Overlay-based Decentralized Federated Learning in Bandwidth-limited Networks

Yudi Huang*
Pennsylvania State University
State College, PA, USA
yxh5389@psu.edu

Tingyang Sun*
Pennsylvania State University
State College, PA, USA
tfs5679@psu.edu

Ting He
Pennsylvania State University
State College, PA, USA
tinghe@psu.edu

ABSTRACT

The emerging machine learning paradigm of decentralized federated learning (DFL) has the promise of greatly boosting the deployment of artificial intelligence (AI) by directly learning across distributed agents without centralized coordination. Despite significant efforts on improving the communication efficiency of DFL, most existing solutions were based on the simplistic assumption that neighboring agents are physically adjacent in the underlying communication network, which fails to correctly capture the communication cost when learning over a general bandwidth-limited network, as encountered in many edge networks. In this work, we address this gap by leveraging recent advances in network tomography to jointly design the communication demands and the communication schedule for overlay-based DFL in bandwidth-limited networks without requiring explicit cooperation from the underlying network. By carefully analyzing the structure of our problem, we decompose it into a series of optimization problems that can each be solved efficiently, to collectively minimize the total training time. Extensive data-driven simulations show that our solution can significantly accelerate DFL in comparison with state-of-the-art designs.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Networks** → **Overlay and other logical network structures**.

KEYWORDS

Decentralized federated learning, network tomography, overlay routing, mixing matrix design.

ACM Reference Format:
Yudi Huang, Tingyang Sun, and Ting He. 2024. Overlay-based Decentralized Federated Learning in Bandwidth-limited Networks . In *International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '24), October 14–17, 2024, Athens, Greece.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641512.3686364

*Both authors contributed equally to the paper. This work was supported by the National Science Foundation under award CNS-2106294 and CNS-1946022, and has also received funding from The Pennsylvania State University, USA, and the Indian Institute of Science, India.

MobiHoc '24, October 14–17, 2024, Athens, Greece

ACM ISBN 979-8-4007-0521-2/24/10…$15.00
https://doi.org/10.1145/3641512.3686364

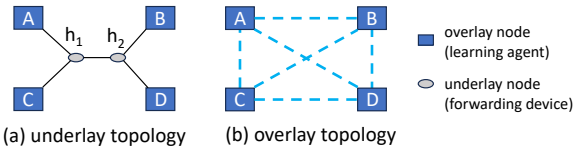


Figure 1: Overlay-based DFL.

1 INTRODUCTION

As a new machine learning paradigm, *decentralized federated learning (DFL)* [20] allows multiple learning agents to collaboratively learn a shared model from the union of their local data without directly sharing the data. To achieve this goal, the agents repeatedly exchange model updates with their neighbors through peer-to-peer connections, which are then aggregated locally [16]. Since its introduction, DFL has attracted significant attention, because compared to centralized *federated learning (FL)* [25], DFL can avoid a single point of failure and reduce the communication complexity at the busiest node without increasing the computational complexity [20].

Meanwhile, FL including DFL faces significant performance challenges due to the extensive data transfer. Although the training data stay local, the agents still need to communicate frequently to exchange local model updates, which often incurs a nontrivial communication cost due to the large model size. Such communication cost can dominate the total cost of the learning task, e.g., accounting for up to 90% of time in cloud-based FL [23], and the problem is exacerbated in other networks that are more bandwidth-limited (e.g., wireless mesh networks [4]). This issue has attracted tremendous interests in reducing the communication cost, including compression-based methods for reducing the amount of data per communication such as [17] and methods for reducing the number of communications through hyperparameter optimization such as [5, 32, 33] or adaptive communications such as [30].

However, most existing works made the simplistic assumption that each pair of logically adjacent agents are also physically adjacent in the underlying communication network. This is not true in *overlay-based DFL*, where the connections between logically adjacent agents (i.e., *overlay links*) can map to arbitrary routing paths in the underlying communication network that may share links (i.e., *underlay links*). For example, a set of learning agents $\{A, B, C, D\}$ may have the physical connectivity in Fig. 1a and the logical connectivity in Fig. 1b. Although connections (A, B) and (C, D) appear disjoint in the overlay, they actually map to paths sharing link (h_1, h_2) in the underlay. Ignoring such link sharing can cause incorrect prediction of the communication time, because concurrent communications over connections with shared links can

take longer than stand-alone communication on each of them, and the problem can be exacerbated by heterogeneous capacities and background loads at underlay links. Most existing works ignored such complications by assuming the communication time to be proportional to the maximum number of neighbors an agent communicates with [5, 9, 19, 32], which generally leads to suboptimal designs in the case of overlay-based DFL.

We want to address this gap *without requiring explicit cooperation from the underlay network*, i.e., the agents can neither directly observe the internal state of the underlay (e.g., routing topology, link capacities) nor control its internal behavior. Such scenarios arise naturally when the agents are interconnected by a public network. In particular, we are interested in running DFL over bandwidth-limited networks, which models many application scenarios such as HetNets [4], device-to-device networks [34], IoT networks [28], underwater networks [27], and power line communication networks [13]. In contrast to high-bandwidth networks such as inter-datacenter networks [24], bandwidth-limited networks are more sensitive to communication demands generated by DFL and are thus in greater needs of proper designs. To this end, we propose an optimization framework for overlay-based DFL that jointly designs the *communication demands* and the *communication schedule* within the overlay, without explicit cooperation from the underlay. Building upon recent advances in network tomography [10] and mixing matrix design [5], we cast the problem into a set of tractable optimizations that collectively minimize the total time in achieving a given level of convergence.

1.1 Related Work

Decentralized federated learning. Initially proposed under a centralized architecture [25], FL was later extended to a fully decentralized architecture [20], which was shown to achieve the same computational complexity but a lower communication complexity. Since then a number of improvements such as [21] have been developed, but these works only focused on reducing the number of iterations.

Communication cost reduction. There are two general approaches for reducing the communication cost in FL: reducing the amount of data per communication through compression, e.g., [17], and reducing the number of communications, e.g., [33]. The two approaches can be combined for further improvement [30]. Instead of either activating all the links or activating none, it has been recognized that better efficiency can be achieved by activating subsets of links. To this end, [30] proposed an event-triggered mechanism and [5, 32] proposed to activate links with predetermined probabilities. In this regard, our work designs predetermined link activation as in [5, 32], which provides more predictable performance than event-triggered mechanisms, but *we consider a cost model tailored to overlay-based DFL*: instead of measuring the communication time by the number of matchings [5, 32] or the maximum degree [9, 19], we evaluate the minimum time to complete all the activated agent-to-agent communications over a bandwidth-limited underlay, while taking into account heterogeneous capacities and possibly shared links.

Topology design in DFL. The logical topology defining the neighborhoods of learning agents is an important design parameter in DFL that controls the communication demands during training. The impact of this topology on the convergence rate of DFL has been mostly captured through the spectral gap of the mixing matrix [15, 20, 26] or equivalent parameters [32]. Although recent works have identified other parameters that can impact the convergence rate, such as the effective number of neighbors [31] and the neighborhood heterogeneity [19], these results just pointed out additional factors and did not invalidate the impact of spectral gap. Based on the identified convergence parameters, several solutions have been proposed to design the logical topology to balance the convergence rate and the cost per communication round [5, 19, 32], and some solutions combined topology design with other optimizations (e.g., model pruning [15]) for further improvement. In this regard, our work also includes topology design based on a parameter related to the spectral gap, but *we explicitly consider the communication schedule to serve the demands triggered by the designed topology over a bandwidth-limited underlay* to optimize the overall wall-clock time of overlay-based DFL. To our knowledge, the only existing work addressing overlay-based DFL is [24]. However, it assumed a special underlay where the paths connecting learning agents only share links at the first and the last hops, whose capacities are assumed to be known. While this model may suit high-bandwidth underlays such as inter-datacenter networks, it fails to capture the communication cost in bandwidth-limited underlays as addressed in our work.

Network-aware distributed computing. It was known that awareness to the state of the communication underlay is important for data-intensive distributed computing tasks [23]. Several works attempted to solve this problem in the context of cloud networks, assuming either a black-box network [23] or a white-box network [3]. In this regard, our work assumes a black-box underlay as in [23], but unlike the simple heuristics used in these works, we leverage state-of-the-art techniques from *network tomography* [10] to estimate the necessary parameters about the underlay.

1.2 Summary of Contributions

We jointly design the communication demands and the communication schedule for overlay-based DFL in a bandwidth-limited uncooperative underlay, with the following contributions:

(1) We consider, for the first time, communication optimization in DFL on top of a bandwidth-limited underlay network with arbitrary topology. To this end, we propose a general framework for jointly designing the communication demands and the communication schedule (e.g., routing, rates) among the learning agents, without cooperation from the underlay.
(2) We tackle the complexity challenge by decomposing the overall problem into a series of smaller subproblems, that are collectively designed to minimize the total training time to achieve a given level of convergence. Through carefully designed relaxations, we convert each subproblem into a tractable optimization to develop efficient solutions.
(3) We evaluate the proposed solution in comparison with benchmarks based on real network topologies and datasets. Our results show that (i) our design of the communication demands can already reduce the training time substantially without compromising the quality of the trained model, (ii) our design of the communication schedule further increases

the improvement, and (iii) the observations remain valid under realistic inference errors about the underlay.

