Machine Translated by Google

Faculty of Computer Science

UNIVERSIDADE DA CORUÑA

END OF DEGREE WORK
DEGREE IN COMPUTER ENGINEERING
MENTION IN COMPUTING

Development of a library for federated learning under an architecture peer-to-peer

Student: Pedro Guijas Bravo

Address: Daniel Rivero Cebrian

Enrique Fernandez Blanco

A Coruña, September 2022

Machine Translated by Google

To all the people around me, especially my parents and partner.

Machine Translated by Google

Thanks

I would like to thank those who have supported me during the realization of this project, without them this would not have been possible:

- To my girlfriend, Elena, for putting up with me all these years.
- To my parents and grandparents, for being with me despite life's misfortunes.
- To my tutors, for helping me make possible such an interesting project in which I have learned so much.
- To my roommates, for creating an environment of academic performance and disconnect xion at opportune moments.
- To my lifelong friends, for helping me disconnect.
- To my friend Fernando, for always smiling.

Machine Translated by Google

Summary

In the last decade, the evolution of Machine Learning has been very prosperous, requiring the most fruitful models to be fed by large volumes of data. Obtaining and managing this data is often complicated, generally scarce and subject to privacy measures. Federated Learning implies a paradigm shift in the training of Machine Learning models. This new approach makes it possible to perform the learning process on data distributed among a large number of clients.

Despite the fact that this new technique brings numerous advantages, one of its major limitations is the need for a server that orchestrates the entire learning process, providing a single point of failure. Likewise, it will be necessary to have a large infrastructure to make these systems scalable. To try to solve these disadvantages, new approaches called Decentralized Federated Learning will emerge, one of the most promising solutions being the use of peer-to-peer networks.

Given the absence of any support library for Decentralized Federated Learning, this project proposes the development of a general purpose library that allows Fe derated Learning **on** peer-to-peer networks, **using the** Gossip protocol. The use of the Gossip protocol will guarantee fault tolerance in the peer-to-peer network, creating a decentralized, scalable and robust ecosystem.

The library seeks to support all kinds of devices, with special emphasis on its ease of use and future expansion. In addition to allowing deployment, it enables the execution of simulations, making it possible to carry out tests in controlled environments.

For development, the agile SCRUM methodology has been used. The central iterations have been dedicated to the implementation of the system, while the initial and final ones have been devoted to project preparation and testing, respectively. In the various tests carried out, the MNIST and FEMNIST datasets have been used, obtaining results that are really similar to equivalent executions with classical training.

Abstract

In the last decade, the evolution of Machine Learning has been really thriving. The most pro ductive models need to be fed by a large volume of data. Obtaining and managing these data

Machine Translated by Google

is often complicated, as they are generally scarce and subject to privacy measures. Feder ated Learning implies a paradigm shift in the training of Machine Learning models. This new approach allows us to carry out the learning process on data which are distributed among a large number of clients.

In spite of the fact that this new technique has numerous advantages, one of its great limita tions is the need for a server that orchestrates the entire learning process, assuming a single point of failure. Moreover, a large infrastructure will be necessary to make these systems scal able .In order to solve these disadvantages, new approaches called Decentralized Federated Learning will emerge, and one of the most promising solutions will be the use of peer-to-peer
networks.

Given the lack of any library to support Decentralized Federated Learning, this project pro poses the development of a general purpose library that allows **Federated Learning over peer-to-peer networks, using the Gossip protocol.** The use of the Gossip protocol will guarantee fault tolerance in the network, creating a decentralized, scalable and robust ecosys
he has.

The library will seek to support all kinds of devices, with special emphasis on ease of use and future expansion. In addition to allowing deployment, it will enable the execution of simulations, making it possible to perform tests in controlled environments.

The agile SCRUM methodology has been used for this development. The central iterations have been used to the implementation of the system, while the initial and final iterations have been respectively dedicated to project preparation and testing. The MNIST and FEMNIST datasets have been used in the different tests carried out, obtaining results that are really similar to equivalent executions with classic training.

Keywords:

- Machine Learning
- Decentralized Federated Learning
  lyzed
- peer-to-peer
- Protocolo Gossip •

Python •

Pytorch

Keywords:

- Machine Learning
- Decentralized Federated Learning
- peer-to-peer
- Gossip protocol
- Python
- Pytorch

Machine Translated by Google

General index

Machine Translated by Google

GENERAL INDEX

Machine Translated by Google

GENERAL INDEX

Machine Translated by Google

GENERAL INDEX

Machine Translated by Google

index of figures

Machine Translated by Google

INDEX OF FIGURES

Machine Translated by Google

INDEX OF FIGURES

Machine Translated by Google

Table index

Machine Translated by Google

TABLE INDEX

Machine Translated by Google

Chapter 1

Introduction

At present, Artificial Intelligence has become one of the most prolific fields. computer hackers. One of its main branches, Machine Learning (ML), is increasingly popular and used, its impressive advances being the result of more and more complex models, nourished by large amounts of data.

The conventional way of working in ML consists of the centralization of information and computation in the same place, implying limitations. One limitation would be related to the infrastructure, since the maintenance of information and the different learning processes require large resources. Another major limitation focuses on the nature of the information. The data comes from a wide variety of sources, and is usually subject to privacy measures that may even prevent its dissemination.
According to these measures, a potential privacy risk is being generated by using this centralized approach to learning processes.

Thus, in 2016, Google proposed Federated Learning (FL) [1], in Spanish, Aprendiza je Federado. This new approach aims to solve the problems exposed above thanks to the creation of models on distributed data. Its operation is simple, it consists of an iterative process orchestrated by a central server. In each iteration, different nodes will contribute to a global model by aggregating local models. In other words, to obtain the local models, the nodes will perform parallel training using their own data and resources.

Despite the obvious advantages of not explicitly sharing data, the orchestrator server also comes with a number of limitations. These consist of the existence of a single point of failure and a possible bottleneck, considerably limiting scalability. To solve these problems, it is sought that the nodes manage to communicate

Machine Translated by Google

between them, regardless of the server. This set of approaches is called Decentralized Federated Learning [2].

The use of Peer-to-peer (p2p) networks [3.1.3] and Gossip protocols [3] is one of the most promising approaches to Decentralized Federated Learning. The p2p architecture will allow the creation of an infrastructure where the devices or nodes behave collaboratively among equals. As for the Gossip protocol, it will allow the creation of a robust mechanism for the exchange of messages, taking advantage of the redundancy of p2p networks . In this way, a decentralized ecosystem will be obtained, which guarantees both scalability and robustness. It should be noted that fault tolerance will be crucial in real environments, especially if you want to support edge devices, which despite having more and more computing power, are often affected by availability problems related to the use of batteries and telephone networks. Unfortunately, most of the work related to this approach is focused on research, and there are no libraries that allow the easy creation of such systems.

Consequently, motivated by the non-existence of a general purpose library for Decentralized Federated Learning, an implementation of this type will be developed in this project. Being a general library, it will try to support all kinds of devices, looking for the lightness of the system focused on supporting low-performance devices. Likewise, a specific topology will not be established, trying for the user to decide the most suitable one for his system. With regard to communications, they will try to minimize, including cryptographic methods to guarantee confidentiality and reliability in them.

Seeking convenience and flexibility, the library will be user-oriented, being completely agnostic to existing ML frameworks , ensuring their easy inclusion.
Specifically, this project will focus on Artificial Neuron Networks (RR.NN.AA), since Federated Learning has barely been explored with other kinds of models.

Likewise, in order to validate and test the created systems, the library will not only be deployment oriented, but simulated environments will also be contemplated. The simulation will facilitate the development and use of the library, allowing an easy transition to deployment.

Finally, a real practical example will be carried out where the operation and qualities of the library can be validated and demonstrated.

Machine Translated by Google

1.1 Objectives

The objective of this project will be the **creation of a library that supports Federated Learning on p2p networks, using the** Gossip protocol. In order to meet this objective, the following specific sub-objectives have been identified that must be addressed in order to carry out the first one:

- **Study:** Federated Learning Study, p2p networks and Gossip protocols.
- **Simplicity and Ease:** It will try to bring these two principles, both to the development of the library, and to its use by the end user.
- **Modularity and Abstraction:** Concepts that together with the previous ones will facilitate the expansion of functionalities, both for initial and future developers.

  A design based on these two concepts will allow, among other things, the easy incorporation of new ML frameworks and federated aggregation algorithms.
- **Robustness:** The possibility of failures in the nodes must be taken into account, controlling their occurrence without harming the system.
- **Privacy:** As will be discussed in the fundamentals chapter 3, the privacy by design of the FL will not be enough , since it could be reverse engineered.
  Therefore, the bookstore must provide an additional layer of privacy.
- **Research and Deployment:** It will be intended that the library be suitable for these two types of users, allowing easy migration to deployment.
- **Testing:** The developed system should be carefully tested and tested, tra
  Trying to get closer to real environments.

1.2 Structure of the Report

The memory has been structured around the following chapters:

1. **Introduction:** Current chapter. An attempt has been made to introduce the project.
2. **Market Study:** The different existing alternatives to the bookstore will be analyzed that you want to develop.

Machine Translated by Google

3. **Fundamentals:** The theoretical and technological elements applied in the project will be carefully explained. It will try to justify the reasons for choice as well as familiarize the reader.

4. **Methodology and Planning:** This chapter will deal with the methodology to be followed, the initial planning of the project and its follow-up. In addition, a study of the feasibility and risks of the project will be carried out.

5. **Library development:** Details the library development process. Said process will be divided into sprints as established by the methodology to be used, SCRUM. The first and last sprints will focus on project planning and test development respectively.

6. **Conclusions:** In this chapter the conclusions drawn in the
tion and monitoring of the project.

7. **Future Work:** This chapter will present the different lines of future work, not included in the objectives of the project.

Machine Translated by Google

Chapter 2

Market study

The objective of this chapter is to compile and analyze the different libraries and frameworks that support federated learning.

When creating a product, it is necessary to carry out a market study beforehand. In this way, it will avoid developing already created solutions, looking for the weak points of the existing ones to try to solve them and provide improvements.

Next, the comparison criteria will be defined, the different frameworks will be analyzed and compared and it will end with a conclusion.

2.1 Comparison criteria

In this section, the characteristics on which the market study will focus will be briefly defined, creating criteria that facilitate a comparative analysis. These characteristics will be the following:

- **Ease of Use:** In other words, usability.
- **Ease of Expansion:** That is, the extensibility of the library.
- **Dependence on some ML** framework : The coupling of the library to some ML framework , as well as the ability to include new .
- **Possibility of simulations and deployments:** The ability of the library to be executed on real environments and simulations of said environments. Simulation is ideal for pre-deployment system validation and research purposes.

Machine Translated by Google

- **Possibility of supporting models beyond the RR.NN.AA:** Although it may not be implemented, the design of the library must be flexible enough to be able to include new types of ML models.
- **Centralized architecture:** Type of architecture, focusing on whether or not it is a centralized architecture.
- **Mechanisms for the distribution of the model code:** That is to say, if mechanisms are provided for the diffusion between devices of the code and operation of the models, not only of its parameters.
- **Mechanisms for data organization:** These mechanisms will allow the selection of data, making possible, among other things, the coexistence of different datasets in the devices. This is a feature that will enable the creation of more versatile federated environments.
- **Device control mechanisms:** These will control both access to the network as well as resources within it.
- **Additional privacy mechanisms:** Techniques outside of Federated Learning that guarantee data privacy.

2.2 Analysis and comparison of existing solutions

Next, the main libraries and frameworks will be briefly introduced, showing in table 2.1 the differences between them according to the established criteria.

- **Flower** [4]: Open-source framework for the creation of Federated Learning systems. Despite not providing immensely extensive functionality, this library is extremely friendly, extensible, and easy to use.
- **OpenFL** [5]: Open-source framework to support Federated Learning developed by Intel. OpenFL is designed to be a flexible, extensible, and easy-to-learn tool for data scientists.
- **NVIDIA Flare** [6]: Open source and extensible framework created by Nvidia. It allows adapting ML systems to a very secure and complete federated environment.
- **PySyft** [7]: Open-source framework developed by OpenMined. PySyft provides a secure and private environment for Data Science in Python. PySyft applies a multitude of techniques among which Federated Learning stands out.

Machine Translated by Google

- **Tensorflow Federated** [8]: Framework developed by the Tensorflow team. This only allows simulating the Federated Learning algorithms, not being possible the deployment of said systems.
- **FATE** [9]: Open-source project started by Webank. It implements multiple secure computing protocols to allow the creation of models complying with data protection regulations. Its extremely broad and complex ecosystem, combined with the language barrier of its development team, makes it difficult to use such a framework.
- **PaddleFL** [10]: Open-source framework for Federated Learning developed by the Chinese search engine Baidu. PaddleFL is based on PaddlePaddle [11], being suitable for both research environments and highly distributed deployments.

of the.

	bookstore to develop	Flower	OpenFL	Nvidia Flare	PySyft	Tensorflow Federated	FATE	PaddleFL
Easy to use	high	high	Media	Middle-low	High-Medium	Media	Low	Middle-low
Ease of Expansion	high	high	Media	high	Media	Low	Low	Media
dependent on someone framework	No	No	No	No	No	TensorFlow	FederatedML (own)	PaddlePaddle
Centralized Architecture	No	And	And	And	And	And	And	And
Mechanisms for distribution of the code of the model	No	No	And	And	And	No	And	And
Mechanisms for data organization	No	No	And	And	And	No	And	And
mechanisms of device control	No	No	And	And	And	No	And	No
mechanisms of additional privacy	And	And	And	And	And	No	And	And
Possibility of simulations and deployments	And	And	And	And	And	Only Simulation	And	And
Possibility of support to models beyond las RR.NN.AA	And	And	No	And	No	No	And	No

Table 2.1: Comparative table of Federated Learning libraries

As can be seen in the table, at the time of carrying out the market study, **there is no public library or** framework **that addresses an implementation of Decentralized Federated Learning,** with the little existing work being private and completely oriented towards research.

Machine Translated by Google

2.3 Conclusion

Federated Learning libraries are generally immature, sparse, and with a relatively steep learning curve. Table 2.1 highlights that most existing frameworks aim to create complex and complete systems, including mechanisms and technologies outside of Federated Learning. Obviously, these will not be dealt with in this project, but their future inclusion will be taken into account.

On the contrary, Flower is the only framework with a similar value proposition to the library to be developed: a simple, extensible and user-oriented library. In this way, bearing in mind that there is no library with a decentralized architecture and inspired by Flower's approach, we will seek to create a library with similar characteristics that supports Federated Learning on p2p networks . Likewise, since development time is quite limited, emphasis will be placed on facilitating future extensibility by third parties.

Machine Translated by Google

Chapter 3

Fundamentals

This chapter will try to explain the theoretical and practical foundations of the project. HE will try to familiarize the reader with the contents covered.

3.1 Theoretical Foundations

It will begin by briefly explaining the RR.NN.AA, since its understanding is necessary to deal with Federated Learning. This will be discussed below, seeking to contextualize it, explain it and treat its situation in decentralized environments. Immediately afterwards, the contents related to the operation of the Gossip protocol and the p2p architecture will be addressed , on which the Federated Learning system will be created.

3.1.1 Artificial Neural Networks

Artificial Neuron Networks (RR.NN.AA) [12] are computational models inspired by the central nervous system. This class of models learn by themselves, through supervised training, that is, their knowledge is not explicitly programmed.

As its name suggests, networks are made up of massively interconnected artificial neurons, following a specific architecture. Each neuron is a processing unit, the output of each neuron being determined by function 3.1. Neurons internally perform the weighted sum of their inputs by given weights (wij), w0 is a term called bias, this will always be added. In addition, to avoid that the concatenations of neurons result in an operation of the same structure as that of the neuron (3.1), the activation function ÿ will be applied, in charge of eliminating the linearity between

Machine Translated by Google

operations.

$$yi = ÿi((ÿmi\ wijxij\ ) + wi0) \quad (3.1)$$
$$j=1$$

where:

yi = output value of neuron i.

ÿi = activation function for neuron i.

mi = number of inputs to neuron i.

wij = weight for input j of neuron i, wi0 will be the bias.

xij = input value j of neuron i.

The learning process is responsible for finding the weights of the model that produce the desired behavior. In this way, the training will be treated as a problem of minimization of the error produced by the network. Algorithm 1 details the learning, noting that it is an iterative process in which each epoch (E) seeks to improve the weights of the model (wij). To control convergence, the improvements (get_upgrade(wij ;train_data)) are smoothed by the learning rate (ÿ), whose value is generally in the interval (0,1].

Algorithm 1 Process Learning and una Neural Network Artificial

for each epoch from 1 to E **do**
  wij = wij + ÿ ÿ get_upgrade(wij ;train_data) **end**
for

Gradient descent algorithms [13] along with error backpropagation [14] are used as a mechanism to determine improvements over weights, due to their great scalability and flexibility. These will use a set of data representative of the problem to be solved to calculate the error of the model, and consequently, the gradients of the weights with respect to said error.

It should be mentioned that when a lot of data is available for training, it is not usual to supply the learning algorithm with the whole data set at once. This usually produces problems of stagnation of the model. To induce greater randomness, the samples are supplied to the optimization algorithm in batches . Likewise, another frequent variation is the use of the Stochastic Gradient Descent (SGD) algorithm [15], since it introduces more randomness to the training, producing very good results in practice.

Machine Translated by Google

3.1.2 Federated Learning

Federated Learning [1] is a new approach to traditional ML that dispenses with centralizing training data in a single machine or data center. This novel paradigm allows multiple devices to collaboratively train a model, without the need for training data to leave the device.

As mentioned in the introduction, its operation is simple and iterative, or quested by a central server. Each iteration is known as **a round of Federated Learning,** consisting of the following steps:

1. Transmission of the global model to the different devices.
2. Through local training, the devices will try to improve the model global using their own data.
3. Sending the models obtained on each device to the orchestrator server.
4. Update of the global model by processing the different models premises, this process is called **Federated Aggregation.**

In figure 3.1 the different steps of each round can be seen graphically. Again, it should be stressed that only the RRs will be covered, as Federated Learning has hardly been covered with any other class of ML models.

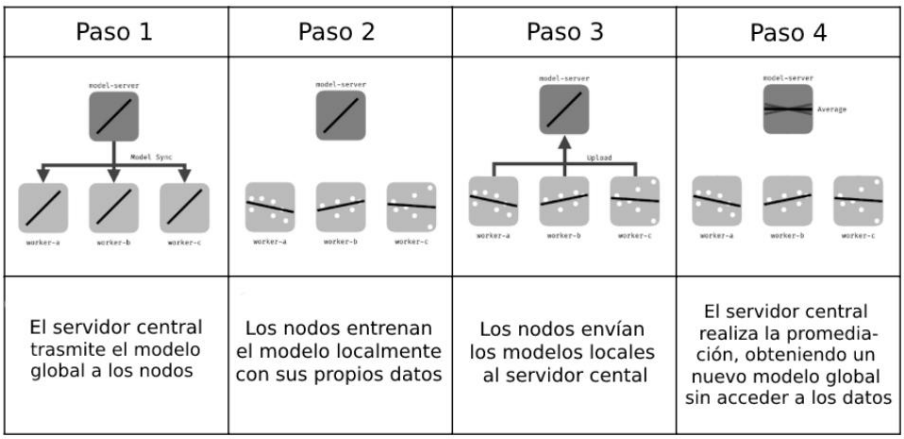


Figure 3.1: Steps in each round of Federated Learning

Consequently, what is really new in Federated Learning will be model aggregation, so the first and main federated aggregation algorithm will be discussed below.

Machine Translated by Google

rada: Federated Average (FedAvg) [16]. Immediately afterwards, the advantages and disadvantages will be addressed, ending with the Decentralized Federated Learning.

Federated Average (FedAvg)

Federated Aggregation is the core of Federated Learning. Understanding the circumstances of its creation, as well as its operation, it will be possible to better understand the way in which the FL operates.

The most fruitful Deep Learning models [17] have been mostly based on SGD along with its extensions or variations (such as Adaptive Moment Estimation (ADAM) [18] or Adaptive Gradient Algorithm (AdaGrad) [19]). Thus, the first contact with federated aggregators is Federated Stochastic Gradient Descents (FedSGD).

With FedSGD, in each round the devices will calculate the gradients locally, being later added by making a weighted average by the number of samples from each device. Having a common initialization at the beginning of each round does not necessarily degrade the performance of the resulting averaged model.

In traditional ML , the cost of communications is practically non-existent, with the highest cost being computational. In a federated environment the opposite happens, so to minimize the cost, it will be tried to increase the computation in the device in favor of decreasing the communications. Seeking to reduce communication by increasing computing, FedAvg emerges, as a generalization of FedSGD, achieving between 10-100 times less communication in comparison. In FedAvg, model weights and not gradients will be shared, with the amount of computation and communications controlled by 3 parameters:

- C: Fraction of Clients. In order not to saturate the communication channels, only one fraction of devices will be improving the model in each round.
- E: Number of epochs or iterations of the learning algorithm per round in each client. In a round you will be able to perform more than one step down the gradient.
- B: Size of the batch used in learning.

Therefore, for a node with n data samples, the number of local updates per round will be u = E ÿ (n/B). If all the samples are used in a single batch and E = 1, it will correspond to FedSGD (u = 1).

Machine Translated by Google

The pseudocode of Federated Learning using FedAvg can be seen in Algorithm 2, for the server side, and in Algorithm 3, for the client side. Briefly, the server will initialize the model (w0) and start the training loop for R rounds.
In each round, it will select a subset of nodes (St) and add the received models using a weighted average by the number of samples, obtaining a new model, wt + 1. As for the device, a local training will be performed using SGD with its data (Pk), a duration of E epochs and a batch size of B. When finished, the weights of the resulting model will be sent to the server.

Algorithm 2 Federated Averaging Server Side (K clients)

initialize w0
for each rount t from 1 to R **do**
  train_set_size ÿ max(C·K, 1)
  St ÿ (subconjunto aleatorio de train_set_size nodos) **for** each client k in St **do** k ÿ
    ClientUpdate(k, wt) w t+1
  end for
  wt+1 ÿ ÿ $\sum_{k=1}^{K}(n_k/n)\cdot w_{t+1}^{k}$
end for

Algorithm 3 Federated Averaging Client Side ÿ

ÿ (split Pk en batches de tamaño B) **for** each local epoch from 1 to E **do for**
  each batch b in ÿ **do**
    w ÿ w ÿ ÿÿl(w; b) # SGD
  end for
end for
return w to server

It should be noted that there are many federated aggregators based on FedAvg that will try to improve its convergence in certain situations. Among all, the following aggregation methods stand out: Federated Learning with Matched Averaging (FedMA) [20], Stochastic Controlled Averaging for Federated Learning (SCAFFOLD) [21], q-Fair Federated Average (q-FedAvg) [22], FedOpt, FedYogi, FedAdam [23], FedProx [24].

Advantages and Deficiencies of Federated Learning

The use of Federated Learning systems has numerous advantages, among which are:

- Use of private and confidential data without having to worry about handling the data. data.

Machine Translated by Google

- Continuous learning process.
- No large infrastructure is required beyond the orchestrator server. The computation will be distributed by the different devices and the data will not need to be stored.

