Decentralized Federated Learning with Unreliable Communications

Hao Ye, Le Liang, and Geoffrey Ye Li

***Abstract*—Decentralized federated learning, inherited from decentralized learning, enables the edge devices to collaborate on model training in a peer-to-peer manner without the assistance of a server. However, existing decentralized learning frameworks usually assume perfect communication among devices, where they can reliably exchange messages, *e.g.*, gradients or parameters. But the real-world communication networks are prone to packet loss and transmission errors. Transmission reliability comes with a price. The commonly-used solution is to adopt a reliable transportation layer protocol, *e.g.*, transmission control protocol (TCP), which however leads to significant communication overhead and reduces connectivity among devices that can be supported. For a communication network with a lightweight and unreliable communication protocol, user datagram protocol (UDP), we propose a robust decentralized stochastic gradient descent (SGD) approach, called Soft-DSGD, to address the unreliability issue. Soft-DSGD updates the model parameters with partially received messages and optimizes the mixing weights according to the link reliability matrix of communication links. We prove that the proposed decentralized training system, even with unreliable communications, can still achieve the same asymptotic convergence rate as vanilla decentralized SGD with perfect communications. Moreover, numerical results confirm the proposed approach can leverage all available unreliable communication links to speed up convergence.**

I. Introduction

Empowered by massive training data, deep learning is emerging as the major driving force behind breakthroughs in a wide range of areas, including computer vision, natural language processing, speech processing, etc. Inspired by these success stories, there is a growing trend to use deep learning to unleash the power of massive data generated at the edge network by such devices as smart phones, wearables, and sensors [1]. Nevertheless, the limited communication bandwidth, together with the desire for user privacy, makes it infeasible to collect all distributed data and then train a model at a single machine. As an alternative, training deep learning models at edge networks in a distributed fashion has gained much attention in, *e.g.,* federated learning [1]. It enables multiple devices to compute multiple devices compute the local mini-batch gradients of the loss function with their local datasets and only exchange model parameters or gradients over a communication network to ensure convergence to the global optimal solution, and protect the privacy of the local devices.

H. Ye was with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA. (email: yehao@gatech.edu). L. Liang is with School of Information Science and Engineering, Southeast University, Nanjing 210096, China. (email: lliang@seu.edu.cn) G. Y. Li is with the ITP Lab, the Department of Electrical and Electronic Engineering, Imperial College London. (email: geoffrey.li@imperial.ac.uk)

Traditional federated learning is designed with a parameter-server architecture as shown in Fig. 1 (a), where a centralized server orchestrates the training process. In each training iteration, the global model is transmitted to the participating edge devices and these devices compute a set of potential model updates based on local data. These updates are then sent to the central server and aggregated into a single global update. Since all the selected devices have to send updates to a single centralized server in each training iteration, the server becomes a communication bottleneck of the system, which makes it difficult to scale to a large number of devices.

In this paper, we focus on the decentralized training paradigm building on the device-to-device (D2D) network, which facilitates collaborative learning made possible by the ability to communicate among the edge devices without the assistance of the server. As shown in Fig. 1 (b), each device stores and optimizes the model parameters with its own data and exchanges messages with neighbours in a peer-to-peer manner to reach a consensus. Decentralized optimization has been investigated intensively since 1980s [2]. The most widely-used decentralized algorithms include gradient and subgradient descent [3], [4], alternating direction method of multipliers (ADMM) [5]–[7], and dual averaging [8]. Recently, decentralized implementations of SGD for training deep learning models have gained much attention [9]–[13]. The decentralized stochastic gradient descent (SGD) algorithm has been investigated in [9] for optimizing non-convex objectives (*e.g.*, training deep neural networks) and proven to have the same asymptotic convergence rate as the centralized SGD. This framework has been extended to deal with variations in communication paradigms. An asynchronous decentralized SGD framework has been developed in [10], where the devices communicate asynchronously to reduce the idle time. In [11], gradient push-sum has been exploited to develop a decentralized SGD under a directed communication network.

A major bottleneck in achieving fast convergence is the limited communication resources, such as bandwidth and power. To reduce the communication cost in decentralized training, prior works on building communication-efficient decentralized training systems have concentrated on compressing the communication messages via sparsification and quantization [14]–[16], skipping the communication rounds by performing a certain number of local updates [12], [17], or communicating in an asynchronous manner [10]. It has been shown in [14] that directly compressing the shared messages leads to non-vanishing quantization errors, which in turn causes failure to converge. The first exact decentralized optimization method with message quantization has been developed in [18]

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3152445, IEEE Journal of Selected Topics in Signal Processing

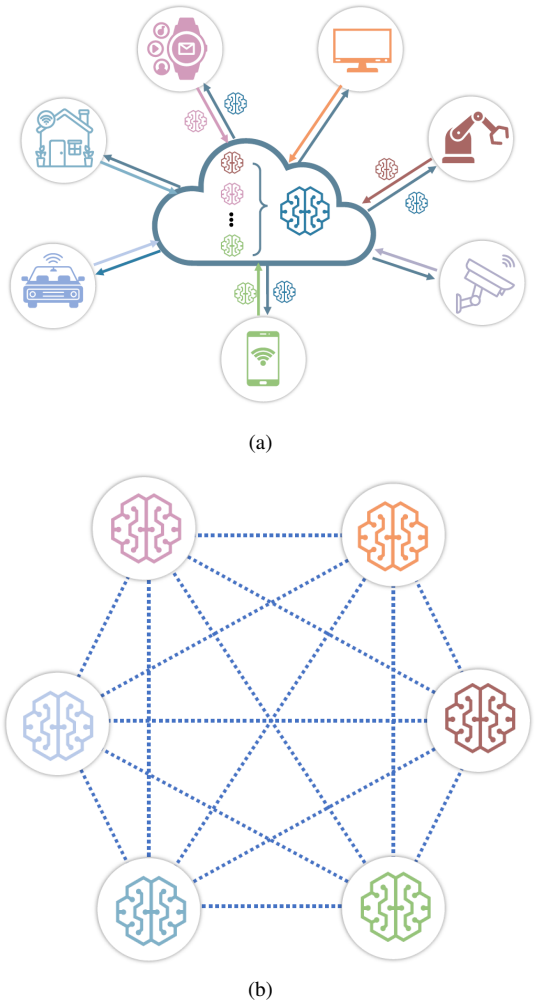


Fig. 1. Centralized vs decentralized federated learning. (a) Centralized federated learning: A central server orchestrates the training process via aggregating the local updates from the devices and broadcasting the updated model. (b) Decentralized federated learning: Each device optimizes the local model with its local data and exchanges messages in a peer-to-peer manner to reach a consensus.

for strongly convex objectives. For non-convex optimization, quantization approaches have been proposed in [14], where the differences of parameters of two consecutive steps are quantized and shared. Moreover, DeepSqueeze in [15] applies an error-compensation method to decentralized settings. ChocoSGD in [16] lets devices estimate remote models with a local estimator, which supports arbitrary quantization by tuning the communication matrix. Apart from compressing the communication messages, another direction to improve the communication efficiency is reducing the communication frequency. The devices take several local SGD update steps before a consensus update with other devices [12], [17].

Nevertheless, most of the existing decentralized optimization methods require a reliable communication network among the devices while the real-world communication systems are prone to packet loss and transmission errors. The transmission errors are pervasive in federated learning due to the harsh wireless channels, which introduce noise, fading, and interference. The default solution is to use a reliable transportation layer communication protocol, *e.g.,* transmission control protocol (TCP), where acknowledgment (ACK), retransmission, and time-out mechanisms are employed to detect and recover from transmission failures. But this transmission reliability comes with a price, which usually incurs considerable communication overheads since the messages may be transmitted multiple times. In addition, the number of neighbours that each device can communicate with is limited to ensure reliability, which also slows down the training speed.

In this paper, a lightweight transmission protocol, user datagram protocol (UDP), is adopted to provide connectionless and unreliable packet delivery service instead of using these reliable but heavyweight communication protocols. Each device communicates with the rest of the network via soft and unreliable communication links, where packet loss and transmission errors occur randomly. A robust decentralized training algorithm, called Soft-DSGD, is developed to deal with the unreliable communications. The devices update their model parameters with partially received messages and the mixing weights in the consensus updates are optimized according to the reliability matrix of different communication links. We prove that the proposed Soft-DSGD under the unreliable communication network achieves the same asymptotic convergence rate as the vanilla decentralized SGD with perfect communications. In addition, numerical results confirm that Soft-DSGD can efficiently leverage all unreliable communication links that are available to accelerate convergence.

The rest of the paper is organized as follows. In Section II, the background information on the decentralized optimization is provided. In Section III, the proposed Soft-DSGD for training deep learning models with unreliable communication networks is presented in detail. The theoretical convergence analysis of Soft-DSGD is shown in Section IV. The simulation results are presented in Section V and the conclusions are drawn in Section VI.

II. Decentralized Optimization

In this section, we briefly introduce the decentralized optimization, including the setting and the applications.

A. Decentralized Optimization and Decentralized SGD

Here, we briefly introduce decentralized training. We consider the following decentralized optimization problem over a network of N devices:

$$f^* := \min_{\mathbf{x}\in\mathbb{R}^d} \left[f(\mathbf{x}) := \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x}) \right],$$

where each component $f_i : \mathbb{R}^d \to \mathbb{R}$ defines the local objective and is only known by the i-th device. The network topology is represented by an undirected connected graph $\mathcal{G}$, where the devices can only communicate along the edges. For deep learning model training, the local objective f_i is given in a stochastic form,

$$f_i(\mathbf{x}) := \mathbb{E}_{\xi_i\sim\mathcal{D}_i} F_i(\mathbf{x}, \xi_i),$$

where $\mathcal{D}_i$ denotes the local data stored at device i, ξ_i is a mini-batch data from $\mathcal{D}_i$, and $F_i(\mathbf{x}, \xi_i)$ is the local loss function with respect to ξ_i.

In the decentralized SGD framework [9], each device maintains its own local parameters, $\mathbf{x}_i \in \mathbb{R}^d$, and computes the local gradient, $\mathbf{g}_i := \nabla F_i(\mathbf{x}_i, \xi_i)$ based on ξ_i sampled from the local dataset. After that, parameters are exchanged with the neighbouring devices via peer-to-peer communications. Formally, the two steps in each training iteration are

1) **SGD update:** $\mathbf{x}_i^{(t+\frac{1}{2})} = \mathbf{x}_i^{(t)} - \gamma \mathbf{g}_i^{(t)}$, where γ denotes the learning rate.
2) **Consensus update:** $\mathbf{x}_i^{(t+1)} = \sum_{j=1}^{N} w_{i,j} \mathbf{x}_j^{(t+\frac{1}{2})}$, where $w_{i,j}$ is the (i,j)-th item of the mixing weights matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$. If two devices i and j are neither neighbors nor identical, then $w_{i,j} = 0$.

To ensure convergence, $\mathbf{W}$ is often set to be symmetric and doubly stochastic, *i.e.,* $\mathbf{W}^T = \mathbf{W}$ and $\mathbf{W}\mathbf{1} = \mathbf{1}$, where $\mathbf{1}$ indicates an N-dimensional vector of all 1's. The spectrum gap of $\mathbf{W}$ is also required to be strictly positive, *i.e.*, $\max\{\|\lambda_2(\mathbf{W})\|, \|\lambda_N(\mathbf{W})\|\} < 1$, where $\lambda_k(\mathbf{W})$ denotes the k-th largest eigenvalue of $\mathbf{W}$.

B. Motivating Applications

The recent increasing attention on the decentralized optimization is mainly driven by a wide range of applications where a network of devices need to cooperate to optimize the common objective. Three examples are included here and more applications of the decentralized optimization can be found in [19].

a) Decentralized estimation: The wireless sensor network and Internet-of-things (IoT) are often utilized for monitoring and estimating uncertain environmental state s. Suppose there are N devices and each has a measurement y_i, which is modeled as a random variable with density $p_i(y_i|s)$. In this case, the maximum likelihood estimate of s can be formulated by the decentralized optimization framework with $f_i = p_i(y_i|s)$.

b) Decentralized resource allocation: The increasing user demand and number of devices impose critical challenges on the wireless resource management schemes, which aim at making the best use of the limited resources (e.g. bandwidth and power). The objective of resource allocation is usually to maximize the summation of a utility function of all communication links, such as spectrum efficiency or energy efficiency. Compared to centralized resource allocation schemes, which collect the local information at a server, conducting the resource allocation in a decentralized manner leads to a smaller communication overhead.

c) Decentralized training of machine learning models: A variety of machine learning approaches can be formulated as an optimization problem over the training data, (*e.g.,* classification or regression). When the data is stored across multiple devices, the training object can be formulated as the decentralized optimization problem, where f_i is the training loss on the subset data on the i-th device. The decentralized SGD and its variants have been utilized for improving the scalability of machine learning models in both datacenters and decentralized networks of devices.

