Titanic: Towards Production Federated Learning with Large Language Models

University of Toronto

Ningxin Su, Chenghao Hu, Baochun Li, Bo Li (HKUST)

Production FL

Not limited to original federated learning…

distributed machine learning

more practical

security

Even the smallest 7B Llama 2 model takes 32 GB of GPU memory (LoRA, batch size of 4, no quantization)

Conventional federated learning (FL) requires sending model updates to the server — but the models are too large!

Llama 2 7B: **27GB** of data to be sent in each round of communication

Challenges of training LLMs under the FL structure:

computation resource constraints

communication overhead

heterogeneity …

Titanic, a new distributed training paradigm that can

fine-tune LLMs

operate within the computation and communication constraints

preserve privacy

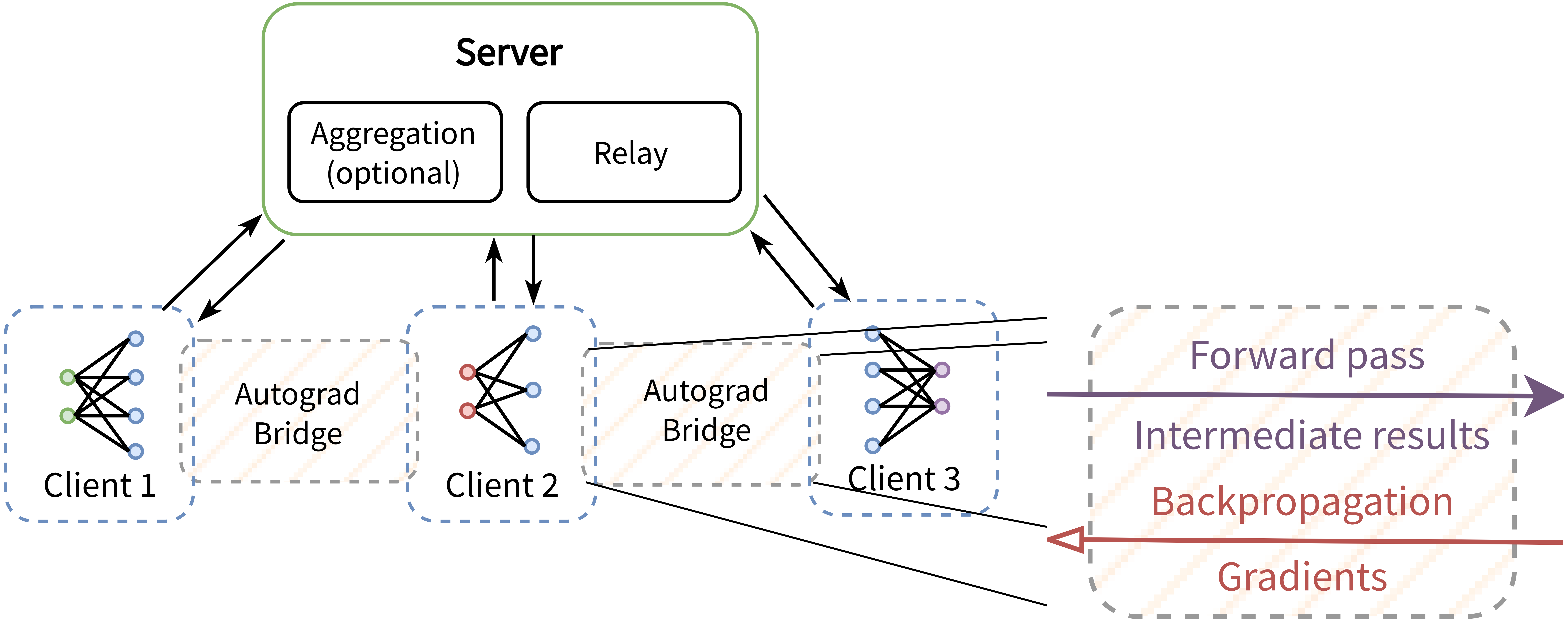
How?

By technically **separating fine-tuning** to clients and server

However, scalability is limited in split learning.