Likewise, the main disadvantages consist of:

- Regarding the orchestrator server, to add the models in a scalable and fault tolerant, a large infrastructure must be available.
- Model aggregation does not have privacy mechanisms by design, so customer information could be obtained through reverse engineering. Mechanisms such as Secure Aggregation [25] or other encryption techniques will be used to solve this problem.
- Loss in model performance against malicious clients and data distributions that cause devices to not have a representative set of samples (non-Independent and Identically Distributed Random Variables (IID) distributions).
- It is a relatively new technology and under study.

Decentralized Federated Learning

Now that we have a basic notion of Federated Learning, it must be taken into account that there are many variations of it, aimed at solving different problems [26]. This subsection will focus on dealing with FL following a serveless approach, that is, without the orchestrator server.

There are different alternatives to create Decentralized Federated Learning systems. The p2p networks with graphs [27] or the use of blockchain as a substitute for the orchestrator server [28] are one of the many solutions. However, this work will only focus on p2p networks and the Gossip protocol, mainly due to its flexibility, lightness, robustness and fault tolerance, allowing its use with low-performance devices, as well as more powerful nodes.

The scientific literature on Decentralized Federated Learning is really scarce, and there are no works that address a solution similar to the one that will be developed in this project. No

Machine Translated by Google

However, a work that deals superficially with two of the fundamental parts of this project should be highlighted: the use of p2p networks in Federated Learning and the aggregation of models in distributed environments [29].

This work proposes a system really similar to the original proposal of Federated Learning. The big differentiation is that each node will receive and add the local models, in the same way that the orchestrator server does. As a result, at the end of each round, all the nodes will have the same version of the model, the result of the aggregation.

In figure 3.2 it can be seen graphically how the nodes send their local models to the rest of the network.

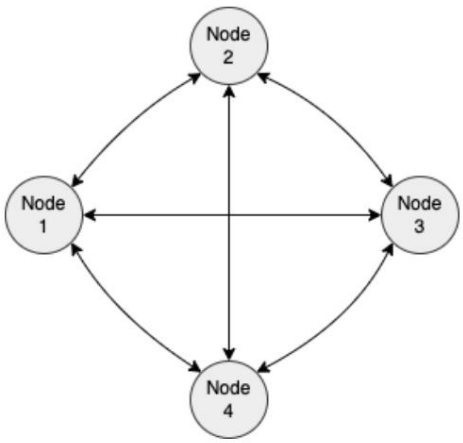


Figure 3.2: Submission of local models using a p2p architecture.

Additionally, keeping in mind the use of the Gossip protocol, this document proposes an optimization for the aggregation of models in decentralized environments. The improvement is based on FedAvg, the most common federated aggregation algorithm and from which most alternatives start.

With FedAvg, the aggregation is a weighted average (formula 3.2), and can be decomposed in an iterative process (formula 3.3). With the iterative method it is possible to obtain intermediate results or partial aggregations that are equivalent to a set of models already added.
In this way, it will seek to send partial aggregations, since they will be condensing information, avoiding sending each model individually. As can be expected, this means significant savings in the cost of communications, especially considering that sending models is the most expensive communication.

$$OF\ G = \ddot{y}_{i=1}^{n} \frac{model_i \ddot{y}\ samples_i}{\ddot{y}_{i=1}^{n} samples_i} \tag{3.2}$$

Machine Translated by Google

$$OF\ G = \ddot{y}_{i=j}^{n} \frac{model_i \ddot{y}\ samples_i}{\ddot{y}_{i=1}^{n} samples_i} + \ddot{y}_{i=1}^{j} \frac{model_i \ddot{y}\ samples_i}{\ddot{y}j_{i=1} samples_i} \ddot{y} \frac{\ddot{y}j \ddot{y}_{i=1}^{n} samples_i}{\ddot{y}_{i=1}^{n} samples_i} \quad (3.3)$$

where:

n = number of models to add.

j = number of models to partially add. (n > j)

In order to average the models correctly, the sets of nodes involved in the partial aggregations must be disjoint sets, and it is necessary to keep track of the remaining models per node.

3.1.3 Peer-to-peer

Peer-to-peer (p2p) is a distributed software architecture. It is made up of different nodes or pairs that behave as equals to each other, forming the so-called network of nodes.

In this architecture, a central element that manages the network will not be necessary, the nodes will communicate with each other, self-organizing it. Thus, the peers are both providers and consumers of resources, in contrast to the traditional client-server architecture model.

High scalability and fault tolerance are the two great advantages of this architecture.
The main drawback lies in the complexity of network administration. Decentralization implies the generation of a lot of traffic and the existence of complex mechanisms for network management.

Despite the fact that the p2p architecture seeks decentralization, there are variants that partially renounce it to mitigate the disadvantages associated with the complicated management of the two. Particularly, in this work only pure p2p networks will be treated , since they guarantee high decentralization and scalability.

Within pure p2p systems , two large groups can be distinguished, **unstructured p2p networks** and **structured p2p networks .**

Machine Translated by Google

Structured p2p **networks**

These p2p systems are organized following a specific topology, guaranteeing that any node can be searched efficiently. In order to route traffic efficiently, nodes must maintain neighbor lists (frequently distributed hash tables), making these networks less robust in the face of high dropout rates, that is, when the frequency of joining and dropping out of nodes is high.

In figure 3.3, some of the main network topologies can be seen. Each architecture has its own characteristics and shortcomings, making them more or less adequate depending on the casuistry of the problem to be solved.

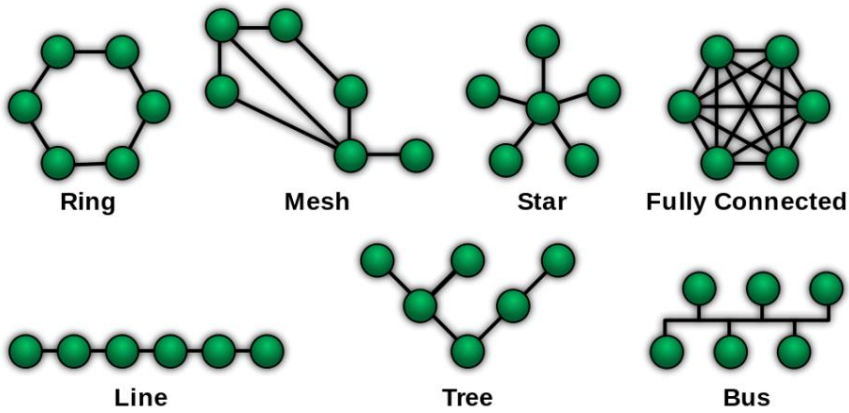


Figure 3.3: Topology types in network
Source: Wikimedia

The rigidity involved in following a specific topology, the fact that these networks are less robust at high dropout rates, and not having the need to efficiently search for a specific device in the network, have led to the fact that this type of network is not used. networks in the project.

Unstructured p2p **networks**

Unstructured p2p networks do not impose a particular structure on the network. So, in order to locate different nodes or information, flooding techniques or protocols based on this technique such as Gossip must be used. Since flooding has a raw and more inefficient approach, it will not be covered in this paper.

Machine Translated by Google

3.1.4 Gossip Protocol

The Gossip protocol is a communication process in p2p networks based on epidemic expansion. It is used to guarantee that a message will be broadcast to all members of the network, being used mainly as a broadcast protocol and an aggregation protocol.

Its operation is iterative and simple, and can be understood as an analogy to the spread of rumors. An item will spread a rumor to a set amount of in

random individuals, in a row, the original individual and those who know the rumor will continue spreading it until everyone knows it, that is, until it converges. As can be guessed, its operation is adequate in large networks with high levels of decentralization.

The main advantages of the protocol include:

- **Scalability:** When carrying out an exponential diffusion, the increase in the number of dispo positive will not cause performance issues.
- **Adaptability:** Thinking of low-performance devices or congested networks, the rate and size of the diffusion can be adapted for each node.
- **Robust and fault tolerant:** Falls or possible errors will not affect the diffusion process. Likewise, the high degree of redundancy in communications may allow us to recover from certain errors.
- **Extremely oriented to distributed systems:** As mentioned, this protocol is especially useful in distributed environments.

Its disadvantages are related to the communication inefficiency relative to unstructured p2p networks . Likewise, the convergence time could be seen as a disadvantage (because it is an iterative protocol), but for this work it will be seen as an advantage for temporarily spacing the communications and not over-saturating the system.

3.1.5 Encryption

Cryptography is used for the purpose of granting privacy in communications, encrypting the messages transmitted. Two types of encryption will be used in this project: asymmetric and symmetric encryption.

Machine Translated by Google

Asymmetric Encryption

Asymmetric encryption or public key encryption is characterized by using two related keys, a public key responsible for encryption and a private key for decryption. The public key must be shared to encrypt messages that only the owner of the associated private key can decrypt.

As can be seen, the main advantage is clear: a simple and secure distribution of keys, limiting itself to only sharing the public one. However, it has clear disadvantages:

- The encrypted message takes up more space than the original.
- Compared to symmetric encryption it is computationally more expensive and slower.
- Some encryption algorithms have a maximum number of bytes to encode, depending on depending strictly on the size of the key.

There are many asymmetric encryption algorithms, being Rivest, Shamir and Adleman (RSA) [30] the one used. This decision is mainly motivated by being the first and most widely used asymmetric encryption algorithm.

Symmetric Encryption

Symmetric encryption is a cryptographic method that is based on the use of the same key for the encryption and decryption of messages. Again, there are many algorithms that allow symmetric encryption, being the algorithm selected for this task Advanced Encryption Standard (AES) [31]. AES has been adopted as an encryption standard by the United States government [32], likewise, it allows a block cipher, making it possible to align the cipher with the block transmission that will be used in the system.

In order to generate a session key safely, asymmetric encryption will be used, since it allows the creation of a secure channel in which the key can be shared. Subsequently, the system will only use this encryption, since it does not present the clear limitations that asymmetric encryption presents.

Machine Translated by Google

3.2 Technological Fundamentals

This section will address the technologies on which the library will be implemented. The reason for their choice will be explained, as well as the particularities of each technology.

3.2.1 Programming Language: Python

Python [33] is an interpreted, multi-paradigm and multi-platform language. It is undoubtedly the most used language for ML, with a huge community and developments related to the sector. Thus, bearing in mind that there is no solution similar to the one to be implemented, this language will be used in order to reach a larger potential audience.

In addition, the syntax and the multitude of libraries that it has will help to abstract from irrelevant details for this project, providing a fast and agile development in comparison with other languages. The most notable libraries that have been used are the following:

- **Threading:** Library used for the concurrency of the system.
- **Pytest:** Allows the creation of unit tests that validate the operation of the system. him.
- **Sphinx:** Software that allows the generation of documentation from the source code.
- **Pycryptodome:** Library that provides the implementation of the main algo cryptographic rhythms.
- **Pytorch:** ML Framework used, in the next section 3.2.2 it will be dealt with in pro its characteristics and reason for choice.
- **Sockets:** Library that allows communication between nodes. Similarly, in section 3.2.3 the reasons for its choice as well as its operation will be discussed.

Finally, it should be mentioned that other languages such as **Julia, Elixir, C++** or R have been raised . As mentioned, discarded in favor of Python essentially for not having such a strong community of ML developers.

Machine Translated by Google

3.2.2 ML libraries

Speaking of frameworks for the creation of RR.NN.AA, Tensorflow [34] and PyTorch [35] are the two great candidates. Both frameworks are open source, allowing easy network programming under a high-level approach and automating the auto-differentiation process. TensorFlow is somewhat more industry-focused, while PyTorch has a reputation for being a research-focused framework.

In this case, PyTorch is better suited to the needs of the project, since it allows easy programming thanks to dynamic computation graphs [36], very similar to how programming is done in python. This makes it very easy to quickly test different models or variations of it, especially thanks to the high levels of abstraction it provides (particularly PyTorch Lightning). Likewise, the community and the adoption of PyTorch in recent years is considerably higher than that of Tensorflow.

3.2.3 Library for handling connections: TCP with sockets

When determining which technology to use for communications, high-level solutions such as FastApi [37] have been proposed. However, sockets have been chosen to handle Transmission Control Protocol (TCP) connections [38].

TCP is the main protocol on which information exchanges in the network are based. It has been chosen to use a low-level protocol due to the flexibility and speed it provides. In addition, the creation of bi-directional full-duplex connections based on data flows will easily adapt to the message passing paradigm to be implemented. However, solutions like FastApi based on TCP as the base protocol introduce unnecessary mechanisms for the project. At the same time, this library is not oriented to message passing, but to the creation of Application Programming Interface (API).

TCP 's design ensures that the communication process can recover from data corruption, loss, duplication, or disorder. Likewise, this protocol allows multiplatform communications in heterogeneous networks, including congestion and timeout control mechanisms.

As mentioned, sockets will be used, since it is the API for the family of Internet protocols (in this case TCP/IP) provided by all modern operating systems. Particularly, the python sockets will be used. This module, in essence, provides access to the socket interface of the operating system under the object-oriented paradigm in python.

Machine Translated by Google

Chapter 4

Methodology and Planning

In the following sections, the feasibility and risk analysis will be detailed first.
Immediately afterwards, the methodology to be followed will be defined, as well as the initial and final planning.
doing with the follow-up of the project.

4.1 Feasibility analysis

As in any project, it will be necessary to carry out an analysis that helps determine the decision of whether to carry it out or not. The use of the analysis Strengths Weaknesses Opportunities Threats (SWOT) [39], will allow to search and analyze, in a proactive and systematic way, all the variables that intervene in the project, in order to determine its feasibility.

Thus, Figure 4.1 shows the SWOT matrix analyzing the situation of the project.
Roughly speaking, the strengths and opportunities will be associated with how innovative and useful the system can be. The usefulness of the system is enhanced taking into account the rise of Artificial Intelligence in all sectors of society. On the other hand, weaknesses and threats are once again subject to the risk involved in creating a new product, such as the lack of information and uncertainty regarding the reception of potential customers.

As discussed in the fundamentals section 3.2, the library is perfectly manageable with the mentioned technologies. Likewise, as will be seen in the following sections, the estimate in time and costs is completely acceptable.

Finally, it is concluded that the project is suitable to be carried out.

Machine Translated by Google

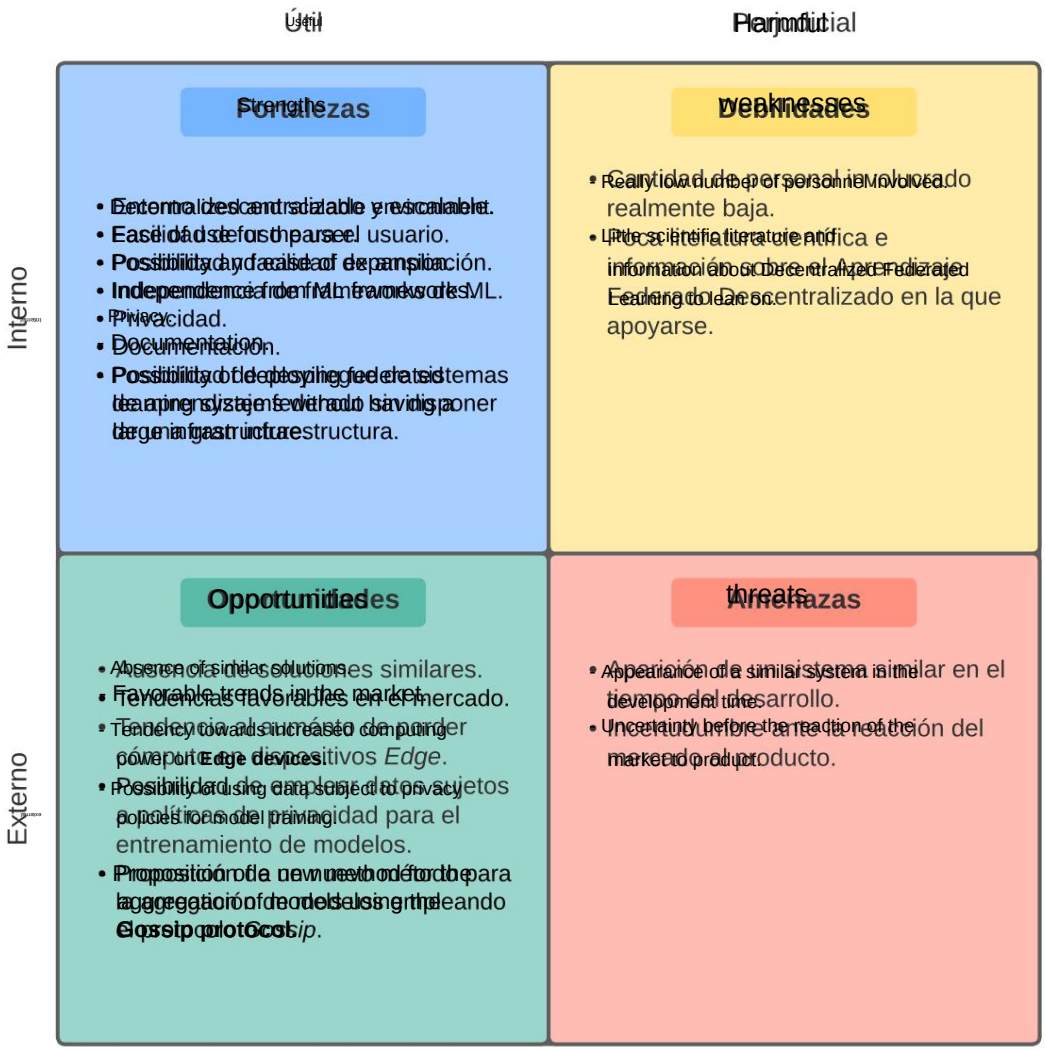


Figure 4.1: Matrix resulting from the SWOT analysis of the project

4.2 Risk analysis

In this section, an analysis of the risks to which the development of the project will be subjected will be prepared. For each risk detected, a probability of occurrence and a level of criticality will be associated. These values will be discrete, the intervals represented by each unit being defined in tables 4.1 and 4.2 respectively. In addition, the consequences that the different risks will cause as well as the palliative or preventive measures to be applied will be discussed.

Machine Translated by Google

spawn level	probability of occurrence
very low	Less than 10%
Low	Between % and 25%
Half	Between 25% and 50%
High	Between 50% and 75%
Very high	more than 75%

Table 4.1: Specification of the probability of occurrence for the different levels of risk.

criticality level	appreciation level
Scarce	Very little is appreciated in the project
Light	Little appreciated in the project
Moderate	It is appreciated in the project
Critical	It is greatly appreciated in the project
Extreme	The project would be canceled

Table 4.2: Specification of the probability of occurrence for the different levels of risk.

In this way, in table 4.3, the risks detected will be listed, being ordered by a descending level of criticality. As can be seen, some risks are unavoidable and others will try to prevent or mitigate their effects.

Risk 1: Appearance of an identical library			
Probability	Low	criticality level	Extreme
Causes	Depending on how similar and solid the bookcase is it could the project is cancelled.		
Precautionary measures	Adoption of agile methodology that allows changing requirements looking for differentiation with the competent bookstore.		
Risk 2: Student withdrawal			
Probability	Media	criticality level	Criticism

Machine Translated by Google

Continuation of table 4.3.

Causes	Depending on the severity, it could lead to long delays in the project.		
Precautionary measures	Assumption of risk.		
Risk 3: Performance of models considerably inferior to classical training			
Probability	Low	criticality level	Critical
Causes	It will mean carefully analyzing the system, trying to detect and fix the errors that cause it.		
Measurements	Continuous testing using toy problems.		
Risk 4: Loss of source code			
Probability	Media	criticality level	Critical
Causes	It will mean redoing all the work.		
Measurements	Use of version control software and backups.		
Risk 5: System too complex			
Probability	Media	criticality level	moderate
Causes	It will mean applying re-engineering, considerably delaying the project.		
Measurements	A multitude of patterns and design principles will be applied.		
Risk 6: Loss or destruction of equipment			
Probability	Low	criticality level	moderate
Causes	It would imply a slight delay and the economic cost of returning to purchase the equipment.		
Measurements	Assumption of risk.		
Risk 7: Withdrawal of a project manager			
Probability	Media	criticality level	moderate
Causes	It might cause some slight delay.		
Measurements	Assumption of risk.		
Risk 8: Rejection of the application for the use of CESGA			

Machine Translated by Google

Continuation of table 4.3.

Probability	Very low	criticality level	moderate
Causes	It will mean dispensing with the results of simulations with high number of nodes.		
Measurements	Assumption of risk.		
Risk 9: Bad estimates in development time			
Probability	high	criticality level	Light
Causes	Depending on the severity, it will induce delays in the project.		
Measurements	Estimation of time with ease, without estimating times too much shortened.		

Table 4.3: Specification of risks underlying the project.

In conclusion, the risks that affect this work, as well as its palliative or preventive measures, are acceptable in order to carry out the project.

4.3 Methodology

To carry out the project, an agile methodology has been used . Specifically, an **adaptation of** SCRUM **[40]** has been used for individual projects. This framework will guarantee rapid achievement of intermediate results through an iterative development divided into sprints. Likewise, SCRUM allows not to specify the requirements initially, and as it is an agile methodology, it will allow a rapid response and adjustment of the project in the event of the possible appearance of new papers or libraries. In this way, SCRUM is highly flexible and fast, fitting perfectly with possible changes in requirements and the nature of the project.

As mentioned, SCRUM is a framework and not a methodology, allowing it to be easily adapted to each project. In this case, it will be slightly modified to fit an individual tutored project.

Machine Translated by Google

4.3.1 Variations on SCRUM

In the project, only the directors and the student will be involved. The student will be responsible for carrying out the work, while the directors will guide, evaluate and validate it. For this reason, the SCRUM roles will have to be adapted. The result is as follows:

- **Development Team :** Role equivalent to that of the original SCRUM, with the exception that there will only be one developer, the student.
- **Responsible for the operation of** SCRUM (Scrum Master): Role carried out by the student, identical to the original SCRUM role.
- **Product Owner** (Product Owner): Role played by the tutors. The Product Owner will manage the backlog and ensure that the work done is appropriate from the perspective of the Final Degree Project (TFG). The user stories will be replaced by **use cases,** since they allow a more detailed specification of what the system should do before the iterations of different actors.

  After finishing each sprint, the Product Owner will intervene evaluating and commenting on the results obtained. Likewise, in case of any setback, sporadic meetings and consultancies will be held. In short, they should always be in contact to cover the gap left by the absence of more opinions.

Due to the availability of the tutors, as well as the fact that the development team is made up of only one member, the daily meetings will be eliminated. However, on the part of the student there must be continuous self-reflection, emphasizing analysis, criticism and adaptation.

Finally, the use of a Kanban board for quick visualization of the project status should be mentioned. In the planning figures, for example in figure 4.3, this board can be seen.

4.3.2 Life Cycle

The life cycle proposed for the elaboration of the project consists of the following phases:

1. **Adding tasks to the** backlog: Firstly, the requirements for desirables will be determined from a high-level point of view. Immediately afterwards, the requirements will be segmented into very specific tasks that will be added to the backlog.

Machine Translated by Google

2. **Organization** of sprints: The backlog tasks will be distributed to different sprints.
3. Sprint **execution :** Development of each sprint. The phases that will make up each sprint will be: **analysis, design, implementation** and **testing.** Its duration will be between 2 and 4 weeks approximately.
4. **Inspection and adaptation:** Analysis and validation of both the results obtained as the process (oriented to improve weaknesses in future sprints).
5. **Analysis by tutors:** Once the sprint is finished, the results will be shown to the tutors. It is one of the most important stages, as it is where feedback from a point of view different from that of the student will be obtained.
6. Return to step **1.** until you finish the backlog tasks and are satisfied with the quality obtained.

