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

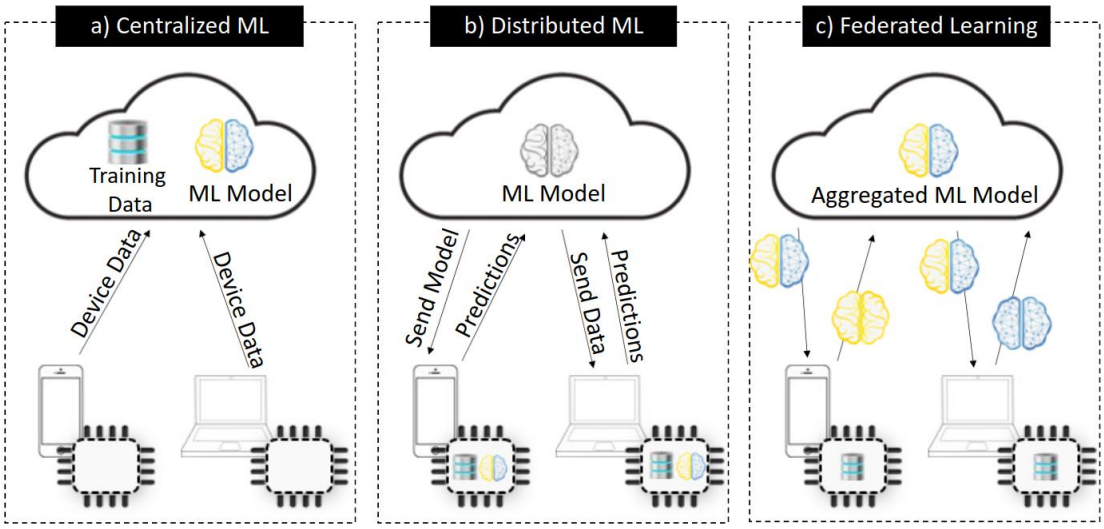
Duration

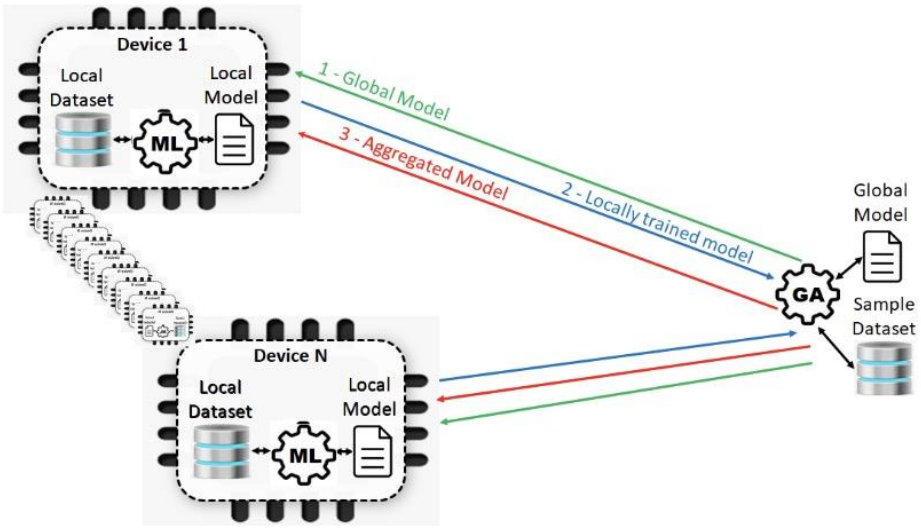
2 weeks

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. Launch the MobFedLS

3.1. Follow these steps:

```
cd MobFedLS
vim .env
```

3.2. Edit the .env file, where **hostname** should be raspberrypi-7**id (**replace id with the id of your board), and in configX.csv, replace **X** with your assigned student number [1,4]:

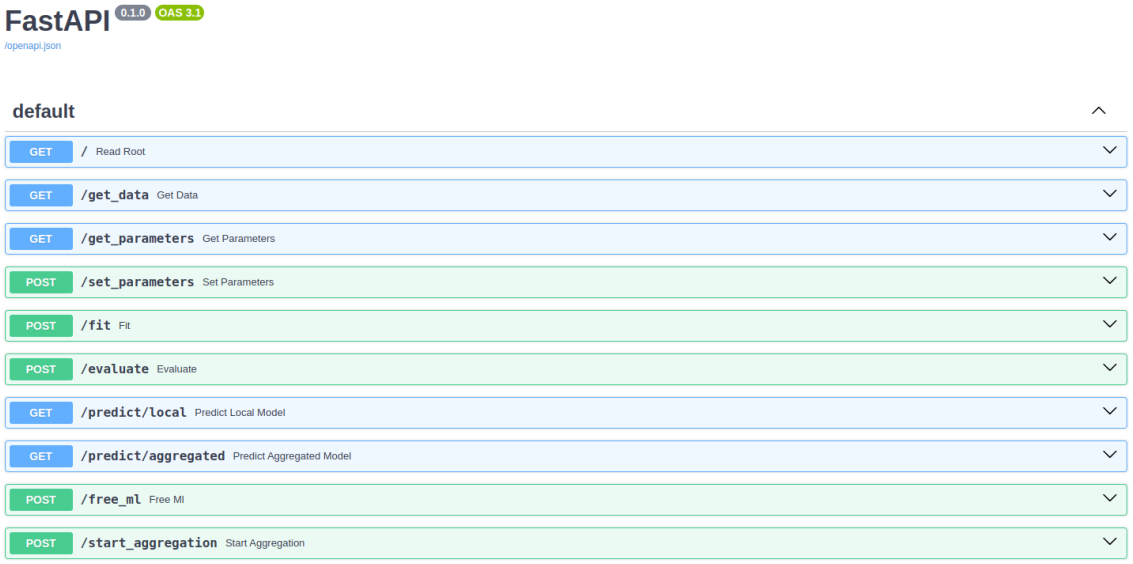
```
MACHINE_ID=hostname
PASSWORD=openlab
ML_APP_IMG=ml-app-mnist
SERVER_IMG=mfl-server
CLIENT_IMG=mfl-ghostclient
ML_BASE_DIR=clientsML_mnist
DATASET_FILE=configX.csv
```

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

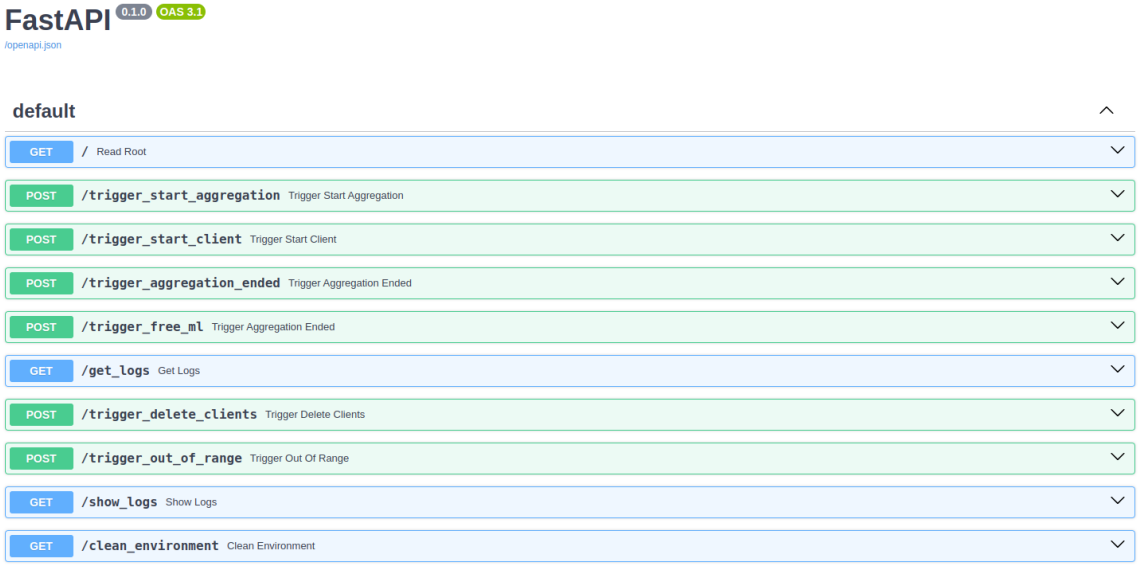
```
docker compose -f base-docker-compose.yaml up -d
```

