Nomad: Accelerating Geo-distributed Learning with Client Transfers

Bart Cox*, Lydia Y. Chen†, Jérémie Decouchant*

*Delft University of Technology, †Université de Neuchâtel

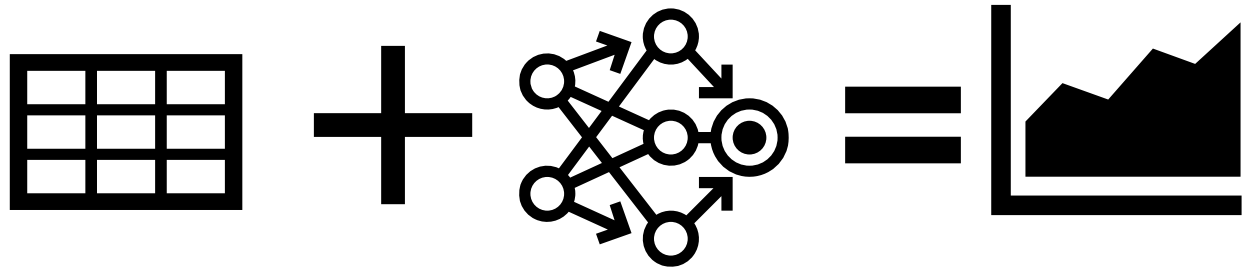
b.a.cox@tudelft.nl

7-11-2025

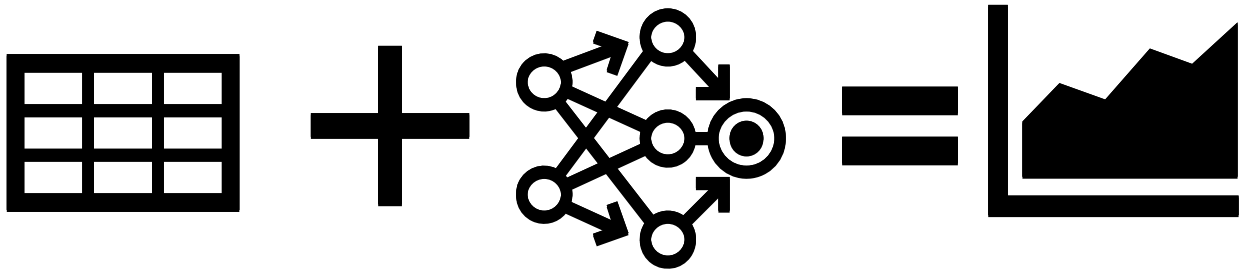
unine

TU Delft Delft University of Technology

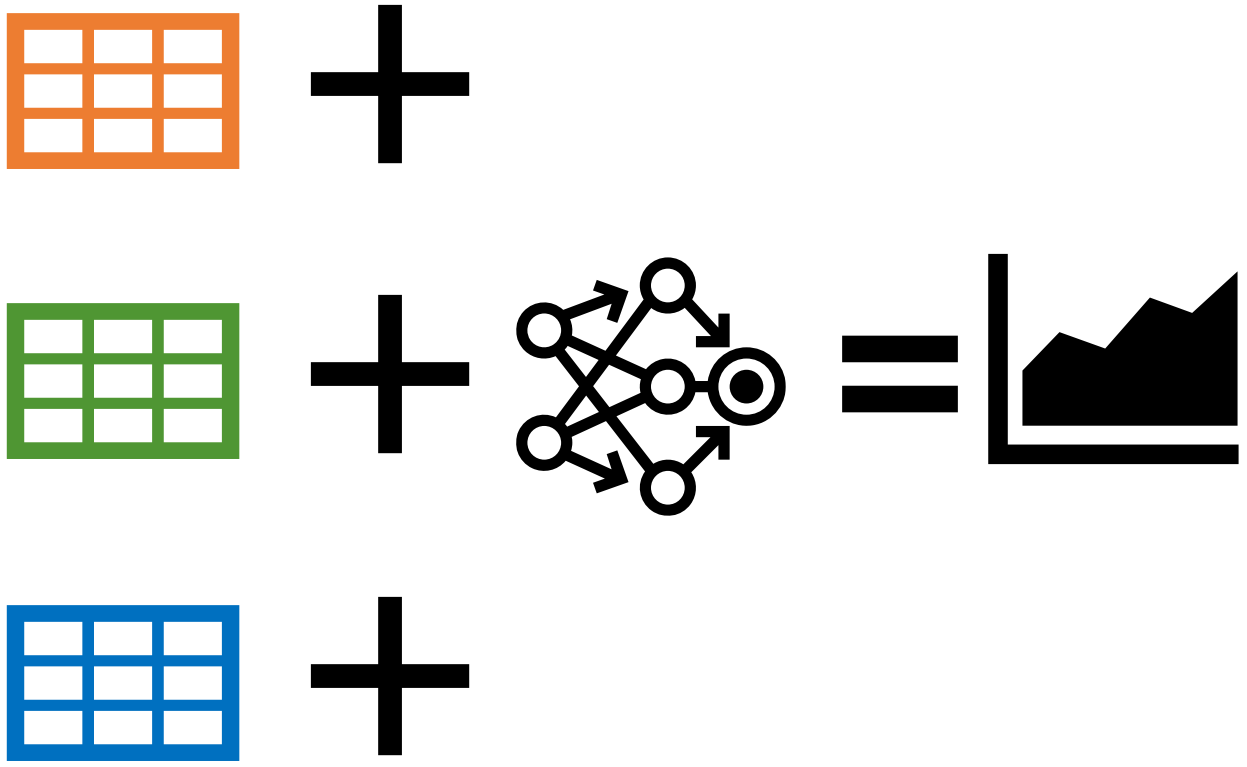
Federated Learning



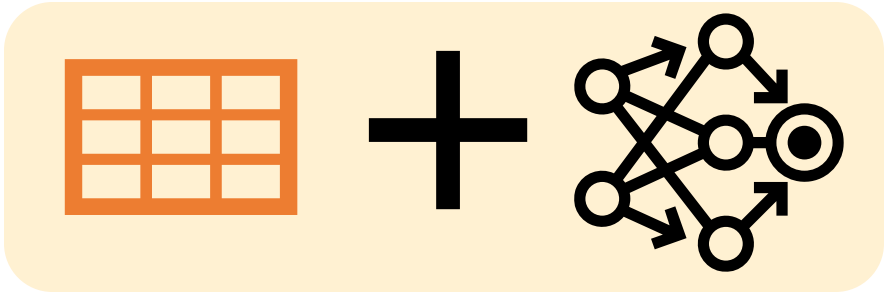
Federated Learning