C. Notation

In the subsequent discussion, we use N and d to denote the number of device and the dimension of the training parameters, respectively. We use $\mathbf{x}_i^{(t)}$ to denote the parameters on device i at time step t. We further define the average

$$\bar{\mathbf{x}}_{(t)} := \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}_i^{(t)}. \tag{1}$$

We also use matrix notation where it is more convenient, *i.e.*

$$\mathbf{X}_t := [\mathbf{x}_1^{(t)}, ..., \mathbf{x}_N^{(t)}]^T \in \mathbb{R}^{N \times d}. \tag{2}$$

In addition, we use $\mathbf{X}_l^{(t)}$ to denote the l-th column of $\mathbf{X}_t$. Similarly, we use $\mathbf{G}_t := [\mathbf{g}_1^{(t)}, ..., \mathbf{g}_N^{(t)}]^T$ and $\nabla \mathbf{F}_t := [\nabla f_1(\mathbf{x}_1), ..., \nabla f_N(\mathbf{x}_N)]^T$ to denote the stochastic gradients and gradients, respectively, and $\mathbf{G}_l^{(t)}$ and $\nabla \mathbf{F}_l^{(t)}$ to denote the l-th column of $\mathbf{G}_t$ and $\nabla \mathbf{F}_t$ respectively. We use $\mathbf{1}$ to denote the vector of ones and $\mathbf{J} := \frac{1}{N}\mathbf{1}\mathbf{1}^\mathbf{T}$. We use $\|\mathbf{y}\|$ to denote the Euclidean norm of a vector y. For a matrix $\mathbf{W}$, we will use $w_{i,j}$ or $\mathbf{W}[i,j]$ to denote the i,j-th entry.

III. SOFT-DSGD

In this section, we introduce Soft-DSGD for training deep learning models with unreliable communication networks as shown in Fig 2.

A. Unreliable communications with UDP

Most of the existing decentralized optimization methods require a reliable communication network among the devices while the real-world edge communication systems are prone to packet loss and transmission errors. TCP and UDP are widely-used transportation layer protocols for transmitting data packets over a communication network. TCP is a connection-oriented protocol and requires handshakes to set up an end-to-end communication connection, where messages can be transmitted in both directions. TCP leverages mechanisms including ACK messages, retransmission, and timeouts to guarantee the transmission reliability. The recipient sends ACK messages back to the sender once receiving the messages correctly. Otherwise, the transmitter will resend the packets to the recipient. When TCP is used in broadcast and multi-cast scenarios, reliability is guaranteed for each recipient. Hence, TCP has a considerable communication overhead, especially in unreliable communication networks, such as wireless systems. Compared to TCP, UDP is a lightweight message-based connectionless protocol with much less overhead and does not need to set up a dedicated connection for transmission. Communication is achieved by transmitting packets in one direction from the source to the destination without verifying whether the recipients have received the packets. Due to the unreliability nature of UDP, it is mainly used for delivery best effort traffic, such as multimedia streaming, where occasional packet loss can be tolerated.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3152445, IEEE Journal of Selected Topics in Signal Processing

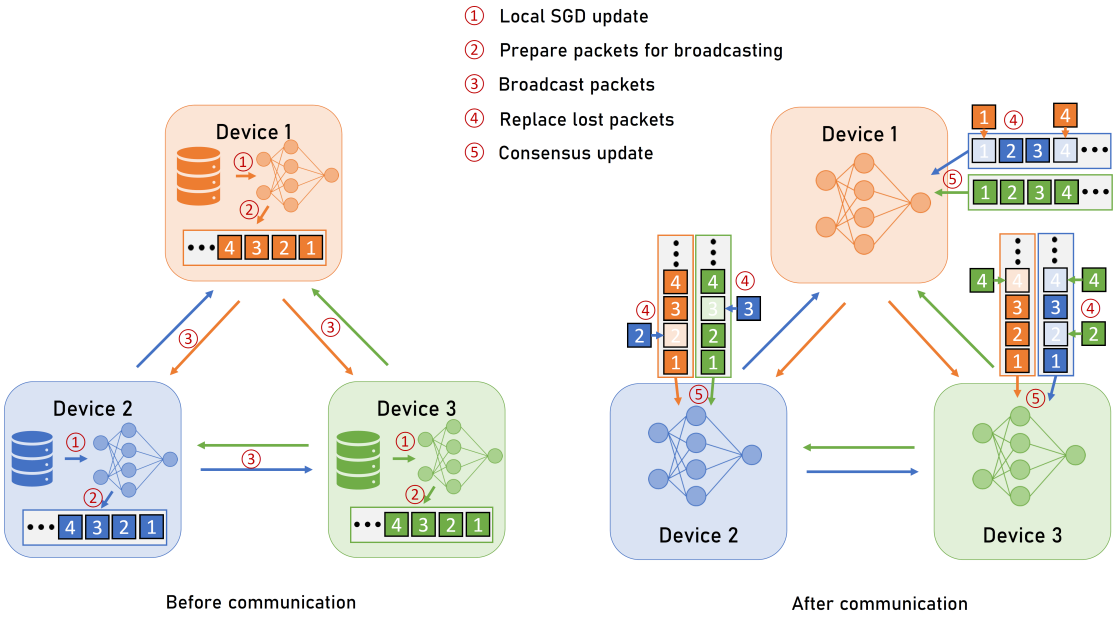


Fig. 2. Pipeline of Soft-DSGD.

In Soft-DSGD, an unreliable and connectionless UDP transmission protocol is adopted. Instead of modeling the communication network as a graph $\mathcal{G}$ where each device can only communicate with its neighbours along the edge as in TCP, we assume a communication network, where each device can send and receive messages from the rest of the network. However, the delivery is not guaranteed in UDP as the transmitted packets are exposed to unreliable communication conditions and no mechanism for reliability enhancement like packet retransmission is implemented. We use a matrix $\mathbf{P} = [p_{i,j}] \in \mathbb{R}^{N\times N}$ to describe the level of link reliability in the communication network, where $p_{i,j}$ represents the probability of successful transmission from the i-th device to the j-th device and $p_{i,i} = 0, \forall i \in \{1, ..., N\}$,. Parameter exchange among the devices consists of the following three steps as follows.

1) **Dividing parameters into packets:** The number of parameters of machine learning models, especially deep neural networks, is usually too huge to be transmitted in a single packet. We assume that the parameters are randomly grouped into multiple packets and transmitted independently.
2) **Broadcasting:** The packets are broadcast to the rest of the network without targeted recipients. With UDP, it is unknown whether the packet can reach its destination with success.
3) **Stochastic receiving:** Due to the unreliability of communication links, packets may be dropped or contaminated randomly when received at any device. With the help of an error detection code, such as a checksum, the receiver can detect errors in the packet, in which case the packet is declared lost and is discarded. Otherwise, the packet is successfully received.

In summary, after the i-th device broadcasts packets of its parameters $\mathbf{x}_i$ to the network, they are randomly received by nodes in the the rest of network. For instance, at the j-th device, the received data, $\mathbf{z}_{i\to j}$, may only include parts of $\mathbf{x}_i$. Therefore, it can be expressed as $\mathbf{z}_{i\to j} = \mathbf{m}_{i\to j} \odot \mathbf{x}_i$, where $\odot$ denotes element-wise multiplication and $\mathbf{m}_{i\to j}(k) = 1$ if the k-th parameter broadcast by the i-th device is successfully received at the j-th device and $\mathbf{m}_{i\to j}(k) = 0$ otherwise.

Algorithm 1 Soft-DSGD Training Algorithm

Require: Initialize local models $\{\mathbf{x}_i\}$, learning rate γ, and mixing weight matrix $\mathbf{W}$.
for $t = 1$ to T **do**
  for $i = 1$ to N **do**
    Sample a mini-batch ξ_i from local dataset $\mathcal{D}_i$.
    Compute the local gradient with $\mathbf{g}_i^t = \nabla F_i(\mathbf{x}_i^{(t)}, \xi_i)$.
    Local SGD update: $\mathbf{x}_i^{(t+\frac{1}{2})} = \mathbf{x}_i^{(t)} - \gamma \mathbf{g}_i^{(t)}$.
    Broadcasting $\mathbf{x}_i^{(t+\frac{1}{2})}$ to the rest of network.
    Receiving messages from other devices: $\mathbf{z}_{j\to i}^{(t+\frac{1}{2})} = \mathbf{m}_{j\to i}^{(t)} \odot \mathbf{x}_j^{(t+\frac{1}{2})}, j \neq i$.
    Replace the missing values with stale data: $\hat{\mathbf{z}}_{j\to i}^{(t+\frac{1}{2})} = \mathbf{z}_{j\to i}^{(t+\frac{1}{2})} + (1 - \mathbf{m}_{j\to i}^{(t)}) \odot \mathbf{x}_i^{(t+1)}$.
    Consensus update: $\mathbf{x}_i^{(t+1)} = w_{i,i}\mathbf{x}_i^{(t+\frac{1}{2})} + \sum_{j=1,j\neq i}^{N} w_{i,j}\hat{\mathbf{z}}_{j\to i}^{(t+\frac{1}{2})}$.

B. Algorithm

Soft-DSGD, illustrated in Algorithm 1, is designed to address unreliability in UDP transmissions. We adopt the vanilla decentralized training framework [9], where each device maintains its own local parameters and conducts a local SGD update as well as a consensus update in each training iteration.

The key challenges are how to conduct a consensus update with only partially received messages from other devices and how to optimize mixing weight matrix $\mathbf{W}$ according to link reliability matrix $\mathbf{P}$.

Filling lost packets with local parameters. To deal with transmission failures in UDP, we replace the lost packets with local parameters at each device. In particular, the i-th device will fill the missing parameters in $\mathbf{z}_{j\to i}$ with those in $\mathbf{x}_i$ and the filled message $\hat{\mathbf{z}}_{j\to i}$ can expressed as

$$\hat{\mathbf{z}}_{j\to i}^{(t+\frac{1}{2})} = \mathbf{z}_{j\to i}^{(t+\frac{1}{2})} + (1-\mathbf{m}_{j\to i}^{(t)}) \odot \mathbf{x}_i^{(t+\frac{1}{2})}. \tag{3}$$

With the filled parameters, the consensus update can be expressed as

$$\mathbf{x}_i^{(t+1)} = w_{i,i}\mathbf{x}_i^{(t+\frac{1}{2})} + \sum_{j=1,j\neq i}^{N} w_{i,j}\hat{\mathbf{z}}_{j\to i}^{(t+\frac{1}{2})} \tag{4}$$

$$= \mathbf{x}_i^{(t+\frac{1}{2})} + \sum_{j=1,j\neq i}^{N} w_{i,j}\mathbf{m}_{j\to i}^{(t)} \odot (\mathbf{x}_j^{(t+\frac{1}{2})} - \mathbf{x}_i^{(t+\frac{1}{2})}). \tag{5}$$

Note that since we use the local parameters to replace the lost values, there is no additional memory required to store the historically received data.

Due to the randomness of the communication network, the consensus update step becomes stochastic. The following lemma, proved in Appendix, introduces two important matrices $\overline{\mathbf{W}}$ and $\overline{\mathbf{W}^2}$, which characterize the first and second moments of the update parameters and also play an important role in the analysis of the convergence rate.

