universidade de aveiro
theoria poiesis praxis

DEPARTAMENTO DE ELECTRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA
MESTRADO EM ENG. DE COMPUTADORES E TELEMÁTICA
ANO 2023/2024

REDES E SISTEMAS AUTÓNOMOS

AUTONOMOUS NETWORKS AND SYSTEMS

PRACTICAL GUIDE 3 – FEDERATED LEARNING

Objectives

- Set up a Federated Learning (FL) cluster
- Use MobFedLS based on Flower to set up the FL cluster
- Perform the training in the clients and aggregation of model in the server
- Communication between clients and server is performed through WiFi ad-hoc network (batman)
- Observe the logs of the clients and the server to check the results of the federated learning training

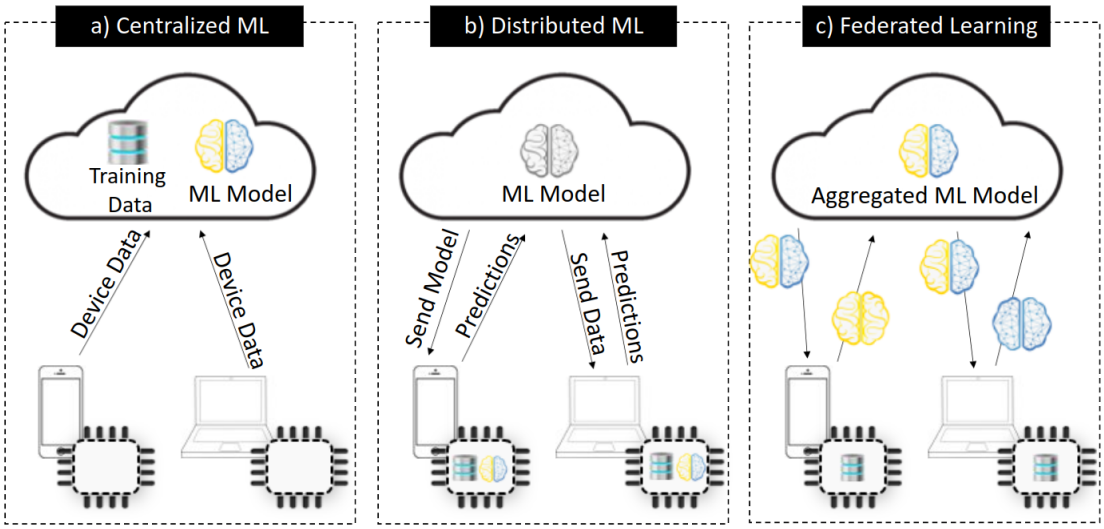
Duration

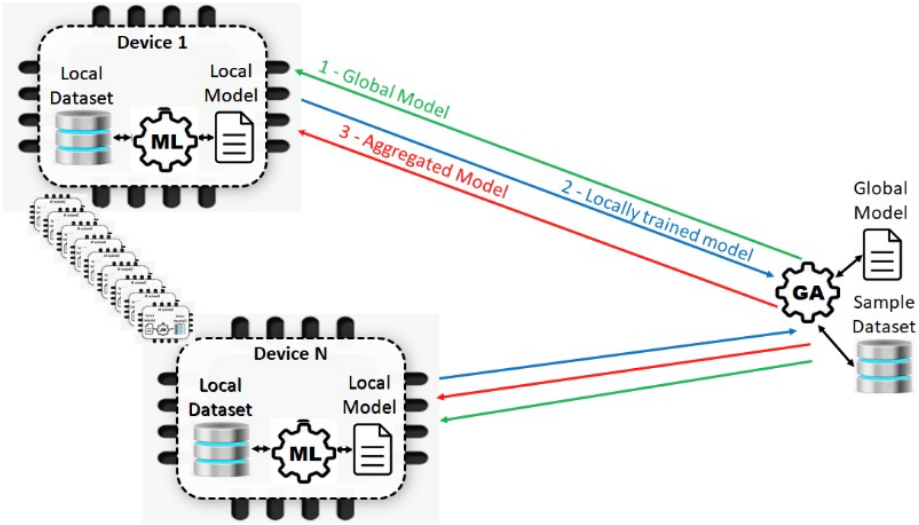
2 weeks

1st week

Introduction

- In order to train a universal model that can be distributed to all edge devices, traditional machine learning for IoT is typically done by uploading all data from each connected device to the cloud, where the entire training process and data prediction is done (**Centralized ML**).
- Another alternative is **Distributed ML** which is a multi-node ML system that builds training models by independent training on different nodes.
- **Federated Learning (FL)** is a machine learning technique that uses local datasets across multiple decentralized edge devices to train an algorithm collaboratively without exchanging data.