3.4. Now open the browser in the following link, replacing **id** according to your board:
http://192.168.3.id:5001/docs - MLapp API
http://192.168.3.id:5101/docs - Manager API

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



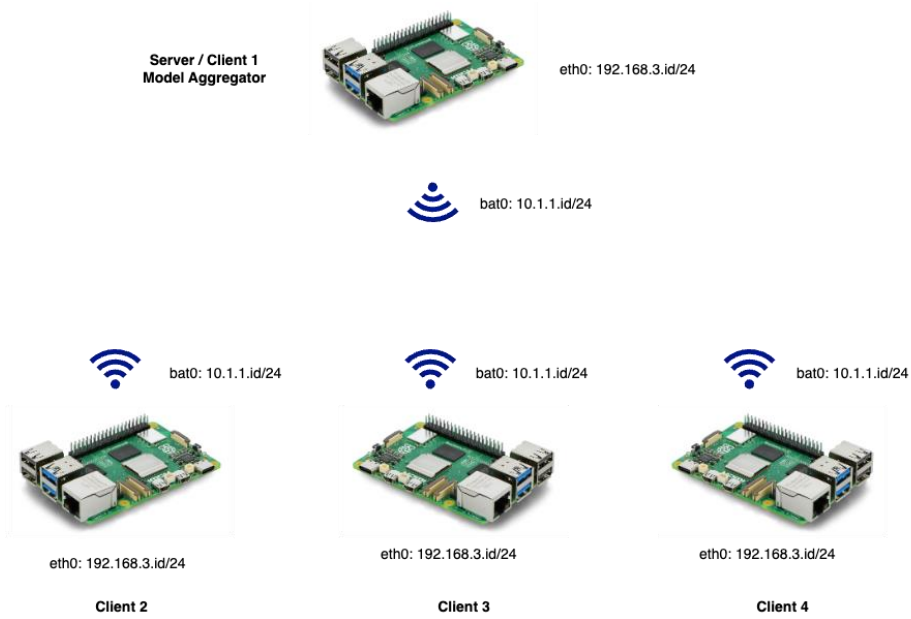
Manager API

3. 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 are part of the cluster.

4. Mount the first cluster of Federated Learning

4.1. First, discuss with the other students of your group, who will be the manager, to form the following the cluster.



4.2. Modify the neighbours file `findNeighbours/neighbours_lists/neighbours_file.json`:

```
{
    "0": "10.1.1.id_rpi_student1:5101",
    "1": "10.1.1.id_rpi_student1:5101",
    "2": "10.1.1.id_rpi_student2:5101",
    "3": "10.1.1.id_rpi_student3:5101",
    "4": "10.1.1.id_rpi_student4:5101"
}
```