Lemma 1. *With the updating rule, the expectations of* $\mathbf{X}_{t+1}$ *and* $(\mathbf{X}_l^{(t+1)})^T(\mathbf{X}_l^{(t+1)})$ *can be expressed as*

$$\mathbb{E}\{\mathbf{X}_{t+1}\} = \overline{\mathbf{W}}(\mathbf{X}_t - \gamma\mathbf{G}_t), \tag{6}$$

and

$$\mathbb{E}\{(\mathbf{X}_l^{(t+1)})^T(\mathbf{X}_l^{(t+1)})\} = (\mathbf{X}_l^{(t)} - \gamma\mathbf{G}_l^{(t)})^T\overline{\mathbf{W}^2}(\mathbf{X}_l^{(t)} - \gamma\mathbf{G}_l^{(t)}), \tag{7}$$

where $\overline{\mathbf{W}}$ *and* $\overline{\mathbf{W}^2}$ *are defined as:*

$$\overline{\mathbf{W}}[i,j] := \begin{cases} w_{i,j}p_{i,j}, & i\neq j \\ 1-\sum_{l=1,l\neq i}^{N} w_{i,l}p_{i,l}, & i=j \end{cases} \tag{8}$$

$$\overline{\mathbf{W}^2}[i,j] := \begin{cases} \sum_{l=1}^{N} w_{i,l}p_{i,l}w_{j,l}p_{j,l} + 2p_{i,j}w_{i,j} \\ -p_{i,j}w_{i,j}\sum_{l=1}^{N}\left(p_{i,l}w_{i,l} + p_{j,l}w_{j,l}\right), & i\neq j \\ 1-2\sum_{l=1}^{N} p_{i,l}\left(w_{i,l} - w_{i,l}^2\right) \\ +\sum_{l=1}^{N}\sum_{m=1,m\neq l}^{N} p_{i,l}w_{i,l}p_{i,m}w_{i,m}, & i=j \end{cases} \tag{9}$$

Remark 1. Lemma 1 illustrates the first- and second-order of statistics of the soft-DSGD updating. In expectation, the consensus updates with an unreliable communication network are equivalent to reliable consensus updates with $\overline{\mathbf{W}}$ as the weight matrix, which is also a doubly stochastic matrix.

Optimizing mixing matrix. In vanilla decentralized SGD, the convergence rate of training largely depends on the mixing matrix $\mathbf{W}$. From Lemma 1, the average mixing weight $\bar{\mathbf{W}}$ depends not only on $\mathbf{W}$ but also on the link reliability matrix $\mathbf{P}$. In this paper, two approaches are exploited to select the mixing matrix $\mathbf{W}$, depending on the availability of the matrix $\mathbf{P}$.

1) If link reliability matrix $\mathbf{P}$ is unknown, each link will be treated equally and uniform mixing weights will be adopted, *i.e.*, $\mathbf{W} = \mathbf{J} = \frac{1}{N}\mathbf{1}\mathbf{1}^T$.
2) If link reliability matrix $\mathbf{P}$ is available, (*e.g.,* maintained at a coordinator), we can optimize $\mathbf{W}$ for faster convergence. As will be shown in the next section, the convergence of Soft-DSGD depends on the largest eigenvalue of matrix $\overline{\mathbf{W}^2} - \frac{1}{N}\mathbf{1}\mathbf{1}^T$. Therefore, we optimize $\mathbf{W}$ to minimize the largest eigenvalue of this matrix:

$$\min_{\mathbf{W}} \quad \lambda_{\max}(\overline{\mathbf{W}^2} - \frac{1}{N}\mathbf{1}\mathbf{1}^T) \tag{10}$$

$$\text{s.t.} \quad 0 \leq w_{i,j} \leq 1, \mathbf{W}^T = \mathbf{W}, \mathbf{W}\mathbf{1} = \mathbf{1} \tag{11}$$

In fact, this optimization problem is convex, as shown in the Appendix. Therefore, it can be solved efficiently.

IV. Convergence Analysis

In this section, we analyze convergence of Soft-DSGD and prove that even in unreliable communication networks, Soft-DSGD achieves the same asymptotic convergence rate as the vanilla decentralized SGD with perfect communications.

A. Assumptions

Assumptions on functions. The functions f and f_i have following properties.

- **(L-smoothness)**. *Each local function* $f_i(\cdot)$ *is smooth and with L-Lipschitzian gradients,* i.e.*, there exists a constant* $L > 0$*, such that* $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$*,*

$$\|\nabla f_i(\mathbf{x}) - \nabla f_i(\mathbf{y})\| \leq L\|\mathbf{x}-\mathbf{y}\|. \tag{12}$$

- **(Bounded variance).** *We assume that there exists constants* $\sigma > 0$ *and* $\zeta > 0$*, such that* $\forall \mathbf{x} \in \mathbb{R}^d$*,*

$$\mathbb{E}\{\|\nabla F_i(\mathbf{x},\xi) - \nabla f_i(\mathbf{x})\|\} \leq \sigma^2, \tag{13}$$

$$\frac{1}{N}\sum_{i=1}^{N}\|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\| \leq \zeta^2. \tag{14}$$

  Hence, σ^2 bounds the variance of stochastic gradients at each device and ζ^2 bounds the discrepancy of data distributions at different devices.
- **(Unbiased stochastic gradients)**. Stochastic gradients obtained at each device are unbiased estimates of the real gradients of the local objectives:

$$\mathbb{E}\{\mathbf{g}_i\} = \nabla f_i(\mathbf{x}_i). \tag{15}$$

These assumptions on functions are widely used in the non-convex decentralized optimization literature [9], [14], [16], and are valid in most of applications.

Assumptions on communication networks. Besides the above assumptions on the functions, we make additional assumptions on the unreliable communication network.

- **(Symmetric matrix)**. *The probability for successful transmission from the* i*-th device to the* j*-th device is*

the same as the probability from the j-th device to the i-th device, i.e., $\mathbf{P}^T = \mathbf{P}$.

- **(Independent and stable links)**. *The packet transmission on different links are independent and the link reliability matrix $\mathbf{P}$ remains fixed during training.*

These assumptions on communication networks are reasonable and easy to be satisfied in practice. Due to channel reciprocity, the link reliability is the same for transmissions in two directions on the same communication link. In addition, the assumption on independence of the links is valid as long as the distance between devices are much larger than the wavelength of the signal and the link reliability remains stable if the devices are static during the training. Note that Soft-DSGD can be directly applied in the dynamic environment, where the link reliability matrix changes with time. We concentrate on the static reliability matrix in order to make the analysis easy to understand.

B. Soft-Consensus Algorithm

To analyze the convergence of Soft-DSGD, we first investigate the consensus algorithm with an unreliable communication network. Suppose the devices are initialized with $\{\mathbf{x}_i^{(0)} \in \mathbb{R}^d, i = 1, ..., N\}$, and only the consensus update steps are conducted in each iteration, *i.e.*,

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \sum_{j=1, j\neq i}^{N} w_{i,j}\mathbf{m}_{i\to j}^{(t)} \odot (\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}) \tag{16}$$

Then, we have the following lemma, proved in Appendix, to capture the resistance of the random communication network.

Lemma 2. *Let $\overline{\mathbf{x}}_t$ denote the average of $\mathbf{X}_t$. Following the consensus updating rule, we have*

$$\mathbb{E}\{\overline{\mathbf{x}}_{t+1}\} = \overline{\mathbf{x}}_t, \tag{17}$$

$$\mathbb{E}\{\|\overline{\mathbf{x}}_{t+1} - \overline{\mathbf{x}}_t\|^2\} \leq \frac{\kappa}{N^2}\sum_{i=1}^{N}\|\mathbf{x}_i^{(t)} - \overline{\mathbf{x}}_t\|^2, \tag{18}$$

where $\kappa = 2\max_i \sum_{j=1}^{N} w_{i,j}^2 p_{i,j}(1-p_{i,j})$.

Remark 2. Lemma 2 shows how the average of $\mathbf{X}_t$ behaves with the consensus updates. In particular, the average is preserved in expectation for each step and the expected deviation is bounded by the variance of $\mathbf{X}_t$. κ captures the resistance of the random communication network. If the communication links are deterministic, *i.e.,* $p_{i,j} = 0$ or 1, κ becomes zero. In this case, the average of $\mathbf{X}_t$ will be preserved for each step.

The convergence of the decentralized SGD depends on the convergence rate of the consensus. The convergence rate of consensus with unreliable communications is shown in the following lemma, proved in Appendix.

Lemma 3. *Following the consensus updating rule, the expectation of $\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i^{(t)} - \overline{\mathbf{x}}_t\|^2$ converges to zero at an exponential rate. In particular, we have*

$$\mathbb{E}\{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}_i^{(t)} - \overline{\mathbf{x}}_t\|^2\} \leq \rho^t \sum_{i=1}^{N}\|\mathbf{x}_i^{(0)} - \overline{\mathbf{x}}_0\|^2, \tag{19}$$

where ρ is the largest eigenvalue of matrix $\overline{\mathbf{W}^2} - \frac{1}{N}\mathbf{1}\mathbf{1}^{\mathbf{T}}$.

Remark 3. If the communication network is reliable, the consensus enjoys an exponential convergence rate. Lemma 3 shows that the consensus can also achieve an exponential convergence rate with unreliable communications under the proposed consensus update policy.

C. Convergence

Based on the above assumptions and lemmas, the convergence rate for the proposed decentralized training algorithm with an unreliable communication network can be demonstrated in the following theorem, proved in the appendix.

Convergence of Soft-DSGD Theorem. *Suppose that all local models are initialized with $\mathbf{x}_0 \in \mathbb{R}^d$. Under Assumptions 1-5, if the learning rate satisfies $\gamma L \leq \min\{1, \sqrt{\rho^{-1}} - 1\}$, then after T iterations, we have*

$$\begin{aligned}\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\bar{\mathbf{x}}_t)\|^2 \leq (&\frac{\mathbb{E}[f(\bar{\mathbf{x}}_T)] - \mathbb{E}[f(\bar{\mathbf{x}}_0)]}{\gamma T} \\ &+ \frac{\gamma L}{N}\sigma^2 + \frac{2\gamma L\kappa}{N}\sigma^2 + \frac{6L\kappa\gamma\zeta^2}{N})\frac{1-D}{1-2D} \\ &+ \left(L^2 + \frac{2L\kappa}{\gamma N} + \frac{2(3N+1)L^3\gamma\kappa}{N}\right) \\ &\left(\frac{2\gamma^2\sigma^2\rho}{1-\rho} + \frac{6\gamma^2\zeta^2\rho}{(1-\sqrt{\rho})^2}\right)\frac{1}{1-2D},\end{aligned} \tag{20}$$

where $D = \frac{6\gamma^2 L^2\rho}{(1-\sqrt{\rho})^2}$, $\kappa = 2\max_i \sum_{j=1}^{N} p_{i,j}(1-p_{i,j})w_{i,j}$, and ρ is the largest eigenvalue of the matrix $\overline{\mathbf{W}^2} - \mathbf{J}$.

The resistance of the unreliable communication network is reflected in terms containing κ. If all the communication links are deterministic with $p_{i,j} = 0$ or 1, then $\kappa = 0$, and the results will be consistent with the convergence bound for vanilla decentralized SGD. In addition, the convergence bound depends on ρ to a large degree, which justifies our mixing weight optimization method.

Furthermore, if the learning rate is configured properly, it can achieve a linear speedup in terms of the number of devices, matching the same rate as vanilla decentralized SGD, as indicated in the following corollary, proved in Appendix.

Corollary. Under the same conditions as the above theorem, if the learning rate γ is set as $\gamma = \sqrt{\frac{N}{T}}$, after total T iterations, we have

$$\frac{1}{T}\sum_{t=1}^{T}\|\nabla f(\bar{\mathbf{x}}_t)\|^2 = \mathcal{O}(\frac{1}{\sqrt{NT}}) + \mathcal{O}(\frac{N}{T}), \tag{21}$$

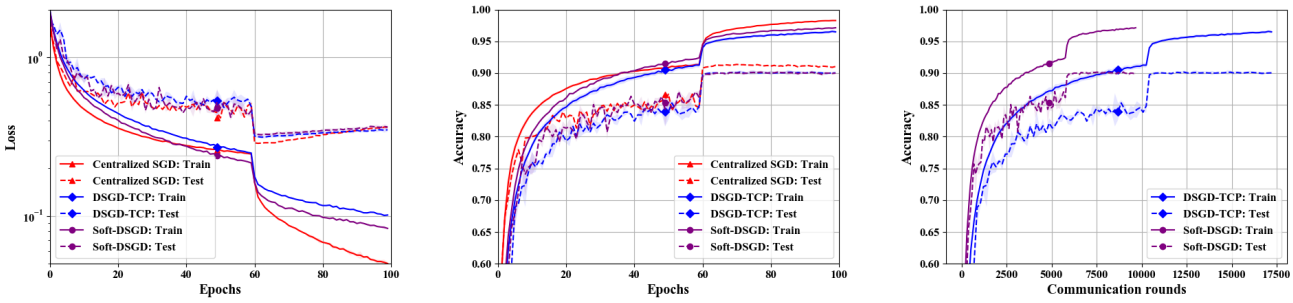
where all other constants are subsumed in $\mathcal{O}$.