We seek a more general paradigm

Titanic - High level



Design principle of **Autograd Bridge**:

Automate model partitioning by client resources constraints, model agnostic

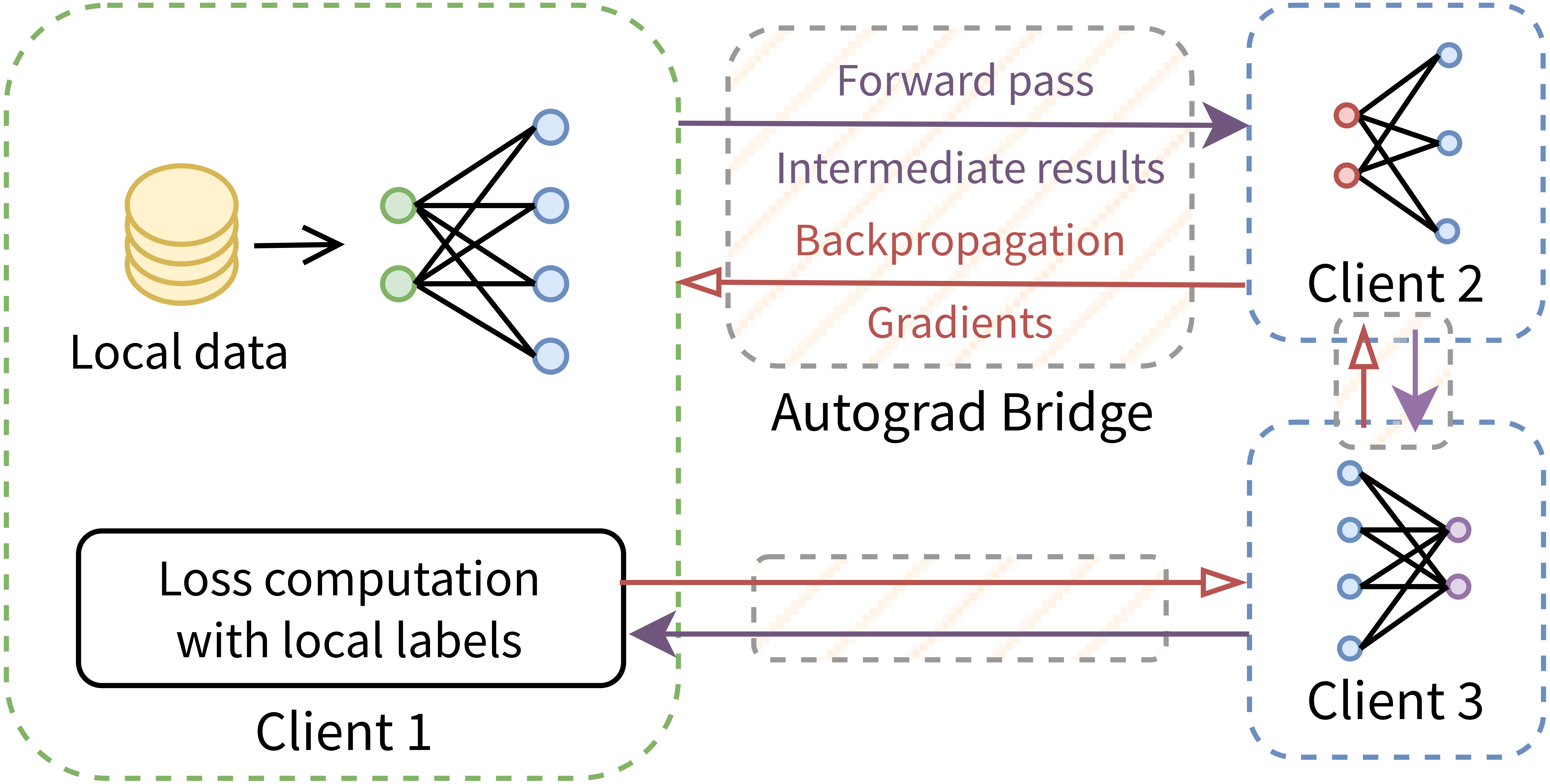
Pytorch is isolated from network transmission (via autograd-bridge)

