Client Selection Approach in Support of Clustered Federated Learning over Wireless Edge Networks

- **Authors: Abdullatif Albaseer, Mohamed Abdallah, Ala Al-Fuqaha, and Aiman Erbad**
- **GLOBECOM 2021 – 2021 IEEE Global Communications Conference**

23CSE362 - EDGE COMPUTING
REVIEW - 1

Team: 23_XpertEdge

- CB.SC.U4CSE23432 : MOHANA KRISHNAN M V
- CB.SC.U4CSE23407 : AMAL GODWIN J
- CB.SC.U4CSE23545 : S DEEPAN
- CB.SC.U4CSE23546 : SAURAV GOPINATH E K

INTRODUCTION

Today’s applications—from smartphones to medical sensors—depend on smart, connected devices.

These devices generate valuable data but must balance **real-time performance** with **privacy**.

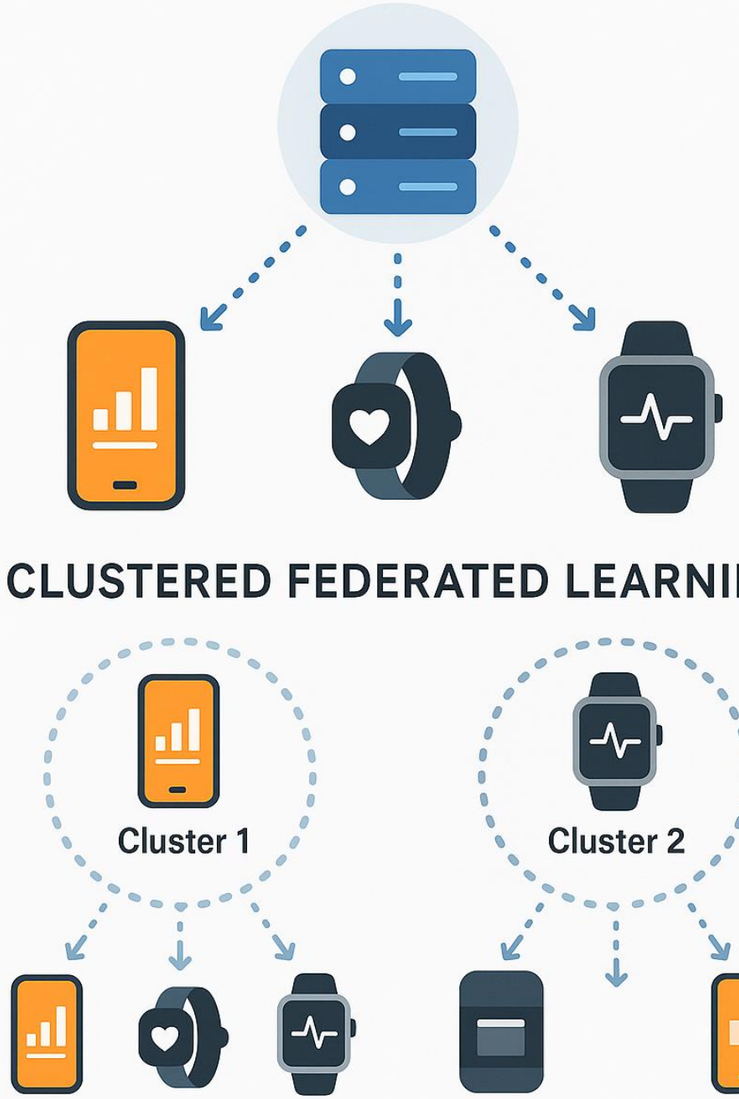
What Is Federated Learning (FL)?

- A decentralized AI paradigm where each device trains **locally** on its own data.
- Devices share **only model updates** (no raw data) with a central server.
- Key benefits:
  - **Privacy preservation**
  - **Bandwidth efficiency**
  - **Scalable edge compute**

Why Clustered Federated Learning (CFL)?

- In Reality, device data is **non-IID**—each user’s data distribution differs.
- A single global model may **underperform** on diverse datasets.
- CFL addresses this by:
  - **Grouping devices** based on the **similarity of their updates**
  - Training **specialized models** for each cluster
  - Delivering **better personalization** and **higher accuracy**

WHAT IS FEDERATED LEARNING?