Consistency with vanilla decentralized SGD. Recall that vanilla decentralized SGD converges at the asymptotic rate of $\mathcal{O}(\frac{1}{\sqrt{NT}}) + \mathcal{O}(\frac{N}{T})$ [9]. Hence, the decentralized SGD with unreliable communications can achieve the same asymptotic convergence rate as vanilla decentralized SGD that assumes a reliable communication network. Therefore, the asymptotic convergence is not negatively affected by unreliability in the communication network.

V. Experiments

In this section we evaluate the performance of Soft-DSGD with unreliable communication networks.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3152445, IEEE Journal of Selected Topics in Signal Processing



(a) (b) (c)

Fig. 3. (a) Loss vs epochs. (b) Accuracy vs epochs. (c) Accuracy vs communication rounds

A. Experimental setup

We conduct experiments on image classification tasks for evaluation. We train ResNet-20 models [20] on the CIFAR-10 dataset [21], which contains $50,000$ images for training and $10,000$ images for testing. We set the weight decay to 0.0001 and mini-batch size to 32 per device. The initial learning rate is 0.1 and decays by a factor of 10 after 60 epochs. The momentum of 0.9 is used.

Geometric random graphs are generated to represent a communication network. The probability of successful transmission for each link is defined in a way that decays with the distance between the devices, *i.e.,* $p_{i,j} = p_{j,i} = k^{(\frac{d_{i,j}}{r})^2}$, where $d_{i,j}$ represents the distance between the i-th device to the j-th device.

We implement Soft-DSGD with PyTorch and train models with an Nvidia 1080Ti GPU. To simulate the random communication network, a random mask $\mathbf{m}_{i\to j}$ is generated before each communication to determine which part of the data be obtained by the receiver. Since the packet size ($\sim 10^2$) is usually much smaller compared to the number of parameters ($\sim 10^6$) and the random partition of the parameters into packets can be different for each device and time step as long as the random seeds are available for the receivers, the communications for each dimension of parameters can be approximated to be independent. We simulate $\mathbf{m}_{i\to j}$ by sampling from $i.i.d.$ Bernoulli distribution regardless of the packet size.

The proposed approach is compared with vanilla decentralized SGD using TCP, where a communication graph $\mathcal{G}(V, E)$ is constructed first and the devices only exchange information with their neighbours. The neighbours of a device are determined by a probability threshold p_δ. Only links with a success probability larger than p_δ are kept while other links are discarded. After the communication graph $\mathcal{G}(V, E)$ is constructed, the Metropolis-Hastings mixing weights are employed [22], *i.e.,*

$$w_{i,j} = \begin{cases} 0, \quad i \neq j, & (i,j) \in E, \\ \dfrac{1}{(\max\{\deg(i), \deg(l)\} + 1)}, \quad i \neq j, & (i,j) \notin E, \\ 1 - \displaystyle\sum_{l=1, l\neq i}^{N} \dfrac{1}{(\max\{\deg(i), \deg(l)\} + 1)}, & i = j, \end{cases}$$

where $\deg(i)$ denotes the number of neighbours of the i-th device.

With TCP as the communication protocol, the receiver will send the ACK to the transmitter once it successfully receives the packet. Otherwise, the transmitter will resend the last packet. If there are multiple neighbors, the transmitter needs to collect the ACK messages from all its neighbours to ensure reliability.

B. Empirical results

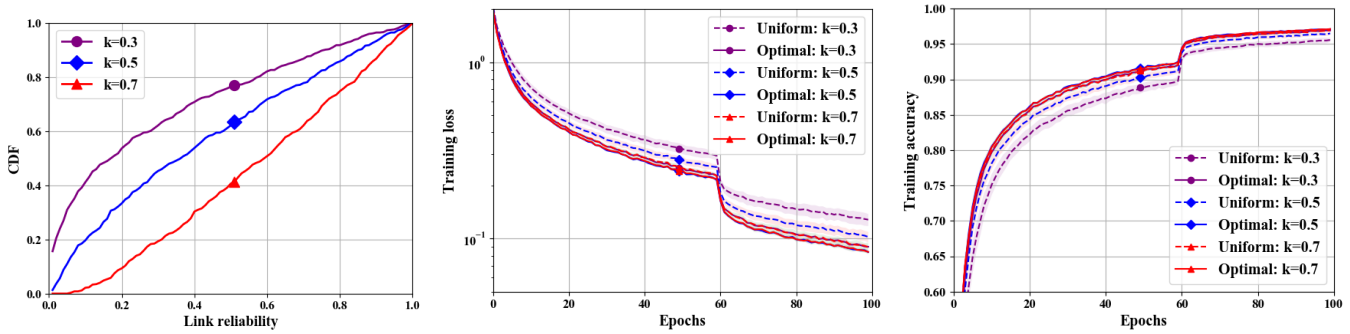
a) Effectiveness of Soft-DSGD: We first compare the Soft-DSGD with vanilla decentralized SGD using TCP. To get reliable results, we run the algorithms on five randomly generated communication graphs. Each graph consists of 16 devices, which are randomly located in a unit square.

Fig. 3 (a) compares the iteration-wise training and testing loss of the Soft-DSGD with the centralized SGD and vanilla decentralized SGD with TCP, which is built on a communication graph using $p_\delta = 0.7$. Fig. 3 (b) shows the iteration-wise training and test accuracy of three algorithms. From both figures, the centralized SGD converges fastest. The training of the Soft-DSGD has a faster convergence than the vanilla decentralized SGD with reliable communication protocols. This is because the Soft-DSGD leverages information from all unreliable communication links and updates with the partially received messages while the vanilla decentralized SGD only updates with messages from a limited number of neighbours.

Fig. 3 (c) shows the convergence with respect to the number of communication rounds. For the Soft-DSGD, the number of communication rounds is the same as the number of iterations, while the packets need to be retransmitted in case of packet loss or transmission errors with TCP. Therefore, the required number of communication rounds is larger comparing with the Soft-DSGD. From the figure, the communication rounds required for TCP to reach 90% training accuracy (or 85% test accuracy) are as twice many as the communication rounds required for the Soft-DSGD.

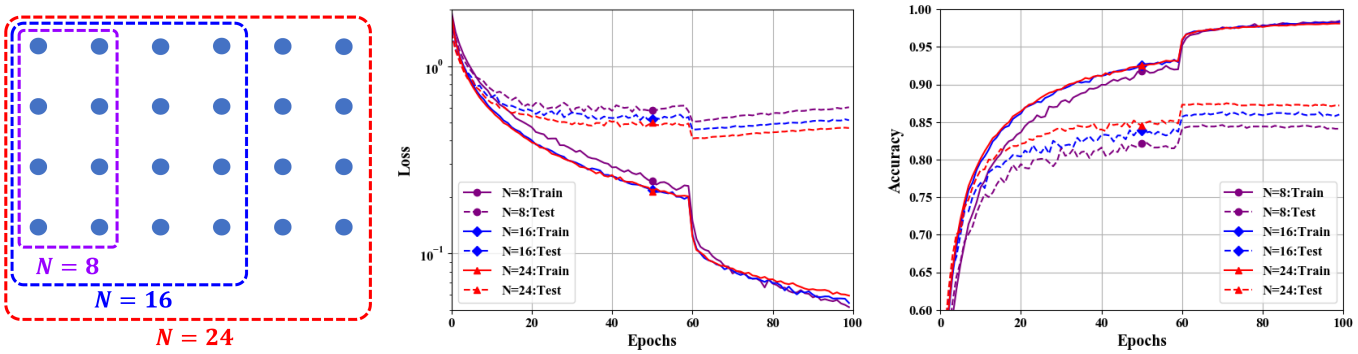
b) Optimal weights vs uniform weights.: We further investigate the effects of optimizing the mixing weight matrix with respect to link reliability matrix. We evaluate the Soft-DSGD using optimal and uniform weights with different link reliability matrices. In particular, we continue to use the randomly communication graphs in a unit square. We keep the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3152445, IEEE Journal of Selected Topics in Signal Processing



(a) (b) (c)

Fig. 4. (a) CDF of link reliability (b) Training loss vs epochs. (c) Training accuracy vs epochs



(a) (b) (c)

Fig. 5. (a) Three communication networks with 8, 16, and 24 devices. (b) Loss vs epochs. (c) Accuracy vs epochs.

positions of devices fixed and change k to generate different link reliability matrices. The cumulative distribution function (CDF) of the link reliability is shown in Fig. 4 (a) with different k.

To compare two weights setting approaches, Fig. 4 (b) and Fig. 4 (c) illustrate the training losses and training accuracy with uniform and optimal mixing weights under different link reliability matrices. We do not include vanilla SGD here because the some of communication graphs are not connected for TCP when $k = 0.3$, while Soft-DSGD still works well. From the figures, the training procedures with both types of weights slow down with the degradation of the communications links. In addition, When $k = 0.7$ and the average reliability of links is high, the performance gap between the uniform and optimal weights is quite small. But as the links quality degrades with $k = 0.5$ and $k = 0.3$, the performance gap between two approaches increases. This is because when there are many links with little probability of transmission success, uniform weights still assign equal weights to these links, which impedes the convergence.

c) Scalability: We also evaluate the performance of the Soft-DSGD with different network sizes. In particular, three communication networks are constructed, consisting of 8, 16, and 24 devices, respectively. As shown in Fig. 5 (a), to ensure the consistency of communication link reliability distribution, the positions of devices are in grids of 4×2, 4×4, and 4×6, respectively. The distances of adjacent devices are same (set as 0.3) in the three networks and each device contains $3,000$ training samples, which are randomly selected from the training set. The loss and accuracy curves of the training and test are shown in Fig. 5 (b) and Fig. 5 (c), respectively. From the figures, both the convergence rate of training and the model performance are improved as the number of devices participated increases, because the increase of number of devices leads to denser communication connections and the increase in the overall training samples.

VI. Conclusion and Future Work

In this paper, we have designed a robust decentralized training framework, called Soft-DSGD, to deal with unreliable communication links. Instead of using connection-orientated protocols, such as TCP, to ensure the reliability but with considerable overhead, Soft-DSGD uses lightweight and unreliable communication protocols, such as UDP, with low communication overhead, but is robust to communication failures. We prove that Soft-DSGD, even with unreliable communications, converges at the same asymptotic rate as the vanilla decentralize SGD over a reliable communication network. From the numerical experiments, Soft-DSGD can leverage information collected from all unreliable communication links to accelerate convergence. There are several interesting future directions to further extend the Soft-DSGD algorithm. For instance, it is interesting to consider directed and asynchronous communication networks, where the devices can update and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3152445, IEEE Journal of Selected Topics in Signal Processing

communicate at different rates. In addition, combining Soft-DSGD with the compression techniques can potentially speed up the distributed training procedure.