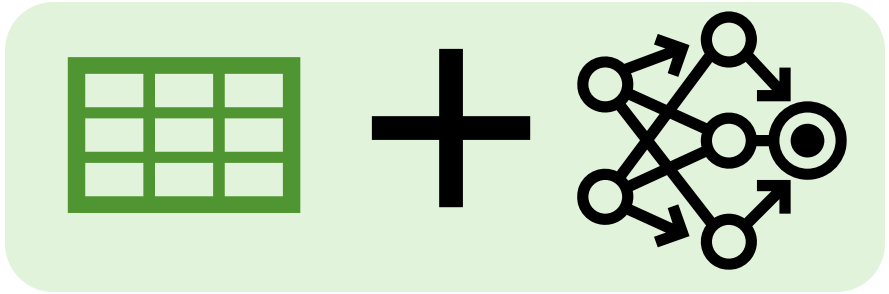


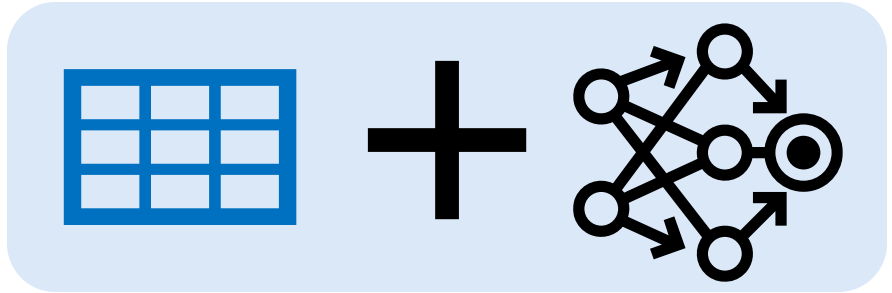
Federated Learning

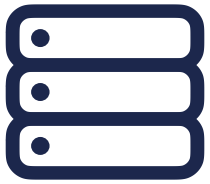


Federated Learning

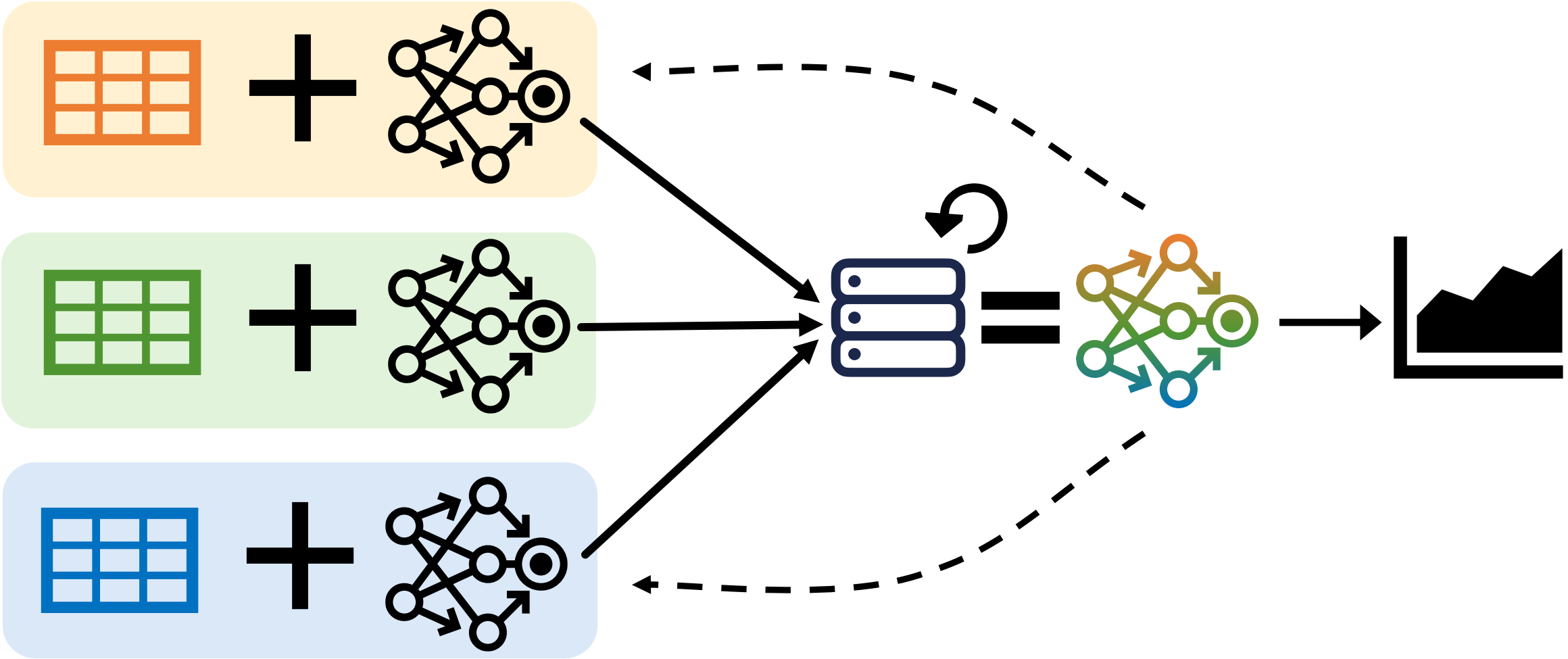






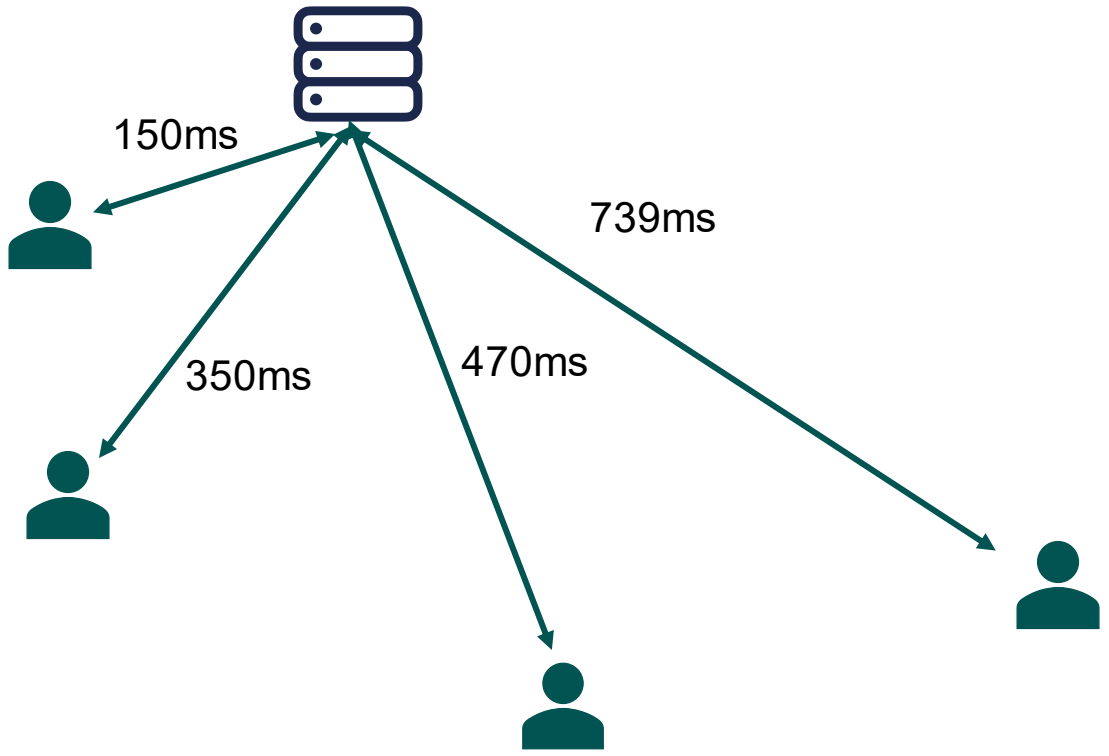


Federated Learning



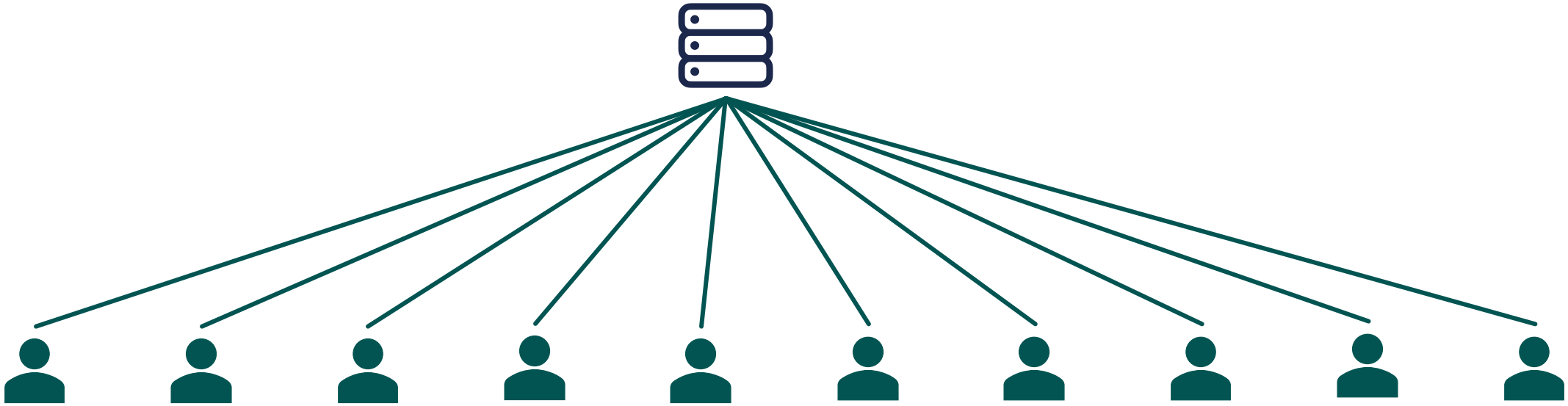
Asynchronous Learning

- Clients interact asynchronous with the server
- No waiting for the slowest client
- But, model staleness