Roadmap. Section 2 describes our overall problem, Section 3 presents our solution and analysis, Section 4 presents our performance evaluation, and Section 5 concludes the paper. **All the proofs can be found in [12, Appendix A].**

2 PROBLEM FORMULATION

2.1 Notations

Let $\boldsymbol{a} \in \mathbb{R}^m$ denote a vector and $\boldsymbol{A} \in \mathbb{R}^{m\times m}$ a matrix. We use $\|\boldsymbol{a}\|$ to denote the ℓ-2 norm, $\|\boldsymbol{A}\|$ to denote the spectral norm, and $\|\boldsymbol{A}\|_F$ to denote the Frobenius norm. We use $\mathrm{diag}(\boldsymbol{a})$ to denote a diagonal matrix with the entries in $\boldsymbol{a}$ on the main diagonal, and $\mathrm{diag}(\boldsymbol{A})$ to denote a vector formed by the diagonal entries of $\boldsymbol{A}$. We use $\lambda_i(\boldsymbol{A})$ $(i = 1, \ldots, m)$ to denote the i-th smallest eigenvalue of $\boldsymbol{A}$.

2.2 Network Model

Consider a network of m learning agents connected through a logical *base topology* $G = (V, E)$ ($|V| = m$), that forms an overlay on top of a communication underlay $\underline{G} = (\underline{V}, \underline{E})$. Unless otherwise stated, both overlay and underlay links are considered directed. Each underlay link $\underline{e} \in \underline{E}$ has a finite capacity $C_{\underline{e}}$. Each overlay link $e = (i, j) \in E$ indicates that agent i is *allowed* to communicate to agent j during learning, and is implemented via a routing path $\underline{p}_{i,j}$ from the node running agent i to the node running agent j in the underlay. We assume that if $(i, j) \in E$, then $(j, i) \in E$ (agents i and j are allowed to exchange results). The routing paths are determined by the topology and the routing protocol in the underlay. Let $l_{i,j}$ denote the propagation delay on $\underline{p}_{i,j}$. We assume that neither the routing paths nor the link capacities in the underlay are observable by the overlay, but the propagation delays between overlay nodes (e.g., $l_{i,j}$) are observable[1].

2.3 Decentralized Federated Learning (DFL)

Consider a DFL task, where each agent $i \in V$ has a possibly non-convex objective function $F_i(\boldsymbol{x})$ that depends on the parameter vector $\boldsymbol{x} \in \mathbb{R}^d$ and the local dataset $\mathcal{D}_i$, and the goal is to find the parameter vector $\boldsymbol{x}$ that minimizes the global objective function $F(\boldsymbol{x})$, defined as

$$F(\boldsymbol{x}) := \frac{1}{m}\sum_{i=1}^{m} F_i(\boldsymbol{x}). \tag{1}$$

For example, we can model the objective of empirical risk minimization by defining the local objective as $F_i(\boldsymbol{x}) := \sum_{\boldsymbol{s}\in\mathcal{D}_i} \ell(\boldsymbol{x}, \boldsymbol{s})$, where $\ell(\boldsymbol{x}, \boldsymbol{s})$ is the loss function for sample $\boldsymbol{s}$ under model $\boldsymbol{x}$, and the corresponding global objective is proportional to the empirical risk over all the samples.

We consider a standard decentralized training algorithm called D-PSGD [20], where each agent repeatedly updates its own parameter vector and aggregates it with the parameter vectors of its neighbors to minimize the global objective function. Specifically, let $\boldsymbol{x}_i^{(k)}$ $(k \geq 1)$ denote the parameter vector at agent i after $k-1$ iterations and $g(\boldsymbol{x}_i^{(k)}; \xi_i^{(k)})$ the stochastic gradient computed by agent i in iteration k (where $\xi_i^{(k)}$ is the mini-batch). In iteration k, agent i updates its parameter vector by

$$\boldsymbol{x}_i^{(k+1)} = \sum_{j=1}^{m} W_{ij}^{(k)} \boldsymbol{x}_j^{(k)} - \eta g(\boldsymbol{x}_i^{(k)}; \xi_i^{(k)}), \tag{2}$$

where $\boldsymbol{W}^{(k)} = (W_{ij}^{(k)})_{i,j=1}^{m}$ is the $m \times m$ *mixing matrix* in iteration k, and $\eta > 0$ is the learning rate. To be consistent with the base topology, $W_{ij}^{(k)} \neq 0$ only if $(i, j) \in E$. The update rule in (2) has the same convergence performance as $\boldsymbol{x}_i^{(k+1)} = \sum_{j=1}^{m} W_{ij}^{(k)} (\boldsymbol{x}_j^{(k)} - \eta g(\boldsymbol{x}_j^{(k)}; \xi_j^{(k)}))$ [20], but (2) allows each agent to parallelize the parameter exchange with neighbors and the gradient computation.

The mixing matrix $\boldsymbol{W}^{(k)}$ plays an important role in controlling the communication cost, as agent j needs to send its parameter vector to agent i in iteration k if and only if $W_{ij}^{(k)} \neq 0$. According to [20], the mixing matrix should be *symmetric with each row/column summing up to one*[2] in order to ensure convergence for D-PSGD. The symmetry implies a one-one correspondence between distinct (possibly) non-zero entries in $\boldsymbol{W}^{(k)}$ and the *undirected overlay links*, denoted by $\widetilde{E}$ (i.e., each $(i, j) \in \widetilde{E}$ represents a pair of directed links $\{(i, j), (j, i)\} \in E$), and thus $W_{ij}^{(k)}$ can be interpreted as the *link weight* of the undirected overlay link $(i, j) \in \widetilde{E}$. The requirement of each row summing to one further implies that $W_{ii}^{(k)} = 1 - \sum_{j=1}^{m} W_{ij}^{(k)}$. In the vector form, the above implies the following decomposition of the mixing matrix

$$\boldsymbol{W}^{(k)} := \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha}^{(k)})\boldsymbol{B}^{\top}, \tag{3}$$

where $\boldsymbol{I}$ is the $m \times m$ identity matrix, $\boldsymbol{B}$ is the $|V| \times |\widetilde{E}|$ incidence matrix[3] for the base topology G, and $\boldsymbol{\alpha}^{(k)} := (\alpha_{ij}^{(k)})_{(i,j)\in\widetilde{E}}$ is the vector of link weights. It is easy to verify that $W_{ij}^{(k)} = \alpha_{ij}^{(k)}$. This decomposition reduces the design of mixing matrix to the design of link weights $\boldsymbol{\alpha}^{(k)}$ in the overlay, where agents i and j need to exchange parameter vectors in iteration k *if and only if* $\alpha_{ij}^{(k)} \neq 0$. Thus, we say that *the (undirected) overlay link* (i, j) *is activated* in iteration k (i.e., both (i, j) and (j, i) are activated) if $\alpha_{ij}^{(k)} \neq 0$.

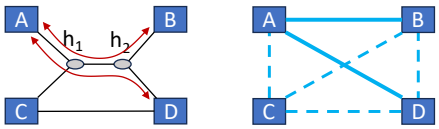
2.4 Design Objective

Our goal is to *jointly design the communication demands between the agents and the communication schedule about how to service these demands* so as to minimize the total (wall-clock) time for the learning task to reach a given level of convergence. The challenges are two-fold: (i) the design of communication demands faces the tradeoff between communicating more per iteration and converging in fewer iterations versus communicating less per iteration and converging in more iterations, and (ii) the design of communication schedule faces the lack of observability and controllability within the underlay network. Below, we will tackle these challenges

[1] This can be obtained by measuring the delays of small probing packets.

[2] In [20], the mixing matrix was assumed to be symmetric and *doubly stochastic* with entries constrained to $[0, 1]$, but we find this requirement unnecessary for the convergence bound we use from [18, Theorem 2], which only requires the mixing matrix to be symmetric with each row/column summing up to one.

[3] This is defined under an arbitrary orientation of each link $\boldsymbol{e}_j \in \widetilde{E}$ as $B_{ij} = +1$ if $\boldsymbol{e}_j$ starts from i, -1 if $\boldsymbol{e}_j$ ends at i, and 0 otherwise.



(a) underlay routes (b) activated overlay links

Figure 2: Underlay-aware communication schedule optimization (learning agents: $\{A, B, C, D\}$; underlay nodes: $\{h_1, h_2\}$).

by combining techniques from network tomography and mixing matrix design.

