronous aggregation in **FedBuff** introduces additional control dimensions, which could also benefit from our stochastic modeling insights.

References

- 1 François Baccelli and Pierre Brémaud. *Elements of Queueing Theory*. Springer, 2003. doi: 10.1007/978-3-662-11657-9.
- 2 Pierre Brémaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation and Queues*. Texts in Applied Mathematics. Springer Cham, 2020.
- 3 Pierre Brémaud. *Probability Theory and Stochastic Processes*. Universitext. Springer Cham, 2020. doi:10.1007/978-3-030-40183-2.
- 4 Jeffrey P. Buzen. Computational algorithms for closed queueing networks with exponential servers. *Communications of the ACM*, 16:527–531, 1973.
- 5 Yujing Chen, Yue Ning, Martin Slawski, and Huzefa Rangwala. 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, 2020.
- 6 Zheyi Chen, Weixian Liao, Kun Hua, Chao Lu, and Wei Yu. Towards asynchronous federated learning for heterogeneous edge-powered internet of things. *Digital Communications and Networks*, 7(3):317–326, 2021.

- 7 Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical japanese literature. 2018. [arXiv:cs.CV/1812.01718](#).
- 8 Li Deng. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- 9 Rustem Islamov, Mher Safaryan, and Dan Alistarh. Asgrad: A sharp unified analysis of asynchronous-sgd algorithms. *arXiv preprint arXiv:2310.20452*, 2023.
- 10 James R Jackson. Networks of waiting lines. *Operations research*, 5(4):518–521, 1957.
- 11 Anastasiia Koloskova, Sebastian U Stich, and Martin Jaggi. Sharper convergence guarantees for asynchronous sgd for distributed and federated learning. *Advances in Neural Information Processing Systems*, 35:17202–17215, 2022.
- 12 Jakub Konečný, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.
- 13 Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.
- 14 Louis Leconte, Matthieu Jonckheere, Sergey Samsonov, and Eric Moulines. Queuing dynamics of asynchronous federated learning, 2024. [arXiv:2405.00017](#).
- 15 John D. C. Little. Little’s law as viewed on its 50th anniversary. *Operations Research*, 59(3):536–549, 06 2011. doi:10.1287/opre.1110.0940.
- 16 Maksim Makarenko, Elnur Gasanov, Rustem Islamov, Abdurakhmon Sadiev, and Peter Richtarik. Adaptive compression for communication-efficient distributed training. *arXiv preprint arXiv:2211.00188*, 2022.
- 17 Yuzhu Mao, Zihao Zhao, Guangfeng Yan, Yang Liu, Tian Lan, Linqi Song, and Wenbo Ding. Communication-efficient federated learning with adaptive quantization. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 13(4):1–26, 2022.
- 18 Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.
- 19 John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Michael G. Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. *ArXiv*, abs/2106.06639, 2021.
- 20 Linping Qu, Shenghui Song, and Chi-Ying Tsui. Feddq: Communication-efficient federated learning with descending quantization. *arXiv preprint arXiv:2110.02291*, 2021.
- 21 R. Serfozo. *Introduction to Stochastic Networks*. Stochastic Modelling and Applied Probability. Springer New York, 1999.
- 22 Rajan Suri. A concept of monotonicity and its characterization for closed queueing networks. *Operations Research*, 33(3):606–624, 1985. doi:10.1287/opre.33.3.606.
- 23 Alexander Tyurin and Peter Richtárik. Dasha: Distributed nonconvex optimization with communication compression, optimal oracle complexity, and no client synchronization. *arXiv preprint arXiv:2202.01268*, 2022.
- 24 Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.
- 25 Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*, 2019.
- 26 Chenhao Xu, Youyang Qu, Yong Xiang, and Longxiang Gao. Asynchronous federated learning on heterogeneous devices: A survey. *arXiv preprint arXiv:2109.04269*, 2021.

A Assumptions

Our analysis is grounded in the assumptions presented in Section 2.1, which align with those established in [14]. These assumptions are detailed as follows:

A1 Lower Boundedness: The objective function f is bounded from below by some real number f^* , meaning $f(w) \geq f^*$ for all $w \in \mathbb{R}^d$.

A2 Gradient Smoothness: Each client's local function f_i has an L -Lipschitz continuous gradient, where $L > 0$. Mathematically, for any vectors $w, \mu \in \mathbb{R}^d$:

$$\|\nabla f_i(w) - \nabla f_i(\mu)\| \leq L\|w - \mu\|.$$

A3 Stochastic Gradient Properties: For each client i , the stochastic gradient $g_i(w)$ is an unbiased estimator of the gradient $\nabla f_i(w)$ with bounded variance $\sigma^2 > 0$. That is, for all $w \in \mathbb{R}^d$:

$$\mathbb{E}[g_i(w) - \nabla f_i(w)] = 0 \text{ and } \mathbb{E}[\|g_i(w) - \nabla f_i(w)\|^2] \leq \sigma^2.$$

A4 Bounded Client Heterogeneity: There exist constant $M > 0$, such that for all $w \in \mathbb{R}^d$:

$$\|\nabla f(w) - \nabla f_i(w)\|^2 \leq M^2.$$

A5 Stationary Regime: The dynamics of the closed Jackson network are assumed to be in their stationary regime. Specifically, the random n -dimensional vector $\xi(t)$, where the i -th component $\xi_i(t)$ represents the number of tasks at client i at time t , follows its stationary distribution $\pi_{n,m}$.

B Proof of Proposition 2.1

The result follows by observing that $(X_t, t \in \mathbb{N})$ tracks the state at departure times of a Jackson network [10, 21] with n clients (usually called *servers*) and m tasks (often called *customers* or *jobs*), with service rate vector μ and relative arrival rate vector p . In particular, (4) follows from Chapter 1 (Definition 1.8 and Theorem 1.12) and Section 4.8 (Definition 4.34 and Example 4.38) in [21].

C Proof of Theorem 3.1

Consider the asynchronous FL framework of Section 2.1. After proving preliminary results in Lemma C.1 and Corollary C.2, we prove (6) in Appendix C.1 and (7) in Appendix C.2.

► **Lemma C.1.** For each $i \in \{1, 2, \dots, n\}$, for each $t \in \mathbb{N}_{>0}$ we have

$$R_{i,t+1} = \begin{cases} \max(0, R_{i,t} - 1) & \text{if } A_{t+1} \neq i, \\ \min\{r \geq R_{i,t} : C_{t+1+r} = i\} & \text{if } A_{t+1} = i. \end{cases}$$

Proof. Let $i \in \{1, 2, \dots, n\}$ and $t \in \mathbb{N}_{>0}$.

The following preliminary results stem from the definition (3) of $R_{i,t}$ and the observation that the sequence $r \in \mathbb{N} \mapsto \sum_{s=t}^{t+r} \mathbb{1}[C_s = i]$ is nondecreasing with increments 0 or 1 and takes value $\mathbb{1}[C_t = i] \in \{0, 1\}$ at $r = 0$ (equal to 0 if $X_{i,t-1} + \mathbb{1}[A_t = i] = 0$ by definition of C_t):

■ We can replace the equal sign with a larger-than-or-equal sign in the definition of $R_{i,t}$:

$$R_{i,t} = \min \left\{ r \in \mathbb{N} : \sum_{s=t}^{t+r} \mathbb{1}[C_s = i] \geq X_{i,t-1} + \mathbb{1}[A_t = i] \right\}. \quad (14)$$

■ We have

$$\sum_{s=t}^{t+R_{i,t}} \mathbb{1}[C_s = i] = X_{i,t-1} + \mathbb{1}[A_t = i]. \quad (15)$$

- Lastly, we can verify that $A_t = i$ implies $C_{t+R_{i,t}} = i$.

Now focusing on $R_{i,t+1}$, we have successively:

$$\begin{aligned}
R_{i,t+1} &\stackrel{(a)}{=} \min \left\{ r \in \mathbb{N} : \sum_{s=t+1}^{t+1+r} \mathbb{1}[C_s = i] \geq X_{i,t} + \mathbb{1}[A_{t+1} = i] \right\}, \\
&\stackrel{(b)}{=} \min \left\{ r \in \mathbb{N} : \sum_{s=t}^{t+1+r} \mathbb{1}[C_s = i] \geq X_{i,t-1} + \mathbb{1}[A_t = i] + \mathbb{1}[A_{t+1} = i] \right\}, \\
&\stackrel{(c)}{=} \min \left\{ r \geq \max(0, R_{i,t} - 1) : \sum_{s=t}^{t+1+r} \mathbb{1}[C_s = i] \geq X_{i,t-1} + \mathbb{1}[A_t = i] + \mathbb{1}[A_{t+1} = i] \right\}, \\
&\stackrel{(d)}{=} \min \left\{ r \geq \max(0, R_{i,t} - 1) : \sum_{s=t+R_{i,t}+1}^{t+1+r} \mathbb{1}[C_s = i] \geq \mathbb{1}[A_{t+1} = i], \right\}
\end{aligned}$$

where (a) and (c) follow from (14), (b) from (1), and (d) by injecting (15). Consistently with the result accounted in the lemma, we conclude by making a case disjunction:

- If $A_{t+1} \neq i$: The right-hand side of the inequality in (d) is 0, hence the inequality is already satisfied by choosing the smallest authorized value for $R_{i,t}$.
- If $A_{t+1} = i$: The right-hand side of the inequality in (d) is 1. Since the sum $\sum_{s=t+R_{i,t}+1}^{t+1+r} \mathbb{1}[C_s = i]$ is 0 (empty) when $r = R_{i,t} - 1$, remains 0 as long as $C_{t+1+r} \neq i$, and increases to 1 whenever $C_{t+1+r} = i$, the result follows. ◀

► **Corollary C.2.** For each $i \in \{1, 2, \dots, n\}$, the function $t \in \mathbb{N}_{>0} \mapsto t + D_{i,t}$ defines a bijective increasing mapping from the set $\{t \in \mathbb{N}_{>0} : A_t = i\}$ onto the set $\{t \in \mathbb{N}_{>0} : C_t = i\}$.