In this guide, we will work with a framework built to operate several clients and a server in a Federated Learning scheme – the MobFedLS, based on Flower (https://www.mdpi.com/2076-3417/13/4/2329)

1. Prepare the cluster

1.1. Like in the previous guide, we will form at least 4 batman networks in the classroom, each with 4 nodes (one Raspberry node per student).

1.2. Modify the batman script with the same settings as the other 3 students in your group:

```
vi batman_installation_master/create_batman_interface.sh
```

1.3. Then run the script:

```
./create_batman_interface.sh wlan0 10.1.1.id/24
```

1.4. Validate that all the nodes have connectivity, simply with the `ping` command.

2. Prepare Visual Studio Code

2.1. Install VSCode if you don't have it yet.

2.2. Install the extension Remote SSH

2.3. Install the extension Remote Explorer

2.4. Use Remote Explorer to connect to the Raspberry using ssh and the same credentials.

2.5. Follow the steps until you reach the terminal and validate you are interacting with the Raspberry through VSCode

3. Launch the MobFedLS through VSCode

3.1. Follow these steps:

```
cd MobFedLS
touch .env
```

3.2. In VSCode edit the .env file, where **hostname** goes should be your board hostname, which is raspberrypi-**id**, replace id with the id of your board, and in datasetX.csv, replace X with your assigned student number [1,4]:

```
MACHINE_ID=Hostname
PASSWORD=openlab
SERVER_IMG=mfl-server
CLIENT_IMG=mfl-ghostclient
DATASET_FILE=datasetX.csv
ALGORITHM=FedAvg
```

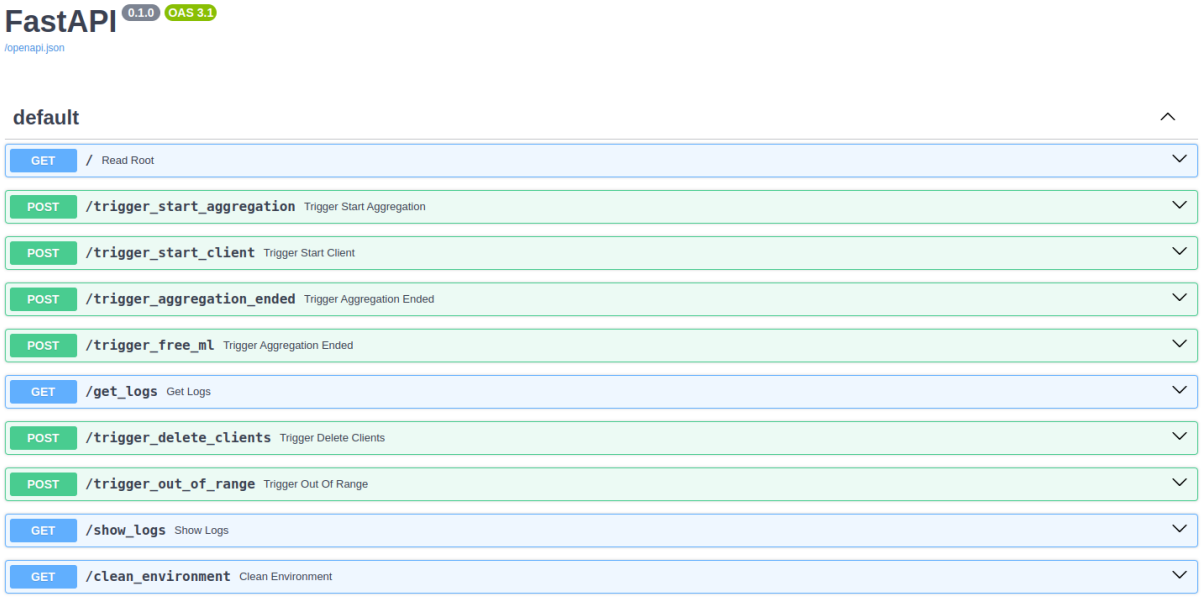
3.3. Now launch the base infrastructure containers of the framework, which are the manager, around, MLapp, with:

```
docker compose -f base-docker-compose.yaml up -d
```

3.4. Now open the browser in the following link, replace id with the id of your board:
http://192.168.3.id:5001/docs - MLapp
http://192.168.3.id:5101/docs - Manager

You should be able to see the following environment. We will explore the usage of several API endpoints to operate the MobFedLS (description of each one in the Appendix).

