ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Department of Computer Science
Network Security Group

ETH Zurich
CAB F
Universitätstrasse 6
CH-8092 Zurich

Network-Aware Communication for Distributed Low-Communication Training

1 Introduction

Training large language models typically relies on tightly coupled clusters with high-bandwidth, low-latency interconnects. This requirement makes it difficult to aggregate computational resources distributed across datacenters, geographic regions, or administrative domains, where network capacity is lower, latency is higher, and available bandwidth may vary over time.

Distributed Low-Communication training (DiLoCo) [1] provides an alternative training paradigm for such environments. Instead of synchronizing gradients after every optimization step, workers independently perform many local optimization steps and periodically exchange model updates. This substantially reduces communication frequency and makes geographically distributed training feasible. More recently, Decoupled DiLoCo [2] further removes global synchronization barriers by allowing independent learners to asynchronously communicate parameter fragments to a central synchronizer.

These approaches substantially relax the communication requirements of distributed training, but the remaining network communication is largely treated as a transport mechanism rather than an optimization opportunity. In geographically distributed deployments, learners may be connected through paths with substantially different bandwidth, latency, congestion, and reliability. At the same time, DiLoCo-style training provides unusual flexibility in when updates must arrive: communication occurs between relatively long periods of local computation, and Decoupled DiLoCo can aggregate parameter fragments without waiting for every learner. Consequently, communication can potentially be scheduled, prioritized, and routed around network conditions without directly stalling computation.

This project investigates how to make the communication layer of DiLoCo-style distributed training explicitly network-aware. The goal is to characterize its communication requirements, identify opportunities to overlap and schedule communication, and design mechanisms that adapt update transmission to heterogeneous and time-varying network conditions. The broader objective is to determine how distributed training algorithms and wide-area networks can jointly exploit the communication flexibility exposed by low-communication and asynchronous training.

2 Project Description

The project will study the interaction between DiLoCo-style training and heterogeneous wide-area networks. It will first characterize the communication generated by DiLoCo and Decoupled DiLoCo, then develop network-aware mechanisms for scheduling and transporting model updates. The project is organized into the following tasks.

Task 1: Characterizing DiLoCo Communication

Build an experimental environment for DiLoCo or Decoupled DiLoCo and characterize the communication generated during distributed training. The analysis should identify the network requirements and timing constraints associated with model synchronization.

Key questions include:

- How much data is transferred during each outer synchronization, and how does this scale with model size and the number of learners?
- How much communication can be overlapped with local training before network performance affects training progress?
- How do bandwidth, latency, jitter, packet loss, and transient congestion affect synchronization time and overall training throughput?
- How do the communication patterns differ between synchronous DiLoCo and fragment-based asynchronous Decoupled DiLoCo?

The resulting measurements should establish when network performance becomes a bottleneck despite DiLoCo's reduced communication frequency and identify the degrees of freedom available to a network-aware communication layer.

Task 2: Network-Aware Update Scheduling

Design mechanisms that exploit the temporal flexibility of DiLoCo communication. Instead of treating each synchronization as an indivisible transfer, the system should reason about when different model updates or parameter fragments need to reach their destination.

Possible mechanisms include:

- Overlapping update transmission with subsequent local optimization steps.
- Prioritizing parameter fragments according to synchronization deadlines or learner progress.
- Dynamically controlling transmission concurrency based on available network capacity.
- Delaying, reordering, or pacing transfers to avoid transient congestion without delaying the next required synchronization.

For Decoupled DiLoCo, the project should investigate whether its fragment-level communication

and relaxed synchronization semantics expose additional scheduling opportunities. In particular, the communication layer may prioritize updates that help the synchronizer reach its aggregation quorum earlier while deprioritizing transfers whose arrival does not affect immediate training progress.

Task 3: Path-Aware Communication

Extend the communication layer to environments where multiple network paths are available between learners and the synchronizer. Rather than relying on a fixed path or conventional per-flow load balancing, the system should explicitly match DiLoCo communication to path characteristics.

This task may explore:

- Selecting paths based on measured bandwidth, latency, congestion, and reliability.
- Assigning different parameter fragments or learner updates to different paths.
- Spreading communication across multiple paths when additional aggregate capacity outweighs path heterogeneity.
- Adapting path selection as network conditions change during training.

The objective is not necessarily to minimize the completion time of every individual transfer. Instead, path decisions should minimize communication on the critical path of training, taking into account local computation, synchronization deadlines, and the aggregation behavior of DiLoCo.

Task 4: Joint Training and Network Adaptation

Investigate whether information from the training algorithm can be combined with network measurements to make better communication decisions.

For example, a scheduler could jointly consider:

- The progress and expected completion time of individual learners.
- Which parameter fragments are required for the next synchronization or aggregation step.
- The number of updates required to satisfy a synchronization quorum.
- Current and predicted path capacity and latency.
- The expected effect of delaying a transfer on overall training progress.

Based on these signals, the project will develop a scheduling policy that determines *what to send, when to send it, and over which path.* Depending on project progress, this may range from heuristic scheduling policies to an explicit optimization or completion-time model.

Task 5: Experimental Evaluation

Evaluate the proposed mechanisms under controlled and realistic network conditions. Experiments should vary both training and network characteristics, including model size, number of learners, synchronization frequency, bandwidth, latency, path heterogeneity, and transient network degradation.

The evaluation should compare against standard DiLoCo or Decoupled DiLoCo communication and quantify metrics such as:

- Training throughput and time-to-train.
- Communication time exposed on the training critical path.
- Network utilization and transferred bytes.
- Robustness to heterogeneous and time-varying network conditions.
- Sensitivity to scheduling, path selection, and synchronization parameters.

An important goal is to distinguish improvements from simply providing additional network capacity. The evaluation should therefore include controlled baselines that isolate the benefits of network-aware scheduling and path selection.

Background and Skills

The project lies at the intersection of **computer networks**, **distributed systems**, and **distributed machine learning**. A successful candidate will benefit from familiarity with:

- Computer networks, including bandwidth, latency, congestion, and transport protocols.
- Distributed systems and asynchronous communication.
- Distributed machine learning and common parallel training architectures.
- Python and experience with machine-learning frameworks such as PyTorch or JAX.
- Experimental methodology and performance analysis.

Prior experience with large-scale model training, GPU communication, RDMA, or wide-area networking is advantageous but not required.

References

[1] A. Douillard, Q. Feng, A. A. Rusu, R. Chhaparia, Y. Donchev, A. Kuncoro, M. Ranzato, A. Szlam, and J. Shen, "Diloco: Distributed low-communication training of language models," 2024. [Online]. Available: https://arxiv.org/abs/2311.08105

[2] A. Douillard, K. Rush, Y. Donchev, Z. Charles, N. Fallen, A. Dubey, I. Gog, J. Dean, B. Woodworth, Z. Garrett, N. Keating, J. Bishop, H. Prior, E. Yvinec, A. Szlam, M. Ranzato, and J. Dean, "Decoupled diloco for resilient distributed pre-training," 2026. [Online]. Available: https://arxiv.org/abs/2604.21428