4.4 Planning

Despite the dynamism of the methodology used, for the initial estimation and organization of the project, a **tentative** Gantt **chart** has been created , which is detailed in tables A.1, A.2 and A.3. The sprints proposed in this temporary planning will be dealt with in the next subsection. As can be seen, the start of the project is dated **February 21, 2022,** with **July 27, 2022** being the estimated completion date.

Regarding the costs, tables 4.4, 4.5 and 4.6 detail the human, material and total costs, respectively. It should be noted that the salaries for each profile have been obtained by averaging the Spanish salary offer on talent.com.

Equipment	Dedicated time (h)	Price (€/ h)	Total cost (€)
Analyst	224	15,36	3360
Programmer	920	14,42	13432
Director	108	25,73	2778,84
		Total	19570,84

Table 4.4: Table with the estimated costs of personnel in initial planning

Machine Translated by Google

Equipment	Time used (h)	Price (€/ h)	Total Cost (€)
MacBook Pro 13”	1252	0,169	211,588
Desktop computer	40	0,219	8,76
Raspberry Pi 4	40	0,007	0,28
Blind Knot	3	1,92	5,76
		Total	226,388

Table 4.5: Table with the estimated costs of the necessary equipment

Estimated cost associated with personnel	19570,84
Estimated cost associated with the equipment	226,388
Total	19797,228

Table 4.6: Table with the estimated initial total costs of the project

4.4.1 Sprints

As mentioned in the previous section 4.3, the development of the library will be divided into brief periods of time, resulting in an incremental iterative development. Next, the time intervals or sprint initially proposed for the project will be detailed. It should be noted that during development, this planning may be modified.

Sprint **0: Project preparation**

Initial sprint of the project and one of the most dense. In this, the theoretical bases and technologies to be applied will be studied. These tasks will be essential to analyze the feasibility and risks of the project as well as carry out a market study. Finally, the tentative initial planning of the project has been carried out. Figure 4.3 shows the tasks that will make up the sprint.

Machine Translated by Google

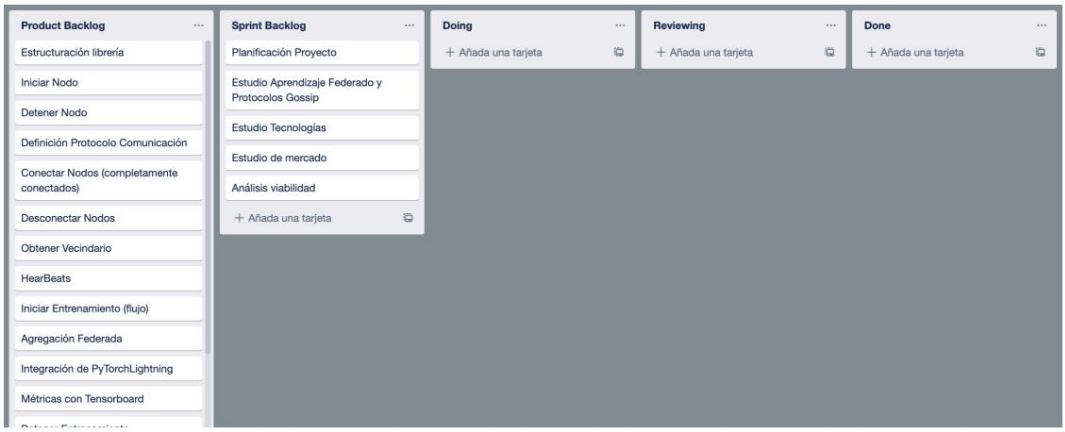


Figure 4.2: Kanban board for visualizing the SCRUM methodology in the preparation sprint.

Sprint **1: Fully connected** p2p architecture

First sprint dedicated to development. In this, it will seek to structure the library and create a base p2p system on which to implement Federated Learning in future iterations. Again, Figure 4.4 shows the different tasks that make up the sprint.

Note that in order to facilitate development, the nodes will be organized in a temporary fully connected architecture.

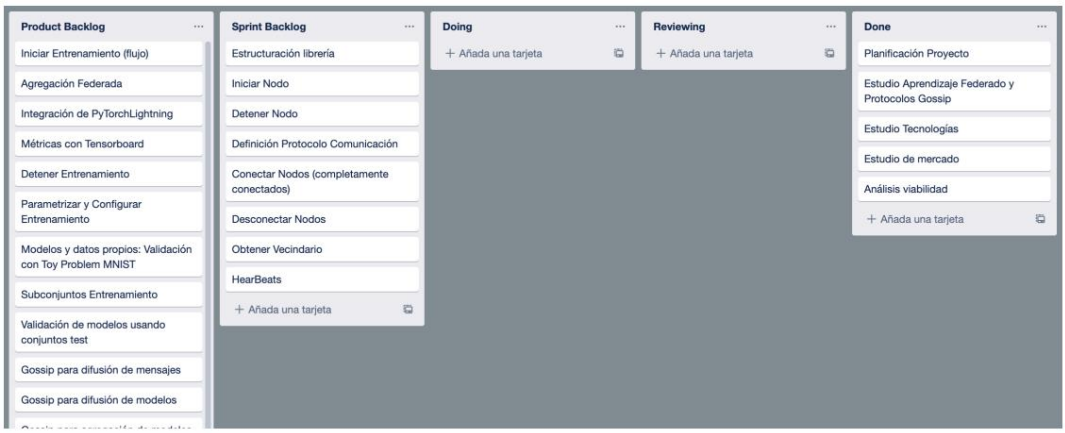


Figure 4.3: Kanban board to visualize the SCRUM methodology in the 1st sprint.

Machine Translated by Google

Sprint **2: Federated Learning**

Sprint for the inclusion of Federated Learning in the p2p network. Observe in figure 4.5 the different tasks that make up the sprint.

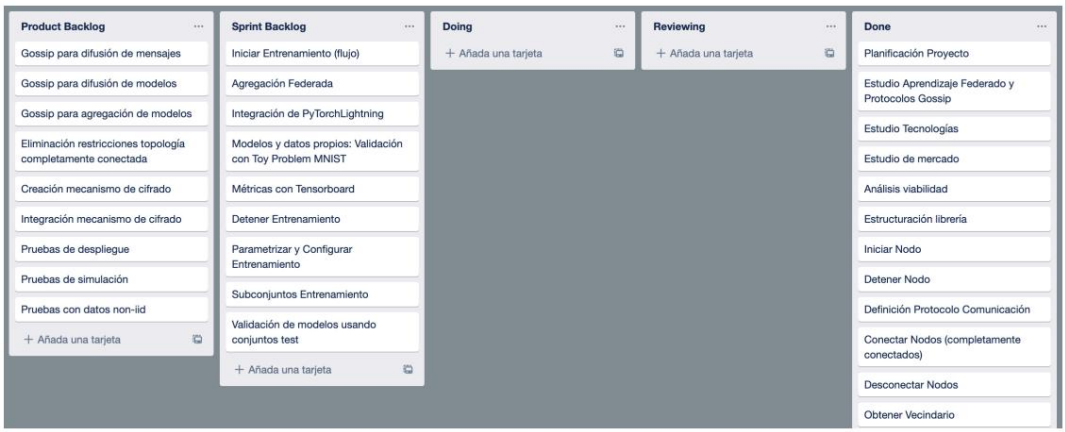


Figure 4.4: Kanban board to visualize the SCRUM methodology in the 2nd sprint.

Sprint **3: Protocolo** Gossip

The third development sprint of the project will aim to implement and integrate the Gossip protocol in p2p network communications . Similar to previous sprints , Figure 4.5 details the tasks for this iteration.

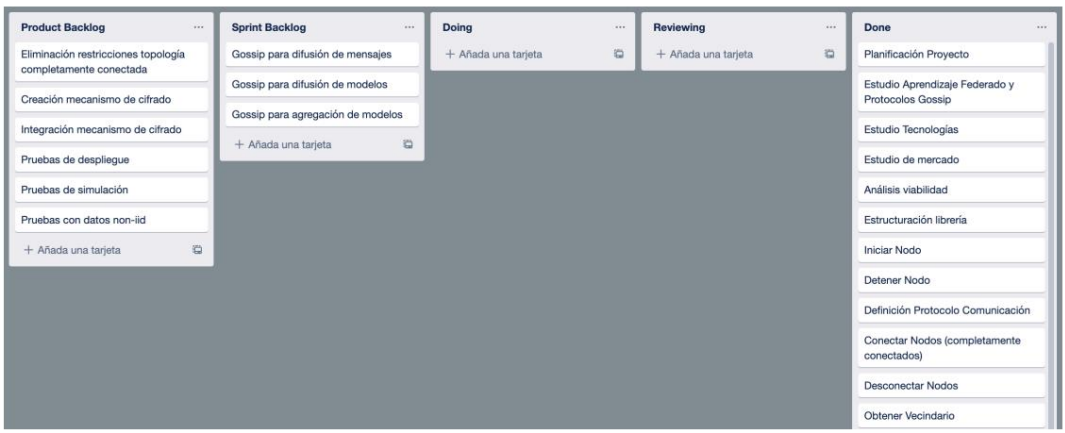


Figure 4.5: Kanban board for visualizing the SCRUM methodology in the 3rd sprint.

Machine Translated by Google

Sprint **4: Topology change**

The objective of this 4th sprint will be to eliminate the fully connected topology temporarily implemented in the 1st sprint. Thanks to already having the Gossip protocol implemented, you will only have to adapt the system, not create functionalities as such. Again, in figure 4.6, you can see the Kanban board with the tasks for this sprint.

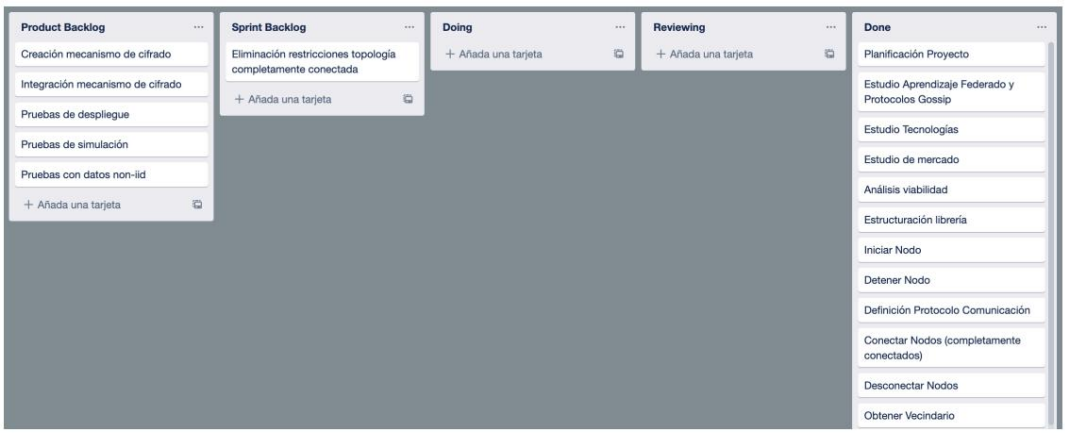


Figure 4.6: Kanban board for visualizing the SCRUM methodology in the 4th sprint.

Sprint **5: Encryption in communications**

Last development sprint . In this, it will seek to encrypt communications between nodes of the p2p system, seeking to provide the system with confidentiality and privacy in communication. See Figure 4.7 for the tasks to be performed.

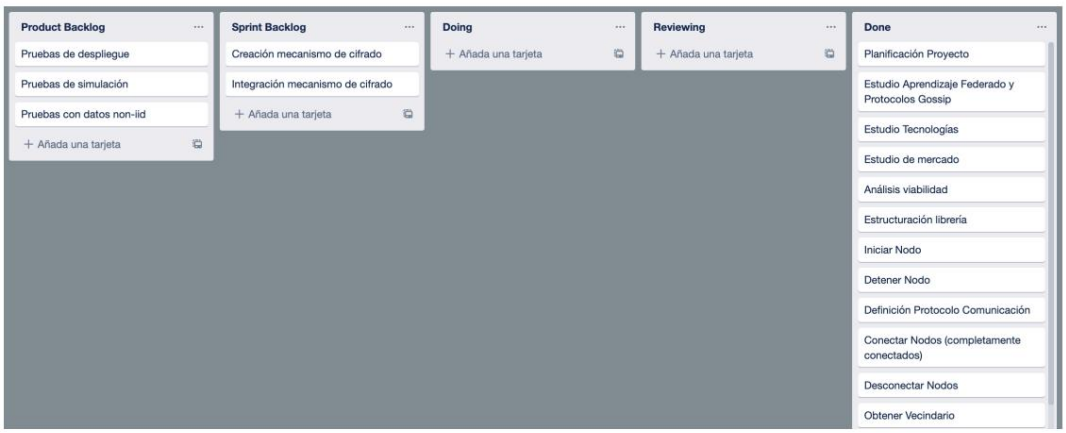


Figure 4.7: Kanban board for visualizing the SCRUM methodology in the 5th sprint.

Machine Translated by Google

Sprint **6: Practical Application**

In this last sprint of the project, different tests will be executed, validating and studying the behavior of the library developed in different environments. Again, in figure 4.5 you can see the Kanban board with the different tasks that make up the sprint.

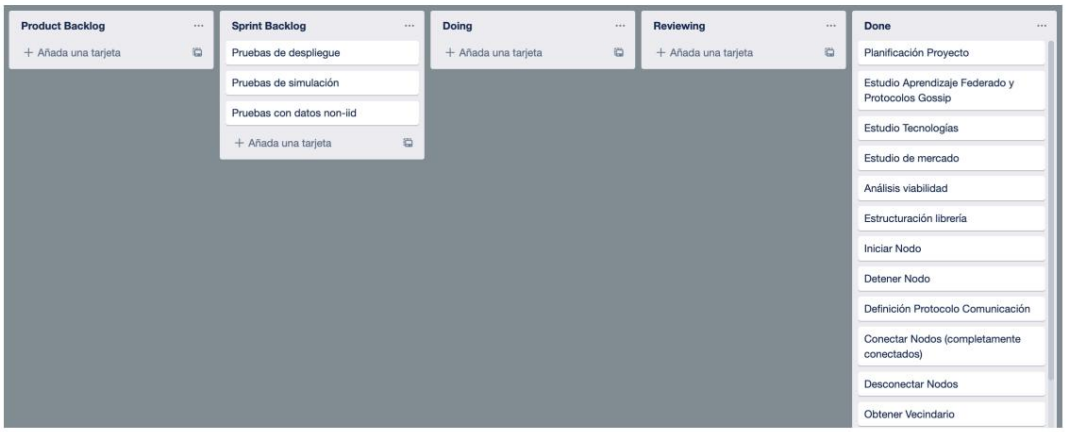


Figure 4.8: Kanban board for visualizing the SCRUM methodology in the 6th sprint.

4.5 Monitoring result

Because there have been no changes to the requirements, the tentative planning has generally been quite successful. However, there has been a delay of one week in the 2nd sprint. Trying not to overload said sprint, the validation tasks with test sets and creation of training sets will be transferred to a new sprint that will precede the third.

To visualize the progress of the project, a Burn Down Chart will be used. This is used in agile methodologies like SCRUM to visualize the pending work. Backlog tasks will be quantized proportionally to their associated amount of work, subtracting that value from the total work when completed. Thus, Figure 4.9 shows the progress of the project compared to its ideal execution. Appreciate the delay that occurs with the start of the 2nd sprint, from which the project will not recover.

Machine Translated by Google

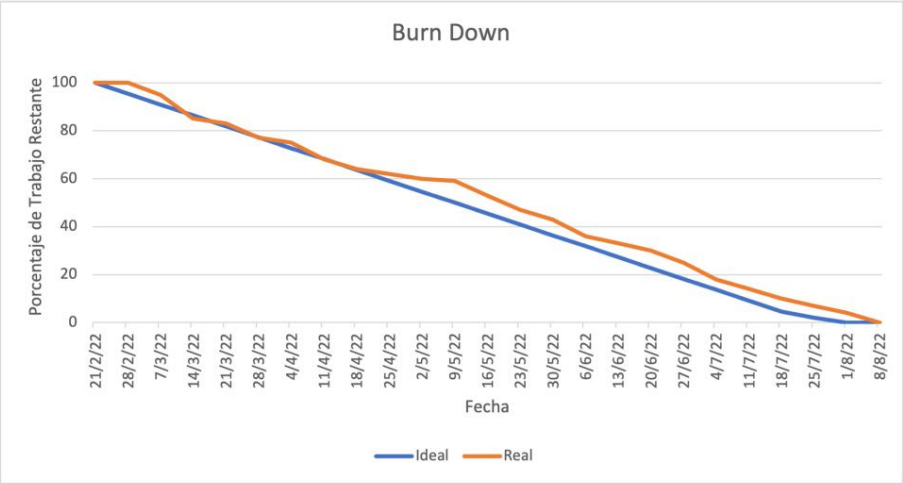


Figure 4.9: Project Burn Down Chart

Similarly, in the follow-up Gantt charts A.4, A.5 and A.6 the variation with the tentatively planned tasks can be seen. Again, the delay induced by the inclusion of the previously mentioned sprint should be appreciated , the other estimates being correct actions.

The costs involved in the delay have been negligible. The human, material and total costs will be detailed in tables 4.7, 4.8 and 4.9 respectively.

Equipment	Dedicated time (h)	Price (€/ h)	Total Cost (€)
Analyst	232	15,36	3563,52
Programmer	987,2	14,42	14235,424
Director	116	25,73	2984,68
		Total	20783,624

Table 4.7: Table with the final costs of personnel in initial planning

Machine Translated by Google

Equipment	Time used (h)	Price (€/ h)	Total Cost (€)
MacBook Pro 13”	1335,2	0,169	225,6488
Desktop computer	40	0,219	8,76
Raspberry Pi 4	40	0,007	0,28
Blind Knot	3	1,92	5,76
		Total	240,449

Table 4.8: Table with the final costs of the necessary equipment

Final cost associated with personnel	20783,624
Final cost associated with the equipment	240,449
Total	21024.0728

Table 4.9: Table with the final total costs of the project

Machine Translated by Google

Chapter 5

Library Development

This chapter will detail the necessary steps for the elaboration and validation of the library of support for Federated Learning on p2p networks. Being consistent with the methodology, there will be a section for each sprint. The central sprints will address the development of the system, devoting the first one to the preparation of the project and the last one to the development of tests on the developed system.

5.1 Sprint 0: Project Preparation

Initial project preparation sprint . In this time interval, various activities have been carried out prior to the development of the library. Therefore, tasks related to planning, market study, feasibility analysis and risk analysis of the project have been carried out, as well as study tasks on the concepts and technologies to be applied (see chapters 2, 3 and 4 respectively) .).

In the next section, only the library requirements will be covered, since most of the work done in this sprint has already been covered in memory.

5.1.1 Requirements Analysis

In this section the requirements of the library will be specified. These requirements are largely related to the objectives of the project.

Machine Translated by Google

Functional Requirements

RF1 The user will be able to manage a node, I understand as such the power to start and stop its operational.

RF2 The user will be able to manage the neighborhood of a node. That is, you can connect and connect from other nodes as well as get your current neighbors.

RF3 The user will be able to manage Federated Learning in the network through a node, starting or stopping it. At startup, the user will need to set the number of rounds and epochs.

RF4 The user will be able to view information on the learning process and status. Also, you can view graphs and training metrics.

RF5 The nodes will self-organize in the learning process. The choice of candidates for model improvement and validation in each round will also be self-managed.

RF6 The user will be able to add or modify the model and the data, allowing an easy integration Gration with any machine learning framework.

RF7 The user will have the possibility of creating and using different federated aggregators of simple way.

RF8 The user can easily set values to the system parameters. These parameters will be relative to gossip, timeouts, and Federated Learning configuration values, among others.

RF9 When starting a node, the user may indicate whether his future activity will form part of a simulated or deployed environment. The main difference will reside in the fact that for the simulated environments, elements that produce an unnecessary overhead for a simulated execution will be dispensed with. For example, in simulated environments, encryption must be dispensed with.

Non-functional Requirements

RNF1 **Availability:** Non-functional requirement implicit in p2p.

RNF2 **Robustness:** The system must be fault tolerant, continuing with its operation despite the appearance errors.

Machine Translated by Google

RNF3 **Scalability:** There should be no performance problems with the increase in the number of nodes.

RNF4 **Privacy:** Since privacy is one of the most desirable features of Federated Learning, additional mechanisms will be included to preserve privacy in communications.

RNF5 **Extensibility:** The library must be designed for a simple extension of its features. Therefore, they will highlight the possible inclusion of new ML libraries as well as the expansion of federated aggregation algorithms.

RNF6 **Usability:** The use of the library must be oriented to the user, allowing an intuitive use of it. The existence of **documentation** will facilitate the initial learning curve.

RNF7 The nodes must be able to be executed by low performance devices.

RNF8 The behavior by the node will be completely asynchronous and non-blocking, being executed in the background.

5.2 Sprint **1: Fully connected** p2p architecture

This sprint will focus on the creation of the initial structure of the library, as well as a simple p2p system (see section 3.1.3). Despite the fact that no learning functionality will yet be incorporated, the design of the network and the communication protocols must take into account their imminent inclusion.

5.2.1 Analysis

The use cases to be implemented in this sprint are found in the use case diagram in Figure 5.1, and are detailed in Table 5.1. It should be noted that only basic use cases for p2p network management are addressed , with only one actor: the library user. This is because the user will be the only one capable of interacting with the p2p network through the nodes that they have.

In order to facilitate the initial development, a completely connected topology will be used. This decision will be temporary, as a fully connected network has very limited scalability.

Machine Translated by Google

CU-1: Nodo starter	
Description	The user starts the node. It involves your creation.
Requirements addressed	RF1, RF9.
Preconditions	The node must not be started.
postconditions	The node will listen for incoming connections.
Alternative flow	The node will not be started as it is already started.
CU-2: Stop Node	
Description	The user stops the operation of a specific node.
Requirements addressed	RF1.
Preconditions	The node must be started.
postconditions	All processes that the node could be executing will be arrested.
Alternative flow	The node will not stop as it is already stopped.
CU-3: Connectar Nodos	
Description	The user connects one node to another. In this case, to guarantee a fully connected topology will also connect to all neighboring nodes of the node against which a connection is to be made.
Requirements addressed	RF2.
Preconditions	The nodes to connect must be started and there must not be a connection among them.
postconditions	The nodes must be connected.
Alternative flow	The connection will not be made.
CU-4: Disconnect Nodes	
Description	The user disconnects one node from another.
Requirements addressed	RF2.
Preconditions	The nodes to disconnect must be started and connected.
postconditions	The nodes should become disconnected.
Alternative flow	The nodes will not be disconnected because they are not connected.

Machine Translated by Google

Continuation of table 5.1.

CU-5: Get Neighborhood	
Description	The user can request the list of neighbors of the node.
Requirements addressed	RF2.
Preconditions	The node must be started.
postconditions	None.
Alternative flow	An empty list will be returned since an unstarted node has no neighbors.

Table 5.1: Table of use cases implemented for the 1st sprint.

