Straggler-Resilient Differentially-Private Decentralized Learning

Yauhen Yakimenka*, Chung-Wei Weng*, Hsuan-Yin Lin*, Eirik Rosnes*, and Jörg Kliewer†
*Simula UiB, N-5006 Bergen, Norway
†Helen and John C. Hartmann Department of Electrical and Computer Engineering,
New Jersey Institute of Technology, Newark, New Jersey 07102, USA

***Abstract*—We consider straggler resiliency in decentralized learning using stochastic gradient descent under the notion of network differential privacy (DP). In particular, we extend the recently proposed framework of privacy amplification by decentralization by Cyffers and Bellet to include training latency—comprising both computation and communication latency. Analytical results on both the convergence speed and the DP level are derived for training over a logical ring for both a skipping scheme (which ignores the stragglers after a timeout) and a baseline scheme that waits for each node to finish before the training continues. Our results show a trade-off between training latency, accuracy, and privacy, parameterized by the timeout of the skipping scheme. Finally, results when training a logistic regression model on a real-world dataset are presented.**

I. Introduction

In distributed learning, a finite-sum optimization problem is solved across multiple nodes without exchanging the local datasets directly, thus limiting the data privacy leakage. In federated learning [1]–[3], there is a single node coordinating the overall training process, while in decentralized learning, see, e.g., [4], [5], there is no such coordinating central server—the nodes maintain a local estimate of the optimal model and iteratively updates it by averaging estimates obtained from neighboring nodes corrected on the basis of their local datasets. This can either be done *sequentially* by a *token* that travels over the set of nodes and gets updated by each node who receives it, or in *parallel* by several distinct tokens. Theoretical studies show that the underlying communication topology has a strong impact on the number of epochs needed to converge [6].

Even though data stays local, the computed partial (sub)gradients can leak information on the local datasets [7]. To address this shortcoming, a carefully selected noise term can be added to the computed partial (sub)gradients before they are transmitted to other nodes, referred to as local differential privacy (LDP) [8], [9]. Recently, a novel relaxation of LDP that naturally arises in decentralized learning, referred to as *network* DP (NDP) was introduced [10], which captures the fact that nodes have only a local view of the system. In [10], the authors showed that the privacy-utility trade-off under NDP can be significantly improved upon compared to what is achievable under LDP, illustrating that formal privacy gains can be obtained from full decentralization, complementing previous notions of "amplifying" the privacy by shuffling, subsampling, and iteration [11]–[14]. Differentially-private decentralized learning has been considered in several previous works, see, e.g., [5], [15], [16].

The impact of *stragglers*, i.e., nodes that take a long time to finish their tasks due to random phenomena such as processes running in the background, has been considered extensively in the literature. A simple way to address the straggler problem is to ignore results from the slowest nodes, see, e.g., [5], [17], which can however lead to convergence to a local optimum when the data is heterogeneous [18], [19]. To address this shortcoming, coded computing methods have been proposed in the literature, see, e.g., [20]–[29].

Straggler resiliency and the impact of user data privacy in decentralized training has been studied separately, while this work *simultaneously* studies both. In particular, we consider sequential training along a logical ring and extend the recently proposed framework of privacy amplification by decentralization by Cyffers and Bellet [10] to include the overall latency—comprising both computation and communication latency—under stochastic gradient descent. In sequential training, nodes do not need to be active during the whole training period, which makes it suitable for scenarios where the nodes have limited resources, and therefore remain dormant unless they are triggered to do an update. Specifically, we consider a ring topology, where each node can only communicate with its immediate neighbors upstream and downstream. Our main contributions are summarized as follows.

- We derive analytical results on both the convergence behavior (see Theorem 1) and the DP level (see Theorems 2 and 3) for a skipping scheme (which ignores the stragglers after a timeout) and a baseline scheme that waits for each node to finish before the training continues, for both a fixed and a randomized ring topology. Our results reveal a trade-off parameterized by the timeout of the skipping scheme.
- We determine the optimal timeout that minimizes the time between two consecutive updates of the token, showing that skipping is beneficial for faster convergence for certain popular computational delay models considered in the literature (see Lemma 2 and Section VI-B).
- We show that randomizing the processing order of nodes on the ring provides an improvement in both convergence behavior and privacy in the long run. Hence, randomization is capable of enhancing the privacy-convergence tradeoff (see Section VI-A).

Finally, we present results for logistic regression on a real-world dataset to validate our theoretical findings. Due to lack of space, all proofs are omitted.