Problem Statement: Challenges in Federated Edge Edge Learning

Heterogeneous Edge Devices

Varying CPU speeds (1–9 GHz), transmit powers, and channel qualities create unpredictable performance across devices.

Non-IID Data Distribution

Each device holds its own private, often imbalanced, data distribution, hindering global model convergence and fairness.

Resource Constraints

Limited bandwidth restricts communication to a subset of of devices, compounded by strict latency deadlines per learning learning round.

Privacy Requirement

Raw data must remain on device; only model updates are shared, are shared, necessitating robust privacy-preserving mechanisms. mechanisms.

Why It's an Edge Computing Problem

Data Locality & Privacy

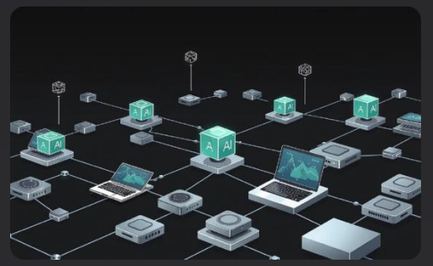
Edge devices generate sensitive data, and regulations (GDPR, HIPAA) forbid uploading raw data to the cloud.

Heterogeneous Hardware & Network

Devices range from low-power sensors to powerful smartphones, with varying network conditions (Wi-Fi, cellular, intermittent connectivity).

Limited Resources & Real-Time Needs

Constrained bandwidth means only some devices can synchronise per round, crucial for latency-sensitive applications like autonomous drones.



Decentralised Intelligence

Offloading compute to the edge reduces reduces cloud load and mitigates central central points of failure, enabling scalable scalable and resilient AI at the network’s network’s periphery.

Proposed Solution: Optimising Federated Learning at the Edge the Edge

1

Greedy Scheduling for Stable Clusters

Once a cluster converges, schedule only its fastest devices for further updates to accelerate fine-tuning without sacrificing sacrificing accuracy.

2

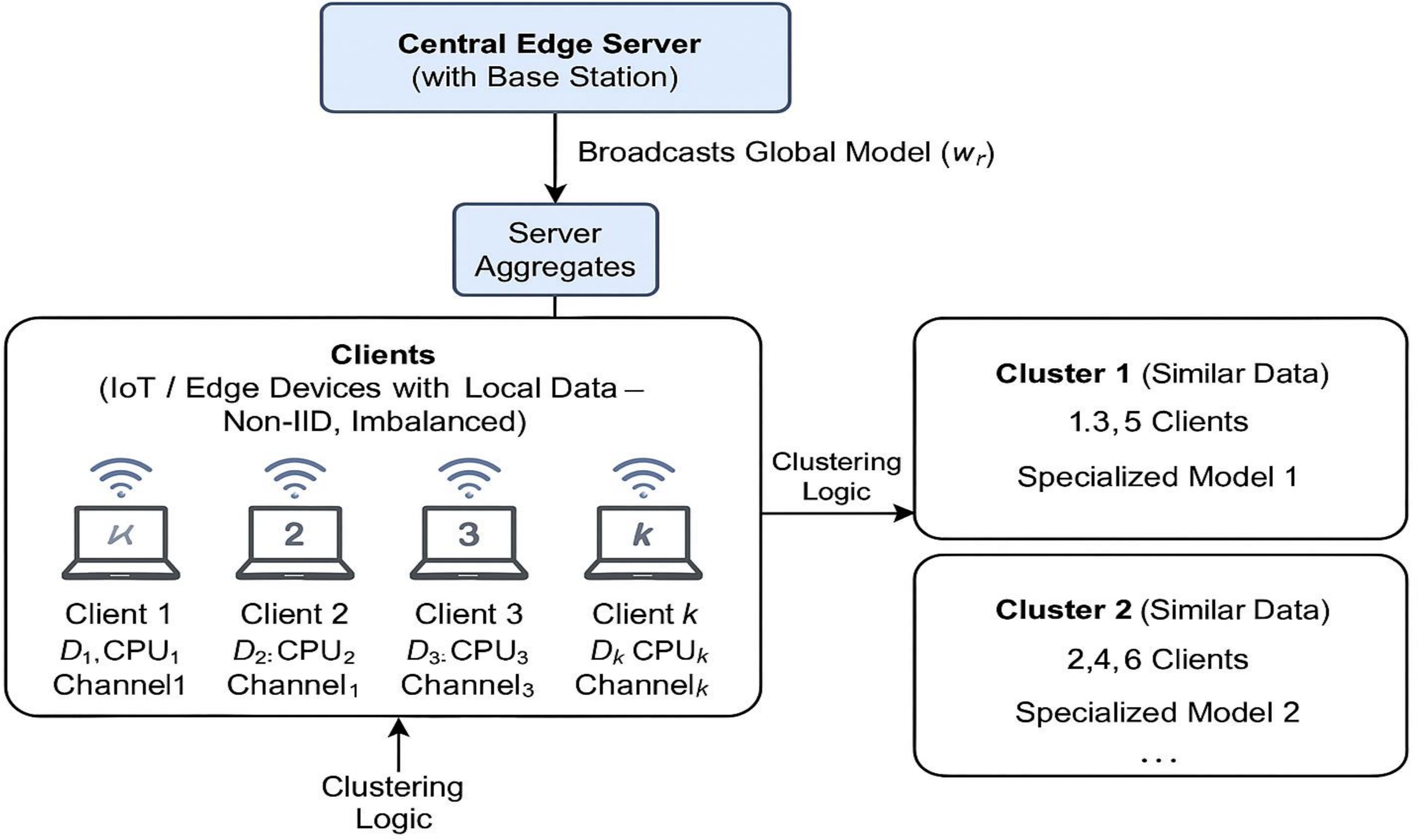
Clustered Federated Learning (CFL)