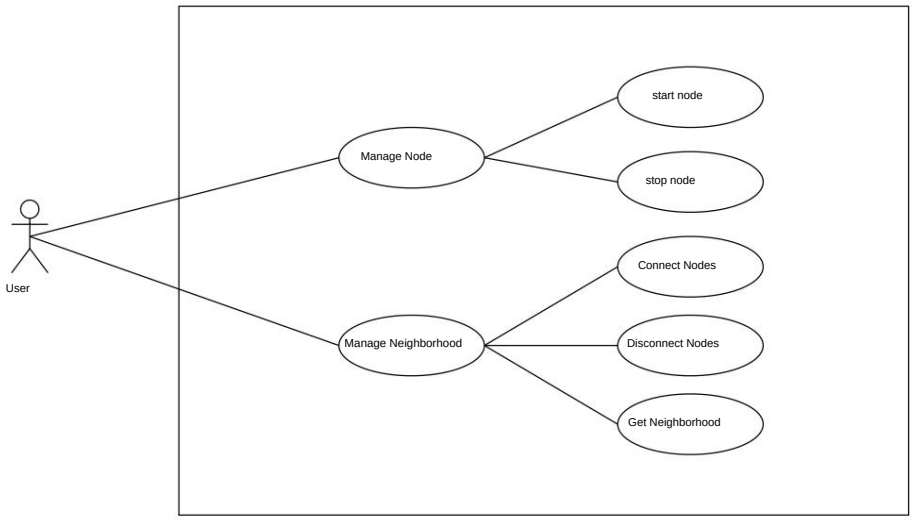


Figure 5.1: Diagram of use cases of the 1st sprint.

5.2.2 Design

An attempt has been made to create a robust and expandable base, designed in a flexible and simple way, seeking to facilitate the use, understanding and modification by the user. In this way, in figure 5.2 the proposed design can be observed, its main classes being the following:

- BaseNode: Base node. It implements basic functionalities of a p2p node.
- NodeConnection: Class in charge of managing the connections between nodes.

Machine Translated by Google

- Heartbeater: Implements a basic heartbeat mechanism.
- CommunicationProtocol: Class in charge of managing the communication between the two. You will define a message protocol so that when a message is valid, it will execute the corresponding code.

  Since the communication between nodes is one of the parts that will be expanded the most, it has been designed using the **command pattern,** looking for easy extensions and modifications in the command repertoire.
- Events, Observable and Observer: In order to minimize coupling, the **observer pattern has been applied.** This will be used to control the different events that can happen in the BaseNode components. Again, the pattern guarantees a trivial scaling process.

To understand in more detail the operation of the nodes, the connection process is detailed in the sequence diagram in Figure B.1 . Special attention should be paid to how the node receives a message for the connection, using CommunicationProtocol for understanding, and the purpose of the loops: the BaseNode loop to create new connections and the NodeConnection loop to receive messages from a connection.

Next, some design aspects related to two of the most important points of the library will be discussed, such as: the communication protocol and robustness against possible falls or failures of this.

Communication protocol

As previously mentioned, it has been decided to use TCP as the base protocol for handling communications (see section 3.2.3 for the justification of the decision). This protocol will therefore be in charge of managing the transmissions between the different nodes. Transmissions will be encoded in 8-bit Unicode Transformation Format (UTF-8) because it allows any unicode character to be represented. The messages proposed for the communications protocol will be treated below:

- BEAT: Message used for the heartbeat mechanism. When it takes a certain amount of time without receiving any, some kind of execution hiccup will be assumed.
- CONNECT <ip> <port> <full>: Message used to establish the connection between two nodes. The address and port of the node that originates the message is indicated to facilitate

Machine Translated by Google

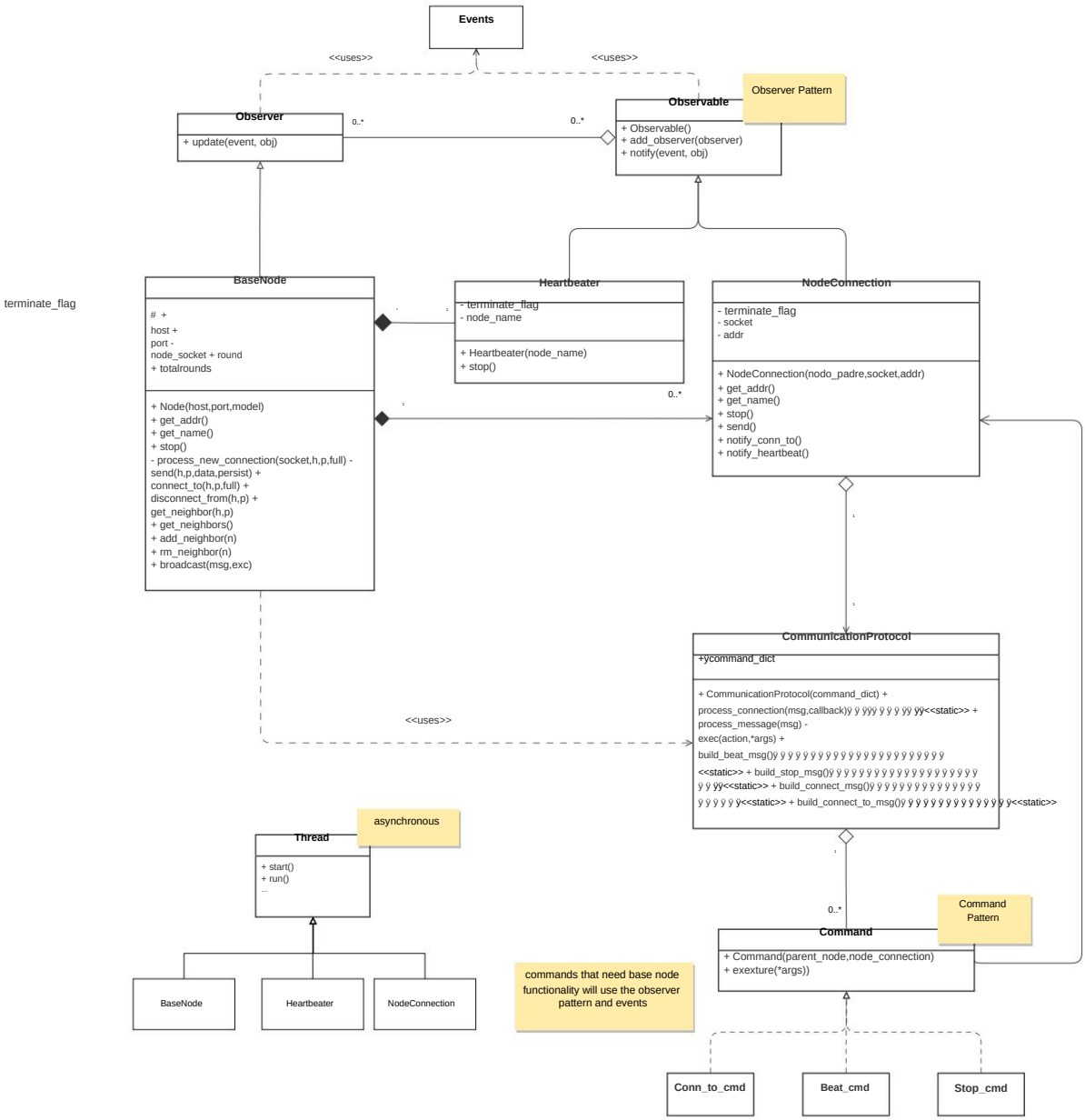


Figure 5.2: Class diagram of the 1st sprint

tar connection handling. If the full option is set, CONNECT_TO messages will be broadcast to all neighbors. Doing this with all connections will result in a fully connected network.

- CONNECT_TO <ip> <port> Message used to tell a node to initiate a connection with another.

- STOP: This message ends the connection between two nodes.

Machine Translated by Google

It should be noted that the CommunicationProtocol class operates as the invoker component within the **command pattern,** receiving messages from TCP connections and computing the corresponding commands. Figure B.2 of the next sprint will show how this component works in a sequence diagram.

Robustness

At this point, as a Federated Learning process is not running yet, abrupt crashes will not have major consequences. However, it has ensured that drops do not harm the system.

With the use of heartbeats and the control in TCP connections , it will be possible to know if a certain node is working correctly, and if not, eliminate it from the neighborhood.

5.2.3 Implementation

First of all, the code has been structured and organized following the python standards [41]. Wheel has been used, one of the tools provided by the python ecosystem to distribute packaged libraries (wheel packaging standard [42]). So, too, highlights the sphinx configuration for the library's documentation.

The threading libraries for asynchronism and socket have been used to manage TCP connections , being a trivial implementation, since the main objective has been the design. However, it is worth noting a couple of elements within the implementation, such as:

- For each neighbor that has a node, one connection will be left open.
- Heartbeats are not processed, they are used to reset an already implemented timeout by the socket class (when receiving a message it is reset).
- If the connection of one node with another is interrupted, it will not rise, breaking the completely connected structure. It has been decided to leave it that way since the fully connected topology is only temporary.
- Receiving messages using sockets works with buffers, that is, when a message is received, a certain number of bytes are actually read from the buffer.

  This operation combined with the use of threads can cause several messages to be sent at the same time, accumulating them in the reception buffer. Multi-message reading and processing has mitigated this problem.

Machine Translated by Google

5.2.4 Tests

Multiple unit tests have been carried out, which have sought to validate the functionality and check the fault tolerance of the basic p2p system. Table 5.2 shows a description of the tests carried out, validating, apart from functionality, fault tolerance. The tests have shown that the system works well, providing a coverage of 91% of the code.

Test 1.1: Connecting nodes	
Description	Two nodes will be started and connected, validating that they are added to the neighborhood.
Results obtained	Two perfectly connected nodes.
Test 1.2: Connection to invalid nodes	
Description	It will try to connect to a node with an invalid address, making sure that no new node is added to the neighbor list.
Results obtained	A node with no new neighbors.
Test 1.3: Connection fully connected nodes	
Description	Four nodes will be connected and disconnected, validating that a fully connected topology.
Results obtained	Fully connected nodes, respecting the topology.
Test 1.4: Multimessage validation	
Description	Multiple messages will be sent to a node in a single delivery. The recipient node You must understand its meaning without making mistakes.
Results obtained	Comprehension of multiple messages without errors.
Test 1.5: Validation of erroneous messages	
Description	Given two connected nodes, an erroneous message will be created, validating that the nodes nodes are disconnected before the occurrence of the same.
Results obtained	Two nodes disconnected.
Test 1.6: Validation of operability without errors when they occur	
Description	Given four fully connected nodes, errors will creep in. in them, looking for the disconnection of nodes so as not to harm the system. The errors introduced are: abrupt closure of the socket and stopping of the heartbeater.
Results obtained	Control of errors by the nodes.

Table 5.2: Explanatory table with unit tests carried out in sprint 1.

Machine Translated by Google

5.3 Sprint 2: Federated Learning

This sprint will aim to implement Federated Learning on the p2p network already created
and.

5.3.1 Analysis

The new use cases to be implemented will essentially allow the start and stop of learning, allowing the modification of elements involved in it and the visualization of results. Figure 5.3 shows the updated use case diagram, being
The new use cases are detailed in Table 5.3.

CU-6: Start Training	
Description	The user will initiate the Federated Learning process on the nodes connected to the network. The number of rounds and epochs must be specified To make. The model to be initialized in the network will be the model of the node to start the process.
Requirements addressed	RF3.
Preconditions	The node must be started and without any learning processes running.
postconditions	The learning process will be initiated in the entire network of nodes.
Alternative flow	The learning will not start.
CU-7: Stop Training	
Description	The user will stop the Federated Learning process on the nodes connected to the network.
Requirements addressed	RF3.
Preconditions	The learning process must be running.
postconditions	All processes related to learning have to end.
Alternative flow	Learning won't stop
CU-8: View Metrics and Information Related to Training	

Machine Translated by Google

Continuation of table 5.3.

Description	The user will be able to consult and view information related to the process of learning as well as information and metrics related to the performance of the Models. The metrics related to the test set will not yet be addressed for the model validation.
Requirements addressed	RF4.
Preconditions	At least one workout must have been started.
postconditions	None.
Alternative flow	There will be no information to be displayed.
CU-9: Establish Model and Data	
Description	The user will be able to establish the data set as well as a model in specific.
Requirements addressed	RF6.
Preconditions	Have a node.
postconditions	The model and data of the node will be established.
Alternative flow	The use case cannot be executed.
CU-10: Establish Federated Aggregator	
Description	The user may set an arbitrary federated aggregation algorithm.
Requirements addressed	RF7.
Preconditions	Have a node.
postconditions	The aggregator of the node will be the established one.
Alternative flow	The use case cannot be executed.
CU-11: Configure Experiment	
Description	The user can establish parameters related to the operation of the system.
Requirements addressed	RF8
Preconditions	Have at least one node.
postconditions	The operating parameters will be those established.

Machine Translated by Google

Continuation of table 5.3.

Alternative flow	The use case cannot be executed.

Table 5.3: Table of use cases implemented for the 2nd sprint.

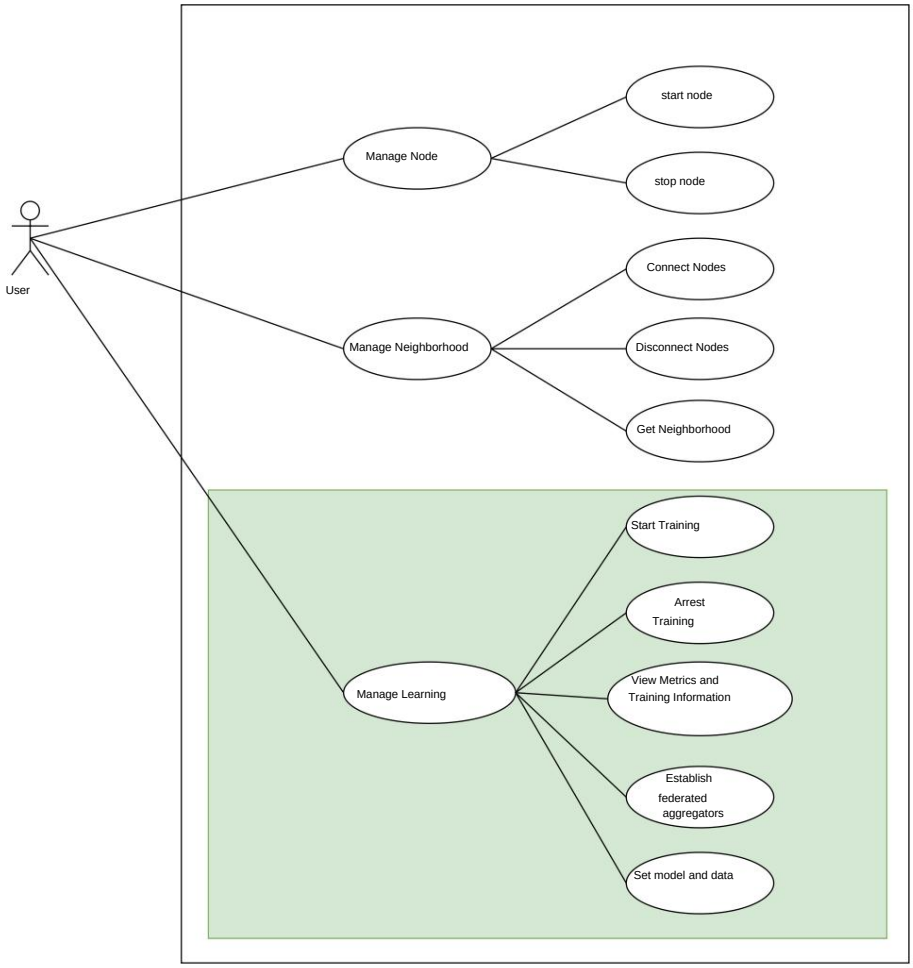


Figure 5.3: Diagram of use cases implemented up to sprint 2. The use cases in the green zone are the new use cases included in the 2nd sprint.

5.3.2 Design

Continuing with the design lines of the previous sprint , an attempt has been made to integrate Federated Learning in a consistent and extensible way, prioritizing good design. In figure 5.4, the corresponding class diagram is detailed, the main added classes being the following:

Machine Translated by Google

- Node: Wide BaseNode allowing the operability of a Federated Learning system over a p2p network.
- NodeLearner: Interface that provides the management, encoding and decoding of ML models . This class is what is known as a template in the **homonymous pattern.**
- LightningLearner: As mentioned in section 3.2, it has been decided to use PyTorch as the ML framework. Therefore, this class is an implementation of NodeLearner using PyTorch Lightning.
- FederatedTensorboardLogger: As its name indicates, it allows logging of all the information related to federated executions. A basic PyTorch logger has been ported to a federated environment.
- Datamodules and Models: The models and data used will be explained in their correspondence. Pondiete apart in the test sprint 5.8.1. Highlight the use of the singleton **pattern** in loading data with MnistFederatedDM to avoid in read operations
  needed in simulated environments.
- Aggregator: Class in charge of performing the aggregation. Its design tries to guarantee a simple extensibility thanks to the **insole pattern.**
- FedAvg: Extends Aggregator, implementing the Fe federated aggregation algorithm dAvg discussed in section 3.1.2.
- Settings: Since we are working with a multitude of variable parameters to be configured by the user, it has been decided to condense them all into the same class.

To further understand the full operation of the system, Figures B.2 and B.3 detail sequence diagrams for two of the most important parts of this sprint: the learning initiation sequence and model aggregation.

The first diagram details the steps of learning and communication between a pair of nodes, where the first node initiates the learning process. Once again, special attention must be paid to how the messages received in NodeConnection are understood using CommunicationProtocol and executed through the corresponding command. As for the second diagram, it shows the process of receiving a model. Two circumstances can be appreciated, the initialization of the model or the aggregation of the model using FedAvg.
In addition, it details how the aggregation timeout starts.

In the same way as in the previous sprint , the most important design aspects related to communication protocol and robustness will be discussed.

Machine Translated by Google

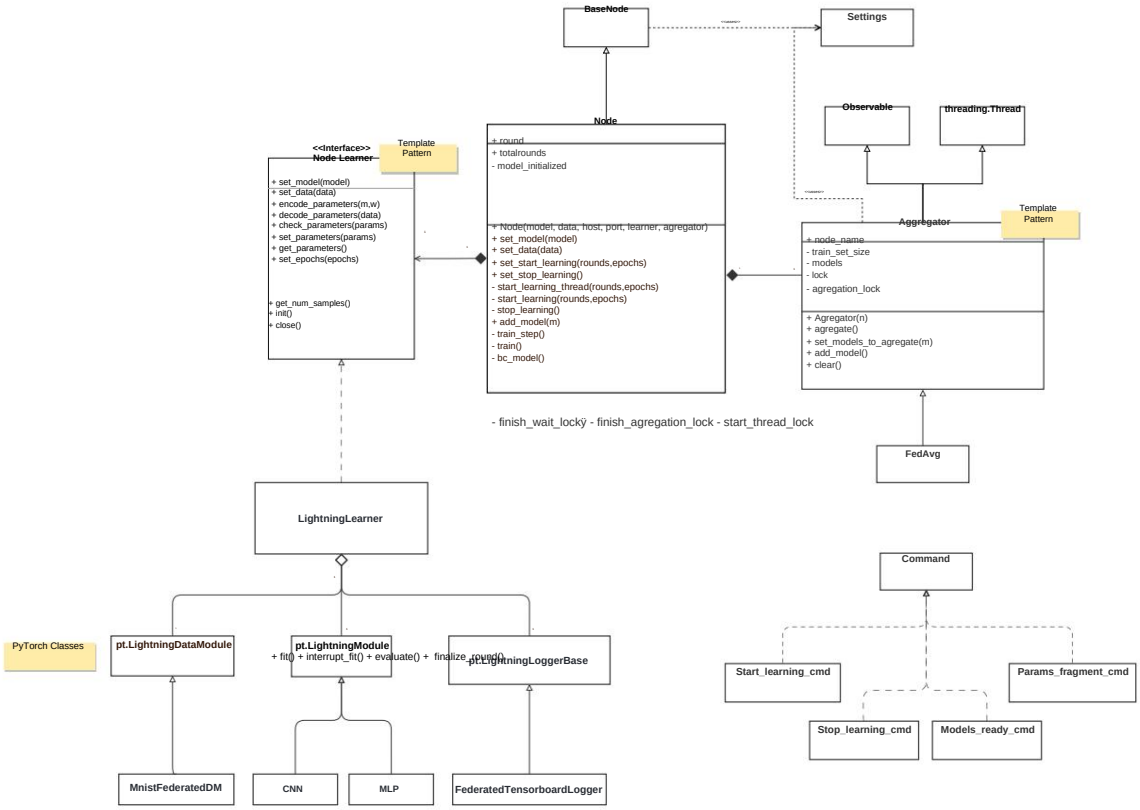


Figure 5.4: Class diagram for the 2nd sprint.

Communication protocol

Because model weights need to be transmitted, the communication protocol will support a new type of message. New messages added to the protocol can be classified
in:

- **Text encoded in** UTF-8: Identical to the messages already created. new messages incorporated are:
  - START_LEARNING <rounds> <epochs>: Indicates the start of the learning process, as well as the number of rounds and epochs per round.
  - STOP_LEARNING: Communicates the abrupt termination of the learning process.
  - MODELS_READY <round>: The node that sends the message indicates that it already has the model added for the round.
- **Serialized Models:** The models and the number of samples used will be serialized, fragmented and sent in different messages. Its structure will be the following:

Machine Translated by Google

- PARAMS <data>: For all fragments except the last one.
- PARAMS <data> \PARAMS: For the last message, it indicates the end of the sending.

Robustness

Starting from the robustness in the connections previously created, this same problem will be dealt with but during the training, contemplating new situations. In this way, the errors that can occur are the following:

- **Node crashes during training:** Nothing can be done. The node will be removed from training.
- **Connection drops during training:** Actually, a node cannot differentiate if another has completely dropped or it has only been the connection between the nodes. themselves.

  Mechanisms have been proposed to control the models added by node, so that at the end of all the nodes, the number of models added is validated, equalizing them if necessary. Finally, following the Keep It Simple, Stupid! (KISS) and considering the imminent implementation of Gossip protocols will mitigate these errors by design, it has been decided not to implement such a mechanism.

  It should be noted that connection drops will not have major consequences. The fact that it starts from a common initialization will minimize future variations between models and the consequences are summed up in a possible worse performance of the model.

- **Incompatible Models:** Error detected in the initialization of the model. The model that has not started the training will be stopped for reasons of incompatibility.
- **Erroneous Binaries:** TCP provides reliable communication, so they should not be There are problems in the transmission of messages.

To deal with unforeseen errors or performance problems by nodes or intermediate network components, timeout mechanisms have been included. In this sprint, it has only been integrated into the aggregation process, starting the countdown from when the first model is added, including the local one.

Machine Translated by Google

5.3.3 Implementation

First of all, the communications and the learning flow have been implemented, performing a simple averaging. In this first contact, the python threading library should be highlighted again, since the aggregation and training processes are processed in the background. Likewise, Locks have been used to make critical sections thread safe and create synchronization mechanisms.

After the implementation of the Federated Learning flow, the different Pytorch Lightning modules have been created. For the transmission of models, different methods have been tested, serialization being the method with the best results. The pic kle library has been used, since it includes methods for serialization and de-serialization of python object structures.

