KING KHALID UNIVERSITY
COLLEGE OF COMPUTER SCIENCE

Decentralized Federated Learning For Enhancing Scalability and Resilience

By

Ali Abdullah Asiri
Mohammed Aslam
Rayan Fahad Alraai
Saud Murayah Alqahtani
Talal Mohammed Albarqi
Abdullah Mohammed Asiri

Supervised by:
Salem Alqahtani
Assistant Professor

Abstract

*In traditional **Federated Learning (FL)**, an aggregator server collects and coordinates updates from client machines, resulting in a potential bottleneck in scalability and a single point of failure. This project aims to evaluate the feasibility of **Decentralized Federated Learning (DFL)** to utilize **peer-to-peer (p2p)** communication among clients to provide both high-performance and resilient system architecture. Using a modified version of the flower framework to implement gRPC protocol, this project allows client updates to flow directly to other clients without an intermediary aggregator server. Results of experiments conducted with the MNIST dataset show that our **DFL** architecture is capable of achieving model convergence, yielding an increase in accuracy from 14% to 33% during three rounds of training while decreasing reliance on a central aggregator server.*

CONTENT

LIST OF FIGURES

1.Introduction

Federated Learning is a method of collaborative machine learning where multiple clients can collectively train a machine learning model without the need for a central repository of their data. Data is processed by the clients themselves, i.e., the devices owned by the users, which include mobile phones, laptops, and IoT nodes. Once the client completes training a portion of the model based on its local data, it transmits only the model update (i.e., the learned model parameters) back to a central server. The central server then combines the updates received from all the clients to form a single, improved model that represents the knowledge gathered from the collective of the participants, while maintaining the confidentiality of each participant's data.

This type of collaborative learning is particularly useful in cases where there are strict requirements for the preservation of client data privacy and/or where client data is sensitive and should not be shared publicly. Because no raw data is transmitted from the client to the server, federated learning protects client data from breaches, misuses and unauthorized access. Furthermore, federated learning facilitates compliance with client data privacy regulations such as GDPR. A further benefit is that federated learning minimizes the amount of data that must be transferred across networks; instead of transferring large volumes of data from client to server, federated learning enables clients to transmit only small updates to the model.

Federated learning is being applied in a variety of real-world applications including predictive keyboards, personalized product recommendations, smart health devices, and distributed IoT systems. An example of federated learning in practice is Google's Gboard keyboard, which utilizes federated learning to create better typing suggestions by combining knowledge from tens of millions of users without collecting their actual typed messages. The idea behind federated learning is that every device contributes knowledge from its data, while the central server only receives aggregated and anonymized improvements to the model and not the original data.

Depending on the devices and organizations involved, federated learning can take several forms. Cross-device federated learning involves a large number of personal devices, each containing small amounts of varying-quality data. Cross-silo federated learning, on the other hand, involves a smaller number of institutions (e.g., hospitals and banks) that collaborate to develop a common model without sharing their respective sensitive datasets. While providing great opportunities for collaborative learning and protecting data privacy, both approaches face similar challenges.

One of the main challenges facing federated learning is addressing non-identically distributed data, i.e., data on different devices, which can lead to unstable training processes. A second challenge lies in the fact that some devices may frequently disconnect or participate sporadically, which presents particular challenges in larger mobile networks. Finally, federated learning requires methods to securely aggregate the model updates to prevent individual updates from being reconstructed and linked to a specific client. Although federated learning still faces many technical challenges, it is developing rapidly and is becoming increasingly important in creating privacy-preserving, scalable, and intelligent systems relying on distributed data continues to evolve rapidly and is becoming a key technology for building privacy-preserving, scalable, and intelligent systems that rely on distributed data.