3 PROPOSED SOLUTION

Our approach is to first characterize the total training time as an explicit function of the set of activated links in the overlay, and then optimize this set. We will focus on a deterministic design that can give a predictable training time, and thus the iteration index k will be omitted. For ease of presentation, we will consider the set of activated overlay links, denoted by $E_a \subseteq \widetilde{E}$, as *undirected links*, as the pair of links between two agents must be activated at the same time.

3.1 Communication Schedule Optimization

Given a set of overlay links $E_a \subseteq \widetilde{E}$ activated in an iteration, each $(i, j) \in E_a$ triggers two communications, one for agent i to send its parameter vector to agent j and the other for agent j to send its parameter vector to agent i. However, directly sending the parameter vectors along the underlay routing paths can lead to suboptimal performance. For example, consider Fig. 2. If $E_a = \{(A, B), (A, D)\}$ but both $\underline{p}_{A,B}$ and $\underline{p}_{A,D}$ traverse the same underlay link (h_1, h_2), directly communicating between the activated agent pairs can take longer than redirecting part of the traffic through other agents (e.g., redirecting $A \to D$ traffic through the overlay path $A \to C \to D$). The same holds if the capacity of the direct path is low, but the capacity through other agents is higher (e.g., if (h_2, D) is a slow link, then redirecting $A \to D$ traffic through C can bypass it to achieve a higher rate). This observation motivates the need of optimizing how to serve the demands triggered by the activated links by routing within the overlay.

3.1.1 Demand Model. Let κ_i denote the size of the parameter vector (or its compressed version if model compression is used) at agent i. A straightforward way to model the communication demands triggered by a set of activated links E_a is to generate two unicast flows for each activated link $(i, j) \in E_a$, one in each direction. However, this model will lead to a suboptimal communication schedule as it ignores the fact that some flows carry identical content. Specifically, all flows originating from the same agent will carry the latest parameter vector at this agent. Thus, the actual communication demands is a set of multicast flows, each for distributing the parameter vector of an activated agent (incident to at least one activated link) to the agents it needs to share parameters with. Let $N_{E_a}(i) := \{j \in V : (i, j) \in E_a\}$. We can express the demands triggered by the activated links E_a as

$$H = \{(i, N_{E_a}(i), \kappa_i) : \forall i \in V \text{ with } N_{E_a}(i) \neq \emptyset\}, \tag{4}$$

where each $h = (s_h, T_h, \kappa_h) \in H$ represents a multicast flow with source s_h, destinations T_h, and data size κ_h.

3.1.2 Baseline Formulation. To help towards minimizing the total training time, the communication schedule should minimize the time for completing all the communication demands triggered by the activated links, within the control of the overlay. To this end, we *jointly optimize the routing and the flow rate within the overlay.* The former is represented by decision variables $z_{ij}^h \in \{0, 1\}$ that indicates whether overlay link (i, j) is traversed by the multicast flow h and $r_{ij}^{h,k} \in \{0, 1\}$ that indicates whether (i, j) is traversed by the flow from s_h to $k \in T_h$, both in the direction of $i \to j$. The latter is represented by decision variables $d_h \geq 0$ that denotes the rate of flow h and $f_{ij}^h \geq 0$ that denotes the rate of flow h on overlay link (i, j) in the direction of $i \to j$. Define constant $b_i^{h,k}$ as 1 if $i = s_h$, -1 if $i = k$, and 0 otherwise. We can formulate the objective of serving all the multicast flows in H (4) within the minimum amount of time as the following optimization:

$$\min_{\boldsymbol{z}, \boldsymbol{r}, \boldsymbol{d}, \boldsymbol{f}} \quad \tau \tag{5a}$$

$$\text{s.t.} \quad \tau \geq \frac{\kappa_h}{d_h} + \sum_{(i,j)\in E} l_{i,j} r_{ij}^{h,k}, \ \forall h \in H, k \in T_h, \tag{5b}$$

$$\sum_{(i,j)\in E: \underline{e} \in \underline{p}_{i,j}} \sum_{h \in H} f_{ij}^h \leq C_{\underline{e}}, \ \forall \underline{e} \in \underline{E}, \tag{5c}$$

$$\sum_{j \in V} r_{ij}^{h,k} = \sum_{j \in V} r_{ji}^{h,k} + b_i^{h,k}, \ \forall h \in H, k \in T_h, i \in V, \tag{5d}$$

$$r_{ij}^{h,k} \leq z_{ij}^h, \ \forall h \in H, k \in T_h, (i, j) \in E, \tag{5e}$$

$$d_h - M(1 - z_{ij}^h) \leq f_{ij}^h \leq d_h, \ \forall h \in H, (i, j) \in E, \tag{5f}$$

$$f_{ij}^h \leq M z_{ij}^h, \ \forall h \in H, (i, j) \in E, \tag{5g}$$

$$r_{ij}^{h,k}, z_{ij}^h \in \{0, 1\}, \ d_h \in [0, M], \ f_{ij}^h \geq 0, \quad \forall h \in H, k \in T_h, (i, j) \in E, \tag{5h}$$

where M is an upper bound on d_h ($\forall h \in H$). Constraint (5b) makes τ an upper bound on the completion time of the slowest flow; (5c) ensures that the total traffic rate imposed by the overlay on any underlay link is within its capacity; (5d)–(5e) are the *Steiner arborescence* constraints [7] that guarantee the set of overlay links with $z_{ij}^h = 1$ will form a Steiner arborescence (i.e., a directed Steiner tree) that is the union of paths from s_h to each $k \in T_h$ (where each path is formed by the links with $r_{ij}^{h,k} = 1$); (5f) implies that $f_{ij}^h = d_h$ if $z_{ij}^h = 1$ and (5g) together with (5h) implies that $f_{ij}^h = 0$ if $z_{ij}^h = 0$, which allows the capacity constraint to be formulated as a linear inequality (5c) instead of a bilinear inequality $\sum_{(i,j)\in E: \underline{e} \in \underline{p}_{i,j}} \sum_{h \in H} d_h z_{ij}^h \leq C_{\underline{e}}$. The optimal solution $(\boldsymbol{z}^*, \boldsymbol{r}^*, \boldsymbol{d}^*, \boldsymbol{f}^*)$ to (5) provides an overlay communication schedule that minimizes the communication time in a given iteration when the set of activated links is E_a.

Complexity: As $|H| \leq |V|$, the optimization (5) contains $O(|V|^2|E|)$ variables (dominated by $\boldsymbol{r}$), and $O(|\underline{E}| + |V|^2(|V| + |E|))$ constraints. Since constraints (5c)–(5f) are linear and constraint (5b) is convex, the optimization (5) is a mixed integer convex programming (MICP) problem and thus can be solved by existing MICP solvers such as Pajarito [22] at a super-polynomial complexity or approximate MICP algorithms such as convex relaxation plus randomized rounding at a polynomial complexity.

3.1.3 Handling Uncooperative Underlay. When learning over an uncooperative underlay as considered in this work, the overlay cannot directly solve (5), because the capacity constraint (5c) requires the knowledge of the routing in the underlay and the capacities of the underlay links. In absence of such knowledge, we leverage a recent result from [10] to convert this constraint into an equivalent form that can be consistently estimated by the overlay. To this end, we introduce the following notion from [10], adapted to our problem setting.

Definition 1 ([10]). A **category of underlay links** Γ_F for a set of overlay links F ($F \subseteq E$) is the set of underlay links traversed *by and only by* the underlay routing paths for the overlay links in F out of all the paths for E, i.e,[4]

$$\Gamma_F := \Big(\bigcap_{(i,j)\in F} \underline{p}_{i,j} \Big) \setminus \Big(\bigcup_{(i,j)\in E\setminus F} \underline{p}_{i,j} \Big). \tag{6}$$

The key observation is that since all the underlay links in the same category are traversed by the same set of overlay links, they must carry the same traffic load from the overlay. Therefore, we can reduce the per-link capacity constraint (5c) into the following *per-category capacity constraint*:

$$\sum_{(i,j)\in F} \sum_{h\in H} f_{ij}^h \le C_F, \ \forall F \subseteq E \text{ with } \Gamma_F \neq \emptyset, \tag{7}$$

where $C_F := \min_{\underline{e}\in\Gamma_F} C_{\underline{e}}$, referred to as the *category capacity*, is the minimum capacity of all the links in category Γ_F. The new constraint (7) is equivalent to the original constraint (5c), as an overlay communication schedule satisfies one of these constraints if and only if it satisfies the other. However, instead of requiring detailed internal information about the underlay (i.e., $(\underline{p}_{i,j})_{(i,j)\in E}$ and $(C_{\underline{e}})_{\underline{e}\in\underline{E}}$), constraint (7) only requires the knowledge of the *nonempty categories* and the corresponding *category capacities*.