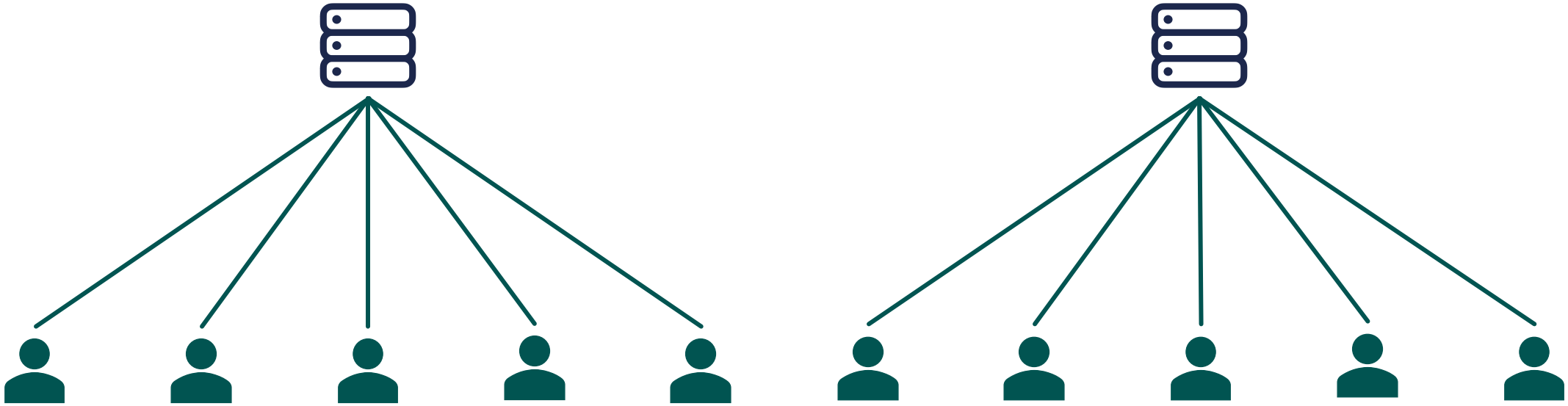
Multiple Servers

- Adding more and more clients can be a heavy burden for the server.
- Using multiple servers can reduce the load on the single server.
- More clients increase the average model staleness



Multiple Servers

- Adding more and more clients can be a heavy burden for the server.
- Using multiple servers can reduce the load on the single server.



Recap

Federated Learning

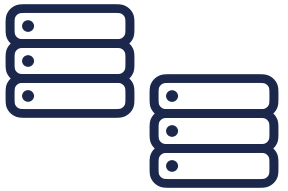
Local Data & learning
Central aggregation

+

Asynchronous

Clients update to server
at own pace

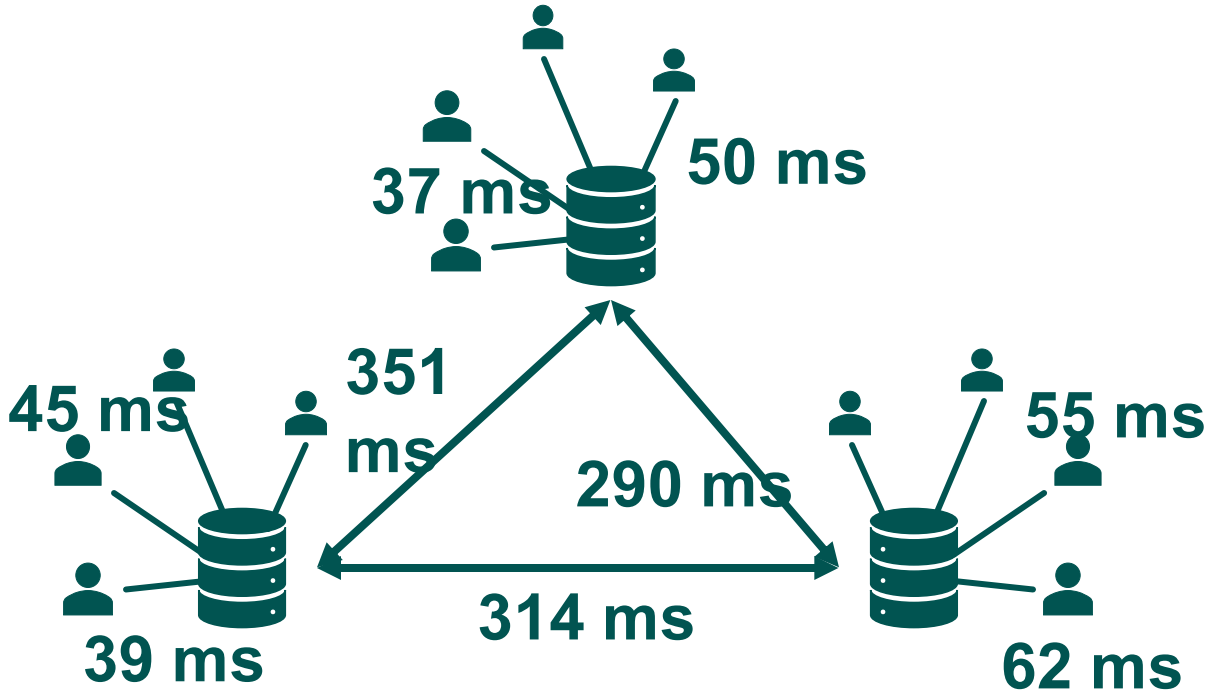
+



Multi-Server

Reduced Server load

Recap



Federated Learning

Local Data & learning
Central aggregation

+

Asynchronous

Clients update to server at own pace

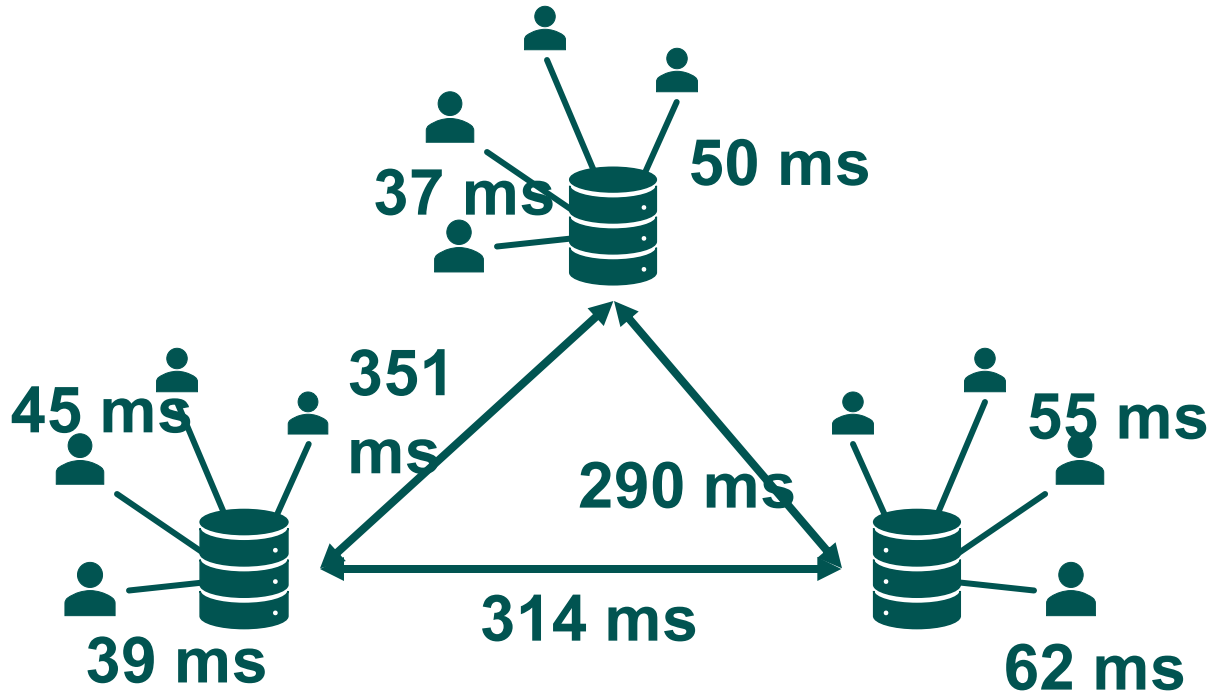
+

Multi-Server

Reduced Server load

Multiple Servers - Spyker [1]

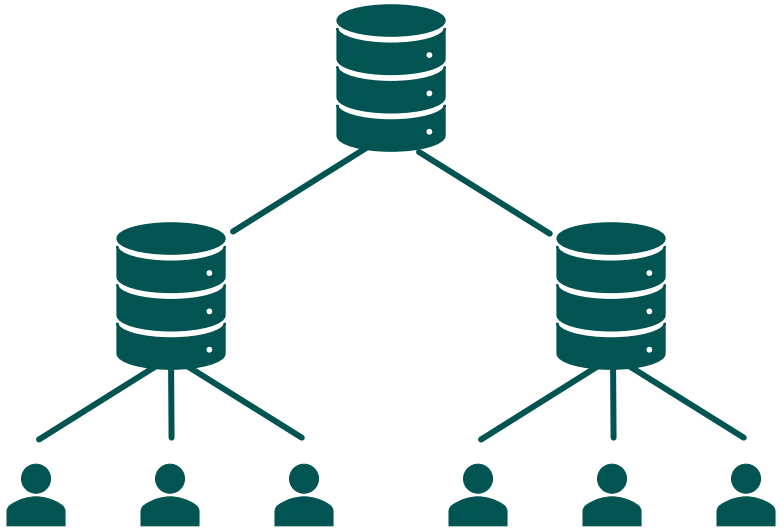
- Reduces average round time
- Low average model staleness
- It allows for asynchronous learning with low average model staleness
- Synchronisation between servers