Proof. Let $i \in \{1, 2, \dots, n\}$. By definition of $D_{i,t}$, we have $D_{i,t} = R_{i,t}$ for each $t \in \mathbb{N}_{>0}$ so that $A_t = i$, hence it suffices to prove the result for $R_{i,t}$ instead of $D_{i,t}$.

For each $t \in \mathbb{N}_{>0}$, $t + R_{i,t}$ is the round at the end of which client i finishes processing the last task that has arrived at this client by the beginning of round t . Lemma C.1 implies that, for each $t \in \mathbb{N}_{>0}$, we have

$$t + 1 + R_{i,t+1} = \begin{cases} t + \max(1, R_{i,t}) & \text{if } A_{t+1} \neq i, \\ \min\{s > t + R_{i,t} : C_s = i\} & \text{if } A_{t+1} = i. \end{cases}$$

It follows directly that the function $t \in \mathbb{N}_{>0} \mapsto t + R_{i,t}$ is nondecreasing and defines an injection from $\{t \in \mathbb{N}_{>0} : A_t = i\}$ onto $\{t \in \mathbb{N}_{>0} : C_t = i\}$. Furthermore, to prove this injection is also a surjection, it suffices to verify that $A_{t+1} \neq i$ implies either $t + 1 + R_{i,t+1} = t + R_{i,t}$ or $C_{t+1+R_{i,t+1}} \neq i$. If $A_{t+1} \neq i$, the only way that $t + 1 + R_{i,t+1} > t + R_{i,t}$ is when $R_{i,t} = 0$, so that the maximum is attained at 1 and $t + 1 + R_{i,t+1} = t + 1$. Using (3), we can verify that $R_{i,t} = 0$ implies $X_{i,t} = 0$ which, combined with $A_{t+1} \neq i$, in turn implies $C_{t+1} (= C_{t+1+R_{i,t+1}}) \neq i$ by definition of C_{t+1} . ◀

C.1 Proof of Equation (6)

Let $i \in \{1, 2, \dots, n\}$. Our goal in this section is to prove that $\mathbb{E}[D_i] = \mathbb{E}[X_i]$. In a nutshell, we will prove that

$$\mathbb{E}[D_i] \stackrel{(a)}{=} \lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t=1}^T D_{i,t} \stackrel{(b)}{=} \lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t=1}^T X_{i,t} \stackrel{(c)}{=} \mathbb{E}[X_i],$$

where all equal signs hold almost surely. Equation (a) will be shown in Lemma C.3 using an ergodicity argument. Equation (b) will be proved using a squeeze argument formulated in Lemmas C.4 and C.5. Lastly, Equation (c) follows from the classical ergodic theorem for irreducible positive-recurrent Markov chains. These arguments are combined at the end of the section to prove (6).

► **Lemma C.3.** *For each $i \in \{1, 2, \dots, n\}$, we have*

$$\lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t=1}^T D_{i,t} = \mathbb{E}[D_i] \quad \text{almost surely.}$$

Proof of Lemma C.3. Equations (2) and (3) show that, for each $i \in \{1, 2, \dots, n\}$, there exists a deterministic function $g_i : (\mathbb{N} \times \{1, 2, \dots, n\} \times \{1, 2, \dots, n\})^{\mathbb{N}} \rightarrow \mathbb{N}$ such that we can write $D_{i,t} = g_i((X_{t+s}, A_{t+s}, C_{t+s}), s \in \mathbb{N})$ for each $t \in \mathbb{N}$. Since the sequence $((X_t, A_t, C_t), t \in \mathbb{N})$ is ergodic, the conclusion follows from [3, Remark 16.1.11]. ◀

► **Lemma C.4.** *Let $i \in \{1, 2, \dots, n\}$. If $X_{i,0} = 0$, then for each $T \in \{1, 2, 3, \dots\}$, we have*

$$\sum_{t=1}^T X_{i,t} \leq \sum_{t=1}^T D_{i,t} \leq \sum_{t=1}^{T+R_{i,T}} X_{i,t}. \quad (16)$$

Proof of Lemma C.4. Let $i \in \{1, 2, \dots, n\}$ and $T \in \mathbb{N}_{>0}$, and assume that $X_{i,0} = 0$. If the set $\{t \in \{1, 2, \dots, T\} : A_t = i\}$ is empty, then all three sums are zero, and the inequalities are trivially satisfied. Therefore, in the remainder, we focus on sample paths for which this set is nonempty.

First observe that, for each $t \in \mathbb{N}$, we have

$$X_{i,s} = \sum_{t=1}^{+\infty} \mathbb{1}[A_t = i, t \leq s < t + D_{i,t}]. \quad (17)$$

Indeed, we have successively:

$$X_{i,s} \stackrel{(a)}{=} \sum_{t=1}^s \mathbb{1}[A_t = i] - \sum_{t=1}^s \mathbb{1}[C_t = i] \stackrel{(b)}{=} \sum_{t=1}^s \mathbb{1}[A_t = i] - \sum_{t=1}^s \mathbb{1}[A_t = i, t + D_{i,t} \leq s],$$

where (a) follows by unfolding (1) and using our assumption that $X_{i,0} = 0$, and (b) follows by observing that, by Corollary C.2, the function $t \mapsto t + D_{i,t}$ defines a bijective (increasing) mapping from the set $\{t \in \mathbb{N}_{>0} : A_t = i\}$ onto the set $\{t \in \mathbb{N}_{>0} : C_t = i\}$. Equation (17) then follows by rearranging the terms.

Now, we can also use the following trick to rewrite $\sum_{t=1}^T D_{i,t}$ in a manner that is similar to (17):

$$\begin{aligned} \sum_{t=1}^T D_{i,t} &= \sum_{t=1}^T \mathbb{1}[A_t = i] D_{i,t} = \sum_{t=1}^T \mathbb{1}[A_t = i] \sum_{s=1}^{+\infty} \mathbb{1}[t \leq s < t + D_{i,t}], \\ &= \sum_{s=1}^{+\infty} \sum_{t=1}^T \mathbb{1}[A_t = i, t \leq s < t + D_{i,t}]. \end{aligned} \quad (18)$$

The lower bound in (16) follows by capping the outer sum at $s \in \{1, 2, \dots, T\}$ in (18) and injecting (17). The upper-bound in (16) follows in a similar spirit:

$$\sum_{t=1}^T D_{i,t} \stackrel{(a)}{=} \sum_{s=1}^{T+R_{i,T}} \sum_{t=1}^T \mathbb{1}[A_t = i, t \leq s < t + D_{i,t}] \stackrel{(b)}{\leq} \sum_{s=1}^{T+R_{i,T}} X_{i,s},$$

where (a) is equivalent to (18), after recalling that tasks leave clients in the same order as they arrive, and (b) follows by expanding the inner sum to $t \in \mathbb{N}_{>0}$. ◀

► **Lemma C.5.** *For each $i \in \{1, 2, \dots, n\}$, we have almost surely that $R_{i,T} = o(T)$ as $T \rightarrow +\infty$.*

Proof of Lemma C.5. Let $i \in \{1, 2, \dots, n\}$. For each $t \in \mathbb{N}$, we have $0 \leq R_{i,t} \leq E_{i,t}$, where $E_{i,t} = \min\{r \in \mathbb{N} : X_{i,t+r} = 0\}$ is the number of rounds that complete, starting from round t , before client i first becomes empty. Since the Markov chain $(X_t, t \in \mathbb{N})$ is positive recurrent and has a finite state space, $\mathbb{E}[E_{i,t}]$ has a finite limit (as the expectation, under the stationary distribution of this Markov chain, of the expected hitting time of the set of states where client i is empty). It follows that $\lim_{t \rightarrow +\infty} \mathbb{E}[E_{i,t}]/t = 0$, which implies that $\lim_{t \rightarrow +\infty} E_{i,t}/t = 0$ almost surely, and therefore that $\lim_{t \rightarrow +\infty} R_{i,t}/t = 0$ almost surely. ◀

Proof of Equation (6). Let $i \in \{1, 2, \dots, n\}$. Let us assume for now that the initial distribution of the Markov chain $(X_t, t \in \mathbb{N})$ is such that $X_{i,0} = 0$ (with probability 1).