978-1-6654-8341-4/22/$31.00 ©2022 IEEE

II. Preliminaries

A. Notation

We use uppercase and lowercase letters for random variables (RVs) and their realization (both scalars and vectors), respectively, and italics for sets, e.g., X, x, and $\mathcal{X}$ represent a RV, a scalar/vector, and a set, respectively. An exception to this rule is τ which denotes the model description, also referred to as the token. Matrices are denoted by uppercase letters, their distinction from RVs will be clear from the context. Vectors are represented as row vectors and the transpose of a vector or a matrix is denoted by $(\cdot)^\top$. The expectation of a RV X is denoted by $\mathbb{E}[X]$. We define $[n] \triangleq \{1, 2, \ldots, n\}$, while $\mathbb{N}$ denotes the set of natural numbers and $\mathbb{R}$ the set of real numbers. The (sub)gradient of a function $f(x)$ is denoted by $\nabla f(x)$, while the ℓ_p-norm of a length-n vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is denoted by $\|x\|_p = \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}$, where $|\cdot|$ denotes absolute value. The base of the natural logarithm is denoted by e, while $\log$ denotes natural logarithm. $\mathcal{N}(\mu, \sigma I_d)$ denotes the d-dimensional Gaussian (uncorrelated) distribution with mean μ and standard deviation σ of each component, where I_d is the identity matrix of size d. $X \sim \mathcal{P}$ denotes that X is distributed according to the distribution $\mathcal{P}$, while $x \sim \mathcal{P}$ denotes a sample x taken from the distribution $\mathcal{P}$. We denote by $\mathcal{D} \sim_u \mathcal{D}'$ the fact that datasets $\mathcal{D} = \cup_{v\in\mathcal{V}}\mathcal{D}_v$ and $\mathcal{D}' = \cup_{v\in\mathcal{V}}\mathcal{D}'_v$ are the same except perhaps for the dataset of the user u, i.e., $\mathcal{D}_v = \mathcal{D}'_v$ for all $v \neq u$, where $\mathcal{V}$ is some set of users. Standard order notation $O(\cdot)$ is used for asymptotic results.

B. Definitions

Definition 1 (k-Lipschitz continuity)**.** *A function $f : \mathcal{W} \to \mathbb{R}$ is* k-Lipschitz continuous *over the convex domain $\mathcal{W} \subseteq \mathbb{R}^d$ if $|f(w) - f(w')| \leq k \|w - w'\|_2$ for all $w, w' \in \mathcal{W}$.*

Definition 2 (β-smoothness)**.** *A function $f\colon \mathcal{W} \to \mathbb{R}$ is* β-smooth *over the convex domain $\mathcal{W} \subseteq \mathbb{R}^d$ if $\|\nabla f(w) - \nabla f(w')\|_2 \leq \beta \|w - w'\|_2$ for all $w, w' \in \mathcal{W}$.*

C. System Model

Consider a decentralized network of n honest-but-curious nodes (users) $\mathcal{V} = \{v_1, \ldots, v_n\}$ with a decentralized dataset $\mathcal{D} = \cup_{v\in\mathcal{V}}\mathcal{D}_v$ where $\mathcal{D}_v = \left\{(x_i^{(v)}, y_i^{(v)})\right\}_{i=1}^{\kappa}$, $(x_i^{(v)}, y_i^{(v)}) \in \mathcal{R} \subseteq \mathbb{R}^{d_{\mathrm{x}}} \times \mathbb{R}^{d_{\mathrm{y}}}$, for some set $\mathcal{R}$ and $d_{\mathrm{x}}, d_{\mathrm{y}} \in \mathbb{N}$, and $\kappa \in \mathbb{N}$, is the private dataset of node $v \in \mathcal{V}$.

The nodes want to compute some function together based on their datasets but want to keep their datasets private. For that, they employ a decentralized protocol where a token $\tau \in \mathcal{W}$, for some convex set $\mathcal{W} \subseteq \mathbb{R}^d$, travels between the nodes according to some predefined (but potentially randomized) path. When receiving the token the r-th time and the global time is h, the node v updates it as $\tau \leftarrow g_r^{(v)}(\tau; \mathtt{state}_v(h))$, and sends it further. Here, $\mathtt{state}_v(h)$ encapsulates all the information available to the node v at time h, e.g., the available data points and the results of previous calculations. It can also include some source of randomness. We assume that the computation in each node v during the r-th visit of the token takes random time $T_r^{(v)}$. Hence, the computation of $g_r^{(v)}(\cdot,\cdot)$ takes time at most $T_r^{(v)}$ as the token may be updated before the entire computation is finished.[1] We consider a model where $T_r^{(v)}$ is comprised of a deterministic constant part (the time it takes to perform an actual computation) and a random part. Additionally, we assume that communication between any two nodes is noiseless and takes constant time χ, and hence the constant part of the computation time can be set to zero without loss of generality. At the end of the protocol, the token τ is distributed among all the nodes, which allow them to calculate the desired result of joint computations. We assume this final distribution takes constant overhead time and we therefore ignore it in further derivations.

For a decentralized protocol $\mathcal{A}$, we denote by $\mathcal{A}(\mathcal{D})$ the (random) transcript of all messages sent or received by all the users, i.e., $\mathcal{A}(\mathcal{D})$ equals $\{(u, w, v) : u \in \mathcal{V}$ sent a message with content w to $v \in \mathcal{V}\}$. However, due to the decentralized nature of $\mathcal{A}$, the user v only has access to the subset of $\mathcal{A}(\mathcal{D})$ consisting of the messages she sent or received, and we denote this view by $\mathcal{O}_v(\mathcal{A}(\mathcal{D})) = \{(u, w, u') \in \mathcal{A}(\mathcal{D}) : u = v$ or $u' = v\}$. For notational convenience, we denote by Ω the set of all possible views, i.e., $\mathcal{O}_v(\mathcal{A}(\mathcal{D})) \in \Omega$ for all possible parameters and realizations.

D. Network Differential Privacy

We accept the notion of NDP introduced in [10].