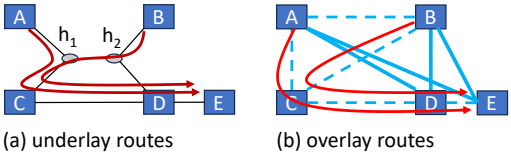
Under the assumption that every underlay link introduces a nontrivial performance impact (e.g., non-zero loss/queueing probability), [10] provided an algorithm that can consistently infer the nonempty categories from losses/delays of packets sent concurrently through the overlay links, under the assumption that concurrently sent packets will experience the same performance when traversing a shared underlay link. Moreover, by leveraging state-of-the-art single-path residual capacity estimation methods, [10] gave a simple algorithm that can accurately estimate the *effective category capacity* $\widetilde{C}_F$ for each detected nonempty category, that can be used in place of C_F in (7) without changing the feasible region. Given the indices of inferred nonempty categories $\widehat{\mathcal{F}}$ and their inferred effective capacities $(\widehat{C}_F)_{F\in\widehat{\mathcal{F}}}$, we can construct the per-category capacity constraint as

$$\sum_{(i,j)\in F} \sum_{h\in H} f_{ij}^h \le \widehat{C}_F, \ \forall F \in \widehat{\mathcal{F}}, \tag{8}$$

which can then be used in place of (5c) in (5) to compute an optimized overlay communication schedule.

Remark: First, the traversal of overlay paths through the overlay links is directional, and the traversal of underlay routing paths through the underlay links is also directional. Correspondingly, the overlay links in a category index F should be treated as directed links (i.e., $(i,j) \in F$ only implies that the underlay links in Γ_F are traversed by the path $\underline{p}_{i,j}$). This is not to be confused with treating the activated links in E_a as undirected links, because each $(i,j) \in E_a$ stands for a parameter exchange between agents i and j. Moreover, we only use the activated links E_a to determine the flow demands H, but any overlay link in E can be used in serving these flows.

[4]Here $\underline{p}$ is interpreted as the set of underlay links traversed by path $\underline{p}$.



(a) underlay routes (b) overlay routes

Figure 3: Challenge for in-overlay aggregation (learning agents: $\{A, B, C, D, E\}$; underlay nodes: $\{h_1, h_2\}$).

3.1.4 Additional Optimization Opportunities and Challenges. The formulation (5) treats each overlay node that is neither the source nor one of the destinations of a multicast flow as a pure relay, but this node is actually a learning agent capable of aggregating the parameter vectors. This observation raises two questions: (i) Can an agent include parameter vectors relayed through it in its own parameter aggregation? (ii) If an agent relays multiple parameter vectors for different sources, can it forward the aggregated vector instead of the individual vectors?

To answer the first question, consider the case in Fig. 2 when A sends its parameter vector $\boldsymbol{x}_A$ to D through the overlay path $A \to C \to D$. If C includes $\boldsymbol{x}_A$ in its own parameter aggregation with a non-zero weight W_{CA}, then by the symmetry of the mixing matrix, A must also include $\boldsymbol{x}_C$ in its parameter aggregation with weight $W_{AC} = W_{CA}$, which is equivalent to activating the overlay link (A, C). As we have left the optimization of the activated links E_a to another subproblem (Section 3.3), there is no need to include relayed parameter vectors in parameter aggregation when optimizing the communication schedule for a given set of activated links.

To answer the second question, consider the case in Fig. 3 when the overlay routes the multicast from A to $\{D, E\}$ (for disseminating $\boldsymbol{x}_A$) over $A \to C \to D \to E$, and the multicast from B to $\{D, E\}$ (for disseminating $\boldsymbol{x}_B$) over $B \to C \to D \to E$. Although instead of separately forwarding $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$, C could aggregate them before forwarding, the aggregation will not save bandwidth for C, as D needs $W_{DA}\boldsymbol{x}_A + W_{DB}\boldsymbol{x}_B$ but E needs $W_{EA}\boldsymbol{x}_A + W_{EB}\boldsymbol{x}_B$, which are generally not the same. Another issue with in-network aggregation (within the overlay) is the synchronization delay introduced at the point of aggregation, and thus in-network aggregation may not reduce the completion time even when it can save bandwidth, e.g., at D. We thus choose not to consider in-network aggregation in our formulation (5). Further optimizations exploiting such capabilities are left to future work.

3.2 Link Weight Optimization

Given the set of activated links $E_a \subseteq \widetilde{E}$, the communication time per iteration has been determined as explained in Section 3.1, but the number of iterations has not, and is heavily affected by the weights of the activated links. This leads to the question of how to minimize the number of iterations for achieving a desired level of

convergence, under the constraint that only the activated links can have non-zero weights.

To answer this question, we leverage a state-of-the-art convergence bound for D-PSGD under the following assumptions:

(1) Each local objective function $F_i(\boldsymbol{x})$ is l-Lipschitz smooth, i.e., $\|\nabla F_i(\boldsymbol{x}) - \nabla F_i(\boldsymbol{x}')\| \leq l\|\boldsymbol{x} - \boldsymbol{x}'\|$, $\forall i \in V$.
(2) There exist constants $M_1, \widehat{\sigma}$ such that $\frac{1}{m}\sum_{i\in V} \mathrm{E}[\|g(\boldsymbol{x}_i;\xi_i) - \nabla F_i(\boldsymbol{x}_i)\|^2] \leq \widehat{\sigma}^2 + \frac{M_1}{m}\sum_{i\in V}\|\nabla F(\boldsymbol{x}_i)\|^2, \forall \boldsymbol{x}_1,\ldots,\boldsymbol{x}_m \in \mathbb{R}^d$.
(3) There exist constants $M_2, \widehat{\zeta}$ such that $\frac{1}{m}\sum_{i\in V}\|\nabla F_i(\boldsymbol{x})\|^2 \leq \widehat{\zeta}^2 + M_2\|\nabla F(\boldsymbol{x})\|^2, \forall \boldsymbol{x} \in \mathbb{R}^d$.

Let $\boldsymbol{J} := \frac{1}{m}\boldsymbol{1}\boldsymbol{1}^\top$ denote an ideal $m\times m$ mixing matrix with all entries being $\frac{1}{m}$.

Theorem 3.1. [18, Theorem 2] Under assumptions (1)–(3), if there exist constants $p \in (0,1]$ and integer $t \geq 1$ such that the mixing matrices $\{\boldsymbol{W}^{(k)}\}_{k=1}^K$, each being symmetric with each row/column summing to one[5], satisfy

$$\mathrm{E}[\|\boldsymbol{X}\prod_{k=k't+1}^{(k'+1)t}\boldsymbol{W}^{(k)} - \boldsymbol{X}\boldsymbol{J}\|_F^2] \leq (1-p)\|\boldsymbol{X}-\boldsymbol{X}\boldsymbol{J}\|_F^2 \tag{9}$$

for all $\boldsymbol{X} := [\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m]$ and integer $k' \geq 0$, then D-PSGD can achieve $\frac{1}{K}\sum_{k=1}^K \mathrm{E}[\|\nabla F(\overline{\boldsymbol{x}}^k)\|^2] \leq \epsilon_0$ for any given $\epsilon_0 > 0$ ($\overline{\boldsymbol{x}}^{(k)} := \frac{1}{m}\sum_{i=1}^m \boldsymbol{x}_i^{(k)}$) when the number of iterations reaches

$$K(p,t) := l(F(\overline{\boldsymbol{x}}^{(1)}) - F_{\inf}) \cdot O\left(\frac{\widehat{\sigma}^2}{m\epsilon_0^2} + \frac{\widehat{\zeta}t\sqrt{M_1+1} + \widehat{\sigma}\sqrt{pt}}{p\epsilon_0^{3/2}} + \frac{t\sqrt{(M_2+1)(M_1+1)}}{p\epsilon_0}\right), \tag{10}$$

where $\overline{\boldsymbol{x}}^{(1)}$ is the initial parameter vector, and $F_{\inf}$ is a lower bound on $F(\cdot)$.

For tractability, we will focus on the case of i.i.d. mixing matrices. In this case, to achieve ϵ_0-convergence, it suffices for the number of iterations to reach $K(p,t)$ as in (10) for $t = 1$. We note that $K(p,1)$ depends on the mixing matrix only through the parameter p: the larger p, the smaller $K(p,1)$. Recall that as explained in Section 2.3, the mixing matrix $\boldsymbol{W}$ is related to the link weights $\boldsymbol{\alpha}$ as $\boldsymbol{W} = \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top$. To restrict the activated links to E_a, we set $\alpha_{ij} = 0$ for all $(i,j) \notin E_a$. Below, we will show that the following optimization gives a good design of the link weights:

$$\min_{\boldsymbol{\alpha}} \ \rho \tag{11a}$$

$$\text{s.t. } -\rho\boldsymbol{I} \preceq \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J} \preceq \rho\boldsymbol{I}, \tag{11b}$$

$$\alpha_{ij} = 0, \ \forall (i,j) \notin E_a. \tag{11c}$$

Corollary 3.2. Under assumptions (1)–(3) and i.i.d. mixing matrices $\boldsymbol{W}^{(k)} \overset{d}{=} \boldsymbol{W}$ that is symmetric with each row/column summing to one, D-PSGD achieves ϵ_0-convergence as in Theorem 3.1 when the number of iterations reaches

$$K(1 - \mathrm{E}[\|\boldsymbol{W} - \boldsymbol{J}\|^2], 1). \tag{12}$$

Moreover, conditioned on the set of activated links being E_a, (12) $\geq K(1-\rho^{*2},1)$, where ρ^* is the optimal value of (11), with "=" achieved at $\boldsymbol{W}^* = \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha}^*)\boldsymbol{B}^\top$ for $\boldsymbol{\alpha}^*$ being the optimal solution to (11).