Let us first prove that the upper and lower bound in Lemma C.4 converge almost surely to the same limit, and that this limit is $\mathbb{E}[X_i]$, that is:

$$\lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t=1}^T X_{i,t} \stackrel{(a)}{=} \mathbb{E}[X_i] \stackrel{(b)}{=} \lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t=1}^{T+R_{i,T}} X_{i,t}. \quad (19)$$

Since $(X_t, t \in \mathbb{N})$ is an irreducible positive-recurrent Markov chain, (a) follows directly from the ergodic theorem [3, Theorem 3.3.2]. The argument for (b) is in a similar spirit, with the extra-complication that the upper bound of summation ($T + R_{i,T}$) is different from the denominator (T). We start by rewriting the upper bound as follows:

$$\frac{1}{T} \sum_{t=1}^{T+R_{i,T}} X_{i,t} = \left(1 + \frac{R_{i,T}}{T}\right) \frac{1}{T + R_{i,T}} \sum_{t=1}^{T+R_{i,T}} X_{i,t}.$$

Equation (b) then follows by combining two arguments: (i) Lemma C.5 implies that $\lim_{T \rightarrow +\infty} 1 + \frac{R_{i,T}}{T} = 1$ almost surely; (ii) since $\lim_{T \rightarrow +\infty} T + R_{i,T} = +\infty$, the ergodic theorem for irreducible positive-recurrent Markov chains again implies that

$$\lim_{T \rightarrow +\infty} \frac{1}{T + R_{i,T}} \sum_{t=1}^{T+R_{i,T}} X_{i,t} = \mathbb{E}[X_i], \quad \text{almost surely.}$$

Now, combining (19) with Lemma C.4 and the squeeze theorem allows us to conclude that

$$\lim_{T \rightarrow +\infty} \frac{1}{T} \sum_{t=1}^T D_{i,t} = \mathbb{E}[X_i], \quad \text{almost surely.}$$

Equation (6) then follows from Lemma C.3.

To prove that (6) also holds without the assumption that $X_{i,0} = 0$ with probability 1, it suffices to recall that the expectations $\mathbb{E}[D_i]$ and $\mathbb{E}[X_i]$ do not depend on the initial distribution. ◀

C.2 Proof of Equation (7)

Let $i, j \in \{1, 2, \dots, n\}$. Our goal is to prove (7), which by (6) is equivalent to

$$\frac{\partial \mathbb{E}[X_i]}{\partial(\log p_j)} = \text{Cov}[X_i, X_j].$$

Recall that the vector X follows the stationary distribution (4), which we can rewrite as

$$\log \mathbb{P}(X = x) = \log \pi_{n,m-1}(x) = \sum_{i=1}^n x_i (\log p_i - \log \mu_i) - \log Z_{n,m-1}, \quad x \in \mathcal{X}_{n,m-1}, \quad (20)$$

where $Z_{n,m-1}$ follows by normalization:

$$\log Z_{n,m-1} = \log \left(\sum_{x \in \mathcal{X}_{n,m-1}} \exp \left(\sum_{i=1}^n x_i (\log p_i - \log \mu_i) \right) \right). \quad (21)$$

Let us first prove the following intermediary result:

$$\frac{\partial \log Z_{n,m-1}}{\partial(\log p_j)} = \mathbb{E}[X_j], \quad \frac{\partial \log \pi_{n,m-1}(x)}{\partial(\log p_j)} = x_j - \mathbb{E}[X_j], \quad x \in \mathcal{X}_{n,m-1}. \quad (22)$$

The first part of (22) follows by taking the partial derivative of (21) and rearranging the terms to retrieve the definition of $\pi_{n,m-1}$:

$$\begin{aligned} \frac{\partial \log(Z_{n,m-1})}{\partial(\log p_j)} &= \frac{1}{Z_{n,m-1}} \frac{\partial Z_{n,m-1}}{\partial(\log p_j)} = \frac{1}{Z_{n,m-1}} \sum_{x \in \mathcal{X}_{n,m-1}} x_j \exp \left(\sum_{i=1}^n x_i (\log p_i - \log \mu_i) \right), \\ &= \sum_{x \in \mathcal{X}_{n,m-1}} x_j \exp \left(\sum_{i=1}^n x_i (\log p_i - \log \mu_i) - \log Z_{n,m-1} \right), \\ &= \sum_{x \in \mathcal{X}_{n,m-1}} x_j \pi_{n,m-1}(x) = \mathbb{E}[X_j]. \end{aligned}$$

Now, the second part of (22) follows by taking the partial derivative of (20) and injecting the previous result:

$$\frac{\partial \log \pi_{n,m-1}(x)}{\partial(\log p_j)} = x_j - \frac{\partial \log Z_{n,m-1}}{\partial(\log p_j)} = x_j - \mathbb{E}[X_j], \quad x \in \mathcal{X}_{n,m-1}.$$

To conclude, it suffices to inject the second part of (22) into the definition of expectation:

$$\begin{aligned} \frac{\partial \mathbb{E}[X_i]}{\partial(\log p_j)} &= \sum_{x \in \mathcal{X}_{n,m-1}} x_i \frac{\partial \pi_{n,m-1}(x)}{\partial(\log p_j)} = \sum_{x \in \mathcal{X}_{n,m-1}} \pi_{n,m-1}(x) x_i \frac{\partial \log \pi_{n,m-1}(x)}{\partial(\log p_j)}, \\ &= \sum_{x \in \mathcal{X}_{n,m-1}} \pi_{n,m-1}(x) x_i (x_j - \mathbb{E}[X_j]) = \text{Cov}[X_i, X_j]. \end{aligned}$$

C.3 Proof of Equations (8) and (9)

Equation (8) was proved in [4, Equation (8)]. Buzen's algorithm [4] as described in the proposition was introduced and shown to yield the correct result in [4, Paragraph "Computation of $G(N)$ "]. (Note that, in [4], N corresponds to our m and M to our n .) All that remains is

to prove (9), which we do in a similar way to the proof of (8) in [4]. To simplify notation, we can focus without loss of generality on the pair $(i, j) = (1, 2)$. We have successively:

$$\begin{aligned}
\mathbb{E}[X_1 X_2] &= \sum_{x \in \mathcal{X}_{n, m-1}} x_1 x_2 \pi_{n, m-1}(x) = \sum_{x \in \mathcal{X}_{n, m-1}} \sum_{k=1}^{x_1} \sum_{\ell=1}^{x_2} \pi_{n, m-1}(x), \\
&= \sum_{\substack{k, \ell=1 \\ k+\ell \leq m-1}}^{m-1} \sum_{\substack{x \in \mathcal{X}_{n, m-1} \\ x_1 \geq k, x_2 \geq \ell}} \pi_{n, m-1}(x) = \frac{1}{Z_{n, m-1}} \sum_{\substack{k, \ell=1 \\ k+\ell \leq m-1}}^{m-1} \sum_{\substack{x \in \mathcal{X}_{n, m-1} \\ x_1 \geq k, x_2 \geq \ell}} \prod_{i=1}^n \left(\frac{p_i}{\mu_i} \right)^{x_i}, \\
&\stackrel{(*)}{=} \frac{1}{Z_{n, m-1}} \sum_{\substack{k, \ell=1 \\ k+\ell \leq m-1}}^{m-1} \left(\frac{p_1}{\mu_1} \right)^k \left(\frac{p_2}{\mu_2} \right)^\ell \sum_{y \in \mathcal{X}_{n, m-1-k-\ell}} \prod_{i=1}^n \left(\frac{p_i}{\mu_i} \right)^{y_i}, \\
&= \frac{1}{Z_{n, m-1}} \sum_{\substack{k, \ell=1 \\ k+\ell \leq m-1}}^{m-1} \left(\frac{p_1}{\mu_1} \right)^k \left(\frac{p_2}{\mu_2} \right)^\ell Z_{n, m-1-k-\ell},
\end{aligned}$$

where $(*)$ follows by making the change of variable $y = x - ke_1 - \ell e_2$, where e_i is the n -dimensional vector with one in component i and zero elsewhere.

D Proof of Theorem 4.1

To prove Theorem 4.1, we build on the virtual iterates framework introduced in [11] and further refined in [14]. Our proof follows the same reasoning as the proof of [14, Theorem 1], with necessary modifications to incorporate the duration of each round. The analysis relies on the assumptions outlined in Section A.