A little more details about implementation

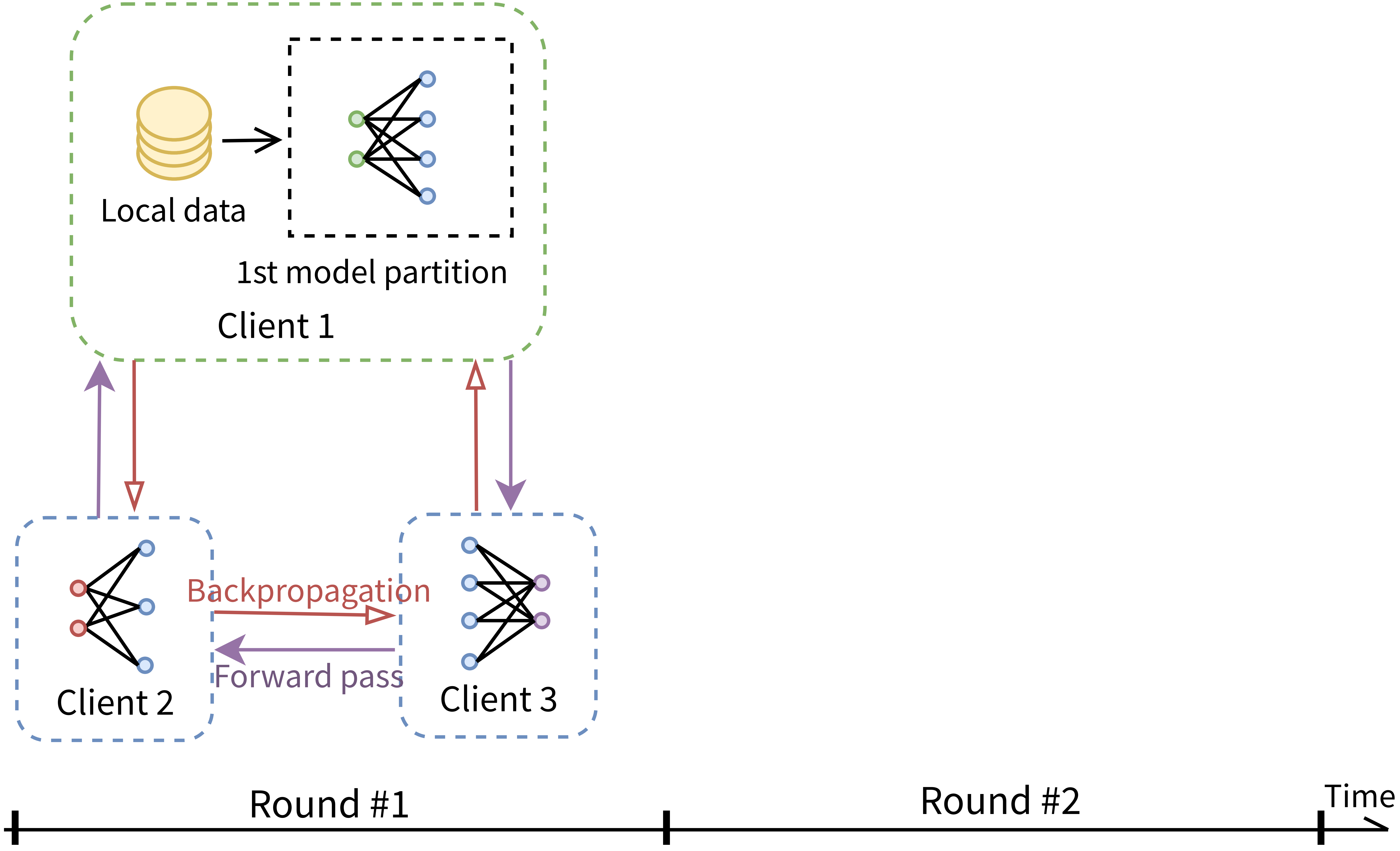
Overwrite *backward( )* to receive gradients cross client without influencing the model