Definition 3 (NDP [10])**.** *A protocol $\mathcal{A}$ satisfies (ε, δ)-NDP if for all pairs of distinct users $u, v \in \mathcal{V}$, all pairs of neighboring datasets $\mathcal{D} \sim_u \mathcal{D}'$, and any $\mathcal{S} \subseteq \Omega$, we have*

$$\Pr[\mathcal{O}_v(\mathcal{A}(\mathcal{D})) \in \mathcal{S}] \leq \mathrm{e}^{\varepsilon} \Pr[\mathcal{O}_v(\mathcal{A}(\mathcal{D}')) \in \mathcal{S}] + \delta,$$

where the notion of neighboring datasets $\mathcal{D} \sim_u \mathcal{D}'$ is defined in Section II-A.

NDP measures how much the information collected by node v depends on the dataset of node u. In the special case that all nodes can observe all messages sent and received, i.e., $\mathcal{O}_v$ is the identity map, NDP boils down to conventional LDP [31]. When processing information in a decentralized manner with no central coordinating entity, and when there is no third party (on top of the topology) observing all messages sent, NDP is a more natural privacy measure than DP or LDP.

III. Empirical Risk Minimization

In this section, we consider the empirical risk minimization problem

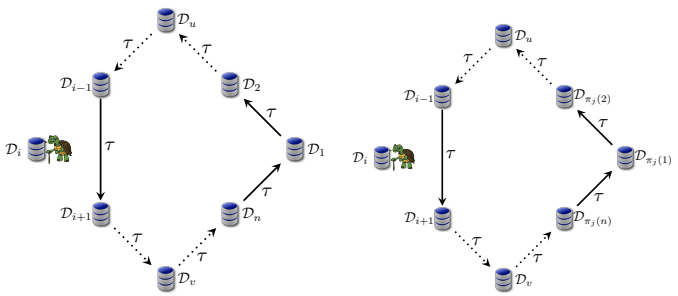
$$\tau^* = \operatorname*{arg\,min}_{\tau\in\mathcal{W}\subseteq\mathbb{R}^d} \left[f(\tau; \mathcal{D}) \triangleq \frac{1}{n} \sum_{v\in\mathcal{V}} f_v(\tau; \mathcal{D}_v) \right], \qquad (1)$$

where $f_v(\tau; \cdot)$ is a k-Lipschitz continuous convex function in its first argument.

A. Skipping Scheme

We suggest the following protocol inspired by projected noisy stochastic gradient descent to solve (1). The token

[1] The RVs $T_r^{(v)}$ are assumed to be independent and identically distributed (i.i.d.) which is in accordance with the literature, where typically stragglers are generated uniformly at random, except for a few works, e.g., [21], [30] that consider a model where a node that becomes a straggler tends to remain a straggler for a long time, violating the i.i.d. assumption on the RVs $T_r^{(v)}$.



(a) SS-Ring. (b) SS-Rand-Ring.

Fig. 1. Illustrating the j-th round in which node v_i is a straggler.

τ keeps the current estimate of the optimal point τ^* and follows a possibly randomized path over the available nodes $\mathcal{V}$. However, to speed up the process, the token waits up to a threshold time $t_{\rm ss}$ and, if the computation has not finished by that time, the token is forwarded further without an update.[2] In our notation, it means that the calculation in each node v is

$$g_r^{(v)}(\tau; \texttt{state}_v(h)) = \begin{cases} \Pi_{\mathcal{W}}\left(\tau - \eta_h\left(\nabla f_v(\tau; \mathcal{D}_v) + N_h\right)\right) & \text{if } T_r^{(v)} \le t_{\rm ss}, \\ \tau & \text{otherwise,} \end{cases} \quad (2)$$

where η_h is the step size (learning rate), $\Pi_{\mathcal{W}}$ denotes the Euclidean projection onto the set $\mathcal{W}$, and N_h is noise with zero mean and standard deviation σ_h. The noise N_h is added in order to protect the privacy of the local datasets, and the standard deviation σ_h is chosen so a certain level of NDP is ensured.[3] In this work, we consider both the gamma distribution (including the exponential distribution) and the Pareto type II (also known as Lomax) distribution for $T_r^{(v)}$, which are well-established models in the literature, see, e.g., [30], [32], [33]. Since we assume that the RVs $T_r^{(v)}$ are i.i.d., we simplify the notation in the following by letting $T \equiv T_r^{(v)}$.

The algorithm stops when a predefined convergence requirement is fulfilled. We refer to the algorithm detailed above as the skipping scheme with parameter $t_{\rm ss}$, which can be optimized in order to reduce either the convergence time and/or the privacy leakage. In the special case of $t_{\rm ss} = \infty$, it reduces to a scheme for which the token always waits. We denote by $p = \Pr[T > t_{\rm ss}]$ the probability of skipping a node. The formal algorithm is given in Algorithm 1, where the output ℓ denotes its execution latency and $\tau_{h_{\max}}$ the final value of the token after $h_{\max}$ steps.

We use Algorithm 1 in two special cases as outlined below and illustrated in Fig. 1. For both schemes, the noise variance is fixed throughout the algorithm, i.e., $\sigma_h = \sigma$, $\forall\, h$, and we assume, for simplicity, that $h_{\max}$ is a multiple of n in the rest of the paper.