For $k \in \mathbb{N}_{>0}$, define the filtration \mathcal{F}_k as:

$$\mathcal{F}_k = \sigma(w_l, A_m, 0 \leq l \leq k, 1 \leq m \leq k-1),$$

which represents the history of updates and client selections up to the end of iteration $k-1$. The virtual iterates μ_k are then defined recursively as:

$$\begin{cases} \mu_0 = w_0, \\ \mu_1 = \mu_0 - \eta \sum_{i=1}^n \frac{\xi_i}{np_i} g_i(w_0), \\ \mu_{k+1} = \mu_k - \frac{\eta}{np_{A_k}} g_{A_k}(w_k), \quad k \geq 1. \end{cases}$$

The difference between μ_k and w_k captures the computation tasks dispatched by the CS, for which the resulting gradients have not yet been received at the end of round $k-1$ (i.e., just after receiving the gradient from client C_{k-1} and before dispatching a new task to client A_k).

Now, consider the system described in Section 2.1, where τ_t denotes the duration of round t , and $\tilde{\tau}_t$ represents the time at which the round t concludes, for $t \in \{0, 1, \dots, T\}$. $\tilde{\tau}_t$ is given by:

$$\tilde{\tau}_t = \sum_{k=0}^t \tau_k, \quad t \in \{0, 1, \dots, T\}.$$

The system operates over a (random) total training time of $\tilde{\tau}_T$.

We begin by presenting the following lemma:

► **Lemma D.1.** *In the framework of Section 2.1, assuming that $\eta \leq \frac{n^2}{8L \sum_{i=1}^n \frac{1}{p_i}}$, the following inequality holds:*

$$\begin{aligned} \frac{1}{T+1} \sum_{k=0}^T \tau_k \|\nabla f(w_k)\|^2 &\leq \frac{4}{\eta(T+1)} \sum_{k=0}^T \tau_k (f(\mu_k) - \mathbb{E}[f(\mu_{k+1}) | \mathcal{F}_k]) \\ &\quad + \frac{2L^2}{T+1} \sum_{k=0}^T \tau_k \|\mu_k - w_k\|^2 + \frac{4\eta LB}{n^2(T+1)} \sum_{i=1}^n \frac{1}{p_i} \sum_{k=0}^T \tau_k, \end{aligned}$$

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

Proof of Lemma D.1. We begin by revisiting an intermediate result from the proof of Lemma 6 in [14], which is verified under the same assumptions as Lemma D.1. This result, restated below, holds for each $k \in \{0, 1, \dots, T\}$:

$$\begin{aligned} \|\nabla f(w_k)\|^2 &\leq \frac{4}{\eta} (f(\mu_k) - \mathbb{E}[f(\mu_{k+1}) | \mathcal{F}_k]) + 2L^2 \|\mu_k - w_k\|^2 + 8\eta L \sum_{i=1}^n \frac{M^2}{n^2 p_i} \\ &\quad + 4\eta L \sigma^2 \sum_{i=1}^n \frac{1}{n^2 p_i}. \end{aligned} \tag{23}$$

To complete the proof, we multiply (23) by $\frac{\tau_k}{T+1}$ and sum the resulting inequality over all $k \in \{0, 1, \dots, T\}$. ◀

► **Lemma D.2.** *In the framework of Section 2.1 and under the condition $\eta \leq \eta_{max}$, where*

$$\eta_{max} = \min \left(\frac{n^2}{8L \sum_{i=1}^n \frac{1}{p_i}}, \min_{k \in \{0, \dots, T\}} \frac{n}{4L} \sqrt{\frac{\tau_k}{m \sum_{i=1}^n \frac{S_{i,k}}{p_i}}} \right),$$

the following inequality holds:

$$\begin{aligned} \frac{1}{8(T+1)} \sum_{k=0}^T \mathbb{E}[\tau_k \|\nabla f(w_k)\|^2] &\leq \frac{1}{\eta(T+1)} \sum_{k=0}^T \mathbb{E}[\tau_k \delta_k] + \frac{\eta^2 L^2 B m}{n^2} \sum_{i=1}^n \frac{1}{p_i^2} \mathbb{E} \left[\frac{1}{T+1} \sum_{k=0}^T S_{i,k} \right] \\ &\quad + \frac{\eta LB}{n^2} \mathbb{E} \left[\frac{\tilde{\tau}_T}{T+1} \right] \sum_{i=1}^n \frac{1}{p_i}. \end{aligned}$$

Here, $\delta_k = f(\mu_k) - \mathbb{E}[f(\mu_{k+1}) | \mathcal{F}_k]$, and $S_{i,k}$ represents the sojourn time (measured in physical time) of the task assigned to client i at round k , if such a task exists; otherwise, it is set to 0. To clarify, $S_{A_k, k}$ corresponds to the sojourn time of the task sent to client A_k at round k , while for all clients $i \neq A_k$, we have $S_{i,k} = 0$.

Proof of Lemma D.2. To prove Lemma D.2, we begin by deriving an upper bound for the second term on the right-hand side of the inequality stated in Lemma D.1.

Let \mathcal{Q}_k represent the multiset of tuples consisting of client indices whose gradient computation tasks are still in progress (either currently being computed or queued) at the beginning of round k (i.e., immediately after receiving the gradient from client C_{k-1} and prior to assigning a new task to client A_k), along with the corresponding round during which these gradients were dispatched to the clients.

The multiset \mathcal{Q}_k can be defined recursively as follows:

$$\mathcal{Q}_1 = \left(\bigcup_{i=1}^n \{(i, 0)\}^{\xi_{i-1}[C_0=i]} \right), \text{ and for all } k \geq 1, \mathcal{Q}_{k+1} = (\mathcal{Q}_k \cup \{(A_k, k)\}) \setminus \{(C_k, I_k)\}.$$

Recall that ξ is the initial state vector, such that ξ_i is the number of tasks initially assigned to client i , for each $i \in \{1, 2, \dots, n\}$, and $|\xi| = m$.

For each $k \in \{1, \dots, T\}$, we have [14, Lemma 9]:

$$\|\mu_k - w_k\|^2 = \left\| -\eta \sum_{(i,j) \in \mathcal{Q}_k} \frac{1}{np_i} g_i(w_j) \right\|^2 \leq \eta^2 |\mathcal{Q}_k| \sum_{(i,j) \in \mathcal{Q}_k} \frac{1}{n^2 p_i^2} \|g_i(w_j)\|^2,$$

where $|\mathcal{Q}_k|$ denotes the cardinality of the multiset \mathcal{Q}_k , and for each $k \in \mathbb{N}_{>0}$, $|\mathcal{Q}_k| = m - 1$.

Now, taking into account the weighted summation over all $T + 1$ rounds, we obtain:

$$\begin{aligned} \frac{1}{T+1} \sum_{k=0}^T \tau_k \|\mu_k - w_k\|^2 &\leq \frac{\eta^2 m}{T+1} \sum_{k=1}^T \tau_k \sum_{(i,j) \in \mathcal{Q}_k} \frac{1}{n^2 p_i^2} \|g_i(w_j)\|^2, \\ &= \frac{\eta^2 m}{T+1} \sum_{k=1}^T \tau_k \sum_{i=1}^n \frac{1}{n^2 p_i^2} \sum_{j=0}^{k-1} \|g_i(w_j)\|^2 \mathbb{1}[(i,j) \in \mathcal{Q}_k], \\ &= \frac{\eta^2 m}{T+1} \sum_{i=1}^n \frac{1}{n^2 p_i^2} \sum_{j=0}^{T-1} \|g_i(w_j)\|^2 \sum_{k=j+1}^T \tau_k \mathbb{1}[(i,j) \in \mathcal{Q}_k]. \end{aligned}$$

The equalities are derived by rearranging the terms. Finally, we observe that for all $j \in \{0, 1, \dots, T-1\}$ and $i \in \{1, 2, \dots, n\}$:

$$\sum_{k=j+1}^T \tau_k \mathbb{1}[(i,j) \in \mathcal{Q}_k] = \min(S_{i,j} - \tau_j \mathbb{1}[A_j = i], \tilde{\tau}_T - \tilde{\tau}_j) \leq S_{i,j} - \tau_j \mathbb{1}[A_j = i] \leq S_{i,j},$$

where $S_{i,j}$ represents the sojourn time of the task assigned to client i at the beginning of round j , provided that client i was the one to receive the task ($A_j = i$), otherwise, $S_{i,j}$ is defined as 0. By taking the expectation, we obtain:

$$\begin{aligned} \mathbb{E} \left[\frac{1}{T+1} \sum_{k=0}^T \tau_k \|\mu_k - w_k\|^2 \right] &\leq \frac{\eta^2 m}{T+1} \sum_{i=1}^n \frac{1}{n^2 p_i^2} \sum_{j=0}^{T-1} \mathbb{E} [\|g_i(w_j)\|^2 S_{i,j}], \\ &\leq \frac{\eta^2 m}{T+1} \sum_{i=1}^n \frac{1}{n^2 p_i^2} \sum_{j=0}^T \mathbb{E} [\|g_i(w_j)\|^2 S_{i,j}]. \end{aligned} \quad (24)$$