After each aggregation, compute cosine similarity of client updates. If the global model nears stationarity but client updates diverge, split into two clusters and train separate models per cluster for better personalisation.

3

Latency-Aware Client Selection

Estimate each device’s compute and communication time. Sort devices by devices by ascending latency and select the fastest while rotating in slower in slower ones for fairness.


Central Edge Server
(with Base Station)
Broadcasts Global Model (w_r)
Server
Aggregates
Clients
(IoT / Edge Devices with Local Data –
Non-IID, Imbalanced)
Client 1
D_1, CPU$_1$
Channel1
Client 2
D_2: CPU$_2$
Channel$_1$
Client 3
D_3: CPU$_3$
Channel$_3$
Client k
D_k CPU$_k$
Channel$_k$
Clustering
Logic
Clustering
Logic
Cluster 1 (Similar Data)
1.3, 5 Clients
Specialized Model 1
Cluster 2 (Similar Data)
2,4, 6 Clients
Specialized Model 2
...

Implementation Strategy

How is it going to be implemented?

It is going to be implemented using SimPy, a Python-based discrete-event simulation simulation library. SimPy models processes (such as clients), resources (like bandwidth slots), and events (such as training and uploading), providing fine-grained fine-grained control over timing, queuing, and concurrency.

Dataset Integration

We use FEMNIST dataset partitions for non-IID client data. The **FEMNIST** dataset is a dataset is a real-world federated dataset consisting of handwritten characters. It characters. It consists of 244,154 training and 61,500 testing images.

Key Implementation Algorithm at Edge

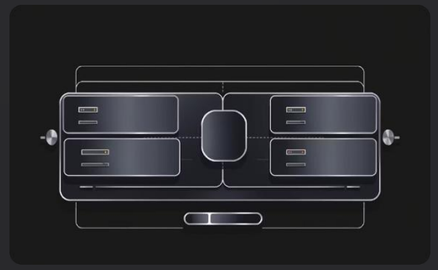
Algorithm Used: The authors designed a **Client Selection Strategy for CFL** which includes:

- Estimating client **latency** (communication + computation).
- Sorting clients by **expected latency**.
- Dividing clients into **clusters** based on gradient similarity using **cosine cosine similarity**.
- Running rounds of FL where:
  - Clustered clients with similar data distributions are grouped.
  - **Greedy scheduling** is used to select clients with **lower latency** from stationary clusters.
  - Clients train locally using **Stochastic Gradient Descent (SGD)**.
  - The server aggregates updates using an efficient scheduling method (Equation 7).