Corollary 3.2 implies that given the set of activated links, we can design the corresponding link weights by solving (11), which will minimize an upper bound (12) on the number of iterations to achieve ϵ_0-convergence. Optimization (11) is a semi-definite programming (SDP) problem that can be solved in polynomial time by existing algorithms [14].

Remark: When $\boldsymbol{W}$ satisfies the additional property of $\boldsymbol{I} \succeq \boldsymbol{W} \succeq -\boldsymbol{I}$, the largest singular value of $\boldsymbol{W}$ is 1 [9], and thus $\|\boldsymbol{W} - \boldsymbol{J}\|$ is the second largest singular value of $\boldsymbol{W}$. In this case, minimizing ρ in (11) (which equals $\|\boldsymbol{W} - \boldsymbol{J}\|$ under the optimal solution) is equivalent to maximizing $\gamma(\boldsymbol{W}) := 1 - \|\boldsymbol{W} - \boldsymbol{J}\|$, which is the *spectral gap* of the mixing matrix $\boldsymbol{W}$ [26]. The spectral gap is the most widely-used parameter to capture the impact of the mixing matrix on the convergence rate [15, 20, 26]. In this sense, our Corollary 3.2 extends the relationship between the spectral gap and the number of iterations to the case of random mixing matrices. As $\gamma(\boldsymbol{W}) \to 0$ (in probability), the number of iterations according to (12) grows at

$$K\left(1 - \mathrm{E}[(1-\gamma(\boldsymbol{W}))^2], 1\right) = O\left(\frac{1}{\mathrm{E}[\gamma(\boldsymbol{W})]}\right), \tag{13}$$

which is consistent with the existing result of $O(1/\gamma(\boldsymbol{W}))$ in the case of deterministic mixing matrix [26]. While other parameters affecting the convergence rate have been identified, e.g., the effective number of neighbors [31] and the neighborhood heterogeneity [19], these parameters are just additional factors instead of replacements of the spectral gap. We thus leave the optimization of these other objectives to future work.

3.3 Link Activation Optimization

Given how to optimize the communication schedule and the link weights for a given set E_a of activated links as in Sections 3.1–3.2, what remains is to optimize E_a itself, which is also known as "topology design" [24] as (V, E_a) depicts a subgraph of the base topology (of the overlay) that is used to determine which agents will exchange parameter vectors during DFL. The set E_a affects both the communication demands (and hence the time per iteration) and the sparsity pattern of the mixing matrix (and hence the number of iterations required). As mentioned in Section 2.4, our goal is to minimize the total training time. For learning over bandwidth-limited networks, the training time is dominated by the communication time [23]. We thus model the total training time by

$$\tau(E_a) \cdot K(E_a), \tag{14}$$

where we have used $\tau(E_a)$ to denote the communication time per iteration according to (5) (with (5c) replaced by (8)), and $K(E_a) := K(1-\rho^{*2},1)$ to denote the number of iterations to achieve a given level of convergence. Our goal is to minimize (14) over all the candidate values of $E_a \subseteq \widetilde{E}$.

Directly solving this optimization is intractable because its solution space is exponentially large and its objective function (14) is not given explicitly. Our approach to address this challenge is to: (i) relax $\tau(E_a)$ and $K(E_a)$ into upper bounds that are explicit functions of E_a, (ii) decompose the optimization to separate the impacts of $\tau(E_a)$ and $K(E_a)$, and (iii) develop efficient solutions by identifying linkages to known problems.

[5] Originally, [18, Theorem 2] had a stronger assumption that each mixing matrix is doubly stochastic, but we have verified that it suffices to have each row/column summing to one.

3.3.1 Relaxed Objective Function. We first upper-bound $\tau(E_a)$ by considering a suboptimal but analyzable communication schedule. Consider a solution to (5) with $z_{ij}^h = 1$ if $i = s_h, j \in T_h$ and 0 otherwise, and $r_{ij}^{h,k} = 1$ if $i = s_h, j = k$ and 0 otherwise, i.e., each parameter exchange corresponding to $(i,j) \in E_a$ is performed directly along the underlay routing paths $\underline{p}_{i,j}$ and $\underline{p}_{j,i}$. To achieve a per-iteration communication time of $\overline{\tau}$, the rate $d_{i,j}$ of sending the parameter vector of agent i to its activated neighbor j must satisfy $d_{i,j} \geq \frac{\kappa_i}{\overline{\tau} - l_{i,j}}$. This is feasible for (5) (with (5c) replaced by (8)) if and only if

$$\sum_{(i,j)\in E_a} \left(\frac{\kappa_i}{\overline{\tau} - l_{i,j}} \mathbb{1}_{(i,j)\in F} + \frac{\kappa_j}{\overline{\tau} - l_{j,i}} \mathbb{1}_{(j,i)\in F} \right) \leq \widehat{C}_F, \ \forall F \in \widehat{\mathcal{F}}, \quad (15)$$

where $\mathbb{1}_\cdot$ denotes the indicator function. The minimum value of $\overline{\tau}$ satisfying (15), denoted by $\overline{\tau}(E_a)$, thus provides an upper bound on the minimum per-iteration time $\tau(E_a)$ under the set of activated links in E_a.

We then upper-bound $K(E_a)$ by upper-bounding the optimal value ρ^* of (11). Consider any predetermined link weight assignment $\boldsymbol{\alpha}^{(0)}$, and let $\boldsymbol{\alpha}^{(0)}(E_a)$ be the corresponding feasible solution to (11) (i.e., $(\alpha^{(0)}(E_a))_{ij} = \alpha_{ij}^{(0)}$ if $(i,j) \in E_a$ and 0 otherwise). Let $\boldsymbol{L}(E_a) := \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha}^{(0)}(E_a))\boldsymbol{B}^\top$ denote the Laplacian matrix for the activated graph. Under this solution, the objective value of (11) is

$$\overline{\rho} := \|\boldsymbol{I} - \boldsymbol{L}(E_a) - \boldsymbol{J}\| \quad (16)$$

$$= \max(1 - \lambda_2(\boldsymbol{L}(E_a)),\ \lambda_m(\boldsymbol{L}(E_a)) - 1), \quad (17)$$

where (16) is from the proof of Corollary 3.2, and (17) is by [5, Lemma IV.2] (where $\lambda_i(\boldsymbol{L}(E_a))$ denotes the i-th smallest eigenvalue of $\boldsymbol{L}(E_a)$). Since $K(1-\rho^2, 1)$ is an increasing function of ρ and $\rho^* \leq \overline{\rho}$, we have

$$K(E_a) := K\left(1 - \rho^{*2}, 1\right) \leq K\left(1 - \overline{\rho}^2, 1\right) =: \overline{K}(E_a). \quad (18)$$

3.3.2 Bi-level Decomposition. Relaxing (14) into its upper bound $\overline{\tau}(E_a) \cdot \overline{K}(E_a)$ provides an objective function that can be easily evaluated for any candidate E_a. However, we still face the exponentially large solution space of $E_a \subseteq \widetilde{E}$. To address this complexity challenge, we decompose the relaxed optimization into a bi-level optimization as follows.

Lemma 3.3. Let β be the maximum time per iteration. Then

$$\min_{E_a \subseteq \widetilde{E}} \overline{\tau}(E_a) \cdot \overline{K}(E_a) = \min_{\beta \geq 0} \beta \cdot \left(\min_{\overline{\tau}(E_a) \leq \beta} \overline{K}(E_a) \right), \quad (19)$$

and the optimal solution E_a^* to the RHS of (19) is also optimal for the LHS of (19).

The bi-level decomposition in (19) allows us to focus on the lower-level optimization

$$\min_{\overline{\tau}(E_a) \leq \beta} \overline{K}(E_a), \quad (20)$$

as the upper-level optimization only has a scalar variable β that can be optimized numerically once we have an efficient solution to (20).