Next, we examine the term $\mathbb{E} [\|g_i(w_j)\|^2 S_{i,j}]$ for all $j \in \{0, 1, \dots, T\}$ and $i \in \{1, 2, \dots, n\}$. Using the law of total expectation, we have:

$$\mathbb{E} [\|g_i(w_j)\|^2 S_{i,j}] = \mathbb{E} (\mathbb{E} [\|g_i(w_j)\|^2 S_{i,j} | w_j]) = \mathbb{E} (\mathbb{E} [\|g_i(w_j)\|^2 | w_j] \mathbb{E} [S_{i,j} | w_j]). \quad (25)$$

To bound $\mathbb{E} [\|g_i(w_j)\|^2 | w_j]$, we utilize the model assumptions. Expanding and simplifying, we get:

$$\begin{aligned} \mathbb{E} [\|g_i(w_j)\|^2 | w_j] &= \mathbb{E} [\|g_i(w_j) - \nabla f_i(w_j) + \nabla f_i(w_j)\|^2 | w_j], \\ &\leq 2\mathbb{E} [\|g_i(w_j) - \nabla f_i(w_j)\|^2 | w_j] + 2\mathbb{E} [\|\nabla f_i(w_j)\|^2 | w_j], \\ &\leq 2\sigma^2 + 2\mathbb{E} [\|\nabla f_i(w_j)\|^2 | w_j] \quad (\text{using Assumption A3}), \\ &\leq 2\sigma^2 + 4\mathbb{E} [\|\nabla f_i(w_j) - \nabla f(w_j)\|^2 | w_j] + 4\mathbb{E} [\|\nabla f(w_j)\|^2 | w_j], \\ &\leq 2\sigma^2 + 4M^2 + 4\mathbb{E} [\|\nabla f(w_j)\|^2 | w_j] \quad (\text{using Assumption A4}). \end{aligned}$$

Substituting this bound into Equation (25) and defining $B = \sigma^2 + 2M^2$, we find:

$$\mathbb{E} [\|g_i(w_j)\|^2 S_{i,j}] \leq 2\mathbb{E} [(2\|\nabla f(w_j)\|^2 + B) S_{i,j}].$$

Plugging this result into inequality (24), we get:

$$\mathbb{E} \left[\frac{1}{T+1} \sum_{k=0}^T \tau_k \|\mu_k - w_k\|^2 \right] \leq \frac{2\eta^2 m}{T+1} \sum_{i=1}^n \frac{1}{n^2 p_i^2} \sum_{j=0}^T \mathbb{E} [(2\|\nabla f(w_j)\|^2 + B) S_{i,j}]. \quad (26)$$

Finally, using the definition of $\tilde{\tau}_T$, we have:

$$\frac{1}{T+1} \sum_{k=0}^T \tau_k = \frac{\tilde{\tau}_T}{T+1}. \quad (27)$$

By taking the expectation of the inequality established in Lemma D.1 and incorporating inequality (26) along with equation (27), we obtain the following:

$$\begin{aligned} \frac{1}{T+1} \sum_{k=0}^T \mathbb{E} [\tau_k \|\nabla f(w_k)\|^2] &\leq \frac{4}{\eta(T+1)} \sum_{k=0}^T \mathbb{E} [\tau_k \delta_k] \\ &\quad + \frac{4\eta^2 L^2 m}{n^2} \sum_{i=1}^n \frac{1}{p_i^2} \frac{1}{T+1} \sum_{k=0}^T \mathbb{E} [(2\|\nabla f(w_k)\|^2 + B) S_{i,k}] \\ &\quad + \frac{4\eta LB}{n^2} \sum_{i=1}^n \mathbb{E} \left[\frac{\tilde{\tau}_T}{T+1} \right] \frac{1}{p_i}. \end{aligned}$$

Therefore, we conclude:

$$\begin{aligned} &\frac{1}{T+1} \sum_{k=0}^T \mathbb{E} \left[\tau_k \left(\frac{1}{4} - 2 \frac{\eta^2 L^2 m}{\tau_k} \sum_{i=1}^n \frac{S_{i,k}}{n^2 p_i^2} \right) \|\nabla f(w_k)\|^2 \right] \\ &\leq \frac{1}{\eta(T+1)} \sum_{k=0}^T \mathbb{E} [\tau_k \delta_k] + \frac{\eta^2 L^2 B m}{n^2} \sum_{i=1}^n \frac{1}{p_i^2} \mathbb{E} \left[\frac{1}{T+1} \sum_{k=0}^T S_{i,k} \right] + \frac{\eta LB}{n^2} \mathbb{E} \left[\frac{\tilde{\tau}_T}{T+1} \right] \sum_{i=1}^n \frac{1}{p_i}. \end{aligned}$$

The result follows by taking:

$$\eta \leq \min_{k \in \{0,1,\dots,T\}} \frac{n}{4L} \sqrt{\frac{\tau_k}{m \sum_{i=1}^n \frac{S_{i,k}}{p_i^2}}}.$$

◀

The last lemma we need to introduce before proving Theorem 4.1 is as follows:

► **Lemma D.3.** *In the framework of Section 2.1, we have:*

$$\mathbb{E} \left[\frac{\tilde{\tau}_T}{T+1} \right] = \frac{1}{\lambda}, \quad (28)$$

$$\mathbb{E} \left[\frac{1}{T+1} \sum_{k=0}^T S_{i,k} \right] = \frac{\mathbb{E}[\xi_i]}{\lambda}, \quad i \in \{1, 2, \dots, n\}, \quad (29)$$

$$\mathbb{E} (\tau_k (f(\mu_k) - \mathbb{E} [f(\mu_{k+1}) | \mathcal{F}_k])) = \frac{1}{\lambda} \sum_{x \in \mathcal{X}_{n,m}} \pi_{n,m}(x) \mathbb{E} (f(\mu_k) - f(\mu_{k+1}) | Y_k = x), \quad (30)$$

where the last equality holds for each $k \in \{0, 1, \dots, T\}$. Here, $Y_k = (Y_{1,k}, Y_{2,k}, \dots, Y_{n,k})$ denotes the state of the network during round k , so that $Y_{i,k} = X_{i,k} + \mathbb{1}[A_k = i]$. λ is the throughput of the Jackson network, that is, the number of rounds per unit of physical time, given as follows, with $Z_{n,m}$ and $Z_{n,m-1}$ as defined in Proposition 2.1:

$$\lambda = \sum_{i=1}^n \mu_i \mathbb{P}(\xi_i > 0) = \frac{Z_{n,m-1}}{Z_{n,m}}. \quad (31)$$

The random vector $\xi \in \mathcal{X}_{n,m}$ follows the distribution $\pi_{n,m}$ given in (4).

Observe that Equation (29) serves as the continuous-time counterpart to Equation (6). Similarly, analogous expressions for Equation (7), (8), and (9) can be readily derived for ξ by following the same reasoning as in Theorem 3.1.

Proof of Lemma D.3. We prove each equation individually.

Proof of Equation (31). We start by proving Equation (31). The throughput λ is defined as the average number of rounds per unit time. Mathematically, for a given physical time window $s \in \mathbb{R}_{>0}$, the throughput can be expressed as:

$$\begin{aligned} \lambda &= \mathbb{E} \left[\frac{1}{s} \sum_{k \geq 0} \mathbb{1}[\tilde{\tau}_k \leq s] \right] = \mathbb{E} \left[\frac{1}{s} \int_0^s \sum_{i=1}^n \mu_i \mathbb{1}[\xi_i(t) > 0] dt \right], \\ &= \sum_{i=1}^n \mu_i \mathbb{P}(\xi_i > 0) = \sum_{i=1}^n \mu_i \frac{p_i}{\mu_i} \frac{Z_{n,m-1}}{Z_{n,m}} = \frac{Z_{n,m-1}}{Z_{n,m}}. \end{aligned}$$

The first equality follows from the definition of throughput. The second equality uses the stochastic intensity formula [1, Section 1.8.3 "Stochastic Intensity Integration Formula"], considering the point process $\{\tilde{\tau}_k\}_{k \in \mathbb{N}}$ with its stochastic intensity $\sum_{i=1}^n \mu_i \mathbb{1}[\xi_i(t) > 0]$. The third equality arises by interchanging the expectation and the integral under stationarity assumptions. The fourth equality replaces $\mathbb{P}(\xi_i > 0)$ according to [4, Equation (6)]. This concludes the proof.

Proof of Equation (28). By the definition of $\tilde{\tau}_T$, we have:

$$\mathbb{E}[\tilde{\tau}_T] = \sum_{k=0}^T \mathbb{E}[\tau_k].$$

Recall that $Y_t = (Y_{1,t}, Y_{2,t}, \dots, Y_{n,t})$, where $t \in \mathbb{N}$, represents the state of the network at round t , so that $Y_{i,t} = X_{i,t} + \mathbb{1}[A_t = i]$. The sequence $\{Y_t\}_{t \in \mathbb{N}}$ can be recursively defined as follows:

- $Y_0 = \xi$ represents the initial state of the network during the first round of training.
- For each $t \in \mathbb{N}_{>0}$ and $i \in \{1, \dots, n\}$, $Y_{i,t} = Y_{i,t-1} + \mathbb{1}[A_t = i] - \mathbb{1}[C_{t-1} = i]$.