State of the Art: Bridging the Gap in Edge FL

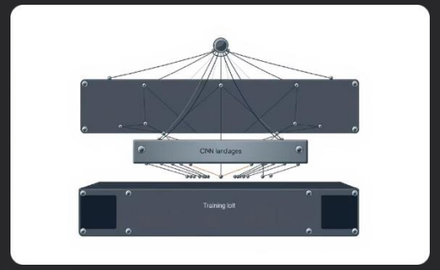
Paper	What They Do	Key Constraints	Advancement in This Paper
FedAvg(McMahan et al., 2017)	Aggregates all client updates into a single global model.	Assumes IID data; involves all clients each round; ignores device/network heterogeneity.	Introduces latency-aware client scheduling and clustered models to handle non-IID data and edge constraints.
FedCS(Nishio & Yonetani, 2019)	Schedules fastest devices to meet per-round latency deadlines.	Trains only one global model; does not address data heterogeneity or personalization.	Adds CFL alongside latency-based selection, yielding specialized models per data-homogeneous cluster.
Clustered FL(Sattler et al., 2020)	Splits clients into clusters via cosine similarity of gradients; trains separate models.	Requires all clients each round; ignores device compute/network constraints.	Integrates heterogeneity-aware scheduling so clustering occurs efficiently, respecting edge resource limits.

Implementation Flow: Phases of Development



Phase 1: Data & Client Setup

Load and normalize FEMNIST dataset, partitioning non-IID data (2 classes per client). Simulate diverse client profiles including CPU frequency, distance, and bandwidth.

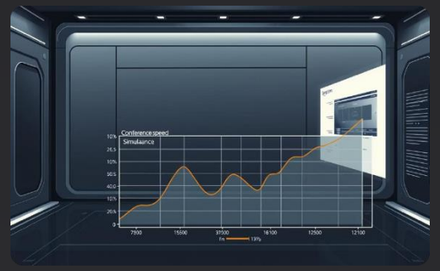


Phase 2: Model & Local Training

Define the CNN architecture (2 convolutional + 2 fully connected layers) and implement `train_local()` to compute weight deltas. SimPy accurately models on-device computation delays.

Phase 3: Client Selection & Clustering Clustering

Clients are selected based on estimated latency, prioritizing the fastest while rotating rotating slower ones for fairness. After FedAvg, if the global model diverges from from client updates, use cosine similarity to to split clients into bi-partitions.



Phase 4: Orchestration & Evaluation

Integrate all phases within a SimPy loop over over 'R' rounds. Log key metrics such as accuracy per cluster, convergence speed, and speed, and overall round latency to generate generate performance plots and detailed detailed tables.

Team Task Split-Up: Edge Modules

Edge Module	Team Member	Key Responsibilities
Client & Resource Simulation	Deepan	Load & preprocess FEMNIST shards; simulate CPU frequency, distance, and bandwidth; implement a latency estimator (compute + communication).
Latency-Aware Scheduling	Amal Godwin J	Design and code select_clients() by latency; implement fairness rotation of slow clients; unit-test scheduling under SimPy.
Cluster Management (CFL)	Mohana Krishnan	Implement split-criteria code cosine-similarity bi-partition; maintain a cluster models list.
Orchestration & Metrics	Saurav Gopinath	Build SimPy federated loop scheduler; collect & log round times and accuracy per cluster; generate convergence and fairness plots.

Summary & Q&A

Problem

Edge devices with non-IID data, limited compute & bandwidth need personalised models without sharing raw data.

Edge Focus

Data stays local, heterogeneity in CPU/network and strict latency & privacy constraints are paramount.

Solution

Latency-aware client scheduling + Clustered Federated Learning (CFL) leads to specialised models per client group.

Implementation

A SimPy-based discrete-event simulation with FEMNIST dataset, dataset, comprising four edge-focused modules.