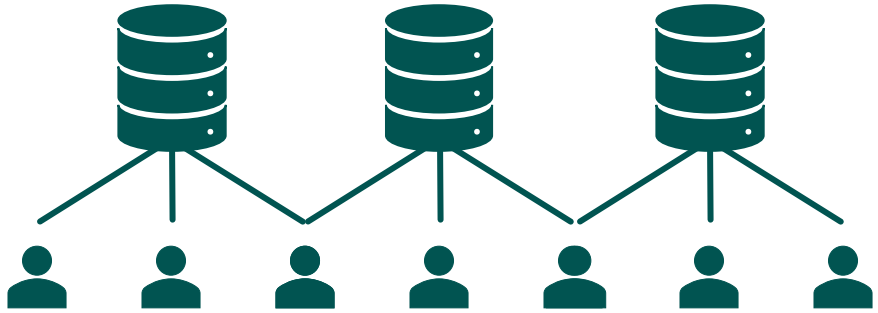
[1] Zuo et al. "Spyker: Asynchronous multi-server federated learning for geo-distributed clients" Middleware 2024

Other related work

Hiearchical FL [2]



Multi-Server with overlapping clients [3]



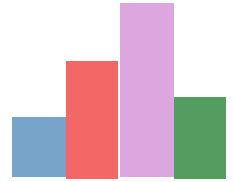
[2] Liu et al. "Client-edge-cloud hierarchical federated learning". ICC 2020
[3] Han et al. "FedMes: Speeding up federated learning with multiple edge servers." IEEE JSAC 2021

Server non-IIDness

- Server distribution is the **composition** of the local client distributions
- This works great in the **synchronous** case or of the clients are equally **fast**

Server non-IIDness

- Server distribution is the **composition** of the local client distributions
- This works great in the **synchronous** case or of the clients are equally **fast**
- What is system properties diverge?
  - For example: latency



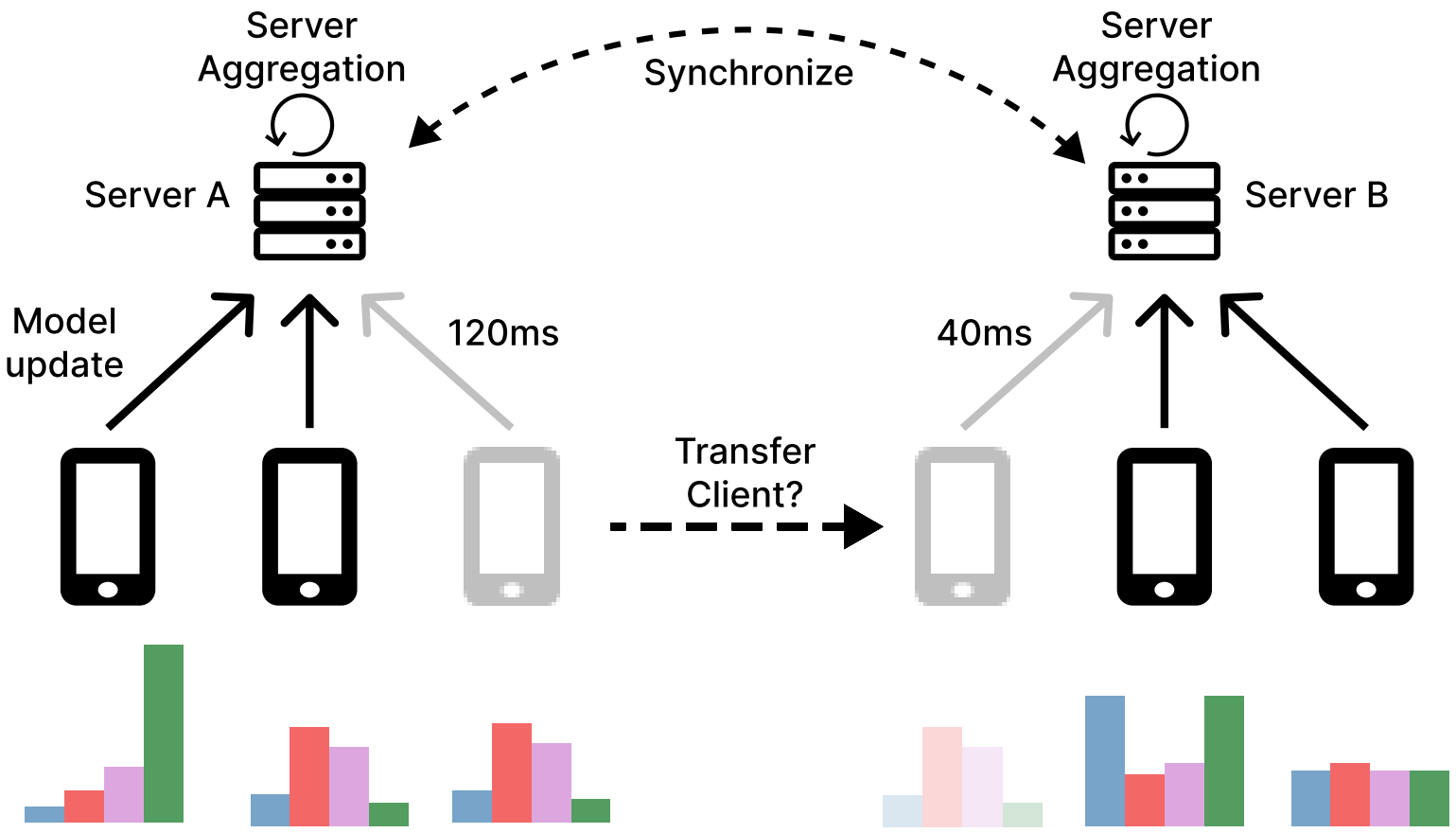
Server non-IIDness

- Model sharpness is used to measure non-IIDness
- Scale model sharpness by the latency