Use WebSockets protocol over HTTPS for generality

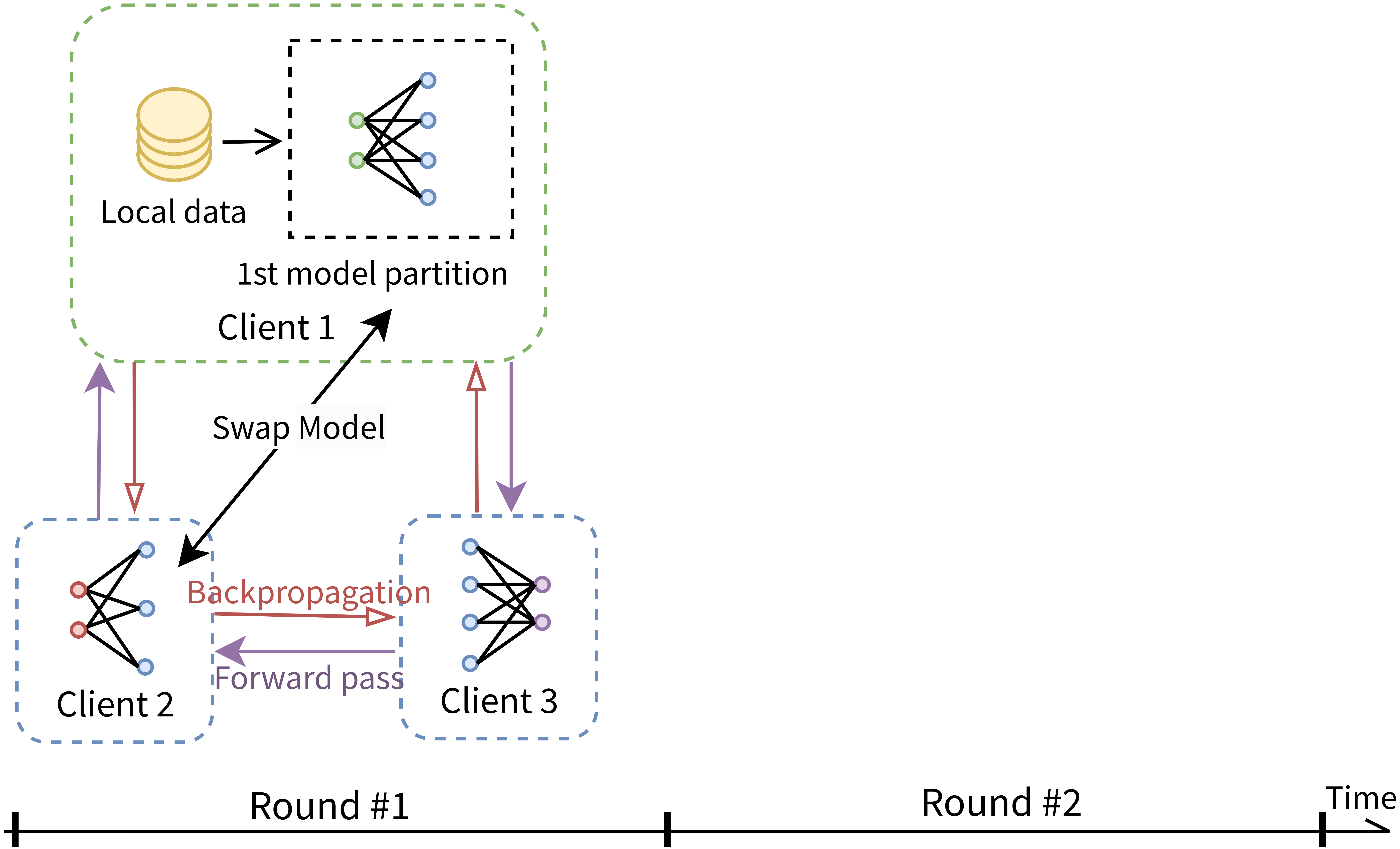
A tale of three cases