- First, we consider an update schedule in which the nodes in $\mathcal{V}$ are processed along a logical ring, i.e., the node path sequence of Algorithm 1 is $(v^{(1)}, \ldots, v^{(h_{\max})}) = ((v_1, \ldots, v_n), (v_1, \ldots, v_n), \ldots, (v_1, \ldots, v_n))$. The corresponding scheme is denoted by SS-Ring.
- Second, we consider a *randomized* version of the logical ring above. In particular, each round over the ring can be seen as a random walk on the set of nodes, but without replacement. For each round, the random walk procedure is restarted. Hence, the node path sequence becomes $(v^{(1)}, \ldots, v^{(h_{\max})}) = \big((v_{\pi_1(1)}, \ldots, v_{\pi_1(n)}), (v_{\pi_2(1)}, \ldots, v_{\pi_2(n)}), \ldots, (v_{\pi_{h_{\max}/n}(1)}, \ldots, v_{\pi_{h_{\max}/n}(n)})\big)$, where $\pi_1, \ldots, \pi_{h_{\max}/n}$ are random permutations over $[n]$. The scheme is denoted by SS-Rand-Ring.

Algorithm 1: Skipping Scheme

Input: Datasets $\mathcal{D}_v$ and k-Lipschitz continuous convex functions $f_v : \mathcal{W} \times \mathcal{R}^\kappa \to \mathbb{R}$, $v \in \mathcal{V}$, in the first argument, noise standard deviation sequence $(\sigma_1, \ldots, \sigma_{h_{\max}})$, node path sequence $(v^{(1)}, \ldots, v^{(h_{\max})})$, learning rate parameter ζ, skipping parameter $t_{\rm ss}$, number of steps $h_{\max}$, and communication latency χ

Output: $(\tau_{h_{\max}}, \ell)$

1 $\tau_0 \leftarrow 0, \quad \ell \leftarrow 0, \quad c \leftarrow 1$
2 $\mathcal{P} \leftarrow$ Comp. lat. model (gamma or Pareto type II)
3 **for** $h \in [h_{\max}]$ **do**
4 $t \sim \mathcal{P}$
5 **if** $t \le t_{\rm ss}$ **then**
6 $\eta_h \leftarrow \zeta/\sqrt{c}$
7 $\tau_h \leftarrow \Pi_{\mathcal{W}}\left(\tau_{h-1} - \eta_h\left(\nabla f_{v^{(h)}}(\tau_{h-1}; \mathcal{D}_{v^{(h)}}) + N_h\right)\right)$, where $N_h \sim \mathcal{N}(0, \sigma_h^2 I_d)$
8 $\ell \leftarrow \ell + \chi + t, \quad c \leftarrow c + 1$
9 **end**
10 **else**
11 $\tau_h \leftarrow \tau_{h-1}, \quad \ell \leftarrow \ell + \chi + t_{\rm ss}$
12 **end**
13 **end**
14 **return** $(\tau_{h_{\max}}, \ell)$

B. Computation and Communication Latency

The average total latency of the skipping scheme in Algorithm 1 is given by the following lemma.

Lemma 1. *The expected total latency for the skipping scheme in Algorithm 1 is*

$$h_{\max}\left(\chi + \int_0^{t_{\rm ss}} t \, \mathrm{d}\Phi_T(t) + t_{\rm ss}\big(1 - \Phi_T(t_{\rm ss})\big)\right),$$

where $\Phi_T(t) \triangleq \Pr[T \le t]$ *and* $\Phi_T(t_{\rm ss}) = 1 - p$.

If the number of hops $h_{\max}$ is large enough, we would expect shorter times between token updates (all other properties being the same) to be beneficial for convergence. In other words, expected time between two consecutive visits to Line 7 in Algorithm 1 should be minimized.[4]

[2] In a real implementation, acknowledgments can be used to identify straggling nodes, e.g., when the token is forwarded to the next node in line and no acknowledgment is received within a threshold time, the token is forwarded to the second next node in line, etc.

[3] The noise is drawn from a Gaussian distribution with zero mean and standard deviation $\sigma_h = \frac{k\sqrt{8\log(1.25/\delta)}}{\epsilon}$, where $\epsilon > 0$ and $0 < \delta < 1$.

[4] The number of hops between two consecutive updates follows the geometric distribution with success probability $1 - p$.

Lemma 2. *The value of $t_{\rm ss}$ that minimizes the average time between two consecutive updates of the token is given by the solution of the optimization problem*

$$\arg\min_{t_{\rm ss}} \frac{\chi + \int_0^{t_{\rm ss}} t \,\mathrm{d}\Phi_T(t) + t_{\rm ss}\big(1-\Phi_T(t_{\rm ss})\big)}{\Phi_T(t_{\rm ss})}.$$

IV. Convergence Analysis

Here, we provide a convergence result for the two considered schemes by adapting the classical convergence result of [34, Thm. 2] to decentralized learning where nodes are processed according to a Markov chain and for which the (sub)gradient estimate in each step is biased, but *converges to unbiased* exponentially fast, which are the main two new technicalities of the proof.[5] Additionally, the number of token updates is random (depending on the skipping probability), and we need to average over it.