The transmission of the models has generated many problems. The serialized parameters have been fragmented so that they occupy the maximum message size, causing reception problems if they coincide with smaller messages in the buffer. This will completely misalign the read fragments inducing de-serialization errors.
To solve this failure, collisions of fragments and messages will be detected. Upon detection of collisions, the text messages will be processed and, subsequently, only the remaining bytes of the model fragment to be processed will be read from the buffer.

Another problem has also arisen in receiving model fragments. This is because the reception of messages in TCP with sockets does not guarantee that a fragment of the same size as the one sent will be read. Thus, it has been necessary to include mechanisms that guarantee the size of the block.

Regarding fault tolerance, contemplating the situations raised in the previous section and the use of timeouts, the failures that could occur have been satisfactorily controlled.

With the learning process finished, we proceeded to create a logger using TensorBoard that would allow the user to view metrics and training information in a simple and visual way. The logger has been implemented starting from the LightningLoggerBase base class, residing its peculiarity in an adaptation to be persistent and consistent before different training sessions in each round of learning.

Finally, it should be mentioned that the number of open files will increase with the size of the experiment, and may exceed the limitations of the operating system. A simple command, ”ulimit -n {MAX_OPEN_FILES}”, would suffice to fix it, be careful when removing security restrictions imposed by the operating system.

Machine Translated by Google

5.3.4 Tests

First of all, it should be made clear that a highly fault-tolerant system causes tests to lose potential, failing only in catastrophic situations.

In a similar way to the previous sprint , different unit tests have been created, which can be seen in table 5.4. The objective of the tests will be to validate the operation without errors of the system, both with normal operation and for executions with induced errors. The analysis of performance and results will be dealt with in the sprint dedicated to tests 5.8.

Test 2.1: Encode and decode parameters.	
Description	Parameters will be encoded and decoded, validating that the parameters decoded are identical to the originals.
Results obtained	Decoded parameters identical to the original.
Test 2.2: Simple averaging using the aggregator.	
Description	Different values will be averaged using the aggregator, validating that the results are the same as expected.
Results obtained	Values obtained identical to those expected.
Test 2.3: Model averaging.	
Description	Different values will be averaged using the aggregator, validating that the results are the same as expected.
Results obtained	Models obtained identical to those expected.
Test 2.4: Convergence validation.	
Description	Parameterized test (N=1/2 X=2). Given a network with N nodes, it will start a training of X rounds. It will be checked that the resulting models end of training are the same for all nodes.
Results obtained	Training completed with identical models for all nodes.
Test 2.5: Interruption of training.	
Description	A training with a high number of rounds will start. Subsequently, training will be stopped.
Results obtained	Training stopped.
Test 2.6: Fall of a node during training.	
Description	Given four connected nodes the learning will start. Immediately afterwards, one of these nodes will suddenly go offline.

Machine Translated by Google

Continuation of table 5.4.

Results obtained	Continuation of the learning process despite the failure of one of the nodes participants.
Test 2.7: Model decoding error.	
Description	Bytes will be added in the model step that cause an error in the decoding of it. It will validate that the affected node is stopped, disconnecting from the rest.
Results obtained	Disconnection of the node affected by the error in the decoding of the model.
Test 2.8: Learning with different models.	
Description	Given two nodes, each with a different model, the learning. The node that does not start learning will be validated to stop, disconnecting from the rest.
Results obtained	Disconnection of the node that does not start the learning process.

Table 5.4: Explanatory table with unit tests carried out in sprint 2.

5.4 Sprint 3: Test Set and Subsets

The goal of this sprint is to add Federated Learning functionality not addressed by the previous iteration.

5.4.1 Analysis

In essence, the learning steps will be modified to allow a correct evaluation of the resulting models using test sets. The subsets will also be included in the training, allowing the parameterization of the participating nodes in each round.

As a direct consequence, this sprint will not implement new use cases, but will slightly modify existing ones. The modified use cases are specified in Table 5.5.

CU-3: Connectar Nodos	
Description	The user connects one node to another. In this case, to guarantee a fully connected topology will also connect to all neighboring nodes of the node against which a connection is to be made.

Machine Translated by Google

Continuation of table 5.5.

Requirements addressed	RF2.
Preconditions	The nodes to connect must be started, there must not be a connection between them and none must be executing a learning process.
postconditions	The nodes must be connected.
Alternative flow	If the preconditions are not met, the connection will not be executed.
CU-6: Start Training use case	
Description	The user will initiate the Federated Learning process on the nodes connected to the network. The number of rounds and epochs must be specified to perform, being other parameters such as the size of the set of read settings training. The model to initialize in the network will be the model of the node that starts the process.
Requirements addressed	RF3, RF5.
Preconditions	The node must be started and without any learning process in progress. execution.
postconditions	None.
Alternative flow	The learning will not start.
CU-8: View Metrics and Information Related to Training	
Description	The user will be able to consult and view information related to the process of learning as well as information and metrics related to the performance of the models.
Requirements addressed	RF4.
Preconditions	Training must have started at least one training.
postconditions	None.
Alternative flow	There will be no information to be displayed.

Table 5.5: Table of use cases implemented for the 3rd sprint.

Machine Translated by Google

5.4.2 Design

Being an extension of the previous sprint , the design part for Federated Learning has already been addressed. However, the inclusion of new messages in the communication protocol and their respective commands should be mentioned:

- METRICS <round> <loss> <metric>: Message used to transmit the results of the evaluation of the model with the test set of each node. Its associated command will be Metrics_cmd.

  Note that the metrics will only be sent if the node has been told that learning is not running in a federated environment.

- VOTE_TRAIN_SET(<node> <punct>)* VOTE_TRAIN_SET_CLOSE: Message used to transmit the votes of each node. The number of votes will be variable, depending on the specified training set size. The command in charge of executing the logic when this class of messages is received will be Vote_train_set_cmd.

Regarding the design of the new functionalities to be implemented, the proposed design decisions will be specified below:

- The method used for the voting mechanisms consists of a consensus based on chance. Each node will select candidates, assigning each one a value. Subsequently, it will send the votes and wait to receive those from the other nodes (a process similar to waiting for models for aggregation). After receiving all the votes, the scores will be added and the nodes with the highest score will be selected to form the training set. Ties will be broken by alphabetical order of the
  name of each node.

- Because the connections of new nodes at the time of voting could cause differences in the training set, it has been decided to block the connections during the training. This event occurs when there are nodes in the network that have already terminated the voting, but a new node connects to one that has not yet terminated it.

- To avoid carrying out two votes, it has been decided that the nodes of the set of between training validate the model prior to training. With the completion of the last round, all nodes will perform a final validation of the model.

Machine Translated by Google

5.4.3 Implementation

For the incorporation of these two functionalities, the sequence of steps in the training has been modified, its inclusion being trivial.

5.4.4 Tests

For this sprint , no new unit tests have been carried out, the ones detailed in Table 5.4 being sufficient to validate the correct functioning of the system. These have been executed again obtaining identical results. Additionally, the metrics have been visualized, checking that they are correct, as well as that only the training subset participates in each round.

It should be remembered that in the sprint for tests 5.8, the results obtained in learning will be analyzed in detail.

5.5 Sprint 4: Protocol Gossip

This sprint will be dedicated to the inclusion of the Gossip protocol. Said protocol will be in charge of disseminating messages and models, adding a great tolerance to faults and implementing a mechanism for the diffusion of messages between peers not directly connected.

5.5.1 Analysis

In the same way as in the previous sprint , new use cases will not be implemented, but those already created will be refined. The affected use cases are detailed in Table 5.7.

CU-6: Start Training	
Description	The user will initiate the Federated Learning process on the nodes connected to the network. The number of rounds and epochs must be specified to perform, being other parameters such as the size of the set of read settings training. The model to initialize in the network will be the model of the node that starts the process. The relative to the Communication between nodes will be carried out using the Gossip protocol.

Machine Translated by Google

Continuation of table 5.6.

Requirements addressed	RF3, RF5.
Preconditions	The node must be started and without any learning process in progress. execution.
postconditions	The learning process will be initiated in the entire network of nodes.
Alternative flow	The learning will not start.
CU-7: Stop Training	
Description	The user will stop the Federated Learning process on the nodes connected to the network. The diffusion of this message will be using the protocol Gossip
Requirements addressed	RF3.
Preconditions	The learning process must be running.
postconditions	All processes related to learning have to end.
Alternative flow	Learning won't stop

Table 5.6: Table of use cases implemented for the 4th sprint.

5.5.2 Design

To detail the design in the inclusion of the Gossip protocol, message broadcasting will be discussed first, followed by broadcasting and model aggregation. In figure 5.5, you can see the class diagram that details the elements included and modified in this sprint.

The Gossip protocol for broadcasting messages is addressed with the creation of a new component (Gossiper). This will forward the messages that the BaseNode is receiving, following a First In, First Out (FIFO) queue. Note that BaseNode's components communicate with it via events and the observer pattern.

The main problem of this protocol is how to treat infinite cycles in messages.
In the first instance, 2 options were considered: dissemination of receipt confirmation messages or message hasing. As the confirmation messages would have a similar size to the messages themselves, it has been decided to calculate the hash of the messages, keeping a list of the last processed ones. In figure B.4, you can see the sequence diagram for the

Machine Translated by Google

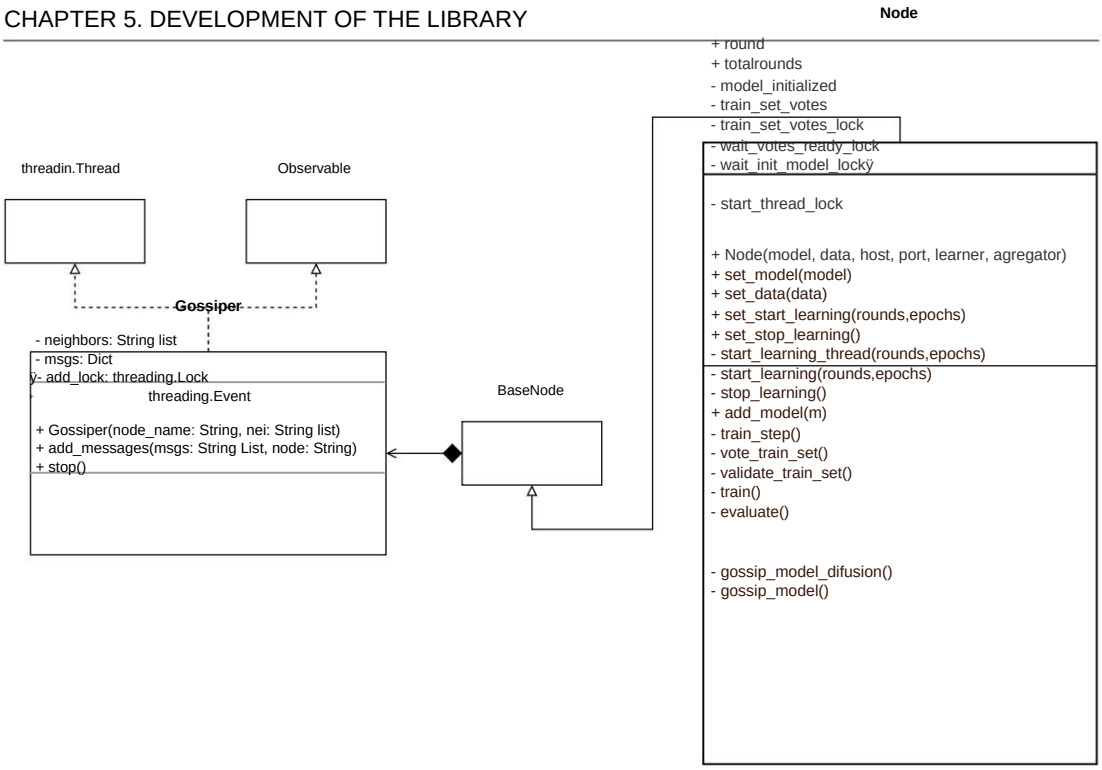


Figure 5.5: Class diagram for the 4th sprint.

diffusion of messages, appreciating the dynamics of selection and sending of pending messages following a determined frequency.

The messages that must be disseminated through the network will be modified, adding a **unique identifier or** hash at the end of it. Said hash will be made up of the message, the instant of creation time and a random number. The modified messages will be: START_LEARNING, STOP_LEARNING, VOTE_TRAIN_SET and METRICS. For the latter two, the creator of the message had to be included, since until now, it was assumed that the person who sent the message was also its creator.

When it comes to model diffusion, different methods were created in Node. They will be used confirmation messages, since in this case, resending models unnecessarily would be highly expensive. Likewise, for the aggregation of models, all the nodes of the training set must be aware of the models that each node has.

terminate_flag:

Figure B.5 shows the sequence diagram for the diffusion of models. Your flow will be exactly the same as broadcast messages, except that models will be sent, using Node methods instead of the Gossiper component. Keep in mind that for the dissemination of models there may be two situations, the main difference being the model that is sent:

Machine Translated by Google

- **Model diffusion:** Process really similar to message diffusion. It only looks for all the nodes to have the same model. Model dissemination will be required in two situations: model initialization and post-aggregation model dissemination. The new message in the communication protocol will be MODEL_INITIALIZED with its respective Model_initialized_cmd command, since MODELS_READY will continue to be used in the same way as before.

- **Model aggregation:** The Gossip protocol will be used for model aggregation. Its operation is identical to broadcast, with the peculiarity that the information it broadcasts will not be identical for all nodes.

  The optimization will be applied to the aggregation proposed in section 3.1.2. In this way, each node will send the partial aggregations that are convenient to its different neighbors.

  Remember that in order to apply this optimization, the set of models that make up the partial aggregation must be disjoint to the set of models already added by a node. This way, immediately after a model is added, the models that have been added will be broadcast over the network. For this, a new MODELS_AGGREGATED message has been created with its respective Models_aggregated_cmd command.
  As for the aggregation itself, it will be identical to the one proposed in sprint 2 (see the sequence diagram in figure B.3).

  Finally, it should be mentioned that if you want to use a federated aggregation algorithm not based on FedAvg, you can approach the aggregation in 2 ways: do not use partial aggregations, that is, disseminating only one model per iteration of the Gossip loop, or redefine the form to operate partial aggregations in Aggregator.

5.5.3 Implementation

The most substantial change that Gossip implies in the reception of models is the change from a passive wait to an active one. That is, in previous sprints , locks were used that awaited the reception of models, validating after each reception if all the nodes had finished.
Now, thanks to the use of Gossip, the waits will be active, sending, if possible, the models that your neighborhood needs. In the same way as in previous versions, there will be timeout mechanisms, being activated in Gossip loops when neighboring nodes get stuck.
For example, a node has spent more than a certain threshold of rounds without completing or notifying a state change.

Regarding the Gossip protocol for the dissemination of messages, it should be noted that there has been

Machine Translated by Google

excessive care with the fact that duplicate messages can not damage the system. Duplicate messages can happen when two NodeConnections receive the same message at the same time.

Finally, regarding the way of specifying the nodes involved in the partial aggregations, instead of transmitting only the model, it has been decided to transmit a triple containing said information: model, nodes involved, number of total samples.

5.5.4 Tests

For this sprint , no specific tests have been carried out, the existing ones being sufficient to guarantee operation without errors. In this way, the tests detailed in tables 5.2 and 5.4 have been executed again, obtaining the same results.

However, in the next sprint the architecture will be generalized, making it impossible to function without the Gossisp protocol. In this way, the unit tests of the next sprint will additionally and indirectly validate the current one.

5.6 Sprint 5: Topology change

The p2p system has been developed using a fully connected topology, seeking only ease of development. From the beginning, flexibility and scalability have been sought when creating the architecture, as well as lightness in the node, aimed at supporting low-performance devices.

5.6.1 Analysis

The system will be adapted to work with any type of topology, forming unstructured networks. The only affected use case is covered in Table 5.7.

Machine Translated by Google

CU-3: Connectar Nodos	
Description	The user connects one node to another. Since the Gossip protocol is used, it is not necessary for two nodes to be directly connected to be able to exchange messages.
Requirements addressed	RF2.
Preconditions	The nodes to connect must be started, there must not be a connection between them and none must be executing a learning process.
postconditions	The nodes must be connected.
Alternative flow	If the preconditions are not met, the connection will not be executed.

Table 5.7: Table of use cases implemented for the 4th sprint.

5.6.2 Design

The fact of already having the Gossip mechanisms designed and implemented might seem enough to migrate to any other topology. However, it will be necessary to have a directory of the nodes connected to the network, since as the system is designed there is no way of knowing. Thus, two options have been proposed to maintain a list of
active neighbors:

1. **Event-driven list:** The idea is to maintain an event-driven list. Node addition and removal messages will be required, needing to maintain the information that nodes are connected to others.
2. **List maintained by** timeouts: Taking advantage of heartbeat messages , they will be modified indicating the sending node and being broadcast using Gossip. The goal will be to maintain a list of active nodes on the network. To keep the list updated, the nodes from which a heartbeat has not been received in a certain time interval will be deleted, allowing their behavior to be parameterized.

   All situations in which the failure of a node can cause starvation, such as model aggregation, voting or diffusion, have their own timeouts. Therefore, despite the fact that it may take a little longer to determine the failure of a node, the simplicity of this mechanism will make it the option to implement.

Regarding the general design of the system, there have been no relevant changes, the diagrams being

Machine Translated by Google

of figures 5.2, 5.4 and 5.5 faithful to the design of the system.

5.6.3 Implementation

The implementation process has not given rise to any excessively important particularities, however, the important implementation decisions to be taken into account will be discussed.
account:

- The training set will be fully connected. After the election of candidates, they will connect with each other. Said decision is motivated to speed up the aggregation process, as well as make it more fault tolerant (more interconnections, more redundancy). Failures or impossibilities in the connections will not imply any inconvenience thanks to the Gossip protocol.
- Gossip is designed in such a way that a node will only serve its directly connected neighbors. Consequently, for networks that are not fully connected, the only global waiting mechanism will be voting at the start of each round.
- Again, as in previous sprints , the most critical point in the system is the first round voting. In this case, a heartbeat message not broadcast throughout the network could cause inconsistencies in the trainset. This will add a parameterizable timeout. This wait will seek to guarantee that the convergence of heartbeat messages has occurred before blocking the participating nodes in the between.
birth.

5.6.4 Tests

In relation to unit tests, new ones have been created, as well as slightly modified some due to the change in topology. Figure 5.8 shows all the tests that validate the changes made in this sprint, with the first three tests being modified.

Likewise, it has been validated that the performance of the resulting models is the same as using fully connected topologies.

Test 1.1: Connecting nodes	
Description	Two nodes will be started and connected, validating that they are added to the neighborhood and the list of nodes in the network.

Machine Translated by Google

Continuation of table 5.8.

Results obtained	Two perfectly connected nodes.
Test 1.3: Connection fully connected nodes	
Description	Four nodes will be connected and disconnected, validating that the topology is maintained and the list of nodes connected to the network is updated.
Results obtained	Fully connected nodes, respecting the topology.
Test 1.6: Validation of operability without errors when they occur	
Description	Given four fully connected nodes, errors will creep in. in them, looking for the disconnection of nodes so as not to harm the network. The errors introduced are: abrupt closure of the socket and stopping of the heartbeater. Affected nodes should be removed from the neighborhood and from the network.
Results obtained	Elimination of failed nodes from the network.
Test 5.1: Connecting nodes not fully connected	
Description	Four nodes will be connected following a ring topology. will be reviewed the neighborhoods, ensuring that the topology is fulfilled, it will also be validated that all nodes are available on the network.
Results obtained	Connected and visible nodes complying with the topology.
Test 5.2: Execution of the learning process in networks with different topologies: Ring, Star, Line	
Description	Four nodes will be connected following in different topologies. It will be validated that training is successful for all nodes.
Results obtained	Successful workouts.

Table 5.8: Explanatory table with unit tests carried out in the 5th sprint.

5.7 Sprint 6: Encryption in Communications

This sprint will focus on the inclusion of mechanisms that guarantee privacy in communications. The main objective of encrypting communications will be to prevent reverse engineering attacks that compromise data.

5.7.1 Analysis

Given the inclusion of encrypted communications, the affected use case is detailed in table 5.9.

Machine Translated by Google

CU-6: Start Training	
Description	The user will initiate the Federated Learning process on the nodes connected to the network. The number of rounds and epochs must be specified to perform, being other parameters such as the size of the set of read settings training. The model to initialize in the network will be the model of the node that starts the process. The relative to the Communication between nodes will be carried out through the protocol Gossip, being encrypted if the user indicates it.
Requirements addressed	RF3, RF5.
Preconditions	The node must be started and without any learning process in progress. execution.
postconditions	The learning process will be initiated in the entire network of nodes.
Alternative flow	The learning will not start.

Table 5.9: Table of use cases implemented for the 6th sprint.

5.7.2 Design

As discussed in Encryption Fundamentals 3.1.5, it will be necessary to establish an asymmetric encryption in order to exchange the shared key of the symmetric encryption. Remember that the pycryptodome library will be used, since it implements the cryptographic algorithms to be used. In figure 5.6, the class diagram corresponding to the inclusion of the encryption algorithms is detailed.

Machine Translated by Google

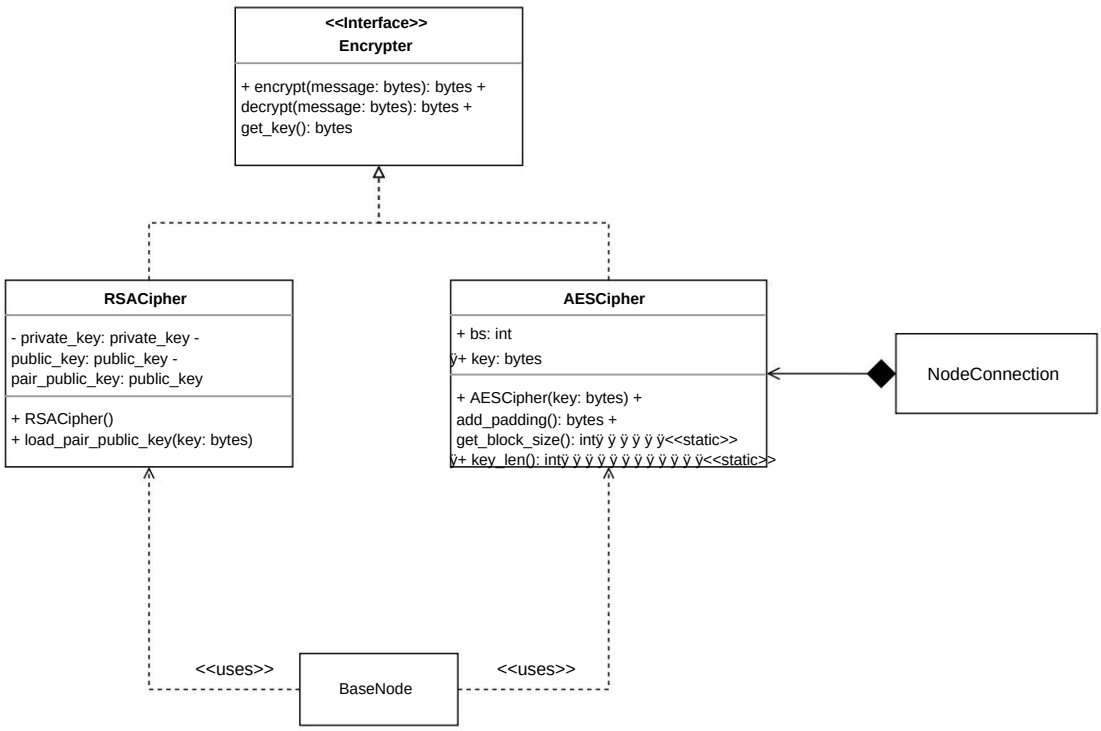


Figure 5.6: Class diagram for the 6th sprint.