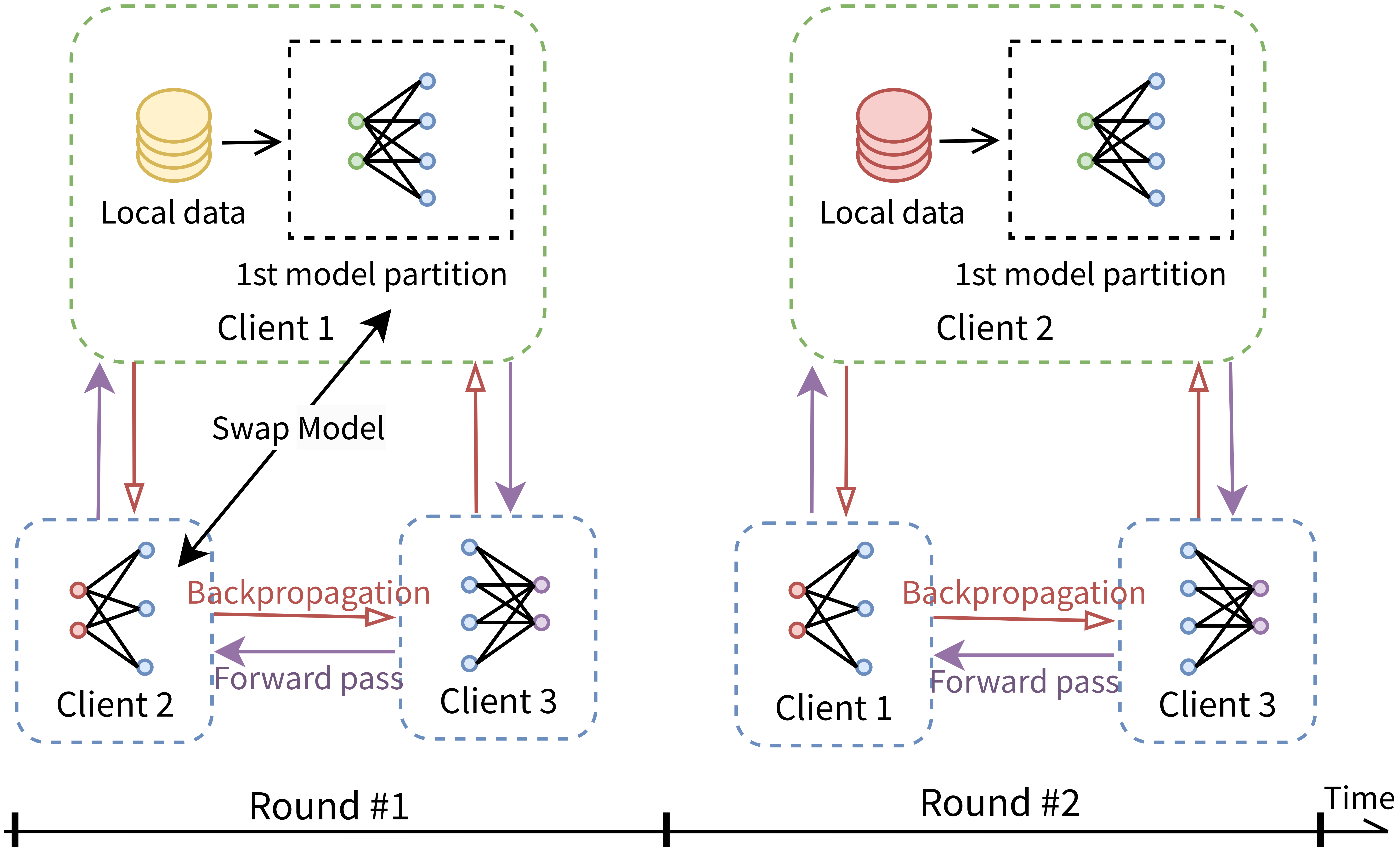
Case 1: Only one client trains with its local data