Algorithm 1: Topology Design via SCA

input : Initial link weights $\boldsymbol{\alpha}^{(0)}$, candidate links $\widetilde{E}$, threshold ϵ.
output : Set of activated links E_a.

initialize $E_s \leftarrow \emptyset$ and $E_o \leftarrow \emptyset$;
while *True* **do**
  obtain $\boldsymbol{y}^*$ by solving the SDP relaxation of (22) with additional constraints that $y_{\tilde{e}} = 0, \forall \tilde{e} \in E_o$ and $y_{\tilde{e}} = 1, \forall \tilde{e} \in E_s$;
  if (22c) *is satisfied by* $\boldsymbol{y}$ *such that* $y_{\tilde{e}} = 1$ *iff* $y^*_{\tilde{e}} \geq \epsilon$ **then**
    Break;
  else
    Find $\tilde{e}_s = \arg\max\{ y^*_{\tilde{e}} : \tilde{e} \in \widetilde{E} \setminus (E_s \cup E_o)$, (22c) is satisfied by $\boldsymbol{y}$ corresponding to $E_s \cup \{\tilde{e}\}\}$;
    $E_s \leftarrow E_s \cup \{\tilde{e}_s\}$;
    Find $\tilde{e}_o = \arg\min\{ y^*_{\tilde{e}} : \tilde{e} \in \widetilde{E} \setminus (E_s \cup E_o)\}$;
    $E_o \leftarrow E_o \cup \{\tilde{e}_o\}$;
return $E_a \leftarrow \{\tilde{e} \in \widetilde{E} :\ y^*_{\tilde{e}} \geq \epsilon\}$;

3.3.3 Algorithms. To solve (20), we encode E_a by binary variables $\boldsymbol{y} := (y_{ij})_{(i,j)\in\widetilde{E}}$, where $y_{ij} = 1$ if $(i,j) \in E_a$ and 0 otherwise. By (15), $\boldsymbol{y}$ is feasible for (20) if and only if $\forall F \in \widehat{\mathcal{F}}$,

$$\sum_{(i,j)\in\widetilde{E}} y_{ij} \left(\frac{\kappa_i}{\beta - l_{i,j}} \mathbb{1}_{(i,j)\in F} + \frac{\kappa_j}{\beta - l_{j,i}} \mathbb{1}_{(j,i)\in F} \right) \leq \widehat{C}_F. \quad (21)$$

Exact solution: We can directly try to solve the lower-level optimization (20) as a convex programming problem with integer variables. Due to the monotone relationship between $\overline{K}(E_a)$ and $\overline{\rho}$, we can equivalently minimize $\overline{\rho}$ as defined in (16) by solving

$$\min_{\boldsymbol{y}} \quad \rho \quad (22a)$$

$$\text{s.t. } -\rho\boldsymbol{I} \preceq \boldsymbol{I} - \sum_{(i,j)\in\widetilde{E}} y_{ij}\boldsymbol{L}_{ij} - \boldsymbol{J} \preceq \rho\boldsymbol{I}, \quad (22b)$$

$$(21),\ \forall F \in \widehat{\mathcal{F}}, \quad (22c)$$

$$y_{ij} \in \{0,1\},\ \forall (i,j) \in \widetilde{E}, \quad (22d)$$

where $\boldsymbol{L}_{ij}$ is the Laplacian matrix representation of link (i,j) with weight $\alpha_{ij}^{(0)}$, i.e., entries (i,j) and (j,i) are $-\alpha_{ij}^{(0)}$ and entries (i,i) and (j,j) are $\alpha_{ij}^{(0)}$ (rest are zero), and (22b) ensures that the auxiliary variable $\rho = \overline{\rho}$ as in (16) under the optimal solution.

The optimization (22) is an integer convex programming (ICP) problem, and thus in theory can be solved by ICP solvers such as [22]. In practice, however, ICP solvers can be slow due to their super-polynomial complexity, and in contrast to the communication schedule optimization (5) that only needs to be solved once, (22) has to be solved many times to optimize the variable β in the upper-level optimization. Thus, we need more efficient algorithms.

Efficient heuristics: To improve the computational efficiency, we have explored two approaches: (i) developing heuristics for (22), and (ii) further simplifying the objective function.

• *Heuristics for* (22)*:* Once we relax the integer constraint (22d) into $y_{ij} \in [0,1]$, (22) becomes an SDP that can be solved in polynomial time [14]. We can thus round the fractional solution into a feasible solution. However, we observe that simple rounding schemes such as greedily activating links with the largest fractional y-values do not yield a good solution (see results for 'Relaxation-ρ'

in Section 4). Thus, we propose an iterative rounding algorithm based on successive convex approximation (SCA) as in Algorithm 1. The algorithm gradually rounds the y-values for a subset of links E_s to 1 and those for another subset of links E_o to 0 to satisfy feasibility. In each iteration, it solves the SDP relaxation of (22) with rounding constraints according to the previously computed E_s and E_o (line 3), and then adds the link with the largest fractional y-value to E_s (lines 7–8) and the link with the smallest fractional y-value to E_o (lines 9–10). The iteration repeats until the integer solution rounded according to a given threshold ϵ is feasible for (22) (line 4).

• *Heuristics based on algebraic connectivity:* Another approach is to simplify the objective based on the following observation.

Lemma 3.4. Minimizing $\overline{\rho}$ in (17) is equivalent to maximizing $\lambda_2(\boldsymbol{L}(E_a))$ (the second smallest eigenvalue) if $\boldsymbol{\alpha}^{(0)}$ satisfies

$$\max_{i\in V} \sum_{j:(i,j)\in\widetilde{E}} \alpha_{ij}^{(0)} + m \cdot \max_{(i,j)\in\widetilde{E}} |\alpha_{ij}^{(0)}| \le 1. \tag{23}$$

Remark: We can satisfy (23) by making $\alpha_{ij}^{(0)}$'s sufficiently small, e.g., $\alpha_{ij}^{(0)} \equiv 1/(2m-1), \forall (i,j) \in \widetilde{E}$.

Under condition (23), we can convert the minimization of $\overline{\rho}$ into a maximization of $\lambda_2(\boldsymbol{L}(E_a))$, known as the *algebraic connectivity*:

$$\max_{\boldsymbol{y}} \quad \lambda_2\Big(\sum_{(i,j)\in\widetilde{E}} y_{ij}\boldsymbol{L}_{ij}\Big) \tag{24a}$$

$$\text{s.t. (21)}, \ \forall F \in \widehat{\mathcal{F}}, \tag{24b}$$

$$y_{ij} \in \{0,1\}, \ \forall (i,j) \in \widetilde{E}, \tag{24c}$$

which selects links to maximize the algebraic connectivity under the linear constraints (24b). A similar problem of maximizing the algebraic connectivity of *unweighted* graphs under *cardinality constraint* (i.e., $\sum_{(i,j)\in\widetilde{E}} y_{ij} \le k$) has been studied with some efficient heuristics [6, 8]. Although our problem (24) addresses a weighted graph and more general linear constraints, the existing heuristics can still be adapted for our problem.

Specifically, as $\lambda_2(\sum_{(i,j)\in\widetilde{E}} y_{ij}\boldsymbol{L}_{ij})$ is a concave function of $\boldsymbol{y}$ [6], relaxing the integer constraint (24c) into $y_{ij} \in [0,1]$ turns (24) into a convex optimization that can be solved in polynomial time, based on which we can extract an integer solution via rounding. We can also adapt the greedy perturbation heuristic in [6] as follows. Let $\boldsymbol{v}(E_a)$ denote the *Fiedler vector* of $\boldsymbol{L}(E_a)$ (i.e., the unit-norm eigenvector corresponding to $\lambda_2(\boldsymbol{L}(E_a))$). It is easy to extend [6, (10)] into

$$\lambda_2(\boldsymbol{L}(E_a \cup \{(i,j)\})) - \lambda_2(\boldsymbol{L}(E_a)) \le \alpha_{ij}^{(0)} \left(v(E_a)_i - v(E_a)_j\right)^2. \tag{25}$$

Based on this bound, we can apply the greedy heuristic to (24) by repeatedly: (i) computing the Fiedler vector $\boldsymbol{v}(E_a)$ based on the current E_a, and (ii) augmenting E_a with the link $(i,j) \in \widetilde{E} \setminus E_a$ that maximizes $\alpha_{ij}^{(0)} \left(v(E_a)_i - v(E_a)_j\right)^2$ subject to (21).

Remark: Even if we can enforce the condition in Lemma 3.4 by suitably setting the initial link weights $\boldsymbol{\alpha}^{(0)}$, the final link weights are determined by the optimization (11) and thus cannot guarantee the equivalence between $\overline{\rho}$ minimization and algebraic connectivity maximization. As a result, our evaluation shows that E_a's designed by the above heuristics based on the algebraic connectivity are less effective than those designed based on $\overline{\rho}$ (see Section 4). However, connecting our problem with algebraic connectivity maximization opens the possibility of leveraging a rich set of existing results, for which the above is just a starting point. In this regard, our contribution is to formally establish this connection and the corresponding condition.