APPENDIX

A. Proof of Lemma 1

Proof. $\mathbf{X}_l^{(t+1)}$, the l-th column of $\mathbf{X}_{t+1}$, can be expressed as $\mathbf{X}_l^{(t+1)} = \widetilde{\mathbf{W}}_l^{(t)}\left(\mathbf{X}_l^{(t)} - \gamma \mathbf{G}_l^{(t)}\right)$, where $\widetilde{\mathbf{W}}_l^{(t)} \in \mathbb{R}^{N\times N}$ is the mixing matrix for l-th dimension. $\widetilde{\mathbf{W}}_l^{(t)}$ is obtained by $\widetilde{\mathbf{W}}_l^{(t)} = \mathbf{W}\odot \mathbf{A}_l^{(t)} + \mathbf{I} - \text{Diag}(\mathbf{W}\mathbf{A}_l^{(t)})$, where $\mathbf{A}_l^{(t)} \in \mathbb{R}^{N\times N}$ denotes and sucessfulness of transmission. If the transmission of the l-th parameter from i to j is successful, then $\mathbf{A}_l^{(t)}[i,j] = 1$ and otherwise, $\mathbf{A}_l^{(t)}[i,j] = 0$. Since the link reliability from i to j is $p_{i,j}$, the expectation of $\widetilde{\mathbf{W}}_l^{(t)}[i,j]$ is $p_{i,j}w_{i,j}$ and the expectation of $\widetilde{\mathbf{W}}_l^{(t)}[i,i] = 1 - \sum_{j=1}^N p_{i,j}w_{i,j}$. Therefore, $\mathbb{E}\{\widetilde{\mathbf{W}}_l^{(t)}\} = \overline{\mathbf{W}}$ and

$$\begin{aligned}\mathbb{E}\{\mathbf{X}_l^{(t+1)}\} &= \mathbb{E}\{\widetilde{\mathbf{W}}_l^{(t)}\}\left(\mathbf{X}_l^{(t)} - \gamma\mathbf{G}_l^{(t)}\right)\\ &= \overline{\mathbf{W}}\left(\mathbf{X}_l^{(t)} - \gamma\mathbf{G}_l^{(t)}\right).\end{aligned}$$

By combining all dimensions, we have

$$\mathbb{E}\{\mathbf{X}_{t+1}\} = \overline{\mathbf{W}}\left(\mathbf{X}_t - \gamma\mathbf{G}_t\right).$$

Similarly, we have

$$\begin{aligned}&\mathbb{E}\{(\mathbf{X}_l^{(t+1)})^T(\mathbf{X}_l^{(t+1)})\}\\ &= (\mathbf{X}_l^{(t)} - \gamma\mathbf{G}_l^{(t)})^T\mathbb{E}\{(\widetilde{\mathbf{W}}_l^{(t)})^T(\widetilde{\mathbf{W}}_l^{(t)})\}(\mathbf{X}_l^{(t)} - \gamma\mathbf{G}_l^{(t)}).\end{aligned}$$

Given $\widetilde{\mathbf{W}}_l^{(t)} = \mathbf{W}\odot\mathbf{A}_l^{(t)} + \mathbf{I} - \text{Diag}(\mathbf{W}\mathbf{A}_l^{(t)})$, we have

$$\begin{aligned}&(\widetilde{\mathbf{W}}_l^{(t)})^T(\widetilde{\mathbf{W}}_l^{(t)})\\ &= (\mathbf{W}\odot\mathbf{A}_l^{(t)})^2 + \text{Diag}^2(\mathbf{W}\mathbf{A}_l^{(t)}) + \mathbf{I} + 2\mathbf{W}\odot\mathbf{A}_l^{(t)}\\ &- \text{Diag}(\mathbf{W}\mathbf{A}_l^{(t)}) - (\mathbf{W}\odot\mathbf{A}_l^{(t)})\text{Diag}(\mathbf{W}\mathbf{A}_l^{(t)})\\ &- \text{Diag}(\mathbf{W}\mathbf{A}_l^{(t)})(\mathbf{W}\odot\mathbf{A}_l^{(t)})\end{aligned}$$

Taking expectation, we can get

$$\overline{\mathbf{W}^2} = \mathbb{E}\{(\widetilde{\mathbf{W}}_l^{(t)})^T(\widetilde{\mathbf{W}}_l^{(t)})\}.$$

This concludes the proof of Lemma 1. ∎

B. Proof of Lemma 2

Proof. (a). According to Lemma 1, if $\mathbf{G}_t$ is zero, then

$$\mathbb{E}\{\mathbf{X}_{t+1}\} = \overline{\mathbf{W}}\mathbf{X}_t.$$

Since $\overline{\mathbf{W}}$ is a doubly stochastic matrix, $\mathbb{E}\{\overline{\mathbf{x}}_{t+1}\} = \frac{1}{N}\mathbf{1}^T\mathbb{E}\{\mathbf{X}_{t+1}\} = \frac{1}{N}\mathbf{1}^T\overline{\mathbf{W}}\mathbf{X}_t = \frac{1}{N}\mathbf{1}^T\mathbf{X}_t = \overline{\mathbf{x}}_t$. (b). We have

$$\begin{aligned}&\|\frac{1}{N}\sum_{i=1}^N \mathbf{x}_i^{(t+1)} - \frac{1}{N}\sum_{i=1}^N \mathbf{x}_i^{(t)})\|^2\\ &= \|\frac{1}{N}\sum_{i=1}^N\sum_{j=1,j\neq i}^N w_{i,j}\mathbf{m}_{i\to j}^{(t)}\odot(\mathbf{x}_j^{(t)} - \mathbf{x}_i^{(t)}))\|^2\\ &= \|\frac{1}{N}\sum_{i=1}^N\sum_{j=1,j\neq i}^N w_{i,j}\mathbf{m}_{i\to j}^{(t)}\odot((\mathbf{x}_j^{(t)} - \bar{\mathbf{x}}_t) - (\mathbf{x}_i^{(t)} - \bar{\mathbf{x}}_t))\|^2\\ &= \|\frac{1}{N}\sum_{i=1}^N(\mathbf{x}_i^{(t)} - \bar{\mathbf{x}}_t)\odot\left(\sum_{j=1,j\neq i}^N w_{i,j}\left(\mathbf{m}_{i\to j}^{(t)} - \mathbf{m}_{j\to i}^{(t)}\right)\right)\|^2.\end{aligned}$$

Each item in $\mathbf{m}_{i\to j}^{(t)} - \mathbf{m}_{j\to i}^{(t)}$ is a variable with mean zero and variance of $2p_{i,j}(1-p_{i,j})$. Therefore, we have

$$\begin{aligned}&\mathbb{E}\{\|\overline{\mathbf{x}}_{t+1} - \overline{\mathbf{x}}_t\|^2\}\\ &= \frac{1}{N^2}\sum_{i=1}^N\left(\|\mathbf{x}_i^{(t)} - \bar{\mathbf{x}}_t\|^2\sum_{j=1,j\neq i}^N 2w_{i,j}^2p_{i,j}(1-p_{i,j})\right)\\ &\leq \frac{\kappa}{N^2}\sum_{i=1}^N\|\mathbf{x}_i^{(t)} - \bar{\mathbf{x}}_t\|^2.\end{aligned}$$

This concludes the proof of Lemma 2. ∎

C. Proof of Lemma 3

Proof. We first consider the l-th dimension. Let $\boldsymbol{\beta}_l(t) = (\mathbf{I} - \mathbf{J})\mathbf{X}_l^{(t)}$. Then we have

$$\boldsymbol{\beta}_l(t) = (\widetilde{\mathbf{W}}_l^{(t)} - \mathbf{J})\boldsymbol{\beta}_l(t-1).$$

Now, taking the expected norm of $\boldsymbol{\beta}_l(t)$ given $\boldsymbol{\beta}_l(t-1)$, we have

$$\begin{aligned}&\mathbb{E}\{\|\boldsymbol{\beta}_l(t)\|^2|\boldsymbol{\beta}_l(t-1)\}\\ &= \boldsymbol{\beta}_l(t-1)^T\mathbb{E}\{(\widetilde{\mathbf{W}}_l^{(t)} - \mathbf{J})^T(\widetilde{\mathbf{W}}_l^{(t)} - \mathbf{J})\}\boldsymbol{\beta}_l(t-1)\\ &= \boldsymbol{\beta}_l(t-1)^T\left(\overline{\mathbf{W}^2} - \mathbf{J}\right)\boldsymbol{\beta}_l(t-1)\\ &\leq \rho\|\boldsymbol{\beta}_l(t-1)\|^2.\end{aligned}$$

Combine all dimensions together and let $\boldsymbol{\beta}(t) = (\mathbf{I}-\mathbf{J})\mathbf{X}_t$, hence $\|\boldsymbol{\beta}(t)\|_F^2 = \sum_{i=1}^N\|\mathbf{x}_i^{(t)} - \overline{\mathbf{x}}_t\|^2$. Then we have

$$\mathbb{E}\{\|\boldsymbol{\beta}(t)\|^2|\boldsymbol{\beta}(t-1)\} \leq \rho\|\boldsymbol{\beta}(t-1)\|^2.$$

Repeat the above procedure over t, then we have

$$\mathbb{E}\{\|\boldsymbol{\beta}(t)\|^2\} \leq \rho^t\|\boldsymbol{\beta}(0)\|^2.$$

∎

D. Proof of Theorem

Proof. Due to the Lipschitz smoothness of f, we have

$$f(\bar{\mathbf{x}}_{t+1}) - f(\bar{\mathbf{x}}_t) \leq \langle\nabla f(\bar{\mathbf{x}}_t), \bar{\mathbf{x}}_{t+1} - \bar{\mathbf{x}}_t\rangle + \frac{L}{2}\|\bar{\mathbf{x}}_{t+1} - \bar{\mathbf{x}}_t\|^2.$$

According to Lemma 2, we have

$$\bar{\mathbf{x}}_{t+1} = \bar{\mathbf{x}}_t - \bar{\mathbf{g}}_t,$$

where $\bar{\mathbf{g}}_t = \frac{1}{N}\sum_{i=1}^N\mathbf{g}_t^{(i)}$. Let $\overline{\nabla f}(\mathbf{x}_t) := \frac{1}{N}\sum_{i=1}^N\nabla f_i(\mathbf{x}_i)$. Taking the expectations of both sides, we have

$$\begin{aligned}&\mathbb{E}f(\bar{\mathbf{x}}_{t+1}) - \mathbb{E}f(\bar{\mathbf{x}}_t)\\ &\leq \langle\nabla f(\bar{\mathbf{x}}_t), \mathbb{E}\{\bar{\mathbf{x}}_{t+1} - \bar{\mathbf{x}}_t\}\rangle + \frac{L}{2}\mathbb{E}\|\bar{\mathbf{x}}_{t+1} - \bar{\mathbf{x}}_t\|^2\\ &= -\gamma\langle\nabla f(\bar{\mathbf{x}}_t), \mathbb{E}\bar{\mathbf{g}}_t\rangle + \frac{L}{2}\mathbb{E}\|\bar{\mathbf{x}}_{t+1} - \bar{\mathbf{x}}_t\|^2\\ &= -\gamma\langle\nabla f(\bar{\mathbf{x}}_t), \overline{\nabla f}(\mathbf{x}_t)\rangle + \frac{L}{2}\mathbb{E}\|\bar{\mathbf{x}}_{t+1} - \bar{\mathbf{x}}_t\|^2.\end{aligned}$$

For the first term, we have

$$\begin{aligned}&\langle\nabla f(\bar{\mathbf{x}}_t), \overline{\nabla f}(\mathbf{x}_t)\rangle\\&= \frac{1}{2}(\|\nabla f(\bar{\mathbf{x}}_t)\|^2 + \|\overline{\nabla f}(\mathbf{x}_t)\|^2 - \|\nabla f(\bar{\mathbf{x}}_t) - \overline{\nabla f}(\mathbf{x}_t)\|^2),\end{aligned}$$

and $\|\nabla f(\bar{\mathbf{x}}_t) - \overline{\nabla f}(\mathbf{x}_t)\|^2$ can be bounded by

$$\begin{aligned}\|\nabla f(\bar{\mathbf{x}}_t) - \overline{\nabla f}(\mathbf{x}_t)\|^2 &= \|\frac{1}{N}\sum_{i=1}^N[\nabla f_i(\bar{\mathbf{x}}_t) - \nabla f_i(\mathbf{x}_t^{(i)})]\|^2\\&\le \frac{1}{N}\sum_{i=1}^N\|\nabla f_i(\bar{\mathbf{x}}_t) - \nabla f_i(\mathbf{x}_t^{(i)})\|^2\\&\le \frac{L^2}{N}\sum_{i=1}^N\|\bar{\mathbf{x}}_t - \mathbf{x}_t^{(i)}\|^2.\end{aligned}$$

For the second term, we have

$$\begin{aligned}&\frac{L}{2}\mathbb{E}\{\|\bar{\mathbf{x}}_{t+1} - \bar{\mathbf{x}}_t\|^2\}\\&= \frac{L}{2}\mathbb{E}\{\|\overline{\mathbf{x}}_{t+1} - (\overline{\mathbf{x}}_t - \gamma\overline{\mathbf{g}}_t) - \gamma\overline{\mathbf{g}}_t\|^2\}\\&= \underbrace{\frac{L}{2}\mathbb{E}\{\|\overline{\mathbf{x}}_{t+1} - (\overline{\mathbf{x}}_t - \gamma\overline{\mathbf{g}}_t)\|^2\}}_{T_1} + \underbrace{\frac{\gamma^2 L}{2}\mathbb{E}\{\|\overline{\mathbf{g}}_t\|^2\}}_{T_2}.\end{aligned}$$