Theorem 1. *If the diameter of $\mathcal{W}$ is $d_{\mathcal{W}}$, the expected difference between the minimum value $f(\tau^*;\cdot)$ and that from Algorithm 1 with an arbitrary learning rate parameter $\zeta > 0$ after $h_{\max}$ steps is bounded as*

$$\mathbb{E}\left[f(\tau_{h_{\max}};\cdot) - f(\tau^*;\cdot)\right] \leq \sum_{h=0}^{h_{\max}} \binom{h_{\max}}{h} (1-p)^h p^{h_{\max}-h} e_h,$$

where $\forall\, h > 0$,

$$e_h \triangleq \frac{(d_{\mathcal{W}}^2 + \zeta^2(k^2+d\sigma^2))(2+\log(h+1))}{\zeta\sqrt{h+1}} + d_{\mathcal{W}} k \sqrt{n}\left(\frac{1}{h+1}\sum_{i=1}^{h+1}|\lambda_1|^i + \sum_{j=1}^{h}\frac{1}{j(j+1)}\sum_{i=h+1-j}^{h+1}|\lambda_1|^i\right)$$

and $e_0 \triangleq d_{\mathcal{W}} k$, $|\lambda_1| = \frac{1-p}{\sqrt{(1+p^2)-2p\cos(\frac{2\pi}{n})}}$ and $0 < p < 1$ for `SS-Ring`*, while $\lambda_1 \triangleq 0$ and $0 \leq p < 1$ for* `SS-Rand-Ring`*.*

Note that the upper bound of Theorem 1 is of order $O\left(\mathbb{E}\left[\log B/\sqrt{B}\right]\right)$, where B is a *positive* binomial RV with probability mass function $\Pr[B=h] = \frac{1}{1-p^{h_{\max}}}\binom{h_{\max}}{h}(1-p)^h p^{h_{\max}-h}$, $h \in [h_{\max}]$, from which it can be proved (using [37, Thm. 1]) that the asymptotic convergence rate equals $O\big(\log(h_{\max})/\sqrt{h_{\max}}\big)$. This rate is the same as that of [34, Thm. 2], while being a $\log(h_{\max})$-factor worse compared to [35, Thm. 1]. The latter is due to 1) the assumption that σ_h decays to zero with h [35, Eq. (16)], and 2) that convergence there is proved for the running average of the token.

Note that the asymptotic behavior of the bound in Theorem 1 is the same for $\lambda_1 = 0$ and $\lambda_1 > 0$. Hence, a biased (sub)gradient estimate that converges to unbiased exponentially fast does not influence the asymptotic convergence rate.

V. Privacy Analysis

In this section, we present results on the privacy level of the skipping scheme for both updating schedules of the token outlined in Section III-A, i.e., for both a fixed and a randomized logical ring on the set of nodes $\mathcal{V}$. We highlight here that compared to [10], that only considers a constant learning rate and also a different randomized path (and no fixed path), our results apply to a decreasing learning rate of the form $\eta_h = \zeta/\sqrt{h}$ (as specified in Algorithm 1).

The proof evolves around upper bounding the Rényi divergence between $\mathcal{O}_v(\mathcal{A}(\mathcal{D}))$ and $\mathcal{O}_v(\mathcal{A}(\mathcal{D}'))$, $\mathcal{D} \sim_u \mathcal{D}'$, for any distinct pair of users u, v, using tools (including a composition theorem for Rényi DP [38, Prop. 1]) from the framework of privacy amplification by iteration [14]. The resulting bound can be transformed into a bound on NDP using [38, Prop. 3] and further optimized. Allowing for a decreasing learning rate constitutes the main technical contribution of the proof.

We first consider the `SS-Ring` scheme. The privacy level $\varepsilon_{\rm ss}$ after a certain number of $h_{\max}/n$ rounds of the token is given by the following theorem.

Theorem 2. *Assume f_v, $v \in \mathcal{V}$, is β-smooth, and let $\varepsilon > 0$ and $0 < \delta < 1$. Then, the* `SS-Ring` *scheme on a ring with n nodes and with learning rate parameter $0 < \zeta \leq 2/\beta$ achieves $(\varepsilon_{\rm ss}, \delta+\delta')$-NDP for all $\delta' \in (0,1]$ with*

$$\varepsilon_{\rm ss} = \varepsilon\frac{\sqrt{\tilde{h}\log(1/\delta)}}{\sqrt{\log(1.25/\delta)}} + \frac{\varepsilon^2\tilde{h}}{4\log(1.25/\delta)},$$

where

$$\tilde{h} \triangleq \left\lceil h_{\max}(1-p)/n + \sqrt{3h_{\max}(1-p)/n \log\left(1/\delta'\right)}\right\rceil,$$

and $0 \leq p < 1$ is the probability of skipping a node.

The following theorem characterizes the privacy level $\varepsilon_{\rm ss}$ of the `SS-Rand-Ring` scheme.

Theorem 3. *Assume f_v, $v \in \mathcal{V}$, is β-smooth, and let $\varepsilon > 0$ and $0 < \delta < 1$. Then, the* `SS-Rand-Ring` *scheme on a ring with n nodes and with learning rate parameter $0 < \zeta \leq 2/\beta$ achieves $(\varepsilon_{\rm ss}, \delta+\delta')$-NDP for all $\delta' \in (0,1]$ with*

$$\varepsilon_{\rm ss} = \frac{\varepsilon^2 a\alpha}{2\log(1.25/\delta)} + \frac{\log(1/\delta)}{\alpha-1},$$

where

$$a \triangleq \frac{1}{n-1}\sum_{r=0}^{\tilde{h}-1}\sum_{d=1}^{n-1}\sum_{h=1}^{d}\frac{h\binom{d}{h}p^{d-h}(1-p)^h}{\gamma_{r,h}},$$

$$\gamma_{r,h} \triangleq 4(1+r\cdot h)\cdot\left(\sqrt{1+r\cdot h+h} - \sqrt{1+r\cdot h}\right)^2,$$

$$\tilde{h} \triangleq \left\lceil h_{\max}(1-p)/n + \sqrt{3h_{\max}(1-p)/n \log\left(1/\delta'\right)}\right\rceil,$$

$$\alpha \triangleq \min\left(\frac{\sqrt{2\log(1/\delta)\log(1.25/\delta)}}{\varepsilon\sqrt{a}} + 1, \frac{1+\sqrt{\frac{16\log(1.25/\delta)}{\varepsilon^2}+1}}{2}\right),$$

and $0 \leq p < 1$ is the probability of skipping a node.