We know that the sequence $\{\tau_k | Y_k\}_{k \in \{0, \dots, T\}}$ consists of independent random variables, such that $\tau_k | Y_k$ is exponentially distributed with a parameter $\sum_{i=1}^n \mu_i \mathbb{1}[Y_{i,k} \geq 1]$, for each $k \in \{0, 1, \dots, T\}$. Conditioning on Y_k , we can write:

$$\mathbb{E}[\tilde{\tau}_T] = \sum_{k=0}^T \sum_{x \in \mathcal{X}_{n,m}} \mathbb{E}[\tau_k | Y_k = x] \mathbb{P}(Y_k = x) = \sum_{k=0}^T \sum_{x \in \mathcal{X}_{n,m}} \frac{1}{\sum_{i=1}^n \mu_i \mathbb{1}[x_i \geq 1]} \mathbb{P}(Y_k = x).$$

The sequence $\{Y_k\}_{k \in \mathbb{N}}$ forms an ergodic, discrete, homogeneous Markov chain. Its stationary distribution can be derived by observing that it corresponds to the jump chain of

the ergodic, time-continuous Markov chain $\{\xi_t\}_{t \geq 0}$. Using Equation (13.56) of Theorem 13.4.5 from [2], the stationary distribution of $\{Y_k\}_{k \in \mathbb{N}}$ is given by:

$$\mathbb{P}(Y_k = x) = \frac{1}{V_{n,m}} \sum_{i=1}^n \mu_i \mathbb{1}[x_i \geq 1] \prod_{j=1}^n \left(\frac{p_j}{\mu_j} \right)^{x_j}, \quad x \in \mathcal{X}_{n,m}, \quad (32)$$

where $V_{n,m}$ is a normalizing constant.

Under the stationarity assumption (A5) and substituting (32) into the previously derived expression for $\mathbb{E}[\tilde{\tau}_T]$, we obtain:

$$\mathbb{E}[\tilde{\tau}_T] = \sum_{k=0}^T \sum_{x \in \mathcal{X}_{n,m}} \frac{1}{V_{n,m}} \prod_{j=1}^n \left(\frac{p_j}{\mu_j} \right)^{x_j} \stackrel{(*)}{=} (T+1) \frac{Z_{n,m}}{V_{n,m}}. \quad (33)$$

Next, we demonstrate that $V_{n,m} = Z_{n,m-1}$. By definition, we have:

$$\begin{aligned} V_{n,m} &= \sum_{x \in \mathcal{X}_{n,m}} \sum_{\substack{i=1 \\ x_i \geq 1}}^n \mu_i \prod_{j=1}^n \left(\frac{p_j}{\mu_j} \right)^{x_j}, \\ &\stackrel{(a)}{=} \sum_{i=1}^n \mu_i \sum_{\substack{x \in \mathcal{X}_{n,m} \\ x_i \geq 1}} \prod_{j=1}^n \left(\frac{p_j}{\mu_j} \right)^{x_j} \stackrel{(b)}{=} \sum_{i=1}^n p_i \sum_{\substack{x \in \mathcal{X}_{n,m} \\ x_i > 0}} \prod_{j=1}^n \left(\frac{p_j}{\mu_j} \right)^{(x-e_i)_j}, \end{aligned}$$

where, for each $i \in \{1, \dots, n\}$, e_i denotes the n -dimensional vector with one in component i and zero elsewhere. (a) is obtained by rearranging the order of summation, while (b) follows from factoring out $\frac{p_i}{\mu_i}$. Applying the variable substitution $y = x - e_i$, the expression simplifies as:

$$V_{n,m} = \underbrace{\sum_{i=1}^n p_i}_{=1} \underbrace{\sum_{y \in \mathcal{X}_{n,m-1}} \prod_{j=1}^n \left(\frac{p_j}{\mu_j} \right)^{y_j}}_{=Z_{n,m-1}} = Z_{n,m-1}.$$

Thus, incorporating this result into equality (*) of (33), and using (31), we conclude that:

$$\mathbb{E} \left[\frac{\tilde{\tau}_T}{T+1} \right] = \frac{Z_{n,m}}{Z_{n,m-1}} = \frac{1}{\lambda}.$$

Proof of Equation (29). To establish Equation (29), consider $i \in \{1, 2, \dots, n\}$ and $k \in \mathbb{N}$. By conditioning on the state of A_k , we have:

$$\mathbb{E}[S_{i,k}] = \sum_{j=1}^n p_j \mathbb{E}[S_{i,k} | A_k = j] = \mathbb{E}[S_{i,k} | A_k = i] p_i.$$

Here, $S_{i,k} | A_k = i$ represents the sojourn time of the task assigned to client i at round k .

Assuming stationarity (A5), and invoking Little's Law [15], we deduce that for each $k \in \mathbb{N}$:

$$\mathbb{E}[\xi_i] = p_i \lambda \mathbb{E}[S_{i,k} | A_k = i],$$

where $p_i \lambda$ is the arrival rate to client i in the closed Jackson network.

Thus, we obtain:

$$\mathbb{E} \left[\frac{1}{T+1} \sum_{k=0}^T S_{i,k} \right] = \frac{\mathbb{E}[\xi_i]}{\lambda}, \quad \forall i \in \{1, 2, \dots, n\},$$

which completes the proof.

Proof of Equation (30). For each $k \in \{0, 1, \dots, T\}$, conditioning on Y_k , we can write:

$$\begin{aligned}
& \mathbb{E}\{\tau_k(f(\mu_k) - \mathbb{E}[f(\mu_{k+1}) | \mathcal{F}_k])\} \\
&= \sum_{x \in \mathcal{X}_{n,m}} \mathbb{E}\{\tau_k(f(\mu_k) - \mathbb{E}[f(\mu_{k+1}) | \mathcal{F}_k]) | Y_k = x\} \mathbb{P}(Y_k = x), \\
&\stackrel{(c)}{=} \sum_{x \in \mathcal{X}_{n,m}} \left(\sum_{\substack{i=1 \\ x_i \geq 1}}^n \mu_i \right)^{-1} \mathbb{E}\{f(\mu_k) - \mathbb{E}[f(\mu_{k+1}) | \mathcal{F}_k] | Y_k = x\} \frac{1}{V_{n,m}} \sum_{\substack{i=1 \\ x_i \geq 1}}^n \mu_i \prod_{j=1}^n \left(\frac{p_j}{\mu_j} \right)^{x_j}, \\
&\stackrel{(d)}{=} \frac{1}{V_{n,m}} \sum_{x \in \mathcal{X}_{n,m}} \prod_{j=1}^n \left(\frac{p_j}{\mu_j} \right)^{x_j} \mathbb{E}\{f(\mu_k) - f(\mu_{k+1}) | Y_k = x\}.
\end{aligned}$$

Step (c) follows from the fact that for all $x \in \mathcal{X}_{n,m}$, the holding times $\{\tau_k | Y_k = x\}_{k \in \mathbb{N}}$ are exponentially distributed with parameter $\sum_{i=1, x_i \geq 1}^n \mu_i$ and are independent of everything else, while step (d) is obtained by simplifying the expression.

Dividing and multiplying by $Z_{n,m}$, we get:

$$\begin{aligned}
& \mathbb{E}\{\tau_k(f(\mu_k) - \mathbb{E}[f(\mu_{k+1}) | \mathcal{F}_k])\} \\
&= \frac{Z_{n,m}}{V_{n,m}} \sum_{x \in \mathcal{X}_{n,m}} \frac{1}{Z_{n,m}} \prod_{j=1}^n \left(\frac{p_j}{\mu_j} \right)^{x_j} \mathbb{E}\{f(\mu_k) - f(\mu_{k+1}) | Y_k = x\}, \\
&= \frac{1}{\lambda} \sum_{x \in \mathcal{X}_{n,m}} \pi_{n,m}(x) \mathbb{E}(f(\mu_k) - f(\mu_{k+1}) | Y_k = x).
\end{aligned}$$

This concludes the proof. \blacktriangleleft

Proof of Theorem 4.1. Taking the inequality from Lemma D.2 and applying the results of Lemma D.3, we obtain:

$$\mathbb{E} \left[\frac{1}{8(T+1)} \sum_{t=0}^T \tau_t \|\nabla f(w_t)\|^2 \right] \leq \frac{\tilde{A}}{\eta\lambda} + \frac{\eta LB}{\lambda n^2} \sum_{i=1}^n \frac{1}{p_i} + \frac{\eta^2 L^2 B m}{\lambda n^2} \sum_{i=1}^n \frac{\mathbb{E}[\xi_i]}{p_i^2},$$

where: $\tilde{A} = \frac{1}{T+1} \sum_{k=0}^T \sum_{x \in \mathcal{X}_{n,m}} \pi_{n,m}(x) \mathbb{E}(f(\mu_k) - f(\mu_{k+1}) | Y_k = x)$. \blacktriangleleft

E Compute G and $\nabla_p G$

► Proposition E.1. *In the framework of Section 2.1, the expression of the upper bound $G(p)$ in terms of routing probabilities is as follows:*

$$G(p) = \frac{A}{\eta(T+1)} + \frac{\eta LB}{n^2} \sum_{i=1}^n \frac{1}{p_i} + \frac{\eta^2 L^2 B m}{n^2} \sum_{i=1}^n \sum_{k=1}^{m-1} \frac{p_i^{k-2}}{\mu_i^k} \frac{Z_{n,m-1-k}}{Z_{n,m-1}}, \quad (34)$$

where the normalizing constants $Z_{n,m}$ for $m \in \{0, 1, \dots, m-1\}$ can be computed explicitly with $O(nm)$ time and memory complexity, as shown in Proposition 2.1.