$$\tilde{V}_l = \sum_{i=1}^{N} m_{il} \cdot f_{i,l} \cdot v_i$$

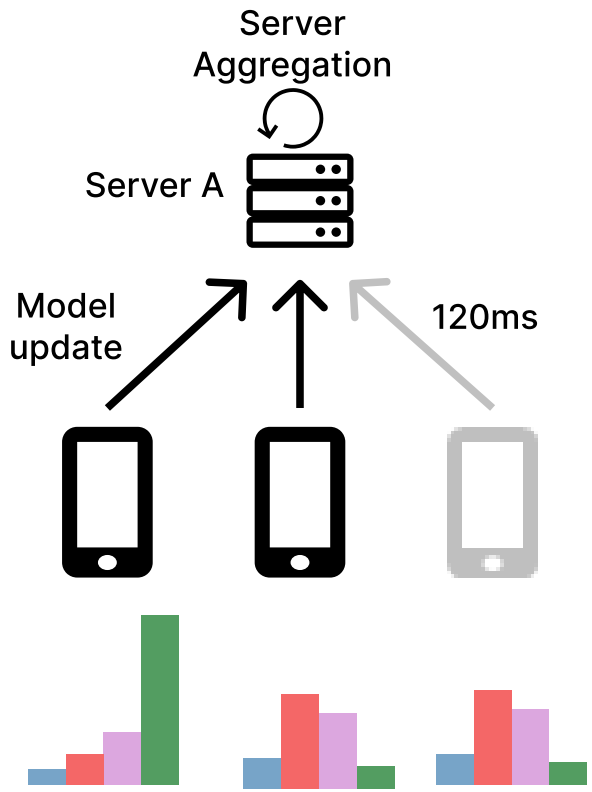
Method

1. Measure imbalance
2. Identify high-impact clients
3. Evaluate transfers
4. Transfer clients



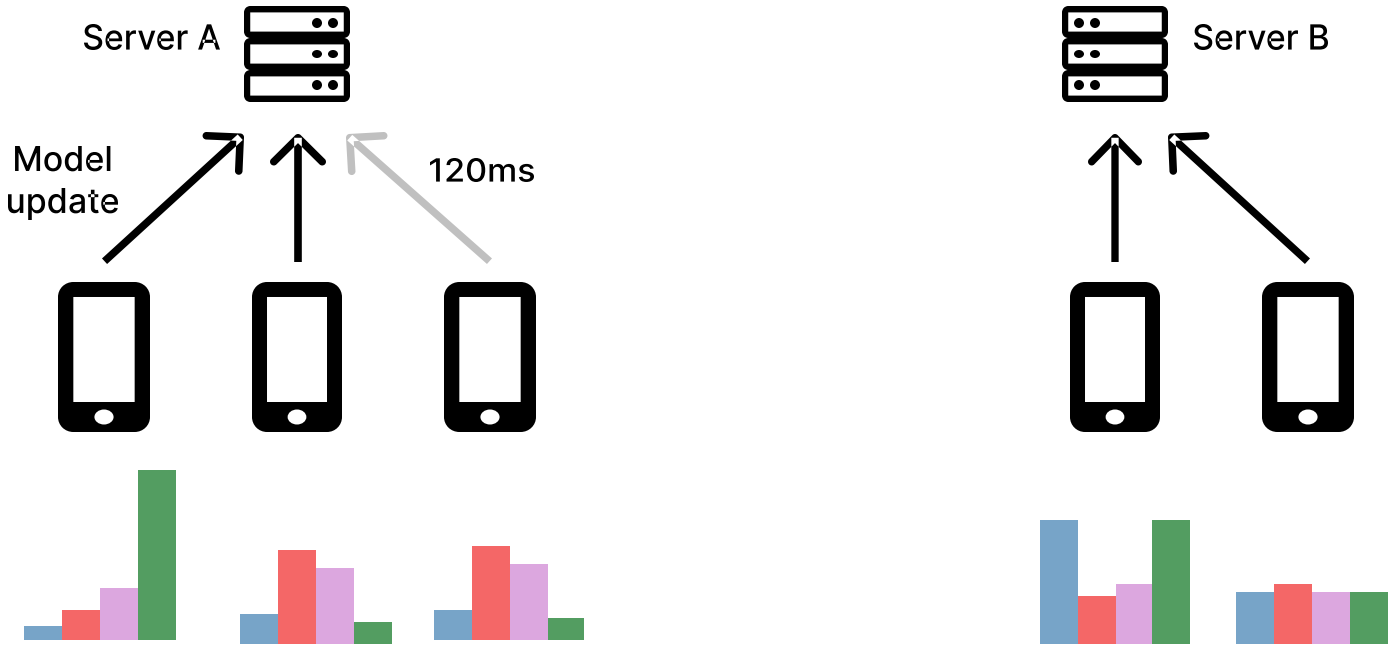
Ranking Clients

- Goal: minimize the effective non-IIDness between servers
- First we rank the clients based on the effect of removing the client from the server



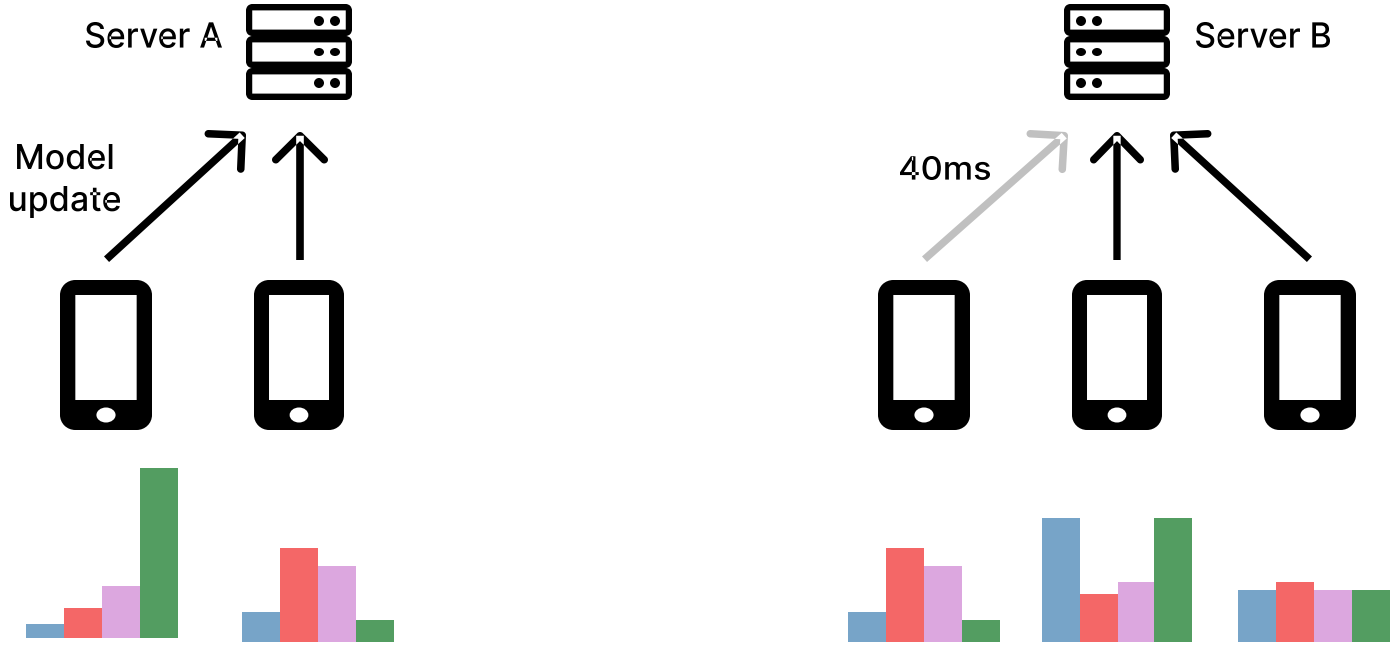
Transferring clients

- Estimate what the impact is of transferring a client from server A to server B



Transferring clients

- Estimate what the impact is of transferring a client from server A to server B



Evaluation

- Baselines:
  - Spyker [1]
  - Random Client Assignment (RCA)
  - Latency-Aware Client Assignment (LACA)
- Datasets
  - MNIST
  - Cifar10
- System setup:
  - 4 servers with {10,40,100} clients

[1] Zuo et al. "Spyker: Asynchronous multi-server federated learning for geo-distributed clients" Middleware 2024

RCA baseline

- Clients are randomly selected and moved to a different random server
- A server with 2 or fewer clients cannot be selected
  - We needs some clients assigned to a server to keep the system running

Algorithm 4: Random Client Reassignment (RCA) with Minimum Server Size Constraint

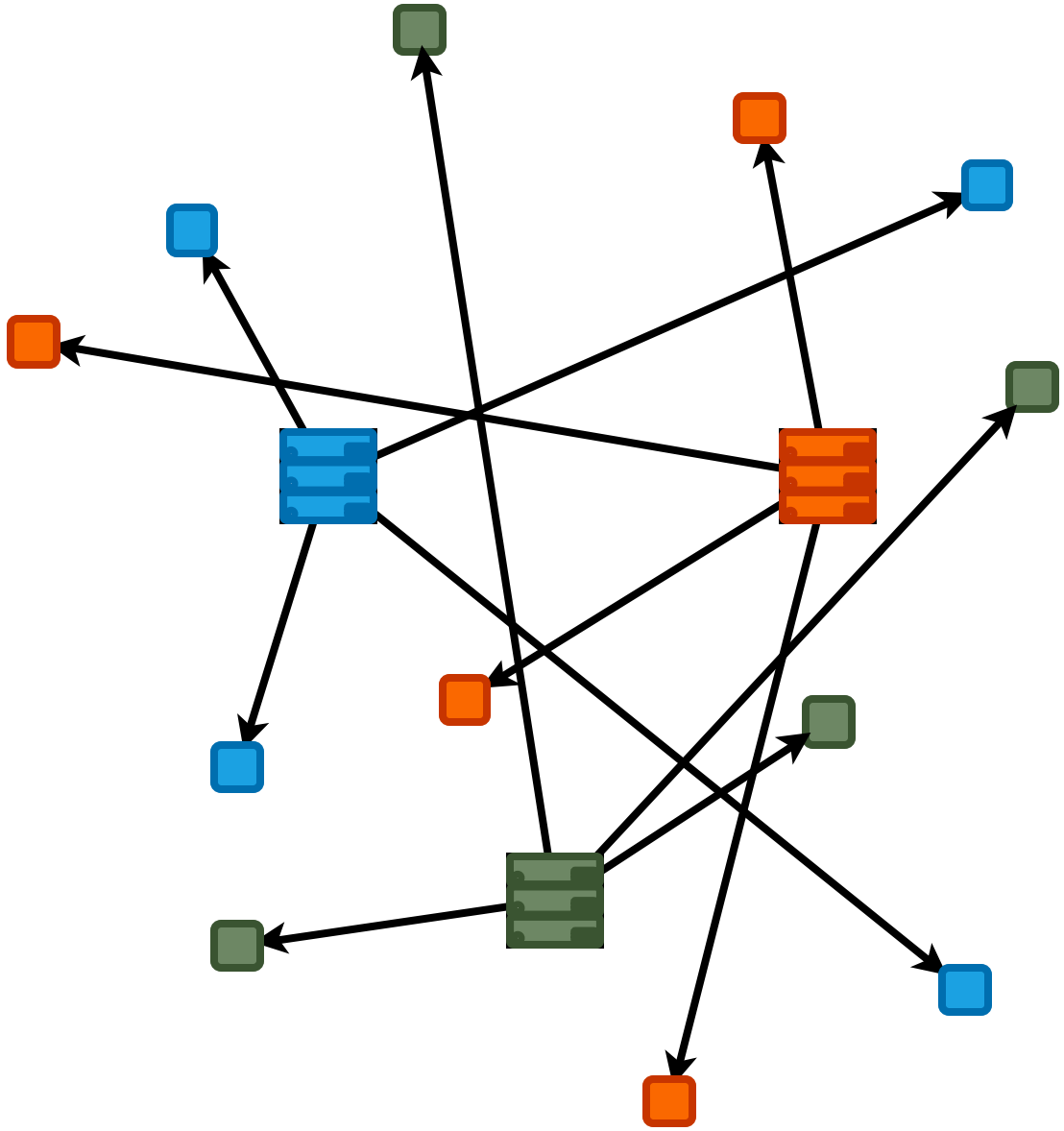
Input : $S = \{S_1, \ldots, S_M\}$: Client index sets for M servers