With Lemma 2, T_1 can be bounded by

$$\begin{aligned}T_1 &\le \frac{L\kappa}{2N^2}\sum_{i=1}^N\mathbb{E}\{\Big(\|\mathbf{x}_i^{(t)} - \gamma\mathbf{g}_i^{(t)} - \bar{\mathbf{x}}_t + \gamma\bar{\mathbf{g}}_t\|^2\Big)\}\\&\le \frac{L\kappa}{N^2}\sum_{i=1}^N\mathbb{E}\{\|\mathbf{x}_i^{(t)} - \bar{\mathbf{x}}_t\|^2\} + \frac{L\kappa\gamma^2}{N^2}\underbrace{\sum_{i=1}^N\mathbb{E}\{\|\mathbf{g}_i^{(t)} - \bar{\mathbf{g}}_t\|^2\}}_{T_3}.\end{aligned}$$

In addition, T_2 can be bounded by

$$\begin{aligned}T_2 &= \frac{\gamma^2 L}{2}\mathbb{E}\{\|\frac{1}{N}\sum_{i=1}^N\Big(\mathbf{g}_i^{(t)} - \nabla f_i(\mathbf{x}_i^{(t)}) + \nabla f_i(\mathbf{x}_i^{(t)})\Big)\|^2\}\\&= \frac{\gamma^2 L}{2N^2}\sum_{i=1}^N\mathbb{E}\{\|\mathbf{g}_i^{(t)} - \nabla f_i(\mathbf{x}_i^{(t)})\|^2\}\\&+ \sum_{i=1}^N\mathbb{E}\{\frac{\gamma^2 L}{2N^2}\|\nabla f_i(\mathbf{x}_i^{(t)})\|^2\}\\&\le \frac{\gamma^2 L}{2N}\sigma^2 + \frac{\gamma^2 L}{2}\mathbb{E}\{\|\overline{\nabla f}(\mathbf{x}_t)\|^2\}.\end{aligned}$$

T_3 is bounded by

$$\begin{aligned}T_3 &= \sum_{i=1}^N\mathbb{E}\{\|\Big(\mathbf{g}_i^{(t)} - \nabla f_i(\mathbf{x}_i^{(t)})\Big) + \Big(\nabla f_i(\mathbf{x}_i^{(t)}) - \nabla f_i(\bar{\mathbf{x}}_t)\Big)\\&+ (\nabla f_i(\bar{\mathbf{x}}_t) - \nabla f(\bar{\mathbf{x}}_t)) - \big(\bar{\mathbf{g}}_t - \overline{\nabla f}(\mathbf{x}_t)\big)\\&- \big(\overline{\nabla f}(\mathbf{x}_t) - \nabla f(\bar{\mathbf{x}}_t)\big)\|^2\}\\&= \sum_{i=1}^N\mathbb{E}\{\|\Big(\mathbf{g}_i^{(t)} - \nabla f_i(\mathbf{x}_i^{(t)}) - \bar{\mathbf{g}}_t + \overline{\nabla f}(\mathbf{x}_t)\Big)\\&+ \Big(\nabla f_i(\mathbf{x}_i^{(t)}) - \nabla f_i(\bar{\mathbf{x}}_t)\Big) + (\nabla f_i(\bar{\mathbf{x}}_t) - \nabla f(\bar{\mathbf{x}}_t))\\&- \big(\overline{\nabla f}(\mathbf{x}_t) - \nabla f(\bar{\mathbf{x}}_t)\big)\|^2\}\\&= \sum_{i=1}^N\mathbb{E}\{\|\frac{N-1}{N}\Big(\mathbf{g}_i^{(t)} - \nabla f_i(\mathbf{x}_i^{(t)})\Big)\\&- \frac{1}{N}\sum_{j=1,j\neq i}^N\Big(\mathbf{g}_j^{(t)} - \nabla f_j(\mathbf{x}_j^{(t)})\Big) + \Big(\nabla f_i(\mathbf{x}_i^{(t)}) - \nabla f_i(\bar{\mathbf{x}}_t)\Big)\\&+ (\nabla f_i(\bar{\mathbf{x}}_t) - \nabla f(\bar{\mathbf{x}}_t)) - \big(\overline{\nabla f}(\mathbf{x}_t) - \nabla f(\bar{\mathbf{x}}_t)\big)\|^2\}\\&= \sum_{i=1}^N\mathbb{E}\{\|\frac{N-1}{N}\Big(\mathbf{g}_i^{(t)} - \nabla f_i(\mathbf{x}_i^{(t)})\Big)\|^2\}\\&+ \sum_{i=1}^N\sum_{j=1,j\neq i}^N\mathbb{E}\{\|\frac{1}{N}\Big(\mathbf{g}_j^{(t)} - \nabla f_j(\mathbf{x}_j^{(t)})\Big)\|^2\}\\&+ \sum_{i=1}^N\mathbb{E}\{\|\Big(\nabla f_i(\mathbf{x}_i^{(t)}) - \nabla f_i(\bar{\mathbf{x}}_t)\Big) + (\nabla f_i(\bar{\mathbf{x}}_t) - \nabla f(\bar{\mathbf{x}}_t))\\&- \big(\overline{\nabla f}(\mathbf{x}_t) - \nabla f(\bar{\mathbf{x}}_t)\big)\|^2\}\\&\le \frac{(N-1)^2}{N^2}\sum_{i=1}^N\mathbb{E}\{\|\Big(\mathbf{g}_i^{(t)} - \nabla f_i(\mathbf{x}_i^{(t)})\Big)\|\}^2\\&+ \frac{1}{N}\sum_{i=1}^N\mathbb{E}\{\|\Big(\mathbf{g}_i^{(t)} - \nabla f_i(\mathbf{x}_i^{(t)})\Big)\|^2\}\\&+ 3\sum_{i=1}^N\mathbb{E}\{\|\Big(\nabla f_i(\mathbf{x}_i^{(t)}) - \nabla f_i(\bar{\mathbf{x}}_t)\Big)\|^2\}\\&+ 3\sum_{i=1}^N\mathbb{E}\{\|(\nabla f_i(\bar{\mathbf{x}}_t) - \nabla f(\bar{\mathbf{x}}_t))\|^2\}\\&+ 3\sum_{i=1}^N\mathbb{E}\{\|\big(\overline{\nabla f}(\mathbf{x}_t) - \nabla f(\bar{\mathbf{x}}_t)\big)\|^2\}\\&\le \frac{N^2 - 2N + 2}{N}\sigma^2 + 3NL^2\sum_{i=1}^N\mathbb{E}\{\|\mathbf{x}_i^{(t)} - \bar{\mathbf{x}}_t\|^2\} + 3N\zeta^2\\&+ 3L^2\sum_{i=1}^N\mathbb{E}\{\|\mathbf{x}_i^{(t)} - \bar{\mathbf{x}}_t\|^2\}\\&\le N\sigma^2 + 3(N+1)L^2\sum_{i=1}^N\mathbb{E}\{\|\mathbf{x}_i^{(t)} - \bar{\mathbf{x}}_t\|^2\} + 3N\zeta^2.\end{aligned}$$

Putting everything back, we have

$$\begin{aligned}&\mathbb{E}\{f(\bar{\mathbf{x}}_{t+1})\} - \mathbb{E}\{f(\bar{\mathbf{x}}_t)\}\\&\le -\frac{\gamma}{2}\mathbb{E}\{\|\nabla f(\bar{\mathbf{x}}_t)\|^2\} - \frac{\gamma}{2}\mathbb{E}\{\|\overline{\nabla f}(\mathbf{x}_t)\|^2\}\\&+ \frac{\gamma L^2}{2N}\sum_{i=1}^N\mathbb{E}\{\|\bar{\mathbf{x}}_t - \mathbf{x}_i^{(t)}\|^2\} + \frac{\kappa L}{N^2}\sum_{i=1}^N\mathbb{E}\{\|\bar{\mathbf{x}}_t - \mathbf{x}_i^{(t)}\|^2\}\\&+ \frac{\kappa\gamma^2 L}{N^2}(N\sigma^2 + (3N+1)L^2\sum_{i=1}^N\mathbb{E}\{\|\bar{\mathbf{x}}_t - \mathbf{x}_t^{(i)}\|^2\} + 3N\zeta^2)\\&+ \frac{\gamma^2 L}{2N}\sigma^2 + \frac{\gamma^2 L}{2}\sum_{i=1}^N\mathbb{E}\{\|\overline{\nabla f}(\bar{\mathbf{x}}_t)\|^2\}\\&= -\frac{\gamma}{2}\mathbb{E}\{\|\nabla f(\bar{\mathbf{x}}_t)\|^2\} - (\frac{\gamma}{2} - \frac{\gamma^2 L}{2})\|\overline{\nabla f}(\mathbf{x}_t)\|^2\\&+ (\frac{\gamma L^2}{2N} + \frac{L\kappa}{N^2} + \frac{(3N+1)L^3\gamma^2\kappa}{N^2})\sum_{i=1}^N\mathbb{E}\{\|\mathbf{x}_t^{(i)} - \bar{\mathbf{x}}_t\|^2\}\end{aligned}$$

$$+\frac{3L\kappa\gamma^2\zeta^2}{N}.$$

Summing over t and taking the average, we can get

$$\begin{aligned}
&\frac{\mathbb{E}\{f(\bar{\mathbf{x}}_T)\}-f(\bar{\mathbf{x}}_0)}{T}\\
&\le -\frac{\gamma}{2T}\sum_{T=1}^{T}\mathbb{E}\{\|\nabla f(\bar{\mathbf{x}}_t)\|^2\}\\
&-(\frac{\gamma}{2T}-\frac{\gamma^2 L}{2T})\sum_{T=1}^{T}\mathbb{E}\{\|\overline{\nabla f}(\mathbf{x}_t)\|^2\}\\
&+(\frac{\gamma L^2}{2N}+\frac{L\kappa}{N^2}+\frac{(3N+1)L^3\gamma^2\kappa}{N^2})\\
&\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N}\mathbb{E}\{\|\mathbf{x}_t^{(i)}-\bar{\mathbf{x}}_t\|^2\}+\frac{\gamma^2 L}{2N}\sigma^2+\frac{\gamma^2 L\kappa}{n}\sigma^2\\
&+\frac{3L\kappa\gamma^2\zeta^2}{N}.
\end{aligned}$$

By minor rearranging, we get

$$\begin{aligned}
&\frac{1}{T}\sum_{T=1}^{T}\|\nabla f(\bar{\mathbf{x}}_t)\|^2\\
&\le\frac{2\mathbb{E}[f(\bar{\mathbf{x}}_T)]-\mathbb{E}[f(\bar{\mathbf{x}}_0)]}{\gamma T}-\frac{1-\gamma L}{T}\sum_{T=1}^{T}\mathbb{E}\{\|\overline{\nabla f}(\mathbf{x}_t)\|^2\}\\
&+(\frac{L^2}{NT}+\frac{2L\kappa}{\gamma N^2 T}+\frac{2(3N+1)L^3\gamma\kappa}{N^2 T})\\
&\sum_{t=1}^{T}\sum_{i=1}^{N}\mathbb{E}\{\|\mathbf{x}_t^{(i)}-\bar{\mathbf{x}}_t\|^2\}+\frac{\gamma L}{N}\sigma^2+\frac{2\gamma L\kappa}{N}\sigma^2\\
&+\frac{6L\kappa\gamma\zeta^2}{N}.
\end{aligned}$$

Next we are going to bound $\sum_{t=1}^{T}\sum_{i=1}^{N}\mathbb{E}\{\|\mathbf{x}_t^{(i)}-\bar{\mathbf{x}}_t\|^2\}$. Letting $\boldsymbol{\beta}(t)=(\mathbf{I}-\mathbf{J})\mathbf{X}_t$, we have $\|\boldsymbol{\beta}(t)\|_F^2=\sum_{i=1}^{N}\|\mathbf{x}_i^{(t)}-\overline{\mathbf{x}}_t\|^2$.