5.7.3 Implementation

As AES is a block cipher algorithm, it must be taken into account that the block size used by the nodes must be a multiple of the block size used by AES. For ease of use and to reduce user errors, the specified block size will automatically be resized to the next valid block size.

For encrypted text messages, additional padding is required to achieve a multiple of the AES block size. It has been decided to include spaces, since it is an irrelevant character for the meaning of the message, saving the task of having to remove the padding. Furthermore, such a UTF-8 encoded character occupies 1 byte, the minimum unit.

Finally, since message encryption adds unnecessary overhead in simulated environments, message encryption will only be used in non-simulated environments.

Machine Translated by Google

5.7.4 Tests

Table 5.10 details the new added unit tests. In view of the results obtained, it can be concluded that in the presence of encrypted communications, the functionality of the system is identical.

Test 6.1: Simple encryption using RSA	
Description	Only one public/private key pair will be created. With these keys, a given encoded text must be encrypted and decrypted.
Results obtained	The decrypted text matches the original.
Test 6.2: Peer-to-peer encryption using RSA	
Description	Two public/private key pairs will be created. Using the keys another pair, it will encrypt and decrypt scrambled text.
Results obtained	The decrypted text matches the original.
Test 6.3: Simple encryption using AES	
Description	A key will be created, this will be used to encrypt and decrypt text encoded.
Results obtained	The decrypted text matches the original.
Test 6.4: Peer-to-peer encryption using AES	
Description	A key will be created that will be shared, the key will be used to encrypt and decrypt encrypted text.
Results obtained	The decrypted text matches the original.
Test 6.5: Model encryption using AES	
Description	A key will be created, the key will be used to encrypt and decrypt a fragmented model.
Results obtained	The decrypted model matches the original.
Test 6.6: Running training using encrypted communications.	
Description	A simple training will be executed between two nodes using communications encrypted.
Results obtained	Satisfactory training without errors.

Table 5.10: Explanatory table with unit tests carried out in the 5th sprint.

Machine Translated by Google

5.8 Sprint 7: Practical Application

Unlike the other sprints, this one will focus solely on testing. These will be much more exhaustive than those carried out in the development sprints , where the results obtained or the performance were not analyzed.

The next section will discuss the datasets and models used for the tests. In the rest, the different tests carried out will be detailed, explained and analyzed, ending with a brief general conclusion.

5.8.1 Employee Datasets

For the execution of tests, the classic dataset Modified National Institute of Standards and Technology dataset (MNIST) [43] and its variation for federated environments, Federated Extended Modified National Institute of Standards and Technology dataset (FEMNIST) will be used. Next, the datasets will be detailed, dealing further with the RR.NN.AA used to solve them.

MNIST

MNIST is a database of 28x28 pixel images of handwritten digits. The dataset consists of 60,000 images for training and 10,000 images for testing.

Since it will be used in a federated environment, the samples will be distributed to each node. The dataset will be divided into disjoint sets, each subset being assigned to a specific node. The subsets will be obtained following an IID distribution, that is, each node will have a representative subset of samples. Therefore, if there are two nodes, MNIST will split in a balanced and random way, distributing the original dataset between these two nodes.

Regarding the models used, the simple two-hidden-layer MultiLayer Perceptron (MLP) detailed in Figure C.1 has proven to be capable enough to get good results. In graph 5.7, it can be seen that approximately 98% accuracy is obtained in the test set.

Machine Translated by Google

FEMNIST

FEMNIST has been created with the aim of creating a benchmarking framework for learning in federated environments (LEAF [44]). It is a dataset derived from the Extended Modified National Institute of Standards and Technology (EMNIST) dataset [45], the successor to MNIST. EMNIST not only significantly increases the number of samples, but also adds new classes. Handwritten uppercase and lowercase letters will be added, making a total of 62 sample types.

To adapt EMNIST to a federated environment, FEMNIST partitions the data based on the writer of the digit or character. As a result, FEMNIST consists of 80.5263 samples distributed among 3,550 users, with an average of approximately 227 samples per user.

Regarding the model used to solve FEMNIST and EMNIST, it has been shown that it is necessary to use a more complex and powerful model such as the Convolutional Neural Network (CNN) [46]. Specifically, the convolutional neural network used is detailed in Figure C.2.

5.8.2 Simulation tests

The first test will consist of the training of a MLP for the resolution of MNIST under the federated and classical paradigm, comparing the performance and results of both executions. It should be noted that for execution in the federated environment, it will be carried out in a 6-node network, using the entire network as trainset.

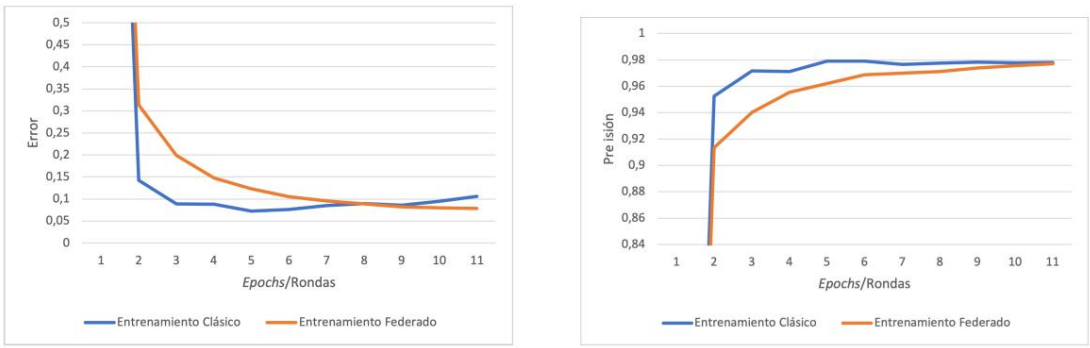
Because to correctly compare both methods the amount of computation must be strictly the same, the federated execution will consist of 10 rounds of 1 epoch per node, that is, in each round all the data from the training set will be used only once to improve the model. Meanwhile, for the classic training, 10 epochs will be run.

Figure 5.7 shows the precision and error obtained on the test set. These metrics will be used to determine the goodness of the model, that is, its inference capacity.
Additionally, figure 5.8 shows the evolution of the error of the nodes in the training set. It should be appreciated how this error is reduced, even when model aggregations occur.

In general, the metrics obtained present a really similar evolution and values for both training sessions. Given the similarity of the results, the gain in time due to the parallelization of the computation is evident.

Machine Translated by Google

Despite the clear gain in parallelization, the costs related to communication must be taken into account, with sending models being the highest cost of the system. Particularly for this experiment, by working on an MLP, a really simple and lightweight model, communication has not been highly costly. However, for the next simulation test, we will work with CNN, with the sending of models being a highly determining factor in the final cost of the system.



(a) Error in the test set

(b) Precision in the test set

Figure 5.7: Comparison of the error and precision on the test set in classical training and federated training for the experiment described in section 5.8.2. The metrics have been obtained at the end of each round in federated training or each epoch in classic training. The amount of computation between samples will be identical for both runs.

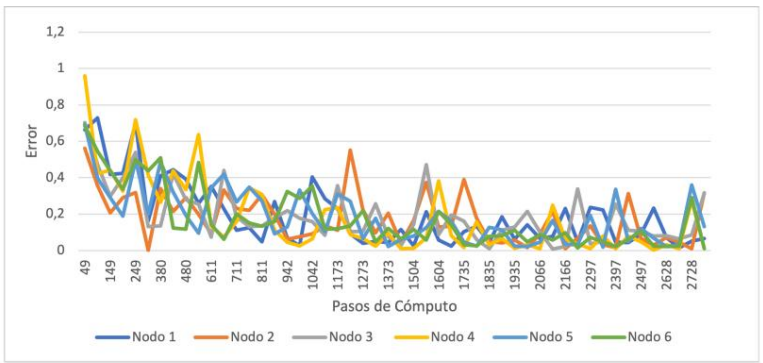


Figure 5.8: Error in the Federated Learning training set for the experiment described in section 5.8.2. Remember that the count is the number of times batches of 32 samples have been used to improve the model. Note that the computation is parallelized.

Machine Translated by Google

5.8.3 Validation of operation in a real environment

The test to be carried out will seek to be identical to the one carried out in the previous section, but in a deployment environment. A total of 6 nodes will be created, being distributed in 3 machines as can be seen in figure 5.9. 2 nodes will reside in the MacBook, 3 in the desktop computer and finally in the Raspberry 1. The nodes will be organized following an online topology.

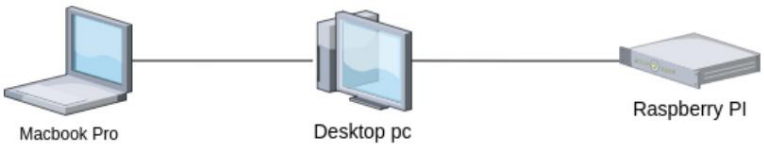


Figure 5.9: Devices involved in deployment testing and their connections.

It should be noted that when having devices with very different benefits, it will be necessary to adjust the parameters of the library accordingly. All the values related to the waiting for the reception of models will be established allowing long waits, since in comparison, the Raspberry presents deficiencies in hardware and consequently in performance.

The execution has been successful, yielding the same results as the previous test (see figures 5.7 and 5.8). In this way, the operation of the system is validated in a real environment, that is, a distributed environment.

5.8.4 Simulation tests with high number of nodes

Due to the lack of tests with a high number of nodes and few samples per node, as well as the lack of tests with moderately complex models, it has been decided to develop a simulation that is closer to a real use case. For this, a node with 64 cores from the Galician Supercomputing Center (CESGA) will be used.

The dataset used, FEMNIST , has been described in section 5.8.1. However, the complete data set has not been used, since only 40 nodes will be created. Regarding the configuration of the federated experiment, the trainset will be 4 nodes, with 10 rounds being executed. Trainset nodes will compute 2 epochs per round.

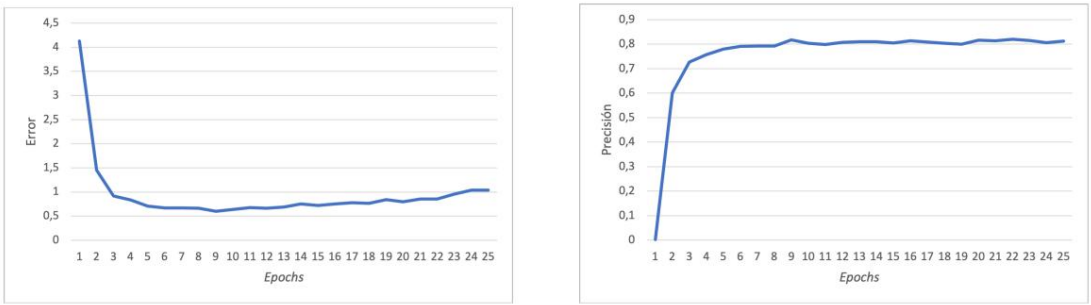
As for the equivalent experiment using classical training, it will again be run on the same data and with the same amount of computation. Taking into account that for each round of Federated Learning, 10% of the da taset samples will be used 2 times, an equivalent classical training should have a duration of 24 epochs. remind yourself

Machine Translated by Google

that in each epoch all the data in the training set will be iterated only once.

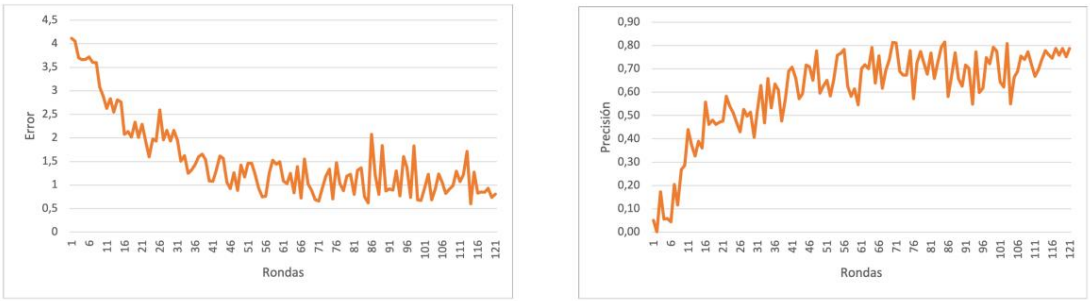
Figures 5.10 and 5.11 will show the evolution of the models, reflected in the error and precision on the test set. Considering only the final values, really similar results have been obtained. On the one hand, the federated training has achieved an error of 0.81 and a precision of 78.2%, with a standard deviation of 0.34 and 13% respectively. Regarding the classical training, an error of 0.79 and an accuracy of approximately 80% have been obtained, appreciating some over-training in the error graph. Unlike other tests carried out where the nodes had more samples, in Federated Learning a noticeable delay in reaching convergence values can be seen.

Finally, regarding the cost of communications, the use of a model 9 times more great that MLP has notoriously increased the volume and cost of communications. To the computational cost of the federated execution must be added the transmission costs of 15 GB of model parameters.



(a) Error (b) Precision

Figure 5.10: Error and precision in the test set throughout classical learning for the experiment described in section 5.8.4.



(a) Error (b) Precision

Figure 5.11: Error and precision in the test set throughout Federated Learning for the experiment described in section 5.8.4.

Machine Translated by Google

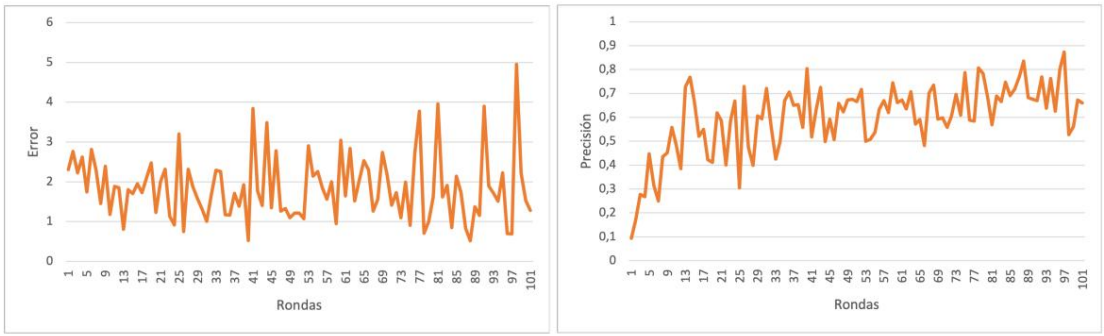
5.8.5 Tests with non-iid data

Because the tests carried out so far have been following IID data distributions, it is necessary to address how the system behaves before non- IID distributions. That is, to analyze the performance of the resulting models when the nodes do not have a set of representative samples of the problem to be solved. Obtaining good results with different distributions essentially depends on the federated aggregator, as will be seen, FedAvg has problems with this type of distributions [47].

In the test to be carried out, the worst data distribution scenario will be recreated. For this scenario, MnistFederatedDM has been modified, ordering the dataset by sample type and distributing it disjointly and orderedly to each node.

For the test, 6 nodes will be created, each having a maximum of 2 sample types. Regarding the Federated Learning parameters, a trainset of 4 nodes will be established, executing 100 rounds. In each round, the nodes will only run 1 epoch.

The metrics on the test set resulting from the execution of the experiment can be seen in Figure 5.12. As can be seen, the results obtained are notoriously inferior to the execution of section 5.8.2. Note that in these 100 rounds there has been no convergence around any value, reaching in the last one an average value of 64.6%, with a really high standard deviation of 20.9%.



(a) Test set error (b) Test set precision

Figure 5.12: Error and precision in the test set for the experiment described in the section 5.8.5.

The main reason for this great differentiation in the performance of the model with this data distribution is due to the fact that the improvement directions proposed by each local training do not have a clear common directionality. In Figure 5.13, it can be seen graphically that for a representative data set, the directions of improvement

Machine Translated by Google

proposals for each node will be similar. However, if the samples are not representative as is the case with the non- IID distribution, the directions of improvement will be offset in the aggregation, resulting in no effective results.

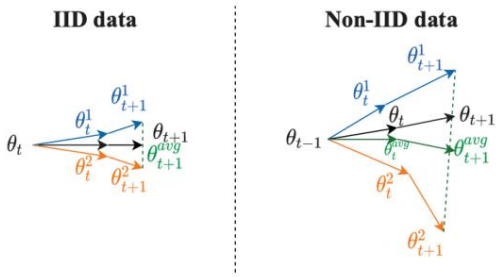


Figure 5.13: Example direction vector enhancement in iid and non-iid data distributions.

5.8.6 Conclusion on the results obtained

The different tests carried out have shown a good behavior of the system. When compared with an equivalent classical training, similar error and precision values have been reached in identical amounts of computation, only two shortcomings have been appreciated: the convergence delay strictly linked to the number of node samples and the problems with unrepresentative data sets produced by non- IID distributions.

The first deficiency is completely acceptable. In the test of section 5.8.4, the convergence has been delayed approximately 48 rounds. A figure, not at all crazy considering the tiny size of the trainset and the low number of samples per node (approximately 227) for this run.

Regarding the problems with non- IID distributions, it must be taken into account that the worst case has been executed with an aggregation algorithm (FedAvg) that has a bad behavior in these situations.

Related to the costs in the communication of Federated Learning, it is considered that they will not imply a disadvantage to the use of the system. However, the diffusion times of the model on the network, or the waits for devices with less features must be taken into account.

In conclusion, if a user who does not have a large infrastructure finds the advantages of Federated Learning attractive, the library will meet all their needs.

Machine Translated by Google

Chapter 6

conclusions

As detailed in the chapter dedicated to development 5, the objective of **developing a support library for Federated Learning on p2p** networks has been more than achieved. Compared to traditional training, performance and really satisfactory results have been achieved.

Related to the potential of FL, it must be taken into account that not only privacy and the paradigm shift in data management are elements to take into account. The parallelization of computing in training will be one of the great characteristics of Federated Learning. As discussed in the introduction, the trend of edge devices to have more and more computing capacity and data, make the FL have an ideal environment to be able to demonstrate its potential in the coming years.

As for the costs of communications, it is considered that although they are a new type of cost to take into account, they do not imply any added disadvantage. As seen in the test section, for a run of 40 nodes and 120 rounds, a total of 15 GB has been transmitted throughout the run, resulting in an average of 0.375 GB in traffic per node, not a crazy number at all.

Despite the great potential and the clear advantages that Federated Learning implies, the main obstacle of this project has been linked from the beginning to its scant trajectory in the scientific literature. Being one of the most critical points the theoretical approach of the feasibility of the project, without the existence of a similar solution. Likewise, it should be taken into account that for the aggregation of models, **an optimization has been proposed that allows the number of communications between nodes to be considerably reduced.**

Regarding the library developed, fault tolerance has been one of the unquestionable and essential requirements, mainly to try to guarantee support for environments

Machine Translated by Google

real. In such environments, the errors will not only be induced by the end devices, but the intermediate network devices will also be a critical point, being able to induce latencies or disconnections between peers. Redundancy is one of the best methods to mitigate this kind of errors, however, there must be a balance with the volume of traffic.

The chosen topology will be a highly determining factor when it comes to issuing unnecessary messages. Considering the implementation of the Gossip protocol, topologies with a high number of neighbors per node such as fully connected networks will involve more traffic compared to topologies with a low number of neighbors such as a bus topology. Naturally, there is no better option, a balance must be found. For example, in networks with a bus topology, a failure in one node will cause the network to split into two unconnected subnets, making communication between them impossible.

Despite the fact that the operation of the library is really satisfactory, there are many sections that need to be improved. As these future improvements are external to the work and its objectives, they will be discussed in section 7 on future work.

In short, the objectives set out in section 1.1, as well as the proposed requirements, have been satisfactorily addressed. The existence of an open source base library like the one developed implies a solid base for future developments, whether it is oriented towards research or the creation of a usable platform.

Machine Translated by Google

Chapter 7

Future work

This chapter will deal with the different lines of work to be carried out on the development done. As mentioned in the introduction chapter, special emphasis has been placed on the extensibility of the library, since the work has focused on the creation of a solid and functional base of Federated Learning on p2p networks .

Among the multitude of lines by which development can continue, those considered to be the main ones will be discussed:

- **New federated aggregation algorithms:** Currently, only FedAvg is implemented . The performance of the model obtained depends to a large extent on the aggregators, making it interesting to have a repertoire of aggregation algorithms to provide versatility to the library. For example, to obtain better models on non- IID distributions, it would be interesting to have algorithms like FedProx [24].

  It should be noted that most of the algorithms start from FedAvg, being possible to perform partial aggregations, however, special care should be taken with aggregation algorithms that are not compatible with partial aggregations.

- **Hot node inclusion:** As discussed in the implementation section of the 3rd sprint 5.4, it has been decided to block the hot inclusion of nodes. Despite the fact that the design is suitable, given the possibility of inconsistencies in the voting, it has been preferred to avoid this problem.

  The inclusion of a mechanism that mitigates the inconsistencies that may be caused in the voting, would allow the hot connection of nodes without any risk apparent.

Machine Translated by Google

- Gossip **loop optimization for sending models:** The current implementation is correct for sending without errors, however, when a node presents problems, models are sent unnecessarily and repeatedly. In order to save bandwidth, the inclusion of a particular mechanism could be sought for nodes that appear to have performance or connection problems. The objective will be to stop or lower the frequency of sending to nodes that are not reacting adequately to sending models.

- **Secure Agregation:** Encryption in communications prevents traffic from users outside the network from being captured, however, malicious nodes may compromise the private information of another node [48] in receiving local models. To solve this problem, it would be interesting to make use of the Secure Multi-Party Computation protocols. Conceivably, Secure Agregation [25] will allow aggregations without revealing any information about the local models of each node.

- **Iterative terminal:** Currently, if you want to control a node in real time, you must use the python iterative terminal or consider creating a script. Consequently, seeking the comfort of the end user, the creation of a command interpreter on the node, obviously iterative, could be considered.

- **Tolerance for incomplete reception of messages:** Similar to what happens with incomplete receptions in the fragments of the model, it could happen that if a large number of messages are sent over the same TCP connection , some message is partially read, causing errors. This can happen if the reception size is exceeded, or if a fragment of a message that is still being sent is received at the reading instant. This error is really unlikely, but it certainly could happen. His solution is trivial, detect when a message is not complete to read the remaining fragment.

- **Tests with a high number of nodes:** Although sockets and TCP allow local deployments and simulations, if you want to do large-scale tests, TCP introduces unnecessary overhead in simulations. For this reason, the creation of an alternative communication protocol for local simulations that seek to study the system with a high number of nodes is proposed.

- **Deployment-oriented work:** As it has been observed, most of the effort has been dedicated to the simulation part, only validating the same operation in the deployment. However, if you want to use the library in a production environment, you should study and verify even more rigorously the operation of the system. In addition, aspects related to operation should be improved, such as

Machine Translated by Google

they could be:

- **Comparison of node settings:** To try to prevent errors, you should validate that the nodes are operating with the same settings.
- **Associate votes to rounds:** In order to avoid inconsistencies caused by not two malfunctioning, the votes should be associated to the moment or round that they were issued.