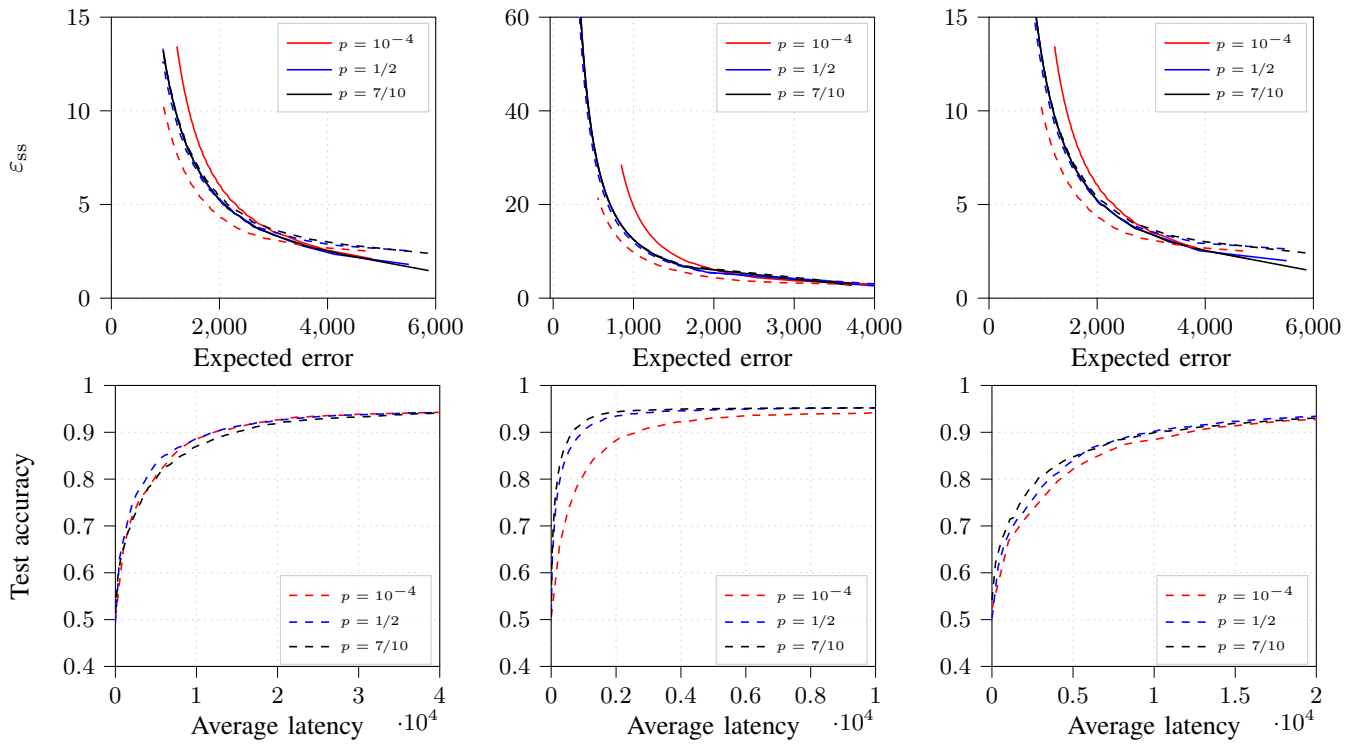
VI. Numerical Results

Here, we first perform a comparison based on the analytical results from Sections IV and V, before turning to training a logistic regression model using the dataset in [39].

A. Convergence Versus Privacy

For fixed values of $n = 10$, $\varepsilon = 1$, $\delta = 10^{-6}$, $\delta' = 1/10$, $d = 8$, $d_{\mathcal{W}} = 10$, $k = 1$, $\zeta = 3/100$, and $\chi = 1/100$, the upper plots of Fig. 2 show the privacy level $\varepsilon_{\rm ss}$ (obtained from Theorems 2

[5] There are several previous works that provide convergence results for Markov chain (noisy) stochastic gradient descent, e.g., [35], [36]. However, all of these works require that σ_h decays to zero with h, which means a significantly higher leakage of private data. The main technical contribution of our result is the circumvention of this assumption. Note that, as in [34, Thm. 2], f_v, $v \in \mathcal{V}$, is not required to be β-smooth or even k-Lipschitz continuous, as we only need the (sub)gradients to be bounded (which follows from k-Lipschitzness).



(a) Exponential with mean 1.

(b) Gamma with shape $1/4$ and scale 1.

(c) Pareto type II with shape 3 and scale 2.