MLapp API



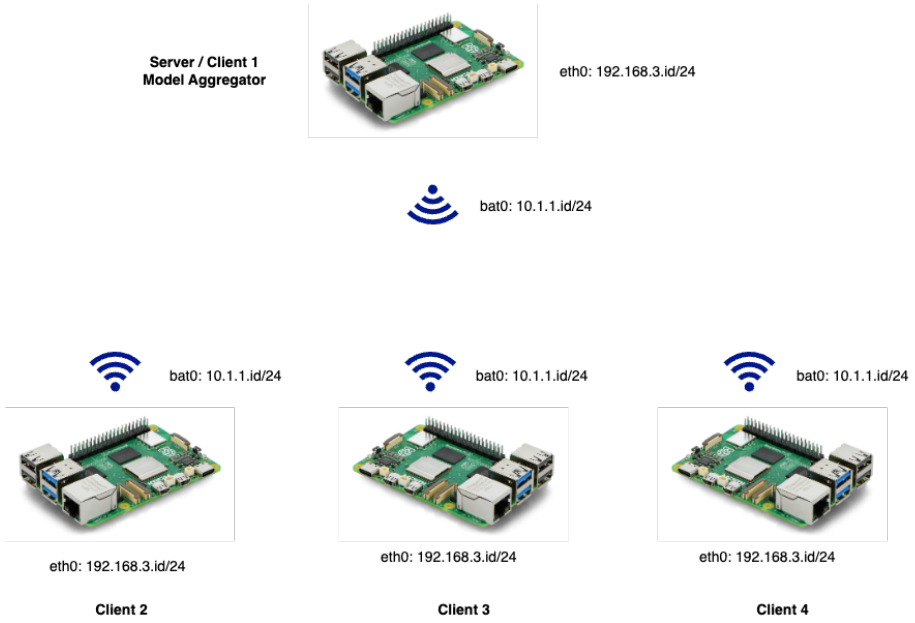
Manager API

4. Clean the environment (Optional, if needed)

If there is any container hanging with an exit code, instead of running, the environment must be cleaned. In order to do that run `/clean_environment` endpoint in the Manager API. This must be done in all nodes that part of the cluster.

5. Mount the first cluster of Federated Learning

5.1. First, discuss with the other groups, who will be the server and who will be the clients, to form the following the cluster.



5.2. Now for each node to know about the others, you'll have to:

5.2.1. Create a neighbors file with:

```
touch findNeighbours/neighbours_lists/neighbours_file.json
```

5.2.2. Modify it in VSCode with following lines

```
{
    "0": "10.1.1.α:5101",
    "1": "10.1.1.α:5101",
    "2": "10.1.1.β:5101",
    "3": "10.1.1.λ:5101",
    "4": "10.1.1.τ:5101"
}
```

Where each line has the ip of the batman network of each node, and:

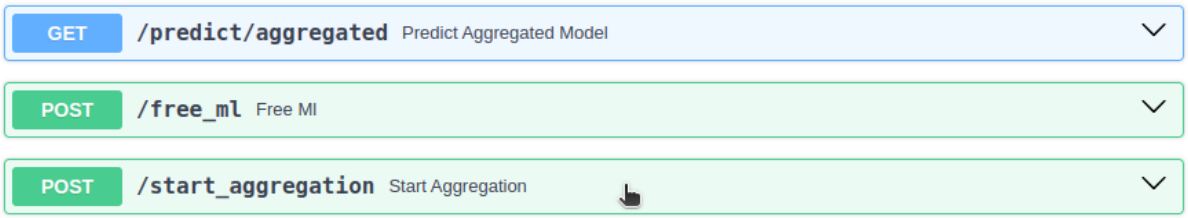
α corresponds to the id of the board from student number 1, that gets to be the server node and it also has a client.

β corresponds to the board from student number 2.