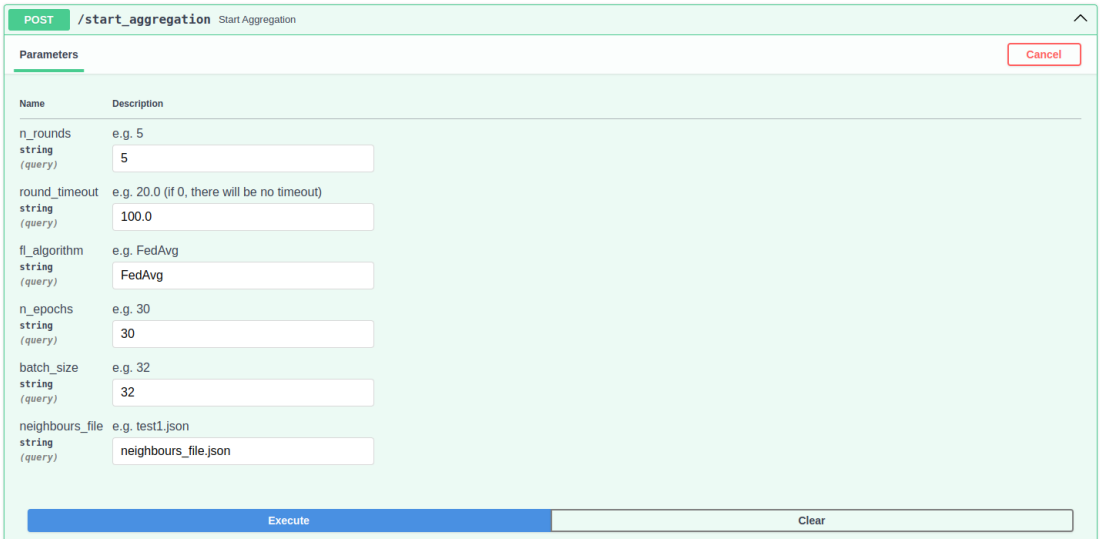
Where each line has the ip of the batman network of each node of the cluster.

4.3. To initialise the parameters according to the dataset of each client, use the endpoint `/get_data` and then the endpoint `/fit` to do a previous training of the model.

4.4. In order to start the aggregation, the **manager board** must use the endpoint `/start_aggregation` (MLapp API):

Fill the following fields with correct information:
neighbours_file: neighbours_file.json
round_timeout: 100.0

Leave the other fields with the defaut information.



In parallel, in a new terminal window, run the following:

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

Finally, in the MLapp web app, press Execute.

5. 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 samples of images of digit.
- Each client is training the model for a subset of samples of only few digits. Example: client 1 is training data only with samples of digits 1 and 2.
- In the end, all the individual models are aggregated in the server, and the aggregated model is returned to the clients. With the aggregated model, all the clients will be able to predict all the digits.

5.1. 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**).

6. Check the performance of the models before and after the aggregation

6.1. Generate plots with `/predict/local` and `/predict/aggregated/` with the field **plot_graphs** set to true.

This endpoint will create plots where you can observe the performance of the model to predict certain digits.

6.2. Use the **scp** command to copy the the generated plots to your computer, so you are able to visualize them. Example:

```
scp nap@192.168.3.id:~/MobFedLS/logs/date/
predict_after_agg_model_config1_run0.png ~/.
```

6.3. Additionally, with the output of the endpoint check the accuracy of each model. Check how different the values of accuracy are before and after the aggregation.

7. Repeate the federated learning process with other conditions

Important note: Between each federated process with different characteristics do a `docker compose down`**, and then** `docker compose -f base-docker-compose.yaml up -d`**, followed by running the endpoints /get_data and /fit to reset the parameters of each client model.**

7.1. With the batman network stable:

Change the values in with the **configX.csv** to modify the number of samples of each digit. Check if the accuracy values change accordingly.

7.2. Experiences with nodes having anomalies:

- Connect less then 4 clients from the beginning - Deliberately put a wrong ip on the neighbours_file in the keys “2”, “3”, or “4”.

  There will be a connection try to that manager and a timeout (to observe this, check the logs of the manager container on the server). Check the accuracy results. Comment about the performance of the aggregated model in this condition.

- Connect 4 clients from the beginning and during the training process (when the ghost client container is running) shutdown one of the boards.

  When one client is disconnected from the server in the middle of a federation round the server will wait until the round_timeout if one or more clients are not answering. and only then proceeds to the next round. Check the logs to observe

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
POST	/fit	This method is used in the FL process to train the ML-App
POST	/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