To identify the best routing strategy, we aim to minimize this upper bound using a gradient descent algorithm. This requires computing the gradient of the function $G(p)$ with respect to the routing probability vector, as detailed in the following proposition, which is a consequence of Theorem 3.1.

► **Proposition E.2.** For each $j \in \{1, \dots, n\}$, we have

$$\frac{\partial G}{\partial p_j} = \frac{\eta LB}{n^2 p_j} \left(-\frac{1}{p_j} + \eta mL \left(\sum_{i=1}^n \frac{\text{Cov}[X_i, X_j]}{p_i^2} - \frac{2\mathbb{E}[X_j]}{p_j^2} \right) \right),$$

where $\mathbb{E}[X_j]$ and $\text{Cov}[X_i, X_j]$ can be computed explicitly with $O(nm)$ time and memory complexity by applying Buzen's algorithm, as shown in Proposition 2.1.

To ensure that the parameters $p = (p_i, i \in \{1, 2, \dots, n\})$ satisfy probability constraints, we introduce auxiliary parameters $\Theta = (\theta_i, i \in \{1, 2, \dots, n\})$ and apply the softmax function:

$$p_j = \frac{e^{\theta_j}}{\sum_{i=1}^n e^{\theta_i}}, \quad j \in \{1, \dots, n\}.$$

This method guarantees that the resulting parameters p are valid probabilities.

The gradient of the upper bound with respect to Θ is given by:

$$\frac{\partial G}{\partial \theta_j} = \left\langle \nabla_p G, \frac{\partial p}{\partial \theta_j} \right\rangle, \quad \text{where: } \frac{\partial p}{\partial \theta_j} = p_j(e_j - p), \text{ for each } j \in \{1, \dots, n\}.$$

Here, e_j denotes the n -dimensional vector with one in component j and zero elsewhere, and $\langle \cdot, \cdot \rangle$ represents the dot product in \mathbb{R}^n .

F Compute H and $\nabla_p H$

► **Proposition F.1.** In the framework of Section 2.1, the expression of the upper bound $H(p)$ in terms of the routing probabilities is given by:

$$H = \frac{Z_{n,m}}{Z_{n,m-1}} \left(\frac{\tilde{A}}{\eta} + \frac{\eta LB}{n^2} \sum_{i=1}^n \frac{1}{p_i} \right) + \frac{\eta^2 L^2 B m}{n^2} \sum_{i=1}^n \sum_{k=1}^m \frac{p_i^{k-2}}{\mu_i^k} \frac{Z_{n,m-k}}{Z_{n,m-1}}, \quad (35)$$

where the normalizing constants $Z_{n,m}$ for $m \in \{0, 1, \dots, m\}$ can be computed explicitly with $O(nm)$ time and memory complexity, as shown in Proposition 2.1.

To find the optimal routing strategy, we aim to minimize this upper bound using a gradient descent algorithm. This requires the gradient of $H(p)$ with respect to the routing probabilities, given in the following proposition.

► **Proposition F.2.** For each $j \in \{1, \dots, n\}$, the gradient $\nabla_p H$ with respect to the routing probabilities p_j is:

$$\begin{aligned} \frac{\partial H}{\partial p_j} = & \frac{\tilde{A}\mathbb{E}[\xi_j - X_j]}{\eta \lambda p_j} + \frac{\eta LB}{n^2 p_j \lambda} \left(-\frac{1}{p_j} + \mathbb{E}[\xi_j - X_j] \sum_{i=1}^n \frac{1}{p_i} \right. \\ & \left. + \eta mL \left(-\frac{2\mathbb{E}[\xi_j]}{p_j^2} + \sum_{i=1}^n \frac{\mathbb{E}[\xi_i \xi_j]}{p_i^2} - \mathbb{E}[X_j] \sum_{i=1}^n \frac{\mathbb{E}[\xi_i]}{p_i^2} \right) \right), \end{aligned}$$

where the random vectors $\xi \in \mathcal{X}_{n,m}$ and $X \in \mathcal{X}_{n,m-1}$ are distributed according to $\pi_{n,m}$ and $\pi_{n,m-1}$, respectively, as defined in (4). Furthermore, for all $(i, j) \in \{1, \dots, n\}^2$, the quantities $\mathbb{E}[X_j]$, $\mathbb{E}[\xi_j]$, and $\mathbb{E}[\xi_i \xi_j]$ can be efficiently computed with $O(nm)$ time and memory complexity using Buzen's algorithm, as described in Proposition 2.1.

G Experiments details

G.1 Neural networks architectures

The Fashion-MNIST and KMNIST datasets each contain 70,000 grayscale images, with 60,000 designated for training and 10,000 reserved for testing. These images, sized at 28×28 pixels, are evenly distributed across 10 classes. On the other hand, the CIFAR-10 and CIFAR-100 datasets feature 60,000 color images (RGB) with a resolution of 32×32 pixels. Of these, 50,000 are used for training, while 10,000 are allocated for testing. CIFAR-10 is organized into 10 classes, while CIFAR-100 expands to 100 classes. Notably, all these datasets are class-balanced, ensuring each class has an equal number of images.

For the Fashion-MNIST and KMNIST datasets, we employ a convolutional neural network (CNN) with the following structure:

- Two convolutional layers with 7×7 filters, each followed by a ReLU activation function. The first convolutional layer has 20 channels, while the second has 40 channels.
- A 2×2 max pooling layer.
- A final fully connected layer with 10 neurons, concluded by a softmax activation function.

For the CIFAR-10 and CIFAR-100 datasets, we employ a convolutional neural network (CNN) with the following architecture:

- **Three sequential convolutional blocks**, each consisting of two 3×3 convolutional layers. Each layer is followed by ReLU activation and Group Normalization. The configuration of these blocks is as follows:
 - The first block contains convolutional layers with 32 channels.
 - The second block contains convolutional layers with 64 channels.
 - The third block contains convolutional layers with 128 channels.
 Additionally, each block includes 2×2 max pooling and a dropout layer with a probability of 0.25.
- **A classification block** comprising:
 - A flattening layer.
 - A fully connected layer with 128 neurons.
 - A dropout layer with a probability of 0.25.
 - A final fully connected layer with the number of neurons corresponding to the number of classes (10 for CIFAR-10 and 100 for CIFAR-100), followed by a softmax activation function.

In all experiments, the stochastic gradient for each task is computed using a batch size of 512 data points. The numerical implementation is carried out in PyTorch, and the experiments are performed on an NVIDIA Tesla P100 GPU.

G.2 Closed Jackson network simulation

For each $t \in \{0, \dots, T\}$, recall that Y_t denote the n -dimensional random variable representing the queue lengths during round t , and τ_t represent the duration (in physical time) of round t . The Jackson network simulation proceeds by iterating through the following steps for each round t :

- Sample τ_t from an exponential distribution with rate parameter $\sum_{j=1}^n \mu_j \mathbf{1}\{Y_{j,t} > 0\}$, where μ_j represents the processing rate of client j and $Y_{j,t}$ denotes the queue length of client j at round t .

- Select a client k to complete a task at the end of round t from the set of non-empty queues. The probability of selecting client k is proportional to the processing speeds of the clients, given by the distribution:

$$\left(\frac{\mu_i \mathbb{1}[Y_{i,t} > 0]}{\sum_{j=1}^n \mu_j \mathbb{1}[Y_{j,t} > 0]}, i \in \{1, \dots, n\} \right).$$

- Reassign the task to another client l based on the routing distribution $(p_i, i \in \{1, \dots, n\})$. The queue lengths are then updated as follows:

$$Y_{t+1} = Y_t - e_k + e_l,$$

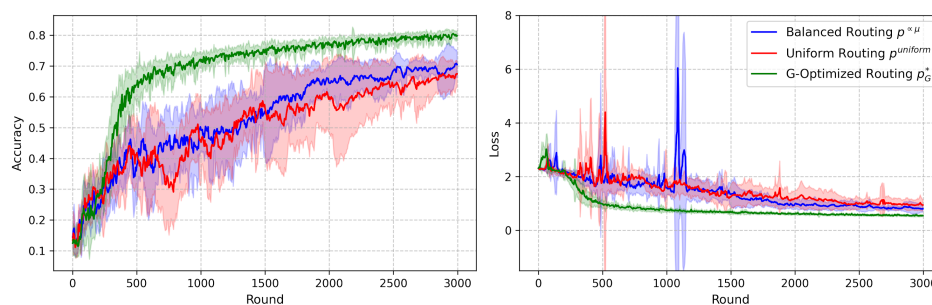
where e_k and e_l are unit vectors indicating that the task is removed from client k and added to client l , respectively. This concludes round t and initiates the next round.