Let's consider the l-th dimension. From Lemma 3, we have

$$\begin{aligned}
\boldsymbol{\beta}_l(t)&=(\widetilde{\mathbf{W}}_l^{(t-1)}-\mathbf{J})\left(\mathbf{X}_l^{(t-1)}-\gamma\mathbf{G}_l^{(t-1)}\right)\\
&=\widetilde{\mathbf{W}}_l^{(t)}\boldsymbol{\beta}_l(t-1)-\gamma(\widetilde{\mathbf{W}}_l^{(t)}-\mathbf{J})\mathbf{G}_l^{(t-1)}\\
&=\cdots\\
&=\prod_{k=1}^{t-1}\widetilde{\mathbf{W}}_l^{(t)}\boldsymbol{\beta}_l(0)-\gamma\sum_{k=1}^{t-1}\left((\prod_{m=1}^{t-1}\widetilde{\mathbf{W}}_l^{(t)}-\mathbf{J})\right)\mathbf{G}_l^{(k)}.
\end{aligned}$$

Since all devices are initialized with $\mathbf{x}_0$, $\boldsymbol{\beta}_l(0)=0$, we have,

$$\begin{aligned}
&\mathbb{E}\{\|\boldsymbol{\beta}(t)\|_F^2\}=\sum_{l=1}^{d}\mathbb{E}\{\|\boldsymbol{\beta}_l(t)\|^2\}\\
&=\gamma^2\sum_{l=1}^{d}\mathbb{E}\|\sum_{k=1}^{t-1}\left((\prod_{m=1}^{t-1}\widetilde{\mathbf{W}}_l^{(t)}-\mathbf{J})\right)\mathbf{G}_l^{(k)}\|^2\\
&=2\gamma^2\underbrace{\sum_{l=1}^{d}\mathbb{E}\|\sum_{k=1}^{t-1}\left((\prod_{m=1}^{t-1}\widetilde{\mathbf{W}}_l^{(t)}-\mathbf{J})\right)(\mathbf{G}_l^{(k)}-\nabla\mathbf{F}_l^{(k)})\|^2}_{T_4}\\
&\quad+2\gamma^2\underbrace{\sum_{l=1}^{d}\mathbb{E}\|\sum_{k=1}^{t-1}\left((\prod_{m=1}^{t-1}\widetilde{\mathbf{W}}_l^{(t)}-\mathbf{J})\right)(\nabla\mathbf{F}_l^{(k)})\|^2}_{T_5}.
\end{aligned}$$

For T_4, we have

$$\begin{aligned}
\mathbb{E}\{T_4\}&=\sum_{l=1}^{d}\sum_{k=1}^{t-1}\mathbb{E}\{\|\left((\prod_{m=1}^{t-1}\widetilde{\mathbf{W}}_l^{(t)}-\mathbf{J})\right)(\mathbf{G}_l^{(k)}-\nabla\mathbf{F}_l^{(k)})\|^2\}\\
&\le\sum_{l=1}^{d}\sum_{k=1}^{t-1}\rho^{t-k}\mathbb{E}\{\|\mathbf{G}_l^{(k)}-\nabla\mathbf{F}_l^{(k)}\|^2\}\\
&\le\sum_{k=1}^{t-1}\rho^{t-k}\mathbb{E}\{\|\mathbf{G}_k-\nabla\mathbf{F}_k\|_F^2\}\\
&\le N\sigma^2\rho(1+\rho+\rho^2+\cdots+\rho^t)\\
&\le\frac{N\sigma^2\rho}{1-\rho}.
\end{aligned}$$

For the second term, let $\mathbf{A}_{q,p}:=\prod_{m=q}^{p}\widetilde{\mathbf{W}}_l^{(t)}-\mathbf{J}$, then

$$\begin{aligned}
\mathbb{E}\{T_5\}&=\sum_{l=1}^{d}\sum_{k=1}^{t-1}\mathbb{E}\{\|\mathbf{A}_{k,t-1}\nabla\mathbf{F}_l^{(k)}\|^2\}\\
&+\sum_{l=1}^{d}\sum_{k=1}^{t-1}\sum_{m=1,m\neq k}^{t-1}\mathbb{E}\{(\mathbf{A}_{k,t-1}\nabla\mathbf{F}_l^{(k)})^T(\mathbf{A}_{m,t-1}\nabla\mathbf{F}_l^{(m)})\}\\
&\le\sum_{l=1}^{d}\sum_{k=1}^{t-1}\rho^{t-k}\mathbb{E}\{\|\nabla\mathbf{F}_l^{(k)}\|^2\\
&+\sum_{l=1}^{d}\sum_{k=1}^{t-1}\sum_{m=1,m\neq k}^{t-1}\mathbb{E}\{\|\mathbf{A}_{k,t-1}\nabla\mathbf{F}_l^{(k)})\|\|(\mathbf{A}_{m,t-1}\nabla\mathbf{F}_l^{(m)})\|\}\\
&\le\sum_{k=1}^{t-1}\rho^{t-k}\mathbb{E}\{\|\nabla\mathbf{F}_k\|_F^2\\
&+\sum_{l=1}^{d}\sum_{k=1}^{t-1}\sum_{m=1,m\neq k}^{t-1}\mathbb{E}\{\frac{1}{2\epsilon}\|\mathbf{A}_{k,t-1}\nabla\mathbf{F}_l^{(k)})\|^2\\
&+\frac{\epsilon}{2}\|(\mathbf{A}_{m,t-1}\nabla\mathbf{F}_l^{(m)})\|^2\}\\
&\le\sum_{k=1}^{t-1}\rho^{t-k}\mathbb{E}\{\|\nabla\mathbf{F}_k\|_F^2\\
&+\sum_{l=1}^{d}\sum_{k=1}^{t-1}\sum_{m=1,m\neq k}^{t-1}\mathbb{E}\{\frac{\rho^{t-k}}{2\epsilon}\|\nabla\mathbf{F}_l^{(k)})\|^2+\frac{\epsilon\rho^{t-m}}{2}\|\nabla\mathbf{F}_l^{(m)})\|^2\}\\
&=\sum_{k=1}^{t-1}\rho^{t-k}\mathbb{E}\{\|\nabla\mathbf{F}_k\|_F^2\\
&+\sum_{k=1}^{t-1}\sum_{m=1,m\neq k}^{t-1}\mathbb{E}\{\frac{\rho^{t-k}}{2\epsilon}\|\nabla\mathbf{F}_k\|_F^2+\frac{\epsilon\rho^{t-m}}{2}\|\mathbf{F}_m\|_F^2.
\end{aligned}$$

By setting $\epsilon=\rho^{\frac{p-q}{2}}$, we have

$$\begin{aligned}
\mathbb{E}\{T_5\}&\le\sum_{k=1}^{t-1}\rho^{t-k}\mathbb{E}\{\|\nabla\mathbf{F}_k\|_F^2\}\\
&+\frac{1}{2}\sum_{k=1}^{t-1}\sum_{m=1,m\neq k}^{t-1}\mathbb{E}\{\sqrt{\rho}^{2t-k-m}\left(\|\nabla\mathbf{F}_k\|_F^2+\|\nabla\mathbf{F}_m\|^2\right)
\end{aligned}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSTSP.2022.3152445, IEEE Journal of Selected Topics in Signal Processing

$$
\begin{aligned}
&\leq \sum_{k=1}^{t-1} \rho^{t-k} \mathbb{E}\{\|\nabla \mathbf{F}_k\|_F^2\} \\
&+ \sum_{k=1}^{t-1} \left(\sqrt{\rho}^{t-k} \mathbb{E}\{\|\mathbf{F}_k\|_F^2\} \cdot \sum_{m=1, m\neq k}^{t-1} \sqrt{\rho}^{t-m} \right) \\
&\leq \sum_{k=1}^{t-1} \rho^{t-k} \mathbb{E}\{\|\nabla \mathbf{F}_k\|_F^2\} \\
&+ \sum_{k=1}^{t-1} \left(\sqrt{\rho}^{t-k} \mathbb{E}\{\|\mathbf{F}_k\|_F^2\} \cdot \sum_{m=1}^{t-1} \sqrt{\rho}^{t-m} - \sqrt{\rho}^{t-k} \right) \\
&\leq \frac{\sqrt{\rho}}{1-\sqrt{\rho}} \sum_{k=1}^{t-1} \sqrt{\rho}^{t-k} \mathbb{E}\|\nabla \mathbf{F}_k\|_F^2.
\end{aligned}
$$

Putting T_4 and T_5 back, we have

$$
\begin{aligned}
&\frac{1}{NT} \sum_{t=1}^{T} \mathbb{E}\{\|\boldsymbol{\beta(t)}\|\} \\
&\leq \frac{2\gamma^2\sigma^2\rho}{1-\rho} + \frac{2\gamma^2\sqrt{\rho}}{(1-\sqrt{\rho})NT} \sum_{t=1}^{T} \sum_{k=1}^{t-1} \sqrt{\rho}^{t-k} \mathbb{E}\|\nabla \mathbf{F}_k\|^2 \\
&= \frac{2\gamma^2\sigma^2\rho}{1-\rho} + \frac{2\gamma^2\sqrt{\rho}}{(1-\sqrt{\rho})NT} \sum_{t=1}^{T} \mathbb{E}\{\|\nabla \mathbf{F}_t\|_F^2\} \sum_{k=1}^{T-t} \sqrt{\rho}^{k} \\
&\leq \frac{2\gamma^2\sigma^2\rho}{1-\rho} + \frac{2\gamma^2\sqrt{\rho}}{(1-\sqrt{\rho})NT} \sum_{t=1}^{T} \mathbb{E}\{\|\nabla \mathbf{F}_t\|^2\} \frac{\sqrt{\rho}}{1-\sqrt{\rho}} \\
&= \frac{2\gamma^2\sigma^2\rho}{1-\rho} + \frac{2\gamma^2\rho}{(1-\sqrt{\rho})^2 NT} \sum_{t=1}^{T} \mathbb{E}\{\|\nabla \mathbf{F}_t\|^2\}.
\end{aligned}
$$

In addition, $\|\nabla \mathbf{F}_t\|_F^2$ can be bounded by

$$
\begin{aligned}
&\|\nabla \mathbf{F}_t\|_F^2 = \sum_{i=1}^{N} \|\nabla f(\mathbf{x}_i^{(t)})\|^2 \\
&= \sum_{i=1}^{N} \|\nabla f_i(\mathbf{x}_i^{(t)}) - \nabla f(\mathbf{x}_i^{(t)}) + \nabla f(\mathbf{x}_i^{(t)}) \\
&- \nabla f(\bar{\mathbf{x}}_t) + \nabla f(\bar{\mathbf{x}}_t)\|^2 \\
&\leq 3 \sum_{i=1}^{N} \|\nabla f_i(\mathbf{x}_i^{(t)}) - \nabla f(\mathbf{x}_i^{(t)})\|^2 \\
&+ 3 \sum_{i=1}^{N} \|\nabla f(\mathbf{x}_i^{(t)}) - \nabla f(\bar{\mathbf{x}}_t)\|^2 \\
&+ 3 \sum_{i=1}^{N} \|\nabla f(\bar{\mathbf{x}}_t)\|^2 \\
&\leq 3N\zeta^2 + 3L^2 \sum_{i=1}^{N} \|\mathbf{x}_t^{(i)} - \bar{\mathbf{x}}_t\|^2 + 3N\|\nabla f(\bar{\mathbf{x}}_t)\|^2.
\end{aligned}
$$

Plugging back we have

$$
\begin{aligned}
&\frac{1}{NT} \sum_{t=1}^{T} \sum_{i=1}^{N} \mathbb{E}\{\|\mathbf{x}_t^{(i)} - \bar{\mathbf{x}}_t\|^2\} \\
&\leq \frac{6N\gamma^2L^2\rho}{(1-\sqrt{\rho})NT} \sum_{t=1}^{T} \sum_{i=1}^{N} \mathbb{E}\{\|\mathbf{x}_t^{(i)} - \bar{\mathbf{x}}_t\|^2\} \\
&+ \frac{6N\gamma^2L^2\rho}{(1-\sqrt{\rho})NT} \sum_{t=1}^{T} \mathbb{E}\|\nabla f(\bar{\mathbf{x}}_t)\|^2 \\
&+ \frac{2\gamma^2\sigma^2\rho}{1-\rho} + \frac{6\zeta^2\gamma^2\sqrt{\rho}}{(1-\sqrt{\rho})}.
\end{aligned}
$$