Fig. 2. Upper plots: privacy level ε_{ss} (bounds from Theorems 2 and 3) versus expected error in function minimization (bound from Theorem 1) for different computation latency models. Lower plots: logistic regression model training, showing accuracy (on the test set) as a function of average latency from Lemma 1. Each curve is an average of 100 independent runs. Upper and lower plots: solid lines are for a fixed ring (`SS-Ring`), while dashed lines are for `SS-Rand-Ring`.

and 3) versus the expected error in function minimization (the bound from Theorem 1), parameterized by the average latency (Lemma 1), which increases toward the upper left corner. We consider the latency models: exponential with mean 1, gamma with shape $1/4$ and scale 1, and Pareto type II with shape 3 and scale 2 (as used in [33]). The probability of skipping $p = \Pr[T > t_{\text{ss}}] \in \{10^{-4}, 1/2, 7/10\}$, since $p = 10^{-4}$ and $7/10$ are close to the values of p given by Lemma 2, respectively 0 and $0.710/0.737$ for the exponential and gamma/Pareto delay models, while $p = 1/2$ is a value in between.

As can be seen from the figure, the trade-offs look similar for all latency models considered. `SS-Rand-Ring` gives better trade-off curves (especially for $p = 10^{-4}$, i.e., virtually no skipping) for smaller values of error in function minimization, while the situation changes for higher values of error (i.e., at the initial stages of Algorithm 1's execution). Hence, path randomization improves the trade-off in the long run, but might be harder to realize in a real-world implementation as it would require a full mesh topology.

On the contrary, the `SS-Ring` curve for $p = 10^{-4}$ is the worst, which means that skipping helps. Also, there is not much difference between the `SS-Ring` curves for $p = 1/2$ and $p = 7/10$ (they are are almost on top of each other and hence difficult to distinguish). Therefore, one should choose the timeout based on faster convergence (cf. Lemma 2). On the other hand, `SS-Rand-Ring` favors smaller values of p (i.e., larger timeout) at the expense of a higher training latency as shown in the next subsection.

B. Empirical Results

We consider training a logistic regression model. For logistic regression the local loss functions are $f_v(\tau, \mathcal{D}_v) = 1/|\mathcal{D}_v| \sum_{(x,y)\in\mathcal{D}_v} \log(1+\mathrm{e}^{-y\tau x^\top})$, where $x \in \mathbb{R}^d$ ($d_{\text{x}} = d$) and $y \in \{-1, 1\}$ ($d_{\text{y}} = 1$). We use a binarized version of the UCI housing dataset [39]. The features are standardized and we further normalize each data point to have unit ℓ_2-norm so that the loss functions $f_v(\tau; \mathcal{D}_v)$ are 1-Lipschitz continuous (i.e., $k = 1$). The dataset is split uniformly at random into a training set with 80% of the data points and a test set with 20% of the points. Moreover, the training dataset is further randomly split across the $n = 10$ nodes in $\mathcal{V}$. We used the `SS-Rand-Ring` scheme with the same parameters as in Section VI-A, but using a mini-batch implementation with batches of size 100 in order to speed up the learning. The chosen mini-batch size is a compromise between the two corner cases: a mini-batch size of 1 is difficult to parallelize, whereas a large mini-batch size may exceed the nodes' limited parallelization capabilities.

The results of the resulting training are shown in the bottom plots in Fig. 2, which show the test accuracy, i.e., the ratio of correct predictions on the test set, versus average latency from Lemma 1 for the same skipping probabilities as in the corresponding upper plots. We observe that skipping achieves a clear speed-up compared to no skipping, except for the exponential delay model, as predicted well by Lemma 2. However, as can be seen from the upper plots, no skipping provides a slightly higher privacy for the `SS-Rand-Ring` scheme.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist. (AISTATS)*, Ft. Lauderdale, FL, USA, Apr. 20–22, 2017, pp. 1273–1282.

[2] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *NeurIPS Workshop Private Multi-Party Mach. Learn. (PMPML)*, Barcelona, Spain, Dec. 9, 2016.

[3] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.

[4] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," in *Proc. 31th Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Long Beach, CA, USA, Dec. 4–9, 2017, pp. 5336–5346.

[5] G. Xiong, G. Yan, R. Singh, and J. Li, "Straggler-resilient distributed machine learning with dynamic backup workers," Feb. 2021, arXiv:2102.06280v1 [cs.LG].

[6] G. Neglia, C. Xu, D. Towsley, and G. Calbi, "Decentralized gradient methods: does topology matter?" in *Proc. 23rd Int. Conf. Artif. Intell. Statist. (AISTATS)*, Virtual Conf., Aug. 26–28, 2020, pp. 2348–2358.

[7] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, Denver, CO, USA, Oct. 12–16, 2015, pp. 1322–1333.

[8] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," Dec. 2017, arXiv:1712.07557v2 [cs.CR].

[9] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farhad, S. Jin, T. Q. S. Quek, and H. V. Poor, "Federated learning with differential privacy: Algorithms and performance analysis," *IEEE Trans. Inf. Forens. Secur.*, vol. 15, pp. 3454–3469, 2020.

[10] E. Cyffers and A. Bellet, "Privacy amplification by decentralization," in *Proc. 25th Int. Conf. Artif. Intell. Statist. (AISTATS)*, Virtual Conf., Mar. 28–30, 2022, pp. 5334–5353.

[11] Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta, "Amplification by shuffling: From local to central differential privacy via anonymity," in *Proc. Annu. ACM-SIAM Symp. Discrete Algorithms (SODA)*, San Diego, CA, USA, Jan. 6–9, 2019, pp. 2468–2479.

[12] V. Feldman, A. McMillan, and K. Talwar, "Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling," in *Proc. 62th Annu. IEEE Symp. Found. Comp. Sci. (FOCS)*, Virtual Conf., Feb. 7–10, 2022, pp. 954–964.