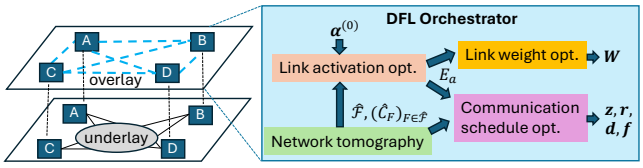


Figure 4: Workflow of overall solution.

3.4 Overall Solution

Fig. 4 illustrates the overall proposed solution as deployed on a centralized orchestrator when initializing DFL tasks, which starts by inferring the necessary information about the underlay using network tomography [10], and then selects the links to activate based on predetermined weights $\boldsymbol{\alpha}^{(0)}$ as in Section 3.3, based on which the link weights are optimized as in Section 3.2 and the communication schedule is optimized as in Section 3.1. Our solution is suitable for the centralized deployment as it only uses predetermined information.

The performance of this solution is guaranteed as follows.

Theorem 3.5. Under the assumption of $\widehat{\mathcal{F}} \supseteq \mathcal{F}$ and $\widehat{C}_F \le C_F$ $(\forall F \in \mathcal{F})$, if the proposed solution activates a set of links E_a^*, then D-PSGD under the corresponding design is guaranteed to achieve ϵ_0-convergence as defined in Theorem 3.1 within time $\overline{\tau}(E_a^*)\cdot\overline{K}(E_a^*)$ for $\overline{\tau}(\cdot)$ and $\overline{K}(\cdot)$ defined in Section 3.3.1.

Remark: Theorem 3.5 upper-bounds the total training time under our design if network tomography detects all the nonempty categories and does not overestimate the category capacities. According to [10], the second assumption holds approximately as the estimation of category capacities is highly accurate, but the first assumption may not hold due to misses in detecting nonempty categories. However, in underlays following symmetric tree-based routing as commonly encountered in edge networks, a new algorithm in [11] can detect nearly all the nonempty categories in networks of moderate sizes. We will empirically evaluate the impact of inference errors on the final performance of DFL in Section 4.

4 PERFORMANCE EVALUATION

We evaluate the proposed algorithms against benchmarks through realistic data-driven simulations in the context of bandwidth-limited wireless edge networks.

4.1 Simulation Setup

4.1.1 Dataset and ML Model. We train a ResNet-50 model with 23,616,394 parameters and a model size of 90.09 MB for image classification on the CIFAR-10 dataset, which consists of 60,000 color images divided into 10 classes. We use 50,000 images for training and the remaining 10,000 images for testing. The dataset undergoes standard preprocessing, including normalization and one-hot encoding of the labels. We set the learning rate to 0.02 and the mini-batch size to 64 for each agent. These settings are

sufficient for D-PSGD to achieve convergence under all evaluated designs. As a sanity check, we also train a 4-layer CNN model based on [25], which has 582,026 parameters and a model size of 2.22 MB, for digit recognition on the MNIST dataset, which comprises 60,000 training images and 10,000 testing images. In both cases, we evenly divide the training data among all the agents after a random shuffle.

4.1.2 Network Topology. We simulate the underlay based on the topologies and link attributes of real wireless edge networks. We consider two important types of networks: (i) WiFi-based wireless mesh networks represented by the Roofnet [2], which has 33 nodes, 187 links, and a data rate of 1 Mbps, and (ii) millimeter-wave-based Integrated Access and Backhaul (IAB) networks used to extend the coverage of 5G networks [1], represented by a hexagon topology with 19 nodes, 56 links, and a data rate of 0.4 Gbps [29]. In each topology, we select 10 low-degree nodes as learning agents (i.e., overlay nodes). We assume the base topology to be a clique among the agents (i.e., any two agents are allowed to communicate), and use the shortest paths (based on hop count) between the agents as the underlay routing paths. See [12] for the simulated topologies.

4.1.3 Benchmarks. We compare the proposed Algorithm 1 ('SCA') against the following benchmarks:

- the baseline of activating all the (overlay) links ('Clique');
- the ring topology ('Ring') commonly adopted by industry;
- the minimum spanning tree computed by Prim's algorithm ('Prim'), proposed by [24] as the state of the art for overlay-based DFL;
- the simplistic heuristic for (22) based on SDP relaxation plus greedy rounding ('Relaxation-ρ');
- the heuristics for (24) based on convex relaxation plus rounding ('Relaxation-λ') or greedy perturbation [6] ('Greedy').

We will first evaluate a basic setting where we provide accurate information about the underlay to all the algorithms, use (11) to optimize the link weights under each topology design, and let all the communications occur directly over the underlay routing paths (i.e., without overlay routing). We will separately evaluate the impact of overlay routing, inference errors, and weight design.

4.2 Simulation Results

Due to space limitation, we will only present the results based on CIFAR-10 and Roofnet, and defer the other results to [12].

4.2.1 Results without Overlay Routing. As the most difficult part of our problem is topology design (i.e., optimization of E_a), we first compare the topology design solutions without overlay routing. As shown in Fig. 5, (i) using sparse topologies rather than the clique can effectively reduce the training time without compromising the performance at convergence, (ii) different topology designs only slightly differ in terms of the convergence rate over epochs, but can differ significantly in terms of the convergence rate over the actual (wall-clock) time, and (iii) the proposed design by 'SCA' notably outperforms the others in terms of training time, while achieving the same loss/accuracy at convergence. Note that the time axis is in log scale. A closer examination further shows that the existing topology designs ('Prim', 'Ring') work relatively well in that they not only converge much faster than the baseline ('Clique') but also outperform the other heuristics based on the optimizations we

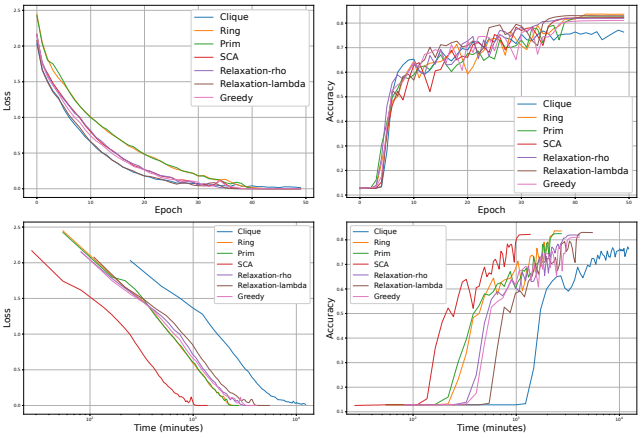


Figure 5: CIFAR-10 over Roofnet: without overlay routing.

	SCA	Relaxation-ρ	Relaxation-λ	Greedy
MNIST	21.31	5.96	6.71	55.84
CIFAR-10	19.55	6.12	6.35	51.74

Table 1: Running times of the proposed algorithms relative to 'Prim' (as ratios) for Roofnet.

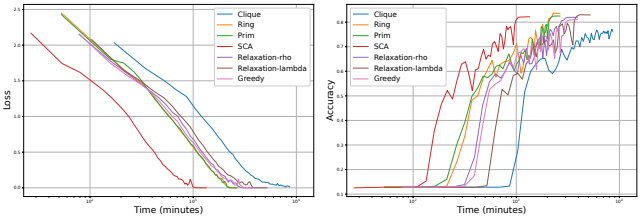


Figure 6: CIFAR-10 over Roofnet: with overlay routing.

formulate ('Relaxation-ρ', 'Relaxation-λ', 'Greedy'). Nevertheless, 'SCA' is able to converge even faster by better approximating the optimal solution to (22). Note that although we have not optimized overlay routing, the proposed algorithm still benefits from the knowledge of how links are shared by routing paths within the underlay (via constraint (21)), which allows it to better balance the convergence rate and the communication time per iteration, while the state-of-the-art design ('Prim') ignores such link sharing. This result highlights the importance of underlay-aware design for overlay-based DFL.

Meanwhile, we note that the simpler heuristics 'Relaxation-ρ' and 'Relaxation-λ' outperform 'SCA' in terms of running time, as shown in Table 1, and all the algorithms based on our optimizations are slower than 'Prim'. This indicates further room for improvement for future work in terms of the tradeoff between the quality of design and the computational efficiency.

4.2.2 Results with Overlay Routing. Fig. 6 shows the results after optimizing the communication schedule under each design by the overlay routing optimization (5). Compared with Fig. 5 (second row), we see that the training time is further reduced for all the topology designs. However, the improvement is only prominent for the dense topology ('Clique') with a reduction of 28%, while the improvement for the other topologies is incremental (2–4%). Intuitively, this is because these sparse topologies generate much less load on the underlay network, and hence leave less room for improvement for overlay routing.