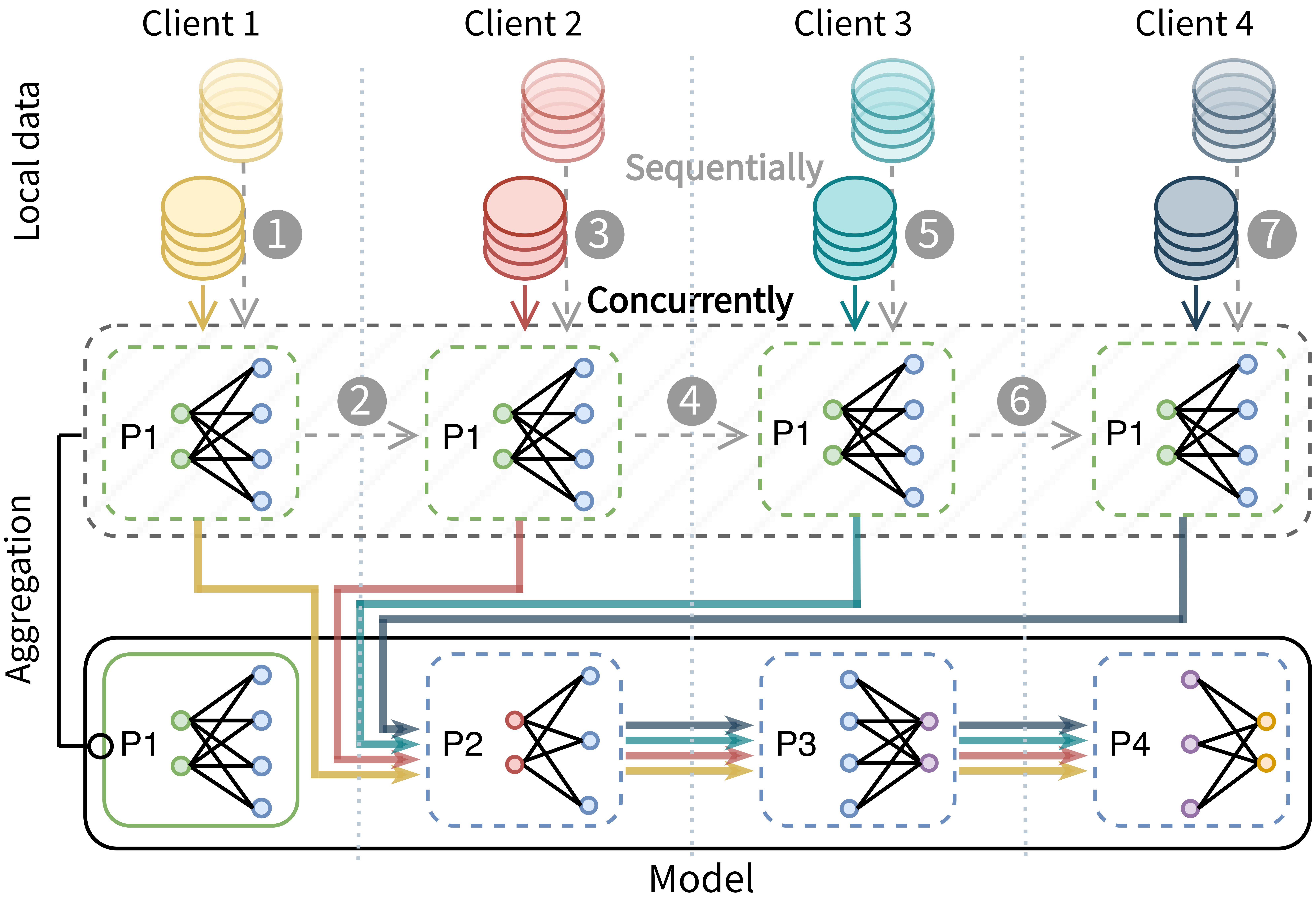
Case 2: Multiple clients train with their local data, without aggregation



Case 2: Multiple clients train with their local data, without aggregation



Case 2: Multiple clients train with their local data, without aggregation



Case 3: Multiple clients train with their local data, with aggregation

Available bandwidth and computation resources across clients or servers can be significantly different.

Client selection becomes a necessary step to ensure optimal performance.

First, assign model partitions $\{P_k\}_{k \in K}$ to clients $\{C_n\}_{n \in N}$, K and N correspond to the total number of model partitions and clients, respectively.

match distance: $d_{kn} = a \cdot D_n^k + b \cdot T_n^k$

D_n^k is the transmission duration; and T_n^k is the training time

affinity: $A_n^k = \dfrac{1}{d_n^k}$

And we also need to take GPU memory into account.

We have several many details here, please find them in paper.

The problem can be transformed into an LP problem;

and the LP relaxation does not affect the integrality of the optimal solution [1].

Mosek [2] solver took less than 40 milliseconds for 100 clients and 32 partitions.

[1] R. R. Meyer, “A Class of Nonlinear Integer Programs Solvable by a Single Linear Program,” *SIAM Journal on Control and Optimization*, vol. 15, no. 6, pp. 935–946, 1977.

[2] MOSEK Optimizer API for Python, https://docs.mosek.com/latest/pythonapi/index.html

Model	Methods	Perplexity
OPT-1.3B	Centralized	**18.02**
	FL with FedAvg	**34.01**
	TITANIC with 4 clients $C_1 \to C_2 \to C_3 \to C_4$	$20.67 \to 19.34 \to 19.38 \to$ **19.17**
	TITANIC with 2 clients $C_1 \to C_2$	$18.77 \to$ **17.23**
Bloom-3B	Centralized	**18.04**
	FL with FedAvg	**30.24**
	TITANIC with 4 clients $C_1 \to C_2 \to C_3 \to C_4$	$25.16 \to 21.49 \to 19.34 \to$ **20.58**
	TITANIC with 2 clients $C_1 \to C_2$	$19.68 \to$ **19.20**
Llama 2-7B	Centralized	**2.54**
	FL with FedAvg	**3.56**
	TITANIC with 4 clients $C_1 \to C_2 \to C_3 \to C_4$	$2.90 \to 2.80 \to 2.93 \to$ **3.34**
	TITANIC with 2 clients $C_1 \to C_2$	$2.30 \to$ **2.85**

Thank you ♡