Regarding network security, the following improvements should be considered:

- **Authentication of access to the network:** User authentication will be necessary to have a control of the users in the network. Above all, taking into account that one of the most widely used service privatization tools, Virtual Private Networks (VPNs), are ineffective due to the use of the Gossip protocol.
- **Attacks:** The occurrence of attacks must be taken into account. Although privacy-related attacks will be covered by the Secure Aggregation protocol, attacks that seek to destabilize the network, or "poisoning" must have some defense mechanism.

Finally, note that the source code of the library will be published on GitHub under a **GNU General Public License v3.0** [49]. This license is widely used in the world of free and open source software, allowing third parties to use, study, share and modify the software created.

Machine Translated by Google

Appendices

Machine Translated by Google

list of acronyms

AdaGrad Adaptive Gradient Algorithm. 12

ADAM Adaptive Moment Estimation. 12

AES Advanced Encryption Standard. 19, 65

API Application Programming Interface. 21

CESGA Supercomputing Center of Galicia. 70

CNN Convolutional Neural Network. 68, 69, 96

EMNIST Extended Modified National Institute of Standards and Technology dataset. 68, 96

FedAvg Federated Average. 12, 13, 15, 59, 72, 73, 76

FedMA Federated Learning with Matched Averaging. 13

FedSGD Federated Stochastic Gradient Descents. 12

FEMNIST Federated Extended Modified National Institute of Standards and Technology data set. 67, 68, 70

FIFO First In, First Out . 57

FL Federated Learning. 1, 3, 12, 14, 74

SWOT Strengths Opportunities Weaknesses Threats. v, 22, 23

IID Independent and Identically Distributed Random Variables. 14, 67, 72, 73, 76

KISS Keep It Simple, Stupid!. 50

Machine Translated by Google

ML Machine Learning. 1–3, 5, 6, 11, 12, 20, 38, 48

MLP MultiLayer Perceptron. 67–69, 71, 96

MNIST Modified National Institute of Standards and Technology dataset. 67, 68, 96

p2p Peer-to-peer. 2, 3, 8, 9, 14–18, 30–32, 36, 38, 40, 44, 45, 48, 60, 74, 76, 93

q-FedAvg q-Fair Federated Average. 13

RR.NN.AA Artificial Neuron Networks. 2, 6, 9, 11, 21, 67, 96

RSA Rivest, Shamir y Adleman. 19

SCAFFOLD Stochastic Controlled Averaging for Federated Learning. 13

SGD Stochastic Gradient Descent. 10, 12, 13

TCP Transmission Control Protocol. 21, 41, 43, 50, 51, 77

TFG Final Degree Project. 27

UTF-8 8-bit Unicode Transformation Format. 41, 65

VPN Virtual Private Network. 78

Machine Translated by Google

Glossary

Independent and Identically Distributed Random Variables In probability theory and statistics, for a set of random variables to be considered independent and identically distributed (iid), each random variable must have the same probability distribution, all being mutually independent.
Focusing on the distribution of samples among the different clients of an environment
federated, a distribution of this type will cause the data sets of each client to be similar, each having a representative set of the problem to be solved. . 14, 80

Secure Multi-Party Computation Secure Multi-Party Computation or privacy-preserving computing is a subfield of cryptography that deals with different entities collaborating with your information while preserving your privacy and confidentiality. Therefore, the objective will be to compute private information from different entities, revealing only the result and not the information provided by each participant.
77

batch When executing an epoch, the total number of samples will not be fed into the machine learning algorithm at once. The samples will be introduced iteratively, the batch being the number of examples that are passed to the algorithm in each learning iteration. . Fri, 10, 12, 13, 69

epoch In the context of machine learning, an epoch can be described as a complete cycle through the entire training data set to improve the model. The epochs indicate the number of iterations that the machine learning algorithm will run during that training.

In the context of this project, epochs will be designated to indicate amount of local computation on nodes. 10, 12, 13, 37, 45, 49, 54, 56, 64, 68, 70 , 71, 82

heartbeat A heartbeat is a periodic signal sent by a hardware or software component.

Machine Translated by Google

ware indicating normal operation with no errors. The heartbeat mechanism is one of the most common techniques in systems to provide high availability and fault tolerance. 41, 43, 61, 62

sprint It is each one of the cycles or iterations that are going to be carried out within a project with a SCRUM methodology. VI, 4, 26–34, 36, 38, 43, 45, 47 , 48 , 50, 52, 53 , 55–60, 62, 63 , 65, 67, 76, 93, 94

unicode Unicode is a character encoding standard designed to facilitate computer processing, transmission, and display of texts from many languages and technical disciplines, as well as classical texts from dead languages. The term Unicode comes from the three objectives pursued: universality, uniformity and uniqueness. 41

automatic differentiation In mathematics and computer algebra , self differentiation mathematics, also called algorithmic differentiation, is a set of techniques for evaluating the derivative of numerical functions specified in a computer program.
Focusing on the artificial intelligence sector, these techniques will be used to perform gradient descent. . 21

edge **devices** Edge devices are physical hardware at remote locations on the network perimeter. These devices have enough memory, processing power, and computing resources to collect data and process it in near real time, with limited assistance from other parts of the network.

Ultimately, an edge device is where data is collected and processed. 2, 74

gradient In mathematics, the gradient is a generalization of the derivative. While a derivative can be defined only in functions of a single variable, returning a scalar, for functions of several variables, the gradient will be used, returning a vector value.

The gradient represents the slope of the tangent line to the graph of a function. That is, the gradient will take higher values at points on the graph with greater increments.
ment. . 10, 12

Machine Translated by Google

Bibliography

[1] B. McMahan and D. Ramage, “Federated learning: Collaborative machine learning without centralized training data,” 2017. [En línea]. Disponible en: https://ai.googleblog. com/2017/04/federated-learning-collaborative.html

[2] R. M. Parizi, A. Dehghantanha, Q. Zhang, and K. Franke, “Decentralized federated lear ning: An introduction and the road ahead,” in Annual Hawaii International Conference on System Sciences. IEEE Computer Society, 2021.

[3] M. Jelasity, “Gossip,” in Self-organising software. Springer, 2011, pp. 139–162.

[4] “Flower: A friendly federated learning framework.” [Online]. Available at: https://flower.dev/

[5] G. A. Reina, A. Gruzdev, P. Foley, O. Perepelkina, M. Sharma, I. Davidyuk, I. Trushkin, M. Radionov, A. Mokrov, D. Agapov, J. Martin, B. Edwards, MJ Sheller, S. Pati, PN Moorthy, S.-h. Wang, P. Shah, and S. Bakas, “Openfl: An open-source framework for federated learning,” 2021. [En línea]. Disponible en: https://arxiv.org/abs/2105.06413

[6] “Nvidia flare.” [Online]. Available at: https://developer.nvidia.com/flare

[7] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat Palmbach, “A generic framework for privacy preserving deep learning,” 2018. [En línea]. Available at: https://arxiv.org/abs/1811.04017

[8] “Tensorflow federated.” [Online]. Available at: https://www.tensorflow.org/federated

[9] "Fate." [Online]. Available at: https://fate.fedai.org/

[10] “Paddlefl.” [Online]. Available at: https://github.com/PaddlePaddle/PaddleFL

Machine Translated by Google

[11] T. W. H. W. Yanjun Ma, Dianhai Yu, “Paddlepaddle: An open-source deep learning platform from industrial practice,” Frontiers of Data and Domputing, vol. 1, no. 1, p. 105, 2019. [En línea]. Disponible en: http://www.jfdc.cnic.cn/EN/abstract/article_2.shtml

[12] RF López and JMF Fernández, Artificial neural networks. Netbiblo, 2008.

[13] S. Ruder, “An overview of gradient descent optimization algorithms,” 2016. [En línea]. Available at: https://arxiv.org/abs/1609.04747

[14] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, “Backpropagation: The basic theory,” Backpropagation: Theory, architectures and applications, pp. 1–34, 1995.

[15] J. Kiefer and J. Wolfowitz, “Stochastic estimation of the maximum of a regression fun ction,” The Annals of Mathematical Statistics, pp. 462–466, 1952.

[16] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, “Communication efficient learning of deep networks from decentralized data,” 2016. [En línea]. Disponible en: https://arxiv.org/abs/1602.05629

[17] J. Schmidhuber, “Deep learning in neural networks: An overview,” Neural Networks, vol. 61, pp. 85–117, jan 2015. [En línea]. Disponible en: https://doi.org/10.1016%2Fj.neunet.2014.09.003

[18] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014. [En línea]. Available at: https://arxiv.org/abs/1412.6980

[19] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization.” Journal of machine learning research, vol. 12, no. 7, 2011.

[20] H. Wang, M. Yurochkin, Y. Sun, DS Papailiopoulos, and Y. Khazaeni, “Federated learning with matched averaging,” CoRR, vol. abs/2002.06440, 2020. [Online]. Available at: https://arxiv.org/abs/2002.06440

[21] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, “SCAFFOLD: stochastic controlled averaging for on-device federated learning,” CoRR, vol. abs/1910.06378, 2019. [En línea]. Disponible en: http://arxiv.org/abs/1910.06378

[22] T. Li, M. Sanjabi, and V. Smith, “Fair resource allocation in federated learning,” CoRR, vol. abs/1905.10497, 2019. [En línea]. Disponible en: http://arxiv.org/abs/1905.10497

[23] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Koneÿný, S. Kumar, and H. B. McMahan, “Adaptive federated optimization,” 2020. [En línea]. Disponible en: https://arxiv.org/abs/2003.00295

Machine Translated by Google

[24] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith, “On the convergence of federated optimization in heterogeneous networks,” CoRR, vol. abs/1812.06127, 2018. [Online]. Available at: http://arxiv.org/abs/1812.06127

[25] K. A. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, “Practical secure aggregation for federated learning on user-held data,” in NIPS Workshop on Private Multi-Party Machine Learning, 2016. [Online]. Available at: https://arxiv.org/abs/1611.04482

[26] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, “A survey on federated learning systems: Vision, hype and reality for data privacy and protection,” IEEE Transactions on Knowledge and Data Engineering, pp. 1–1, 2021. [En línea]. Disponible en: https://doi.org/10.1109%2Ftkde.2021.3124599

[27] A. Lalitha, OC Kilinc, T. Javidi, and F. Koushanfar. Available at: https://arxiv.org/abs/1901.11173

[28] Z. Wang and Q. Hu, “Blockchain-based federated learning: A comprehensive survey,” 2021. [Online]. Available at: https://arxiv.org/abs/2110.02182

[29] T. Wink and Z. Nochta, “An approach for peer-to-peer federated learning,” in 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), 2021, pp. 150–157.

[30] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Commun. ACM, vol. 21, no. 2, p. 120–126, feb 1978. [En línea]. Disponible en: https://doi.org/10.1145/359340.359342

[31] F. I. Processing and A. The, “Announcing the advanced encryption standard (aes).”

[32] E. Barker, “Guideline for using cryptographic standards in the federal government: Cry ptographic mechanisms,” 2016-08-22 2016.

[33] G. Van Rossum and FL Drake Jr, Python reference manual. Center for Mathematics and Informatica Amsterdam, 1995.

[34] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous

Machine Translated by Google

systems," 2015, software available from tensorflow.org. [Online]. Available at: https://www.tensorflow.org/

[35] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf , E. Yang, Z. De Vito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in Advances in Neural Information Processing Systems 32 . 8024–8035. [Online]. Available at: http://papers.neurips.cc/paper/ 9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[36] C. E. Perez, "Pytorch, dynamic computational graphs and modular deep lear ning," Mar 2019. [En línea]. Disponible en: https://medium.com/intuitionmachine/ pytorch-dynamic-computational-graphs-and-modular-deep-learning-7e7f89f18d1

[37] "Fastapi." [Online]. Available at: https://fastapi.tiangolo.com/

[38] LM Crespo Martínez and FA Candelas-Herías, Introduction to TCP/IP: Transmission Systems data carrier. University of Alicante, 1998.

[39] S. Ghazinoory, M. Abdi, and M. Azadegan-Mehr, "Swot methodology: a state-of-the art review for the past, a framework for the future," Journal of business economics and management, vol. 12, no. 1, pp. 24–48, 2011.

[40] J. P. Alexander Menzinsky, Gertrudis López, "Scrum manager," Iubaris Info 4 Media SL, 2016.

[41] "Python enhancement proposals (peps)," Tech. Rep. [Online]. Available at: https://peps.python.org/

[42] D. Holth, "The wheel binary package format 1.0," PEP 427, 2012. [Online]. Available at: https://peps.python.org/pep-0427/

[43] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available at: http://yann.lecun.com/exdb/mnist/

[44] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Koneÿný, H. B. McMahan, V. Smith, and A. Talwalkar, "Leaf: A benchmark for federated settings," 2018. [En línea]. Disponible en: https://arxiv.org/abs/1812.01097

[45] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of MNIST to handwritten letters," CoRR, vol. abs/1702.05373, 2017. [Online]. Available at: http://arxiv.org/abs/1702.05373

Machine Translated by Google

[46] A. Botalb, M. Moinuddin, U. M. Al-Saggaf, and S. S. A. Ali, “Contrasting convolutional neural network (cnn) with multi-layer perceptron (mlp) for big data analysis,” in 2018 International Conference on Intelligent and Advanced System (ICIAS), 2018, pp. 1–5.

[47] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, “Federated learning with non-iid data,” CoRR, vol. abs/1806.00582, 2018. [Online]. Available at: http://arxiv.org/abs/1806.00582

[48] R. Shokri, M. Stronati, and V. Shmatikov, “Membership inference attacks against machine learning models,” CoRR, vol. abs/1610.05820, 2016. [En línea]. Disponible en: http://arxiv.org/abs/1610.05820

[49] “Gnu general public license,” Free Software Foundation. [Online]. Available at: http://www.gnu.org/licenses/gpl.html

Machine Translated by Google

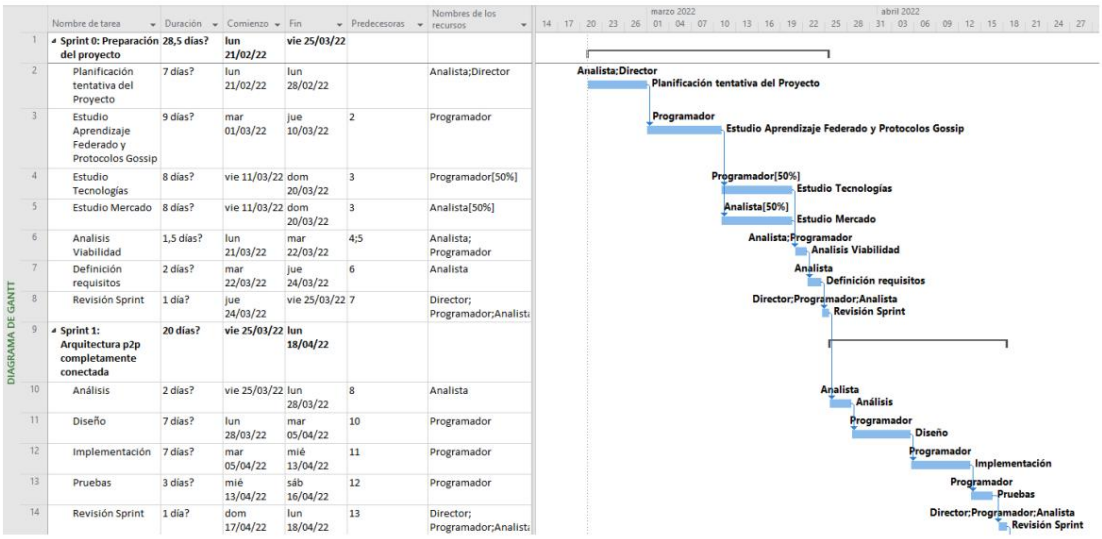
Appendix A

Planning

Tentative initial planning and follow-up will be detailed in the following sections . Of the same.

A.1 **Attempted** Gantt **chart**

The Gantt chart in Figures A.1, A.2 , and A.3 shows the tentative initial schedule.
For more information on the estimated costs or the planning itself, see section 4.4.



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1	**Sprint 0: Preparación del proyecto**	**28,5 días?**	**lun 21/02/22**	**vie 25/03/22**		
2	Planificación tentativa del Proyecto	7 días?	lun 21/02/22	lun 28/02/22		Analista;Director
3	Estudio Aprendizaje Federado y Protocolos Gossip	9 días?	mar 01/03/22	jue 10/03/22	2	Programador
4	Estudio Tecnologías	8 días?	vie 11/03/22	dom 20/03/22	3	Programador[50%]
5	Estudio Mercado	8 días?	vie 11/03/22	dom 20/03/22	3	Analista[50%]
6	Analisis Viabilidad	1,5 días?	lun 21/03/22	mar 22/03/22	4;5	Analista; Programador
7	Definición requisitos	2 días?	mar 22/03/22	jue 24/03/22	6	Analista
8	Revisión Sprint	1 día?	jue 24/03/22	vie 25/03/22	7	Director; Programador;Analista
9	**Sprint 1: Arquitectura p2p completamente conectada**	**20 días?**	**vie 25/03/22**	**lun 18/04/22**		
10	Análisis	2 días?	vie 25/03/22	lun 28/03/22	8	Analista
11	Diseño	7 días?	lun 28/03/22	mar 05/04/22	10	Programador
12	Implementación	7 días?	mar 05/04/22	mié 13/04/22	11	Programador
13	Pruebas	3 días?	mié 13/04/22	sáb 16/04/22	12	Programador
14	Revisión Sprint	1 día?	dom 17/04/22	lun 18/04/22	13	Director; Programador;Analista

Figure A.1: Attempted Gantt chart.
Fragment 1

Machine Translated by Google

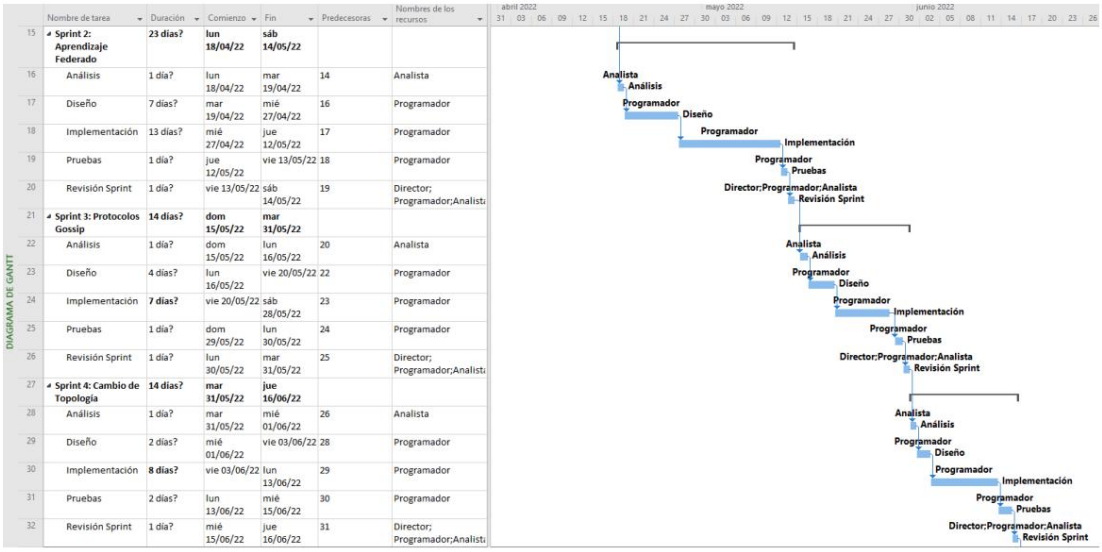


Figure A.2: Attempted Gantt chart.

Fragment 2

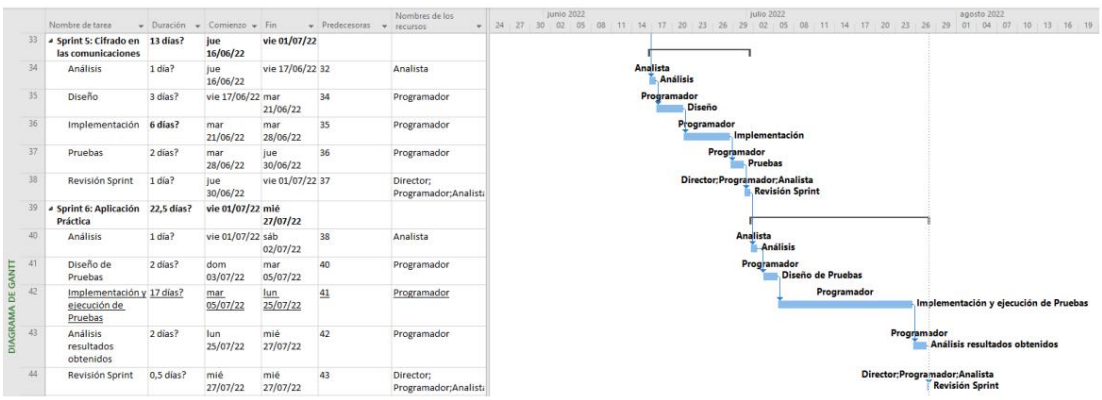


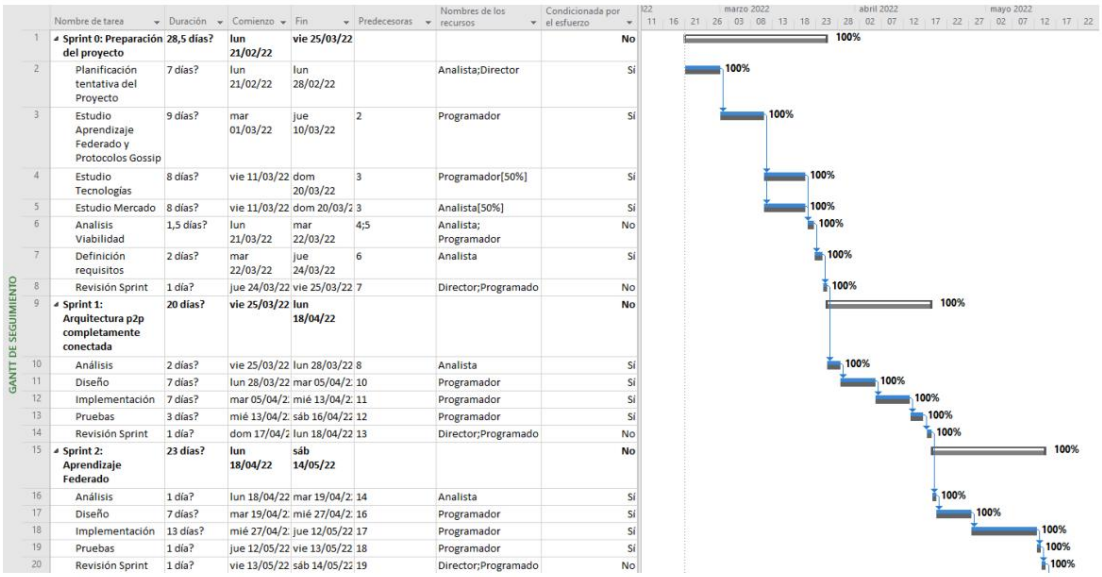
Figure A.3: Attempted Gantt chart.

