universidade de aveiro
theoria poiesis praxis

DEPARTAMENTO DE ELECTRÓNICA, TELECOMUNICAÇÕES E INFORMÁTICA
MESTRADO EM ENG. DE COMPUTADORES E TELEMÁTICA
ANO 2022/2023

REDES E SISTEMAS AUTÓNOMOS

AUTONOMOUS NETWORKS AND SYSTEMS

PRACTICAL GUIDE 3 – FEDERATED LEARNING

Objectives

- Set up a Federated Learning (FL) cluster
- Use FedFramework 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

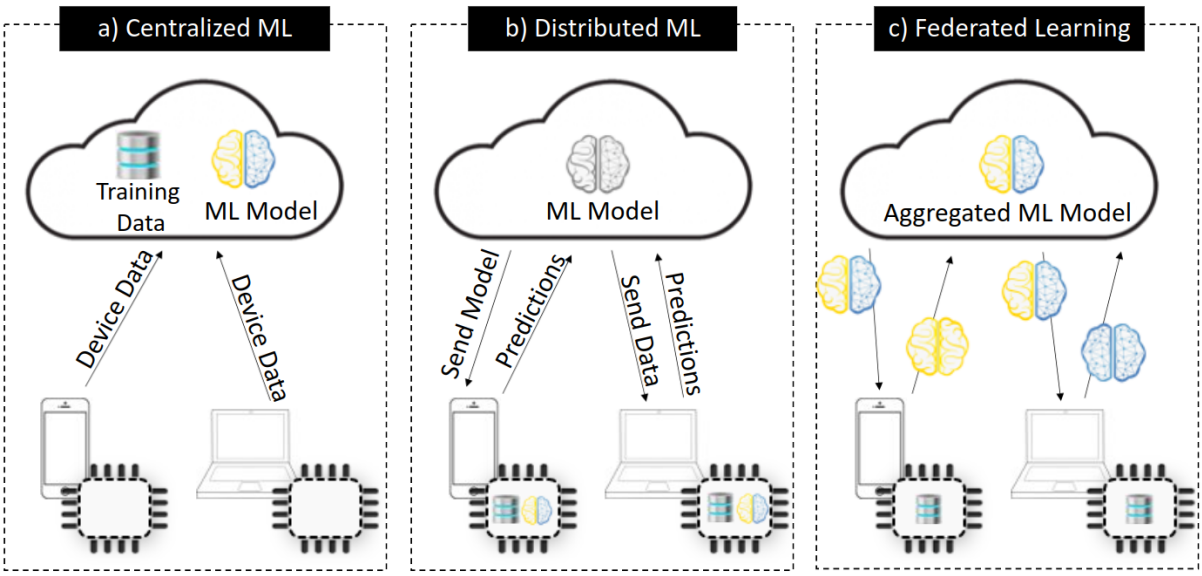
Duration

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 FedFramework, 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 2 batman networks in the classroom, each with 4 groups (one Jetson per group).

1.2. Modify the batman script with the same settings as the other 3 groups nearby:

```
$ vi batman_installation/create_batman_interface.sh
```

1.3. Then run the script:

```
$ ./batman_installation/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 Jetson using ssh and the same credentials.

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

3. Launch the FedFramework through VSCode

3.1. Follow these steps:

```
$ cd fedframework
$ source venv/bin/activate
$ cd FedFramework_aarch64
$ uvicorn app:app --reload
```

3.2. Now open the browser in the following link:
http://127.0.0.1:8000/docs

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

4. Clean the environment

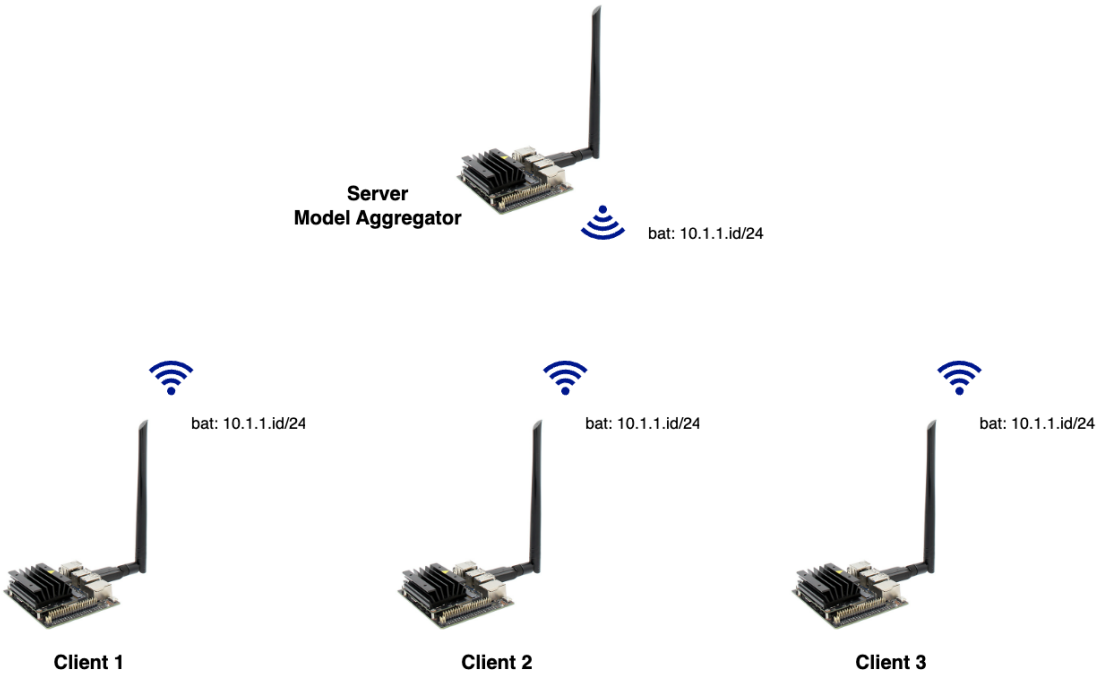
4.1. To start the experience, first we need to clean any previous setup that could have been instantiated in the Jetson before.

4.2. Expand the endpoint `/containers/remove/all`, click "**Try it out**" and then "**Execute**".

4.3. Do the same for the endpoint `/networks/remove`

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. For the container (client or server) to run in the board, we need to create a Docker network. Execute the endpoint `/networks/create`

Use the terminal (**outside of VSCode**) to check if the network was created successfully:

```
$ docker network ls
```

Is `lan0` network part of the list?

Check also with the following command:

```
$ ifconfig
```

Verify that the interface `br-844cdb98274a` is in the list and is part of the network `127.1.0.0/24`.

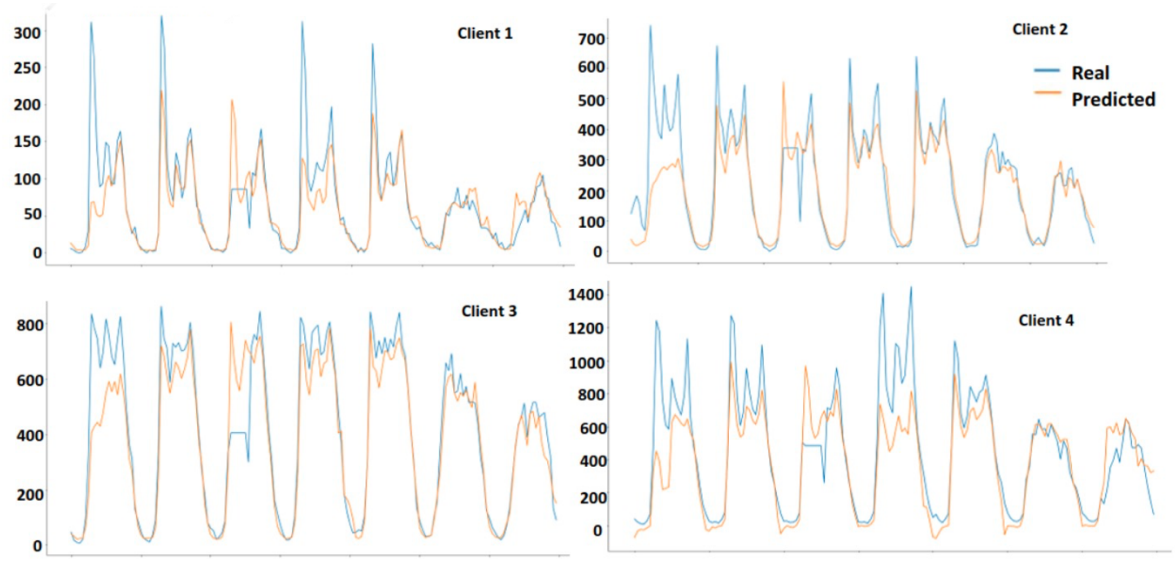
5.3. Now, set your node in the cluster as a server or client (according to the distribution agreed in **5.1**).

If you are a server, execute the endpoint `/containers/create/server`
Use all the default values for the parameters, modify only the following:
n_clients: 3

If you are a client, execute the endpoint `/containers/create/client`
Use all the default values for the parameters, modify only the following:
client_number: X (from 1 to 3, select different number among the groups)
server_ip: 10.1.1.id_server

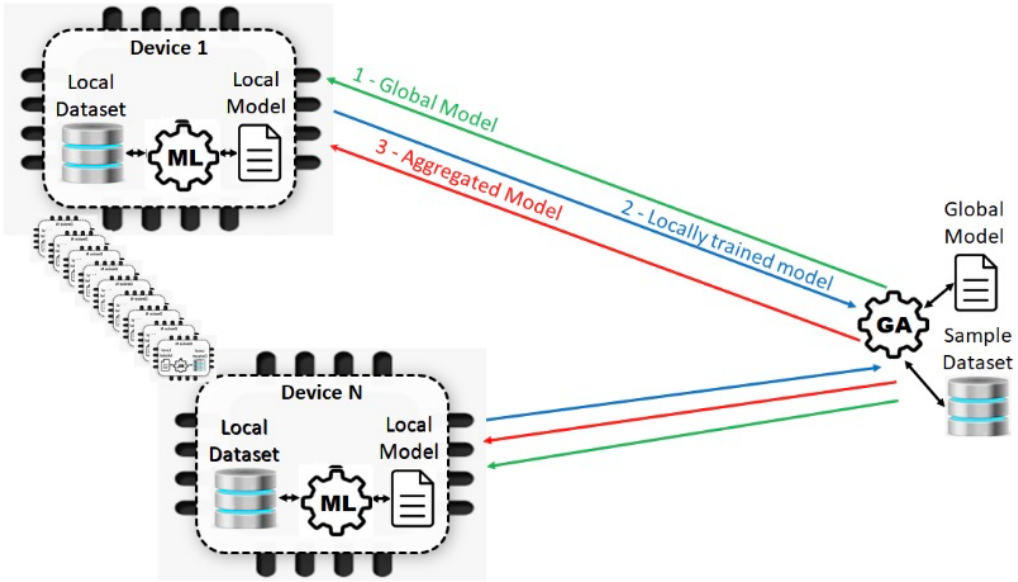
6.

- With all the nodes connected, the clients start training their individual models.
- The FedFramework 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).
- The real data from the sensors gathered by each client/edge node is plotted in the following figures:



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:



5.4. Explore the logs in the clients and in the server through the endpoint `/logs/web`

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. To repeat the training (with the same settings), do the following, in a terminal outside of VSCode:

```
$ docker ps -a
```

Check what is the container id

```
$ docker start container_id
```

The federated learning will execute again automatically.

To change the characteristics of the training (number of rounds, number of clients, number of epochs, etc), you need to clean the environment as done in **4.** and then create the server and clients with the new parameters.

Appendix

Table 1: API Methods.

Networks	
/networks/list	Shows the list of existing networks in docker
/networks/create	Creates the network called "lan0" and with the subnet "72.100.0.0/24"
/networks/remove	Removes the network with the name passed as the name parameter
/networks/inspect	Allows you to inspect the parameters of existing networks
Images	
/images/list	Shows the list of existing images in Docker
/images/create/server	Creates the server's docker image with the name passed as the name parameter
/images/create/client	Creates the client docker image with the name passed as the name parameter
/images/remove	Removes the docker image with the name passed in the name parameter
/images/remove/all	Remove all FedFramework Docker images
Containers	
/containers/list	Shows the list of existing Docker containers
/containers/create/server	Creates the server based on some parameters passed in the request
/containers/create/client	Creates customers based on some parameters passed in the request
/containers/start	Starts an existing container with the name passed as the name parameter
/containers/stop	Terminates an existing container with the name passed as the name parameter
/containers/kill	Kills an existing container with the name passed as the name parameter
/containers/remove	Remove an existing container with the name passed as the name parameter
/containers/remove/all	Remove all existing containers
/containers/restart	Restarts all existing containers
Logs	
/logs	Container logs are saved in files with container names
/logs/web	Container logs are displayed on the web
Extras	
/run	Creates the Network, Server and Client Containers depending on the sent parameters
/remove	Remove all Networks and Containers
/reset	Removes all Networks, Containers and Images