Output : Updated server assignments S

repeat
  Randomly sample $m_{\text{src}} \in \{1, \ldots, M\}$ such that $|S_{m_{\text{src}}}| > 2$
  Randomly sample $i \in S_{m_{\text{src}}}$
  Randomly sample $m_{\text{dst}} \in \{1, \ldots, M\} \setminus \{m_{\text{src}}\}$
until *valid move is found*;
$S_{m_{\text{src}}} \leftarrow S_{m_{\text{src}}} \setminus \{i\}$
$S_{m_{\text{dst}}} \leftarrow S_{m_{\text{dst}}} \cup \{i\}$
return S

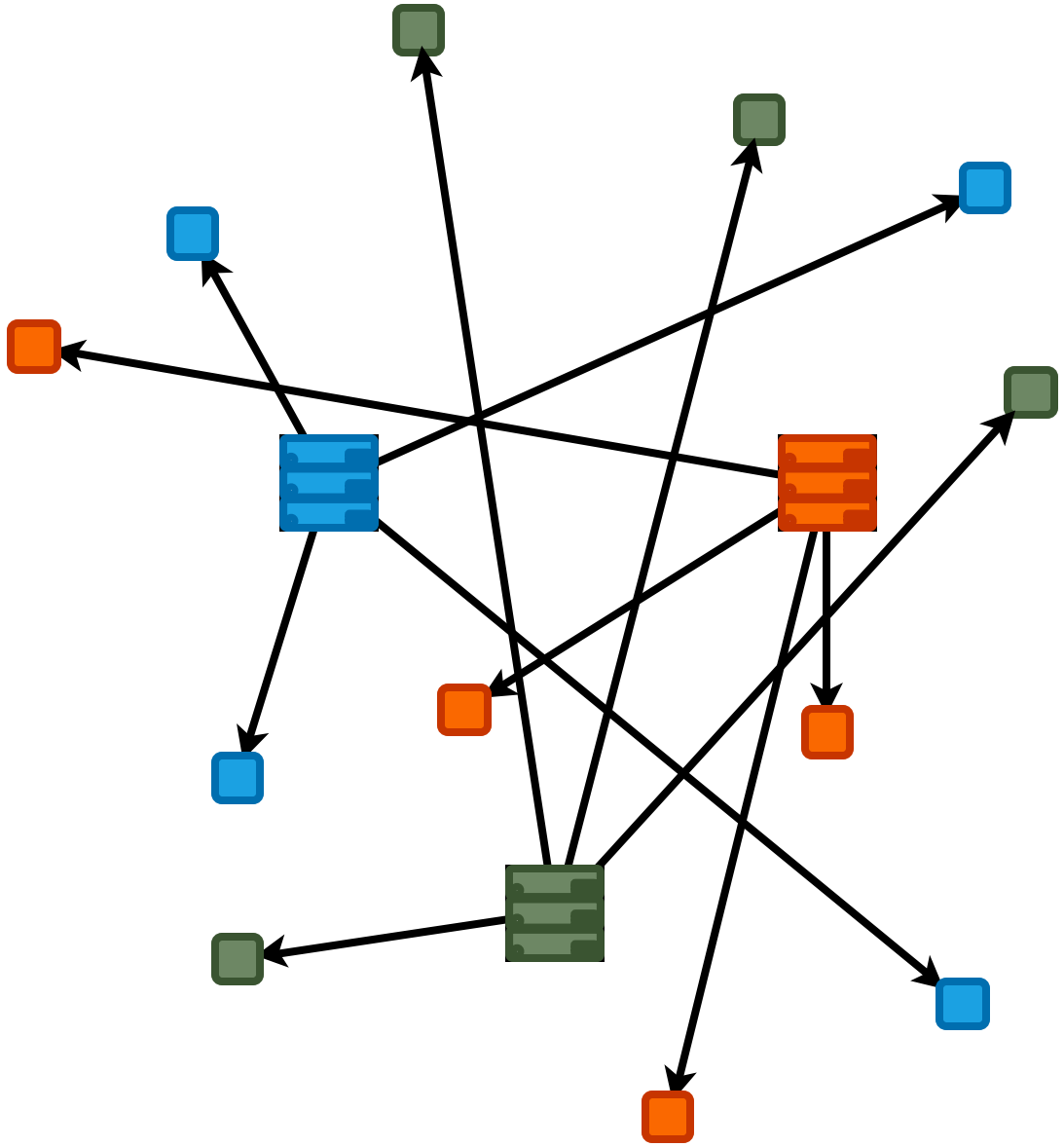
RCA baseline

- Clients are randomly selected and moved to a different random server
- A server with 2 or fewer clients cannot be selected
  - We needs some clients assigned to a server to keep the system running