Fragment 3

Machine Translated by Google

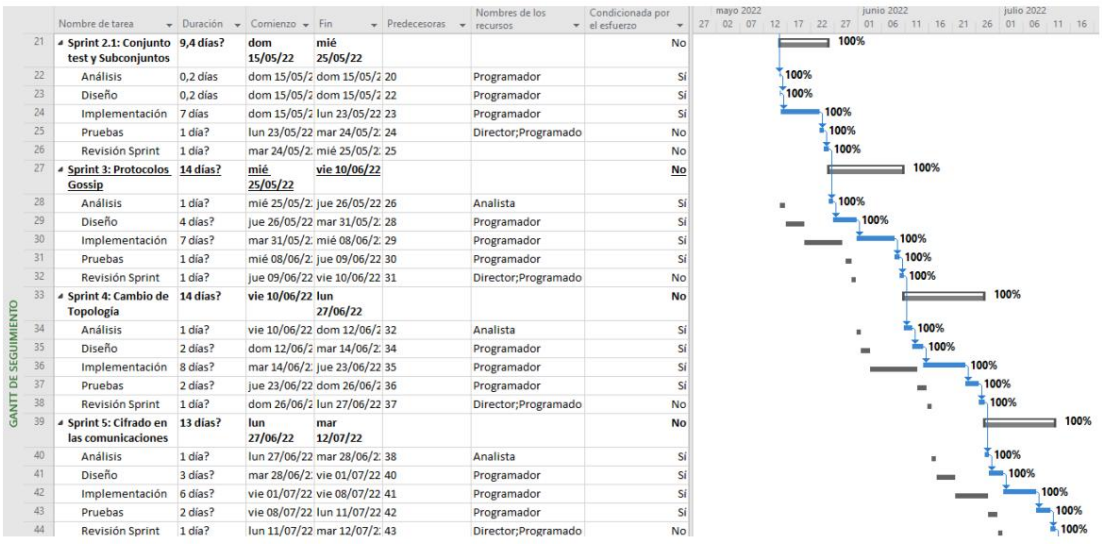
A.2 **Tracking** Gantt Chart

Figures A.4, A.5 and A.6 show the monitoring Gantt chart, it details the deviations from the tentative planning. Again, for more information about monitoring and its costs, see section 4.5.



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos	Condicionada por el esfuerzo
1	**Sprint 0: Preparación del proyecto**	**28,5 días?**	**lun 21/02/22**	**vie 25/03/22**			**No**
2	Planificación tentativa del Proyecto	7 días?	lun 21/02/22	lun 28/02/22		Analista;Director	Sí
3	Estudio Aprendizaje Federado y Protocolos Gossip	9 días?	mar 01/03/22	jue 10/03/22	2	Programador	Sí
4	Estudio Tecnologías	8 días?	vie 11/03/22	dom 20/03/22	3	Programador[50%]	Sí
5	Estudio Mercado	8 días?	vie 11/03/22	dom 20/03/2	3	Analista[50%]	Sí
6	Análisis Viabilidad	1,5 días?	lun 21/03/22	mar 22/03/22	4;5	Analista; Programador	No
7	Definición requisitos	2 días?	mar 22/03/22	jue 24/03/22	6	Analista	Sí
8	Revisión Sprint	1 día?	jue 24/03/22	vie 25/03/22	7	Director;Programado	No
9	**Sprint 1: Arquitectura p2p completamente conectada**	**20 días?**	**vie 25/03/22**	**lun 18/04/22**			**No**
10	Análisis	2 días?	vie 25/03/22	lun 28/03/22	8	Analista	Sí
11	Diseño	7 días?	lun 28/03/22	mar 05/04/2	10	Programador	Sí
12	Implementación	7 días?	mar 05/04/2	mié 13/04/2	11	Programador	Sí
13	Pruebas	3 días?	mié 13/04/2	sáb 16/04/22	12	Programador	Sí
14	Revisión Sprint	1 día?	dom 17/04/2	lun 18/04/22	13	Director;Programado	No
15	**Sprint 2: Aprendizaje Federado**	**23 días?**	**lun 18/04/22**	**sáb 14/05/22**			**No**
16	Análisis	1 día?	lun 18/04/22	mar 19/04/2	14	Analista	Sí
17	Diseño	7 días?	mar 19/04/2	mié 27/04/2	16	Programador	Sí
18	Implementación	13 días?	mié 27/04/2	jue 12/05/22	17	Programador	Sí
19	Pruebas	1 día?	jue 12/05/22	vie 13/05/22	18	Programador	Sí
20	Revisión Sprint	1 día?	vie 13/05/22	sáb 14/05/22	19	Director;Programado	No

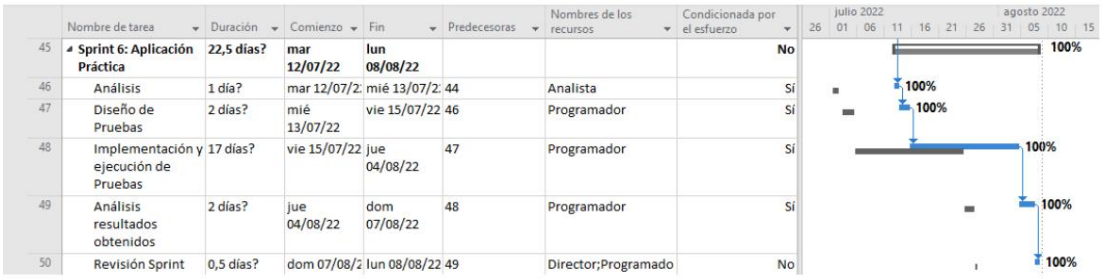
Figure A.4: Tracking Gantt chart.
Fragment 1



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos	Condicionada por el esfuerzo
21	**Sprint 2.1: Conjunto test y Subconjuntos**	**9,4 días?**	**dom 15/05/22**	**mié 25/05/22**			No
22	Análisis	0,2 días	dom 15/05/2	dom 15/05/2	20	Programador	Sí
23	Diseño	0,2 días	dom 15/05/2	dom 15/05/2	22	Programador	Sí
24	Implementación	7 días	dom 15/05/2	lun 23/05/22	23	Programador	Sí
25	Pruebas	1 día?	lun 23/05/22	mar 24/05/2	24	Director;Programado	No
26	Revisión Sprint	1 día?	mar 24/05/2	mié 25/05/2	25		No
27	**Sprint 3: Protocolos Gossip**	**14 días?**	**mié 25/05/22**	**vie 10/06/22**			**No**
28	Análisis	1 día?	mié 25/05/2	jue 26/05/22	26	Analista	Sí
29	Diseño	4 días?	jue 26/05/22	mar 31/05/2	28	Programador	Sí
30	Implementación	7 días?	mar 31/05/2	mié 08/06/2	29	Programador	Sí
31	Pruebas	1 día?	mié 08/06/2	jue 09/06/22	30	Programador	Sí
32	Revisión Sprint	1 día?	jue 09/06/22	vie 10/06/22	31	Director;Programado	No
33	**Sprint 4: Cambio de Topología**	**14 días?**	**vie 10/06/22**	**lun 27/06/22**			**No**
34	Análisis	1 día?	vie 10/06/22	dom 12/06/2	32	Analista	Sí
35	Diseño	2 días?	dom 12/06/2	mar 14/06/2	34	Programador	Sí
36	Implementación	8 días?	mar 14/06/2	jue 23/06/22	35	Programador	Sí
37	Pruebas	2 días?	jue 23/06/22	dom 26/06/2	36	Programador	Sí
38	Revisión Sprint	1 día?	dom 26/06/2	lun 27/06/22	37	Director;Programado	No
39	**Sprint 5: Cifrado en las comunicaciones**	**13 días?**	**lun 27/06/22**	**mar 12/07/22**			**No**
40	Análisis	1 día?	lun 27/06/22	mar 28/06/2	38	Analista	Sí
41	Diseño	3 días?	mar 28/06/2	vie 01/07/22	40	Programador	Sí
42	Implementación	6 días?	vie 01/07/22	vie 08/07/22	41	Programador	Sí
43	Pruebas	2 días?	vie 08/07/22	lun 11/07/22	42	Programador	Sí
44	Revisión Sprint	1 día?	lun 11/07/22	mar 12/07/2	43	Director;Programado	No

Figure A.5: Tracking Gantt chart.
Fragment 2

Machine Translated by Google



	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos	Condicionada por el esfuerzo
45	**Sprint 6: Aplicación Práctica**	**22,5 días?**	**mar 12/07/22**	**lun 08/08/22**			**No**
46	Análisis	1 día?	mar 12/07/2	mié 13/07/2	44	Analista	Sí
47	Diseño de Pruebas	2 días?	mié 13/07/22	vie 15/07/22	46	Programador	Sí
48	Implementación y ejecución de Pruebas	17 días?	vie 15/07/22	jue 04/08/22	47	Programador	Sí
49	Análisis resultados obtenidos	2 días?	jue 04/08/22	dom 07/08/22	48	Programador	Sí
50	Revisión Sprint	0,5 días?	dom 07/08/2	lun 08/08/22	49	Director;Programado	No

Figure A.6: Tracking Gantt Chart Snippet 3

Machine Translated by Google

Appendix B

sequence diagrams

Next, the sequence diagrams that detail the operation of the most important components of the library.

Firstly, the sequence diagram in figure B.1 shows the operation of one of the essential parts of the p2p system, the connection between nodes. This functionality has been implemented in sprint 1, and a detailed explanation of said diagram can be found in section 5.2.2.

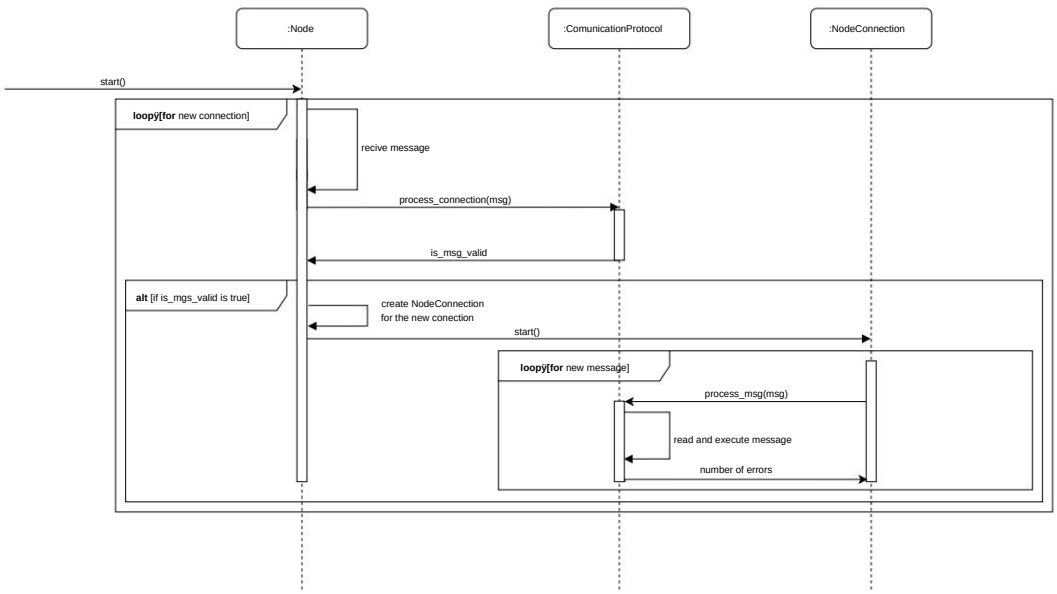


Figure B.1: Sequence diagram of the connection process between nodes.

Machine Translated by Google

APPENDIX B. SEQUENCE DIAGRAMS

Two of the most important parts of federated learning in the library are specified in sequence diagrams B.2 and B.3. The first details the start of learning between two nodes, as well as the steps that make up it. The second diagram shows the model aggregation process. Again, for a more detailed explanation, see the sprint 2 design section 5.3.2.

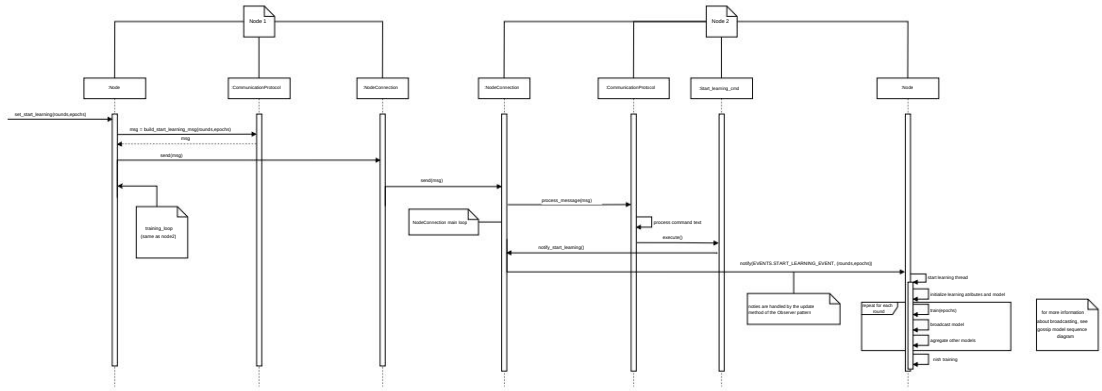


Figure B.2: Sequence diagram for the start of learning.

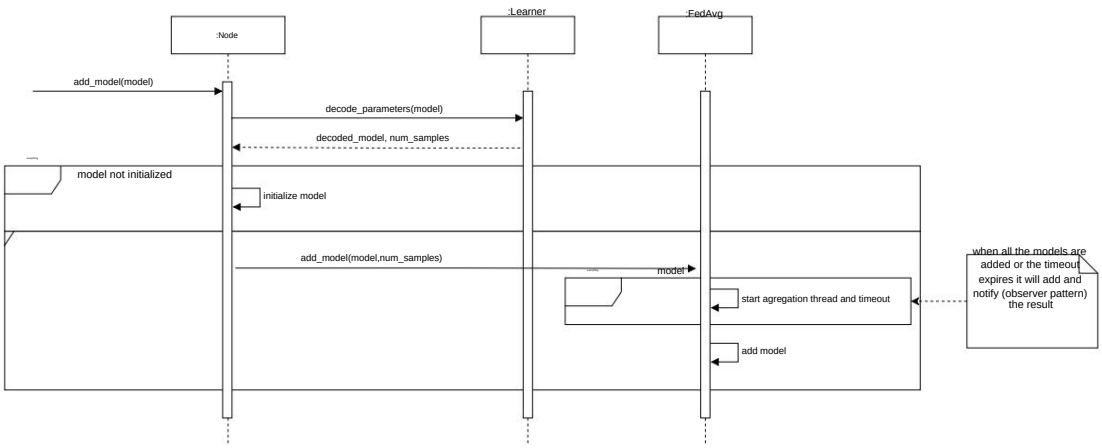


Figure B.3: Sequence diagram for model aggregation.

Finally, figures B.4 and B.5 detail the operation of the Gossip protocol for sending messages and models respectively. Again, a more detailed explanation of them can be found in the design section of sprint 3 5.4.2.

first

Machine Translated by Google

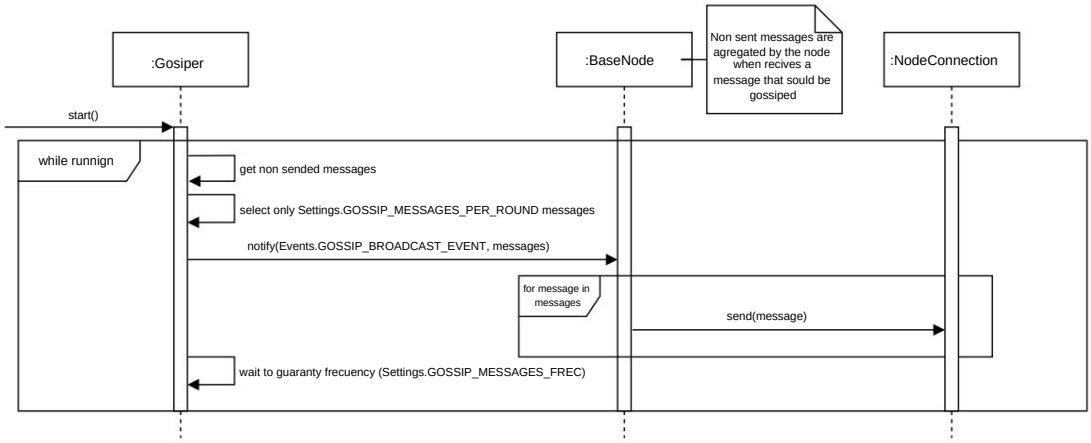


Figure B.4: Sequence diagram for broadcast messages using a Gossip protocol.

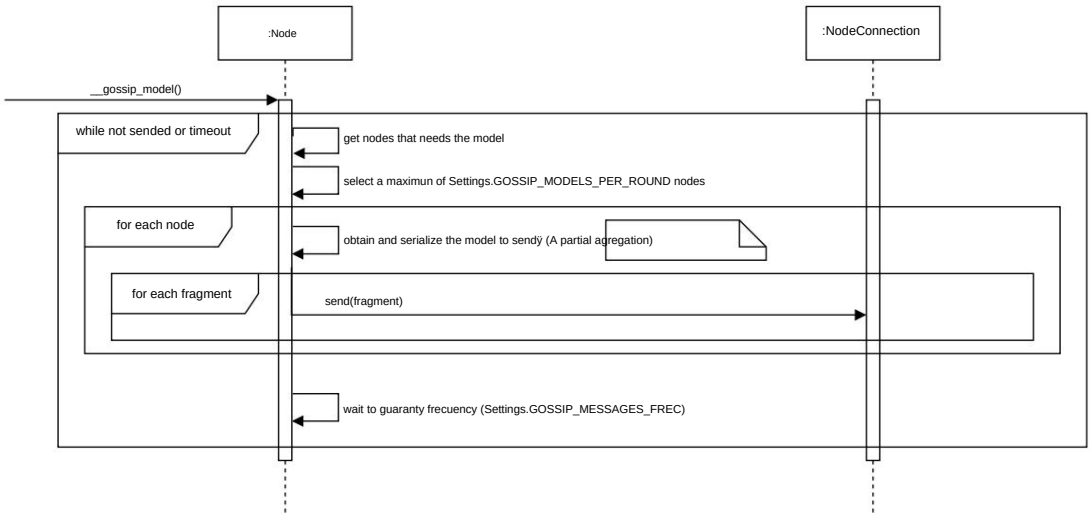


Figure B.5: Sequence diagram for diffusion of models using a Gossip protocol.

Machine Translated by Google

Appendix C

Architectures of the RR.NN.AA **use**

This appendix will be used to explain the models of RR.NN.AA used in the project for the resolution of MNIST and EMNIST.

The first and the simplest is the one detailed in Figure C.1. This model is an MLP, and consists of 4 fully contiguously connected layers. The first, or input layer, will have one neuron per pixel in the image, while the last layer will have as many neurons as there are sample types in the problem to be solved, that is, 10 neurons that represent the numbers from 0 to 9. The two hidden layers of 256 and 128 neurons are used for the model to learn the characteristics that differentiate the existing classes.

Regarding the second model, the specific architecture of the CNN used is detailed in figure C.2. Roughly speaking, convolutional layers will extract features from images, while max pool layers will reduce the size of feature maps. The goal will be to have many low-resolution feature maps that will be fed into an MLP, similar to the model discussed previously. In this way, the concatenation of convolutions and max pool layers will allow obtaining the most salient features of the image, greatly facilitating the work of the classifier (an MLP with a single hidden layer).

Machine Translated by Google

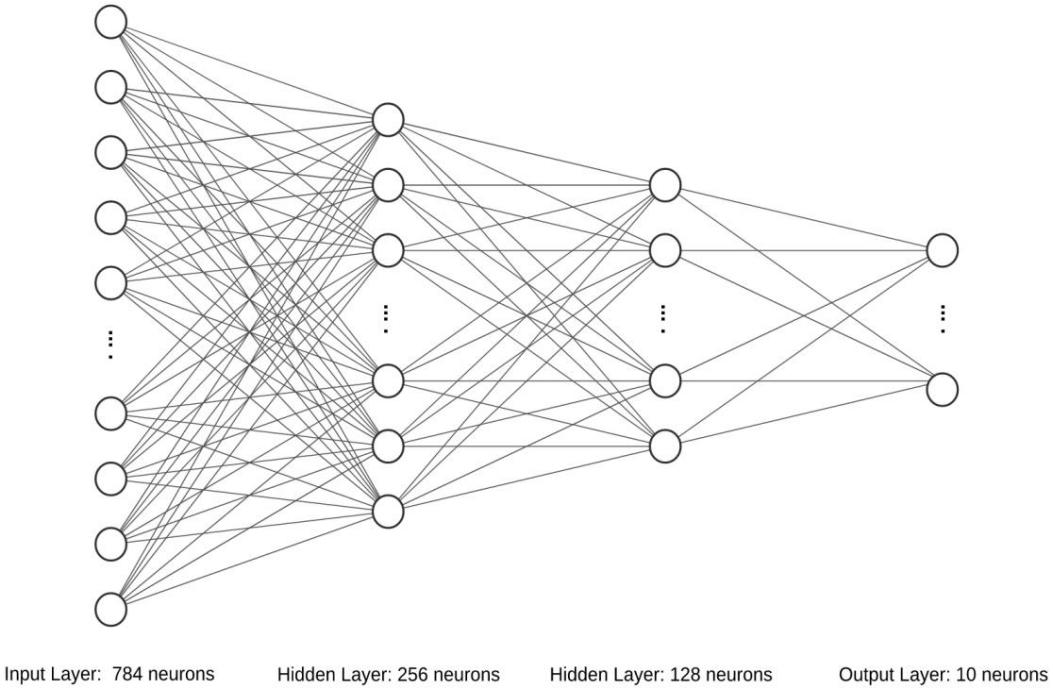


Figure C.1: Concrete architecture of the multilayer perceptron used.

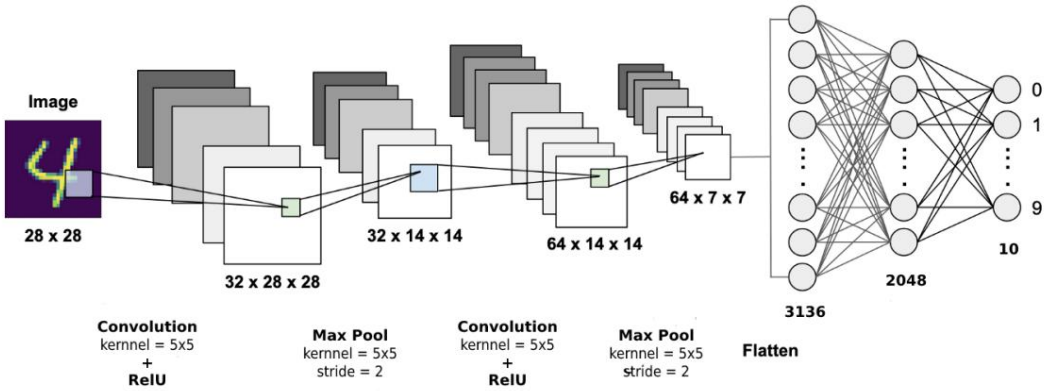


Figure C.2: Concrete architecture of the convolutional neural network used.