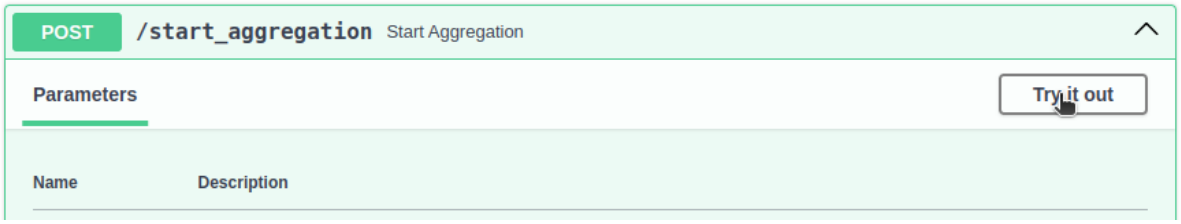
λ corresponds to the board from student number 3.

τ corresponds to the board from student number 4.

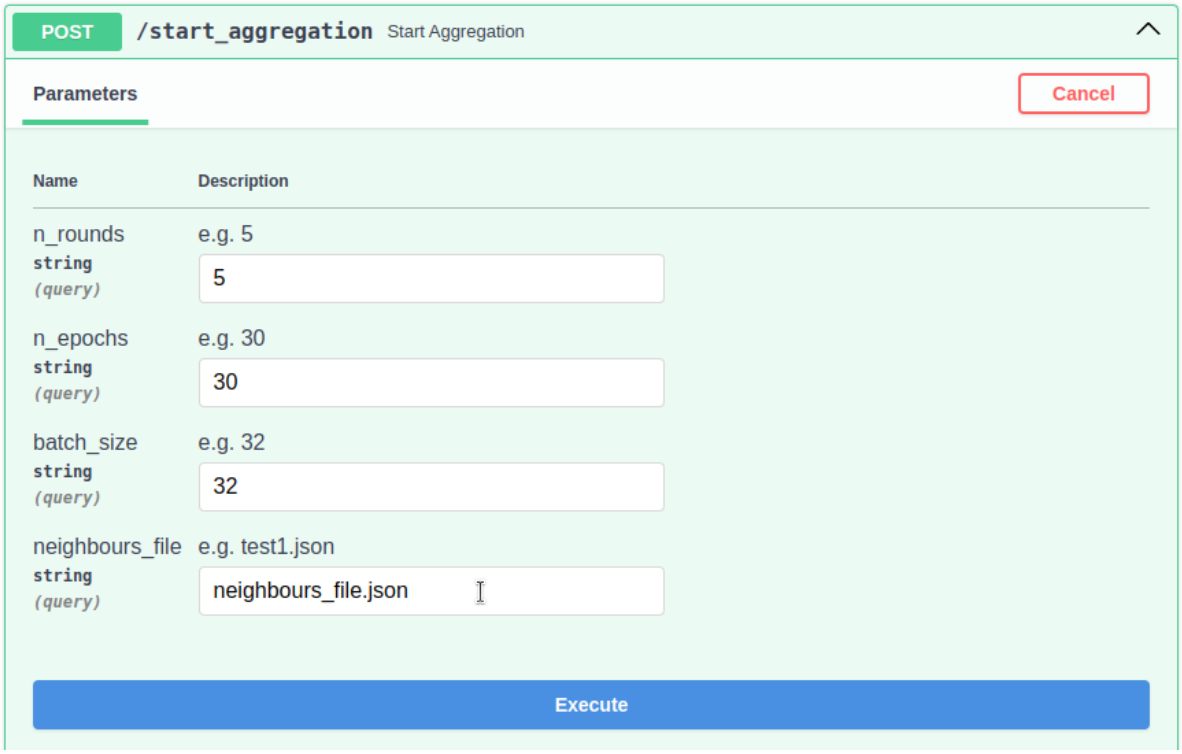
5.3 In order to start the aggregation, the student with the server node must use the endpoint `/start_aggregation` (MLapp API):



Press the try it out button:



In the neighbours file field modify it to the file created previously:

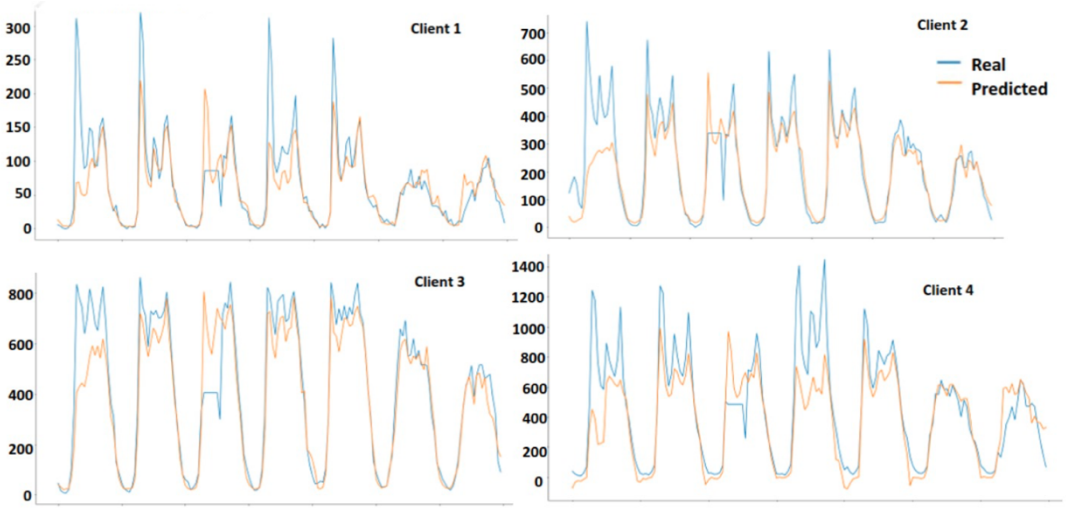


In a new terminal window, run the following:

```
sudo watch -n 1 docker ps
```

Finally, in the MLapp web app, press execute.

6. **Explore the federated learning process**
    - When aggregation is started, and with all the nodes connected, the MLapp in the server node makes a request to the Manager also in the server node, the former now makes use of the neighbors file and makes a request for each node to start the client through their Managers. Then, the Federated Learning process happens between the now created server and clients.
    - The MobFedLS by default has included a dataset of mobility of the city of Aveiro.
    - Each client is an edge node located in a different place of the city, gathering different data about the flow of vehicles (counting vehicles through radar sensors).



S1 - Regular Dataset- Forecast of entries into Aveiro in week 2022-05-16

- The goal for the model is to predict the number of vehicles in certain area of the city at certain time of the day. Each client trains the model with its local data.
- In the end, all the individual models are aggregated in the server, and the aggregated model is returned to the clients, a scheme that is summarized in the following figure:

Explore the logs in the server node through the endpoint `/show_logs` in the Manager API. Check that the training was done in 5 rounds (this was set in the creation of the server, 5 was the default value for **n_rounds**).

7. **Repeate the federated learning process with different characteristics (repeat 5.3 steps)**

    To change the characteristics of the training (number of rounds, number of clients, number of epochs, etc), you need to modify the parameters in start aggregation and when executed the server and the clients will be created with the new parameters.

Appendix

Manager		
HTTP Method	**Endpoint**	**Explanation**
POST	/trigger_start_aggregation	This method is called by the ML-App when the respective MFL-Interface decides that wants to aggregate
POST	/trigger_start_client	This method is used to start a MFL-GhostClient in a Mobile Clients
POST	/trigger_aggregation_ended	This method is called by the MFL-Server to signal the Maestro's MFL-Manager that the FL process has ended
POST	/trigger_free_ml	This method is used to signal the Mobiles MFL-Manager Mobiles to free the ML-App
GET	/get_logs	This method is used to signal the Mobiles MFL-Manager to retrieve the logs of the MFL-GhostClient
POST	/trigger_delete_clients	This method is used to signal the Mobiles MFL-Manager to delete the MFL-GhostClient
POST	/trigger_out_of_range	This method is used by the MFL-GhostClient reaches a timeout without a connection from the MFL-Server
GET	/show_logs	This method is used to see the logs of previous FL runs
GET	/clean_environment	This method is used to clean the infraestructure at any time, if there is a MFL-Server or a MFL-GhostClient stopped with an error

ML-App		
HTTP Method	**Endpoint**	**Explanation**
GET	/get_data	This method is used by the MFL-GhostClient in order to prepare the dataset of the ML-App when the client is starting
GET	/get_parameters	This method is used by to get the current parameters of the ML-App at any point
POST	/set_parameters	This method is used by to set the current parameters on the ML-App at any point
GET	/fit	This method is used in the FL process to train the ML-App
GET	/evaluate	This method is used in the FL process to evaluate sets of parameters on the ML-App
GET	/predict/local	This method is used to predict using the set of parameters before the FL process
GET	/predict/aggregated	This method is used to predict using the set of parameters after the FL process
POST	/free_ml	This method is used by the MFL-Manager to signal that the ML-App can be free
POST	/start_aggregation	This method is used by the ML-App when it decides that wants to start an aggregation