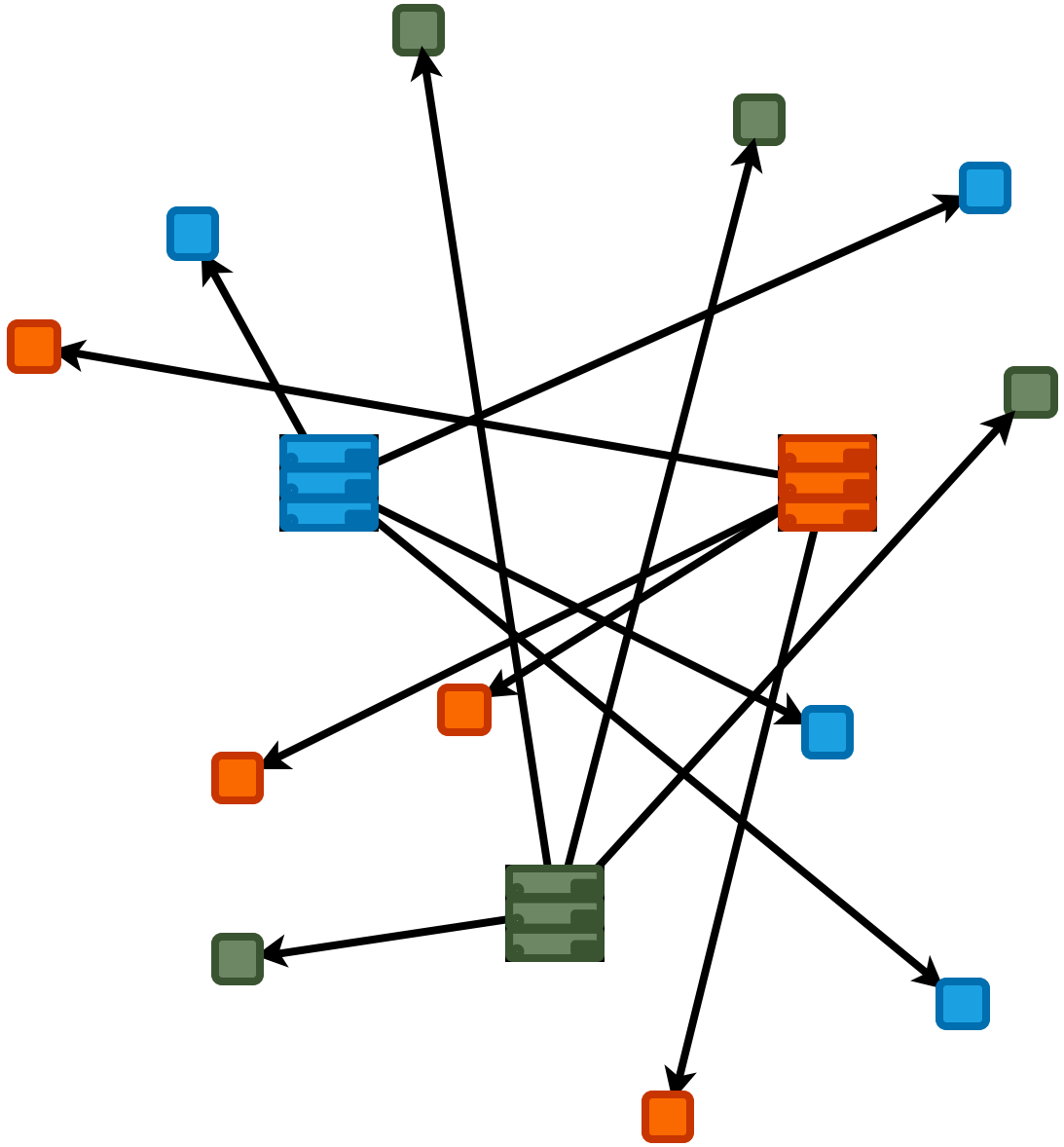
RCA baseline

- Clients are randomly selected and moved to a different random server
- A server with 2 or fewer clients cannot be selected
  - We needs some clients assigned to a server to keep the system running



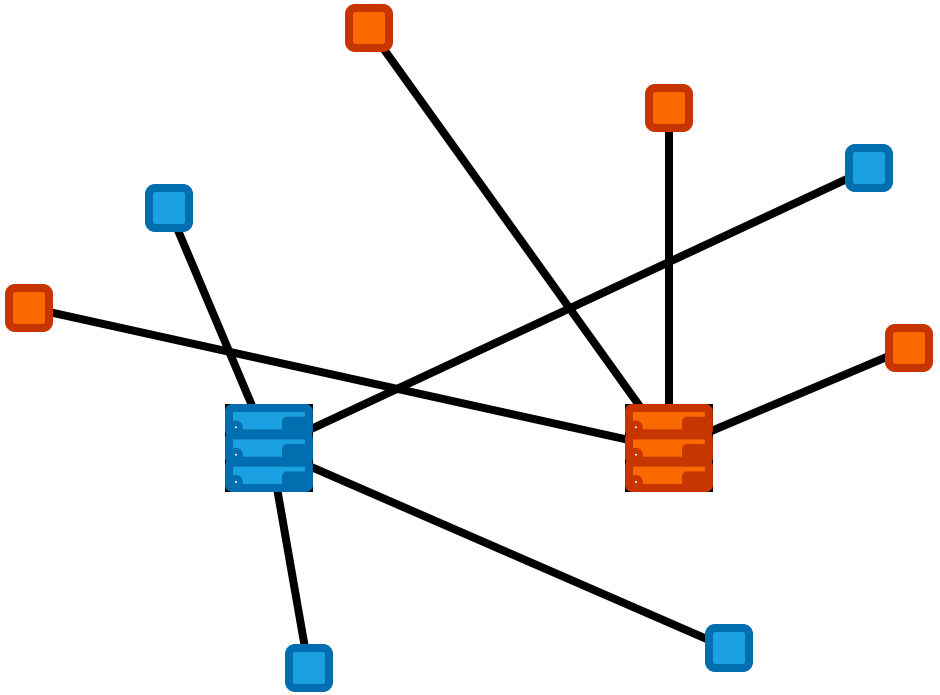
RCA baseline

- Clients are randomly selected and moved to a different random server
- A server with 2 or fewer clients cannot be selected
  - We needs some clients assigned to a server to keep the system running



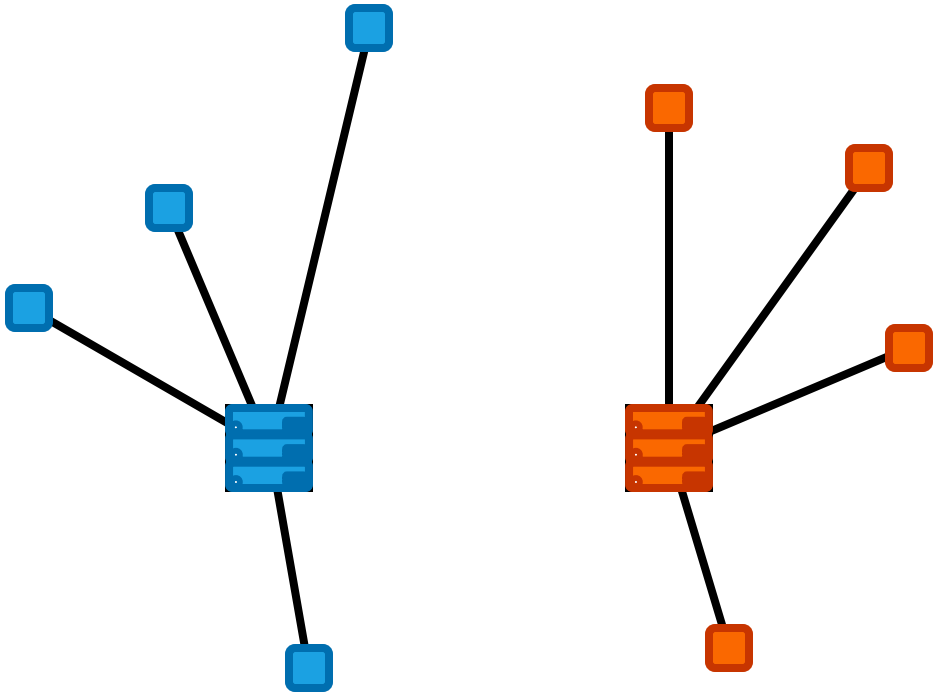
LACA baseline

- Clients are ranked based on the client to server latency
- The optimization goal is to minimize the overall latency in the system
- A server with 2 or fewer clients cannot be selected
  - We needs some clients assigned to a server to keep the system running