After minor rearrangement, we have

$$
\begin{aligned}
&\sum_{t=1}^{T} \sum_{i=1}^{N} \mathbb{E}\{\|\mathbf{x}_t^{(i)} - \bar{\mathbf{x}}_t\|^2\} \\
&\leq \frac{NT}{1-D} \Big(\frac{2\gamma^2\sigma^2\rho}{1-\rho} + \frac{6\gamma^2\zeta^2\rho}{(1-\sqrt{\rho})^2} \\
&+ \frac{6\gamma^2\rho}{(1-\sqrt{\rho})^2 T} \sum_{t=1}^{T} \mathbb{E}\{\|\nabla f(\bar{\mathbf{x}}_t)\|^2\} \Big).
\end{aligned}
$$

where $D = \frac{6\gamma^2L^2\rho}{(1-\sqrt{\rho})^2}$. Plugging back to original, we have

$$
\begin{aligned}
&\frac{1}{T} \sum_{T} \mathbb{E}\{\|\nabla f(\bar{\mathbf{x}}_t)\|^2\} \\
&\leq \frac{\mathbb{E}[f(\bar{\mathbf{x}}_T)] - \mathbb{E}[f(\bar{\mathbf{x}}_0)]}{\gamma T} - \frac{1-\gamma L}{T} \sum_{t=1}^{T} \mathbb{E}\{\|\overline{\nabla f}(\bar{\mathbf{x}}_t)\|^2\} \\
&+ \Big(\frac{L^2}{NT} + \frac{2L\kappa}{\gamma N^2 T} + \frac{2(3N+1)L^3\gamma\kappa}{N^2 T} \Big) \frac{NT}{(1-D)} \\
&\Big(\frac{2\gamma^2\sigma^2\rho}{1-\rho} + \frac{6\gamma^2\zeta^2\rho}{(1-\sqrt{\rho})^2} + \frac{6\gamma^2\rho}{(1-\sqrt{\rho})^2 T} \sum_{t=1}^{T} \mathbb{E}\{\|\nabla f(\bar{\mathbf{x}}_t)\|^2\} \Big) \\
&+ \frac{\gamma L}{N}\sigma^2 + \frac{2\gamma L\kappa}{N}\sigma^2 + \frac{6L\kappa\gamma\zeta^2}{N}
\end{aligned}
$$

Therefore,

$$
\begin{aligned}
&\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\{\|\nabla f(\bar{\mathbf{x}}_t)\|^2\} \\
&\leq \Big(\frac{\mathbb{E}[f(\bar{\mathbf{x}}_T)] - \mathbb{E}[f(\bar{\mathbf{x}}_0)]}{\gamma T} + \frac{\gamma L}{N}\sigma^2 + \frac{2\gamma L\kappa}{N}\sigma^2 + \frac{6L\kappa\gamma\zeta^2}{N} \Big) \frac{1-D}{1-2D} \\
&+ \Big(L^2 + \frac{2L\kappa}{\gamma N} + \frac{2(3N+1)L^3\gamma\kappa}{N} \Big) \Big(\frac{2\gamma^2\sigma^2\rho}{1-\rho} + \frac{6\gamma^2\zeta^2\rho}{(1-\sqrt{\rho})^2} \Big) \frac{1}{1-2D}.
\end{aligned}
$$

Recall that we require that $\gamma L \leq \frac{(1-\sqrt{\rho})}{4\sqrt{\rho}}$. Therefore,

$$
D = \frac{6\gamma^2L^2\rho}{(1-\sqrt{\rho})^2} \leq \frac{3}{8}.
$$

Plugging and setting $\gamma = \sqrt{\frac{N}{T}}$, we have

$$
\begin{aligned}
&\frac{1}{T} \sum_{t=1}^{T} \mathbb{E}\{\|\nabla f(\bar{\mathbf{x}}_t)\|^2\} \\
&\leq \frac{8}{\sqrt{NT}} (\mathbb{E}[f(\bar{\mathbf{x}}_T)] - \mathbb{E}[f(\bar{\mathbf{x}}_0)] + L\sigma^2 + 2L\kappa\sigma^2 \\
&+ 6L\kappa\zeta^2 + 2L\kappa \Big(\frac{2\sigma^2\rho}{1-\rho} + \frac{6\zeta^2\rho}{(1-\sqrt{\rho})^2} \Big)) \\
&+ \frac{8N}{T} \Big(L^2 + \frac{2(3N+1)L^3\kappa}{\sqrt{TN}} \Big) \Big(\frac{2\sigma^2\rho}{1-\rho} + \frac{6\zeta^2\rho}{(1-\sqrt{\rho})^2} \Big) \\
&= \mathcal{O}\Big(\frac{1}{\sqrt{NT}} \Big) + \mathcal{O}\Big(\frac{N}{T} \Big).
\end{aligned}
$$

■

E. Proof of Convexity of Weight Optimization Problem in (13)

Proof. Let $\mathcal{S} = \{\mathbf{W}|0 \leq w_{i,j} \leq 1, \mathbf{W} = \mathbf{W}^T, \mathbf{W1} = \mathbf{1}\}$. For $\mathbf{W}_\alpha$ and $\mathbf{W}_\beta$ from $\mathcal{S}$, and letting $\mathbf{W}_\gamma = \eta\mathbf{W}_\alpha + (1-\eta)\mathbf{W}_\beta$, where $0 \leq \eta \leq 1$, it is easy to verify that $\mathbf{W}_\gamma \in \mathcal{S}$. As shown in Lemma 1, $\overline{\mathbf{W}^2} = \mathbb{E}\{\widetilde{\mathbf{W}}^T\widetilde{\mathbf{W}}\}$, where $\widetilde{\mathbf{W}} = \mathbf{W} \odot \mathbf{A} + \mathbf{I} - \text{Diag}(\mathbf{WA})$ and $\mathbf{A}$ represents the successfulness of transmission. For a given $\mathbf{A}$, we have $\widetilde{\mathbf{W}}_\gamma = \eta\widetilde{\mathbf{W}}_\alpha + (1-\eta)\widetilde{\mathbf{W}}_\beta$.

Hence, we can get

$$\begin{aligned}\widetilde{\mathbf{W}}_\gamma^T\widetilde{\mathbf{W}}_\gamma &= \left(\eta\widetilde{\mathbf{W}}_\alpha + (1-\eta)\widetilde{\mathbf{W}}_\beta\right)^T\left(\eta\widetilde{\mathbf{W}}_\alpha + (1-\eta)\widetilde{\mathbf{W}}_\beta\right)\\ &= \eta^2\widetilde{\mathbf{W}_\alpha}^T\widetilde{\mathbf{W}}_\alpha + (1-\eta)^2\widetilde{\mathbf{W}}_\beta^T\widetilde{\mathbf{W}}_\beta\\ &\quad + \eta(1-\eta)\widetilde{\mathbf{W}}_\alpha^T\widetilde{\mathbf{W}}_\beta + \eta(1-\eta)\widetilde{\mathbf{W}}_\beta^T\widetilde{\mathbf{W}}_\alpha.\end{aligned}$$

We also have

$$\begin{aligned}&\widetilde{\mathbf{W}}_\gamma^T\widetilde{\mathbf{W}}_\gamma - \left(\eta\widetilde{\mathbf{W}}_\alpha^T\widetilde{\mathbf{W}}_\alpha + (1-\eta)\widetilde{\mathbf{W}}_\beta^T\widetilde{\mathbf{W}}_\beta\right)\\ &= -\eta(1-\eta)\left(\widetilde{\mathbf{W}}_\alpha - \widetilde{\mathbf{W}}_\beta\right)^T\left(\widetilde{\mathbf{W}}_\alpha - \widetilde{\mathbf{W}}_\beta\right) \preceq \mathbf{0}.\end{aligned}$$

This implies

$$\widetilde{\mathbf{W}}_\gamma^T\widetilde{\mathbf{W}}_\gamma \preceq \eta\widetilde{\mathbf{W}}_\alpha^T\widetilde{\mathbf{W}}_\alpha + (1-\eta)\widetilde{\mathbf{W}}_\beta^T\widetilde{\mathbf{W}}_\beta.$$

Considering the objective, we can get

$$\begin{aligned}\widetilde{\mathbf{W}}_\gamma^T\widetilde{\mathbf{W}}_\gamma - \frac{1}{N}\mathbf{1}\mathbf{1}^\mathbf{T} \preceq&\eta\left(\widetilde{\mathbf{W}}_\alpha^T\widetilde{\mathbf{W}}_\alpha - \frac{1}{N}\mathbf{1}\mathbf{1}^\mathbf{T}\right)\\ &+ (1-\eta)\left(\widetilde{\mathbf{W}}_\beta^T\widetilde{\mathbf{W}}_\beta - \frac{1}{N}\mathbf{1}\mathbf{1}^\mathbf{T}\right)\end{aligned}$$

Taking the expectation of both sides, we have

$$\begin{aligned}\overline{\mathbf{W}_\gamma{}^2} - \frac{1}{N}\mathbf{1}\mathbf{1}^\mathbf{T} \preceq&\eta\left(\overline{\mathbf{W}_\alpha{}^2} - \frac{1}{N}\mathbf{1}\mathbf{1}^\mathbf{T}\right)\\ &+ (1-\eta)\left(\overline{\mathbf{W}_\beta{}^2} - \frac{1}{N}\mathbf{1}\mathbf{1}^\mathbf{T}\right)\end{aligned}$$

Since λ_{max} is a convex function, we have

$$\begin{aligned}\lambda_{max}\{\overline{\mathbf{W}_\gamma{}^2} - \frac{1}{N}\mathbf{1}\mathbf{1}^\mathbf{T}\} \leq&\lambda_{max}(\eta\left(\overline{\mathbf{W}_\alpha{}^2} - \frac{1}{N}\mathbf{1}\mathbf{1}^\mathbf{T}\right)\\ &+ (1-\eta)\left(\overline{\mathbf{W}_\beta{}^2} - \frac{1}{N}\mathbf{1}\mathbf{1}^\mathbf{T}\right))\\ \leq&\eta\lambda_{max}\left(\overline{\mathbf{W}_\alpha{}^2} - \frac{1}{N}\mathbf{1}\mathbf{1}^\mathbf{T}\right)\\ &+ (1-\eta)\lambda_{max}\left(\overline{\mathbf{W}_\beta{}^2} - \frac{1}{N}\mathbf{1}\mathbf{1}^\mathbf{T}\right)\end{aligned}$$

Hence, the objective of the weight optimization problem is convex. ∎

REFERENCES

[1] Z. Qin, G. Y. Li, and H. Ye, "Federated learning and wireless communications," *IEEE Wireless Communications*, vol. 28, no. 5, pp. 134–140, Sept. 2021.

[2] J. N. Tsitsiklis, "Problems in decentralized decision making and computation." Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, Tech. Rep., Dec. 1984.

[3] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Trans. Automat. Control*, vol. 54, no. 11, pp. 2506–2517, Oct. 2009.

[4] K. Yuan, Q. Ling, and W. Yin, "On the convergence of decentralized gradient descent," *SIAM J. Optim.*, vol. 26, no. 3, pp. 1835–1854, Sept. 2016.

[5] E. Wei and A. Ozdaglar, "Distributed alternating direction method of multipliers," in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2012, pp. 5445–5450.

[6] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the admm in decentralized consensus optimization," *IEEE Trans. Signal Process.*, vol. 62, no. 7, pp. 1750–1761, Feb. 2014.

[7] Z. Zheng, L. Song, Z. Han, G. Y. Li, and H. V. Poor, "Game theory for big data processing: Multileader multifollower game-based admm," *IEEE Trans. Signal Process.*, vol. 66, no. 15, pp. 3933–3945, May 2018.

[8] J. C. Duchi, A. Agarwal, and M. J. Wainwright, "Dual averaging for distributed optimization: Convergence analysis and network scaling," *IEEE Trans. Automat. Control*, vol. 57, no. 3, pp. 592–606, Jun. 2011.

[9] 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. Adv. Neural. Inf. Process. Syst. (NeurIPS)*, vol. 30, Dec. 2017, pp. 5330–5340.

[10] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jun. 2018, pp. 3043–3052.

[11] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jun. 2019, pp. 344–353.

[12] J. Wang and G. Joshi, "Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms," *arXiv preprint arXiv:1808.07576*, 2018.

[13] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "Matcha: Speeding up decentralized sgd via matching decomposition sampling," in *Proc. 2019 Sixth Indian Control Conf.*, 2019, pp. 299–300.

[14] H. Tang, S. Gan, C. Zhang, T. Zhang, and J. Liu, "Communication compression for decentralized training," in *Proc. Adv. Neural. Inf. Process. Syst. (NeurIPS)*, Dec. 2018, pp. 7652–7662.

[15] H. Tang, C. Yu, X. Lian, T. Zhang, and J. Liu, "Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jun. 2019, pp. 6155–6165.

[16] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jun. 2019, pp. 3478–3487.

[17] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, "A unified theory of decentralized sgd with changing topology and local updates," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jun. 2020, pp. 5381–5393.

[18] A. Reisizadeh, A. Mokhtari, H. Hassani, and R. Pedarsani, "An exact quantized decentralized gradient descent algorithm," *IEEE Trans. Signal Process.*, vol. 67, no. 19, pp. 4934–4947, Aug. 2019.

[19] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 427–438, 2012.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[21] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[22] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2508–2530, Jun. 2006.