[13] B. Balle, G. Barthe, and M. Gaboardi, "Privacy amplification by subsampling: Tight analyses via couplings and divergences," in *Proc. 32th Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Montréal, QC, Canada, Dec. 3–8, 2018, pp. 6280–6290.

[14] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta, "Privacy amplification by iteration," in *Proc. 59th Annu. IEEE Symp. Found. Comp. Sci. (FOCS)*, Paris, France, Oct. 7–9, 2018, pp. 521–532.

[15] M. Showkatbakhsh, C. Karakus, and S. Diggavi, "Differentially private consensus-based distributed optimization," Mar. 2019, arXiv:1903.07792v1 [cs.LG].

[16] R. Jin, X. He, and H. Dai, "Decentralized differentially private without-replacement stochastic gradient descent," Sep. 2018, arXiv:1809.02727v3 [cs.LG].

[17] A. Reisizadeh, H. Taheri, A. Mokhtari, H. Hassani, and R. Pedarsani, "Robust and communication-efficient collaborative learning," in *Proc. 33th Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Vancouver, BC, Canada, Dec. 8–14, 2019, pp. 8388–8399.

[18] Z. Charles and J. Konečný, "On the outsized importance of learning rates in local update methods," Jul. 2020, arXiv:2007.00878v1 [cs.LG].

[19] A. Mitra, R. Jaafar, G. J. Pappas, and H. Hassani, "Linear convergence in federated learning: Tackling client heterogeneity and sparse gradients," in *Proc. 35th Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Virtual Conf., Dec. 6–14, 2021, pp. 14 606–14 619.

[20] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, Mar. 2018.

[21] C.-S. Yang, R. Pedarsani, and A. S. Avestimehr, "Timely-throughput optimal coded computing over cloud networks," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput. (MobiHoc)*, Catania, Italy, Jul. 2–5, 2019, pp. 301–310.

[22] S. Li and S. Avestimehr, "Coded computing: Mitigating fundamental bottlenecks in large-scale distributed computing and machine learning," *Found. Trends® Commun. Inf. Theory*, vol. 17, no. 1, pp. 1–148, 2020.

[23] A. Severinson, A. Graell i Amat, and E. Rosnes, "Block-diagonal and LT codes for distributed computing with straggling servers," *IEEE Trans. Commun.*, vol. 67, no. 3, pp. 1739–1753, Mar. 2019.

[24] S. Dutta, V. Cadambe, and P. Grover, ""Short-Dot": Computing large linear transforms distributedly using coded short dot products," *IEEE Trans. Inf. Theory*, vol. 65, no. 10, pp. 6171–6193, Oct. 2019.

[25] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Trans. Inf. Theory*, vol. 66, no. 1, pp. 278–301, Jan. 2020.

[26] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient coding: Avoiding stragglers in distributed learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Sydney, Australia, Aug. 6–11, 2017, pp. 3368–3376.

[27] C. Karakus, Y. Sun, S. Diggavi, and W. Yin, "Straggler mitigation in distributed optimization through data encoding," in *Proc. 31th Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Long Beach, CA, USA, Dec. 4–9, 2017, pp. 5440–5448.

[28] R. Bitar, M. Wootters, and S. El Rouayheb, "Stochastic gradient coding for straggler mitigation in distributed learning," *IEEE J. Sel. Areas Inf. Theory*, vol. 1, no. 1, pp. 277–291, May 2020.

[29] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi, "Beyond inferring class representatives: User-level privacy leakage from federated learning," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Paris, France, Apr. 29 – May 2, 2019, pp. 2512–2520.

[30] A. Severinson, E. Rosnes, S. El Rouayheb, and A. Graell i Amat, "DSAG: A mixed synchronous-asynchronous iterative method for straggler-resilient learning," Nov. 2021, arXiv:2111.13877v1 [cs.DC].

[31] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Local privacy and statistical minimax rates," in *Proc. 54th Annu. IEEE Symp. Found. Comp. Sci. (FOCS)*, Berkeley, CA, USA, Oct. 27–29, 2013, pp. 429–438.

[32] S. Dutta, V. Cadambe, and P. Grover, "Coded convolution for parallel and distributed computing within a deadline," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 25–30, 2017, pp. 2403–2407.

[33] G. Neglia, G. Calbi, D. Towsley, and G. Vardoyan, "The role of network topology for distributed machine learning," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Paris, France, Apr. 29 – May 2, 2019, pp. 2350–2358.

[34] O. Shamir and T. Zhang, "Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Atlanta, GA, USA, Jun. 16–21, 2013, pp. 71–79.

[35] T. Sun, Y. Sun, and W. Yin, "On Markov chain gradient descent," in *Proc. 32th Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Montréal, QC, Canada, Dec. 3–8, 2018, pp. 9918–9927.

[36] G. Ayache and S. El Rouayheb, "Private weighted random walk stochastic gradient descent," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 1, pp. 452–463, Mar. 2021.

[37] M. Žnidarič, "Asymptotic expansion for inverse moments of binomial and Poisson distributions," *Open Statist. Probability J.*, vol. 1, pp. 7–10, Jan. 2009.

[38] I. Mironov, "Rényi differential privacy," in *Proc. 30th IEEE Comput. Secur. Found. Symp. (CSF)*, Santa Barbara, CA, USA, Aug. 21–25, 2017, pp. 263–275.

[39] OpenML, "UCI housing dataset." [Online]. Available: https://www.openml.org/d/823