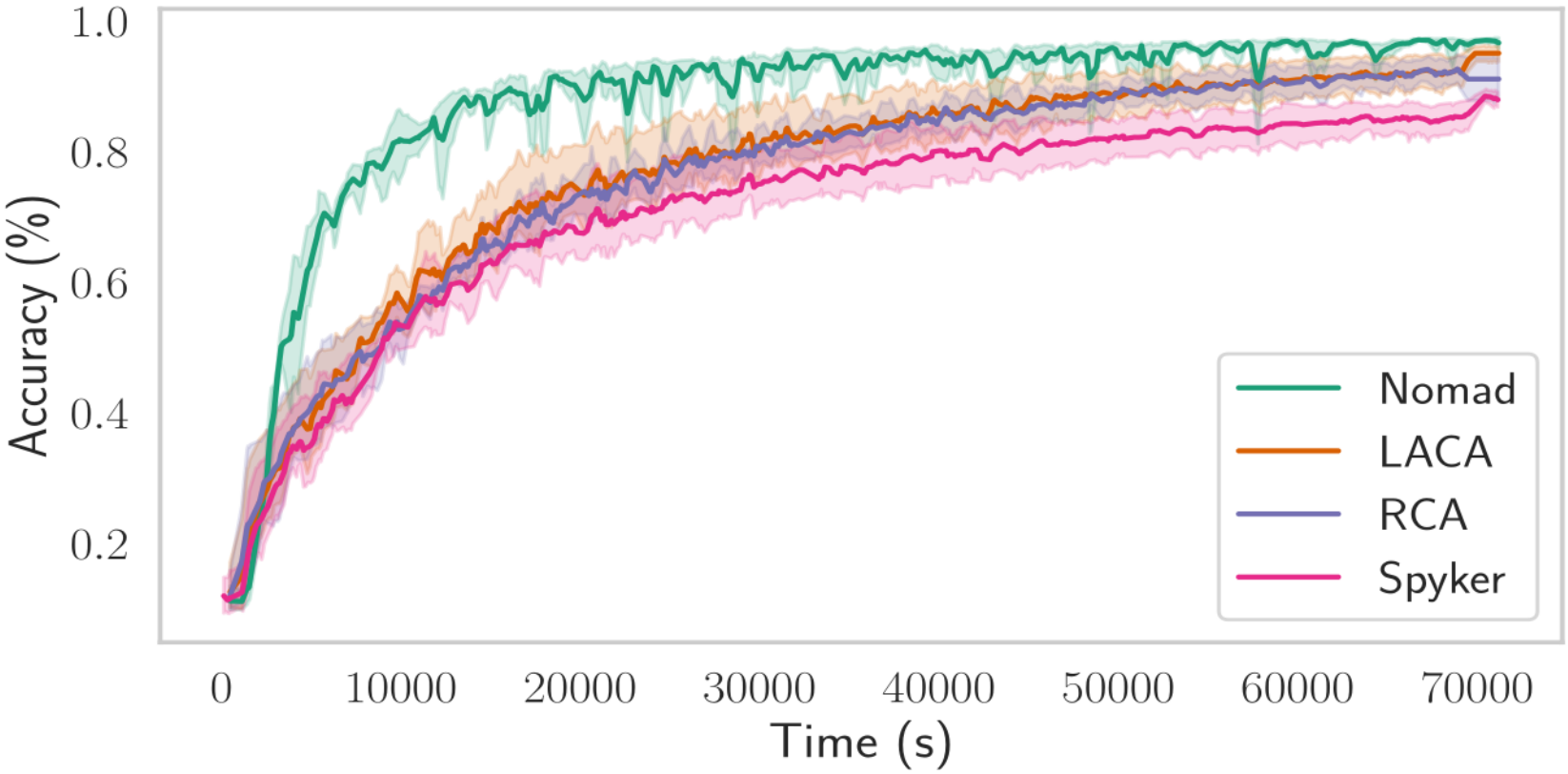
LACA baseline

- Clients are ranked based on the client to server latency
- The optimization goal is to minimize the overall latency in the system
- A server with 2 or fewer clients cannot be selected
  - We needs some clients assigned to a server to keep the system running



Evaluation

- Dataset: MNIST
- 4 servers & 100 clients



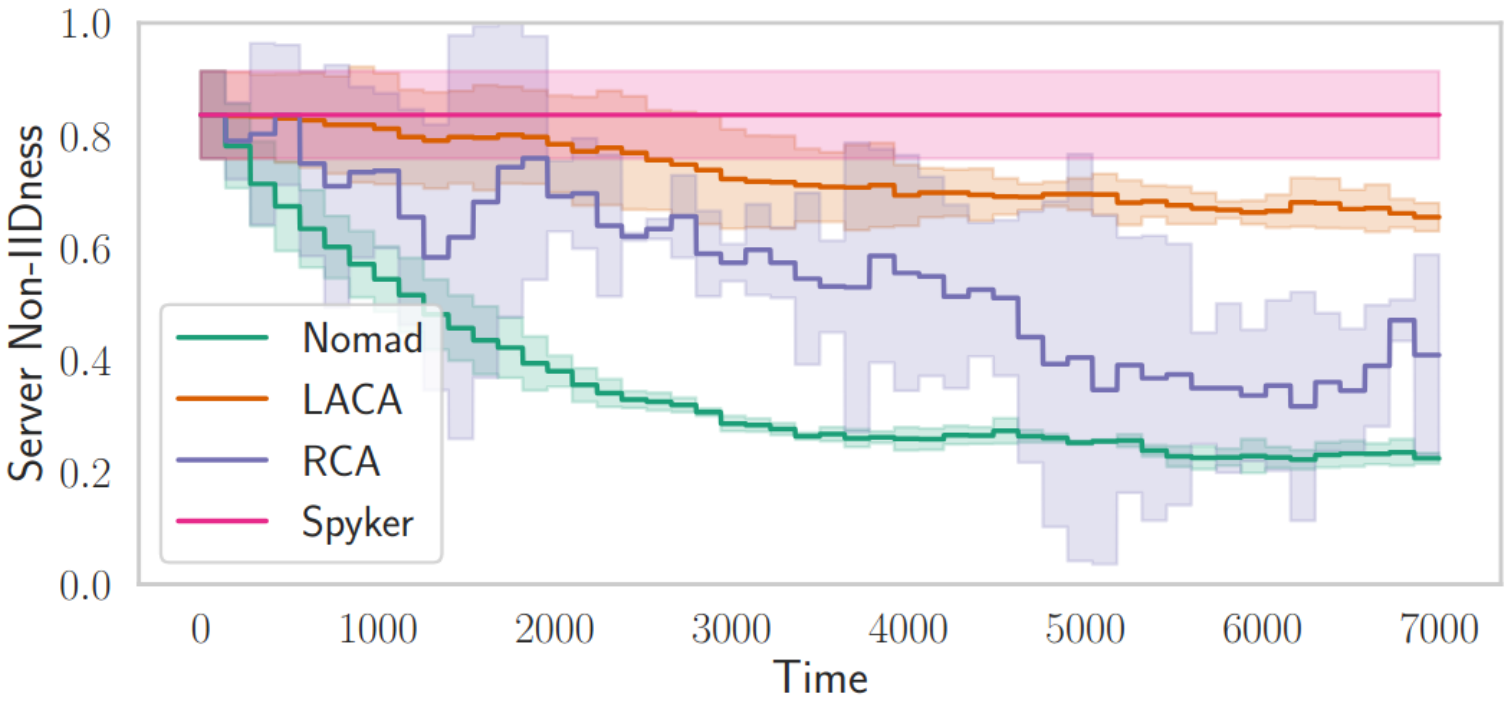
Evaluation

Algorithm	MNIST			Cifar-10		
	$N = 10$	$N = 40$	$N = 100$	$N = 10$	$N = 40$	$N = 100$
Spyker	0.969	0.901	0.871	0.711	0.684	0.642
LACA	0.979	0.966	0.950	0.762	0.713	0.674
RCA	**0.980**	0.937	0.910	**0.776**	0.728	0.709
Nomad (this work)	0.974	**0.971**	**0.969**	0.773	**0.771**	**0.758**

- Overall Nomad outperforms the baselines.
- Noticeable: RCA works well with a low number of clients

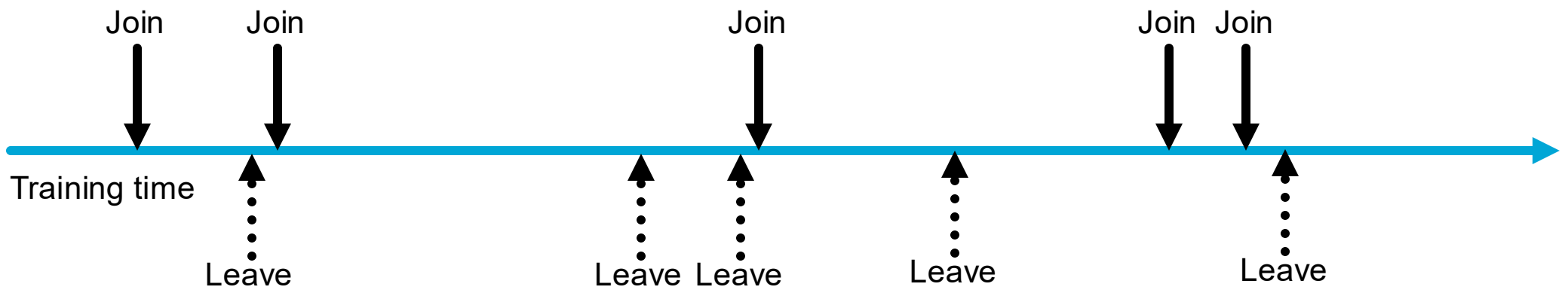
Evaluation

- We track the average Server non-IIDness across training
- LACA is not able the improve significantly
- RCA makes sub-optimal transfers
- Nomad reduces the server non-IIDness



Churn

- What happens when clients are transient?
- Clients can join or leave
- Consider following scenarios:
  - Only joining clients
  - Joining and leaving clients



Churn

- What happens when clients are transient?
- Clients can join or leave
- Consider following scenarios:
  - Only joining clients
  - Joining and leaving clients

Algorithm	Join		Leave & Join	
	50 Events	100 Events	100 Events	200 Events
Spyker	0.533	0.491	0.685	0.681
LACA	0.832	0.792	0.847	0.845
RCA	0.799	0.767	0.829	0.809
Nomad (this work)	**0.851**	**0.827**	**0.873**	**0.855**

Nomad

- Multi-Server Federated Learning
- Dynamically allocate clients to the best server
  - System properties
  - Data properties
- Handle client churn

bacox/Nomad