4.2.3 Results with Inference Errors. While the above results are obtained under the perfect knowledge of the nonempty categories $\mathcal{F}$ and the category capacities $(C_F)_{F\in\mathcal{F}}$, the observations therein remain valid under the inferred values of these parameters, as our inference algorithms proposed in [11] are able to infer these parameters with sufficient accuracy. We thus defer the detailed results to [12] due to space limitation.

4.2.4 Results under Other Weight Design. Instead of solving the SDP (11), one could use alternative designs for link weights. To assess the impact of different weight designs, we conducted additional simulations for the case of Fig. 5–6 using the widely-adopted Metropolis-Hasting weights. The results, provided in [12], show that the Metropolis-Hasting weights introduce a noticeable delay in convergence compared to our proposed weight design.

5 CONCLUSION

We considered, for the first time, communication optimization for running DFL on top of a bandwidth-limited underlay network. To this end, we formulated a framework for jointly optimizing the hyperparameters controlling the communication demands between learning agents and the communication schedule (including routing and flow rates) to fulfill such demands, without cooperation from the underlay. We showed that the resulting problem can be decomposed into a set of interrelated subproblems, and developed efficient algorithms through carefully designed convex relaxations. Our evaluations based on real topologies and datasets validated the efficacy of the proposed solution in significantly reducing the training time without compromising the quality of the trained model. Our results highlight the need of network-application co-design in supporting DFL over bandwidth-limited networks, and our overlay-based approach facilitates the deployment of our solution in existing networks without changing their internal operations.

REFERENCES

[1] 3GPP. 2018. NR; Study on integrated access and backhaul. TR 38.874. https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3232

[2] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. 2004. Link-level Measurements from an 802.11b Mesh Network. In *SIGCOMM*.

[3] Jingrong Chen, Hong Zhang, Wei Zhang, Liang Luo, Jeffrey Chase, Ion Stoica, and Danyang Zhuo. 2022. NetHint: White-Box Networking for Multi-Tenant Data Centers. In *USENIX NSDI*. 1327–1343.

[4] Xianhao Chen, Guangyu Zhu, Yiqin Deng, and Yuguang Fang. 2022. Federated learning over multihop wireless networks with in-network aggregation. *IEEE Transactions on Wireless Communications* 21, 6 (2022), 4622–4634.

[5] Cho-Chun Chiu, Xusheng Zhang, Ting He, Shiqiang Wang, and Ananthram Swami. 2023. Laplacian Matrix Sampling for Communication- Efficient Decentralized Learning. *IEEE Journal on Selected Areas in Communications* 41, 4 (2023), 887–901. https://doi.org/10.1109/JSAC.2023.3242735

[6] Arpita Ghosh and Stephen Boyd. 2006. Growing Well-connected Graphs. In *IEEE Conference on Decision and Control*. 6605–6611.

[7] Michel X Goemans and Young-Soo Myung. 1993. A catalog of Steiner tree formulations. *Networks* 23, 1 (1993), 19–28.

[8] Zhidong He. 2019. Optimization of convergence rate via algebraic connectivity. arXiv:1912.06536 [math.OC]

[9] Yifan Hua, Kevin Miller, Andrea L Bertozzi, Chen Qian, and Bao Wang. 2022. Efficient and reliable overlay networks for decentralized federated learning. *SIAM J. Appl. Math.* 82, 4 (2022), 1558–1586.

[10] Yudi Huang and Ting He. 2023. Overlay Routing Over an Uncooperative Underlay. In *The 24th International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc'23)*. 151–160.

[11] Yudi Huang and Ting He. 2024. Overlay Routing Over an Uncooperative Underlay. https://pennstateoffice365-my.sharepoint.com/:b:/g/personal/tzh58_psu_edu/ESv6_HUIDMdImHnCunvJdQwBv0KV7VxuI6sqG3NSbGuCfw?e=dIZ13s

[12] Yudi Huang, Tingyang Sun, and Ting He. 2024. Overlay-based Decentralized Federated Learning in Bandwidth-limited Networks. arXiv:2408.04705 [cs.LG] https://arxiv.org/abs/2408.04705

[13] Zehan Jia, Ziqi Yu, Haijun Liao, Zhao Wang, Zhenyu Zhou, Xiaoyan Wang, Guoqing He, Shahid Mumtaz, and Mohsen Guizani. 2022. Dispatching and Control Information Freshness-Aware Federated Learning for Simplified Power IoT. In *GLOBECOM*. IEEE, 1097–1102.

[14] Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. 2020. A faster interior point method for semidefinite programming. In *IEEE FOCS*. IEEE, 910–918.

[15] Zhida Jiang, Yang Xu, Hongli Xu, Lun Wang, Chunming Qiao, and Liusheng Huang. 2023. Joint Model Pruning and Topology Construction for Accelerating Decentralized Machine Learning. *IEEE Transactions on Parallel and Distributed Systems* (2023).

[16] Peter Kairouz et al. 2021. *Advances and Open Problems in Federated Learning*. Now Foundations and Trends.

[17] Anastasia Koloskova, Tao Lin, Sebastian U Stich, and Martin Jagg. 2020. Decentralized Deep Learning with Arbitrary Communication Compression. In *The International Conference on Learning Representations (ICLR)*.

[18] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. 2020. A Unified Theory of Decentralized SGD with Changing Topology and Local Updates. In *ICML*.

[19] Batiste Le Bars, Aurélien Bellet, Marc Tommasi, Erick Lavoie, and Anne-Marie Kermarrec. 2023. Refined convergence and topology learning for decentralized SGD with heterogeneous data. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1672–1702.

[20] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. 2017. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 5336–5346.

[21] Yucheng Lu and Christopher De Sa. 2021. Optimal Complexity in Decentralized Training. In *International Conference on Machine Learning (ICML)*.

[22] Miles Lubin. 2017. *Mixed-integer convex optimization: Outer approximation algorithms and modeling power*. Ph. D. Dissertation. Massachusetts Institute of Technology, Sloan School of Management, Operations Research Center.

[23] Liang Luo, Peter West, Jacob Nelson, Arvind Krishnamurthy, and Luis Ceze. 2020. PLink: Discovering and Exploiting Locality for Accelerated Distributed Training on the public Cloud. In *Proceedings of Machine Learning and Systems*, Vol. 2. 82–97.

[24] Othmane Marfoq, Chuan Xu, Giovanni Neglia, and Richard Vidal. 2020. Throughput-optimal topology design for cross-silo federated learning. *Advances in Neural Information Processing Systems* 33 (2020), 19478–19487.

[25] H. McMahan, Eider Moore, D. Ramage, S. Hampson, and Blaise Agüera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *AISTATS*.

[26] Giovanni Neglia, Chuan Xu, Don Towsley, and Gianmarco Calbi. 2020. Decentralized gradient methods: does topology matter?. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2348–2358.

[27] Jiaming Pei, Wenxuan Liu, Lukun Wang, Chao Liu, Ali Kashif Bashir, and Yue Wang. 2023. Fed-IoUT: Opportunities and Challenges of Federated Learning in the Internet of Underwater Things. *IEEE Internet of Things Magazine* 6, 1 (2023), 108–112.

[28] Pinyarash Pinyoanuntapong, Wesley Houston Huff, Minwoo Lee, Chen Chen, and Pu Wang. 2022. Toward scalable and robust AIoT via decentralized federated learning. *IEEE Internet of Things Magazine* 5, 1 (2022), 30–35.

[29] Henrik Ronkainen, Jonas Edstam, Anders Ericsson, and Christer Östberg. 2020. Integrated access and backhaul – a new type of wireless backhaul in 5G. Ericsson Technology Review. https://www.ericsson.com/4ac691/assets/local/reports-papers/ericsson-technology-review/docs/2020/introducing-integrated-access-and-backhaul.pdf

[30] Navjot Singh, Deepesh Data, Jemin George, and Suhas Diggavi. 2022. SPARQ-SGD: Event-triggered and compressed communication in decentralized optimization. *IEEE Trans. Automat. Control* 68, 2 (2022), 721–736.

[31] Thijs Vogels, Hadrien Hendrikx, and Martin Jaggi. 2022. Beyond spectral gap: The role of the topology in decentralized learning. *Advances in Neural Information Processing Systems* 35 (2022), 15039–15050.

[32] Jianyu Wang, Anit Kumar Sahu, Gauri Joshi, and Soummya Kar. 2022. MATCHA: A Matching-Based Link Scheduling Strategy to Speed up Distributed Optimization. *IEEE Transactions on Signal Processing* 70 (2022), 5208–5221.

[33] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. 2019. Adaptive Federated Learning in Resource Constrained Edge Computing Systems. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1205–1221. https://doi.org/10.1109/JSAC.2019.2904348

[34] Hong Xing, Osvaldo Simeone, and Suzhi Bi. 2021. Federated learning over wireless device-to-device networks: Algorithms and convergence analysis. *IEEE Journal on Selected Areas in Communications* 39, 12 (2021), 3723–3741.