G.3 Additional experiments

To evaluate the robustness of the optimized routing strategy p_G^* with respect to G across challenging heterogeneous environments, we conducted additional experiments using the Fashion-MNIST and CIFAR-100 datasets. The first scenario considered is the same as the one presented in Section 5.1, but with a different data partitioning across clients.

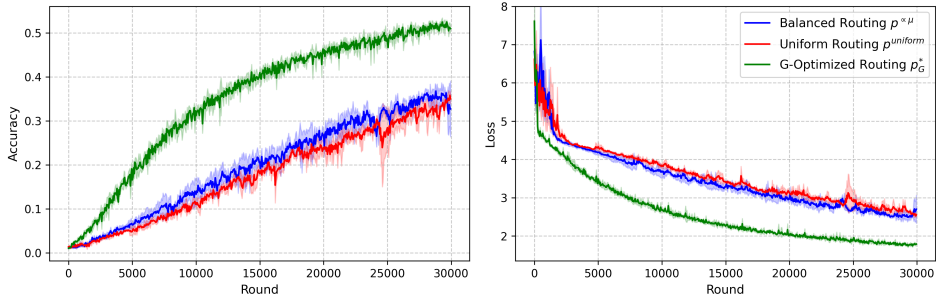
Scenario of Section 5.1 with highly-heterogeneous data distribution

Figure 5 shows the accuracy and loss trajectories over 3,000 rounds of the optimized, uniform, and balanced routing strategies, for the Fashion-MNIST dataset. Clients were each limited to 3 unique image labels out of 10, assigned sequentially and cyclically. Specifically, the first client accessed labels 0, 1, and 2; the second client, labels 3, 4, and 5; and so on, wrapping around to 0 after label 9. This ensured a non-independent and identically distributed (iid) data distribution, as each client could only access a limited subset of labels. All clients were allocated an equal number of images. We ran 10 independent simulations using different random seeds, initializing the central neural networks with identical weights for each routing strategy to eliminate the impact of initialization on convergence. Performance metrics were recorded every 5 rounds on a test dataset.



■ **Figure 5** Performance on the validation set at the CS in the scenario of Section 5.1 under highly-heterogeneous data splits using the Fashion-MNIST dataset. Solid lines show metrics averaged over independent simulations, and shaded areas represent the standard deviation.

Figure 6 compares the accuracy and loss trajectories over 30,000 rounds under the three routing strategies. We applied a similar partitioning approach, restricting each client to 5



■ **Figure 6** Performance on the validation set at the CS in the scenario of Section 5.1 under highly-heterogeneous data splits using the CIFAR-100 dataset. Solid lines show metrics averaged over independent simulations, and shaded areas represent the standard deviation.

unique labels out of the 100 available, assigned sequentially. All clients received an equal number of images. Three independent simulations were conducted with varying random seeds, yet all maintained identical initial weights for the central neural networks across the different routing strategies. Performance metrics were evaluated every 50 rounds on a test dataset.

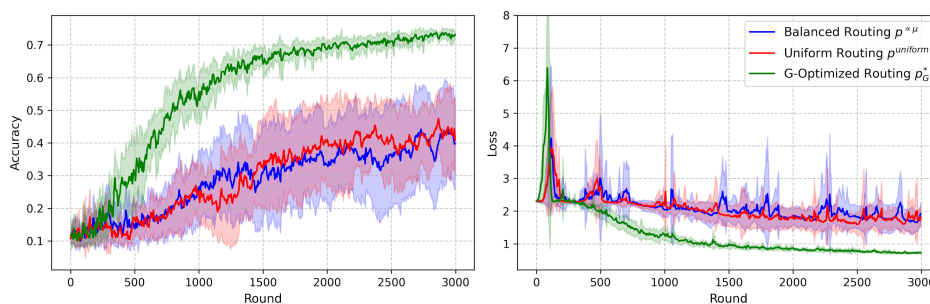
Our results show that the optimized routing strategy ultimately achieves the highest accuracy, with a notably faster performance gain compared to the other methods. In contrast, the uniform and balanced routing strategies exhibit higher volatility across independent simulations, and frequent spikes in loss.

Scenario with disjoint datasets

To further validate these findings, we examined another challenging heterogeneous setting involving a system with 10 clients and 100 tasks. In this scenario, each image label from the 10 labels in the Fashion-MNIST dataset is assigned exclusively to a single client, ensuring that each client has a completely distinct data distribution. Additionally, the service speed of each client i is defined as $\mu_i = e^{i/50}$, resulting in the fastest client being approximately 20% faster than the slowest. The learning rate is set to $\eta = 0.005$, and the smoothness constant is $L = 1$.

In Figure 7, we compare the evolution of accuracy and loss under the optimized routing strategy against the uniform and balanced routing strategies in this more complex environment. Similar to the previous experiment, training spans 3,000 rounds, with metrics recorded on a test dataset at intervals of 5 rounds.

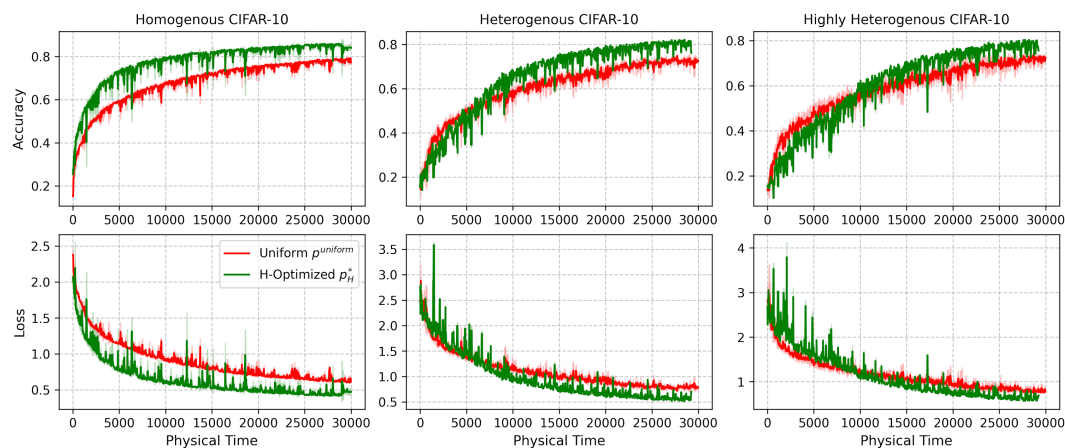
Our results further validate that the optimized routing strategy p_G^* consistently surpasses the alternatives p^{uniform} and $p^{\infty\mu}$, achieving superior accuracy with a consistently faster performance gain. Furthermore, it demonstrates greater robustness and stability in this highly heterogeneous environment, showing a smaller standard deviation in both loss and accuracy across the 10 independent simulations, along with fewer loss spikes compared to the uniform and balanced strategies. Additionally, although the optimized routing often selects the slowest client more frequently, the learning process does not develop a bias toward optimizing the local loss of this client. Instead, it preserves balanced global performance, effectively avoiding overfitting to the image labels associated with any specific client.



■ **Figure 7** Performance on the validation set at the CS with 10 clients for heterogeneous data splits across 100 tasks. Solid lines show metrics averaged over independent simulations, and shaded areas represent the standard deviation.

Scenario of Section 5.2 with CIFAR-10

To further evaluate the effectiveness of H-optimized routing in terms of physical time performance, we conduct experiments on the CIFAR-10 dataset and compare it against uniform routing across three different configurations. The first configuration considers homogeneous datasets across clients, the second involves heterogeneous datasets distributed according to a Dirichlet distribution (as described in Section 5.2), and the third represents a highly heterogeneous setting where each client has access to only three out of the ten image classes. We use the same network setup as in Section 5.2. The results, presented in Figure 8, consistently demonstrate the superiority of H-optimized routing across all configurations, showcasing its ability to effectively balance update frequency while minimizing gradient staleness.



■ **Figure 8** Performance over physical time on the validation set in the scenario of Section 5.2, with $n = 30$ clients and $m = 30$ tasks, evaluated under homogeneous, heterogeneous, and highly heterogeneous data distributions. Standard normalization and data augmentation techniques are applied. The simulation runs for 30,000 time units, recording accuracy and loss every 10 rounds for p^{uniform} and every 50 rounds for p_H^* . Each experiment is repeated independently five times. Solid lines show metrics averaged over independent simulations, and shaded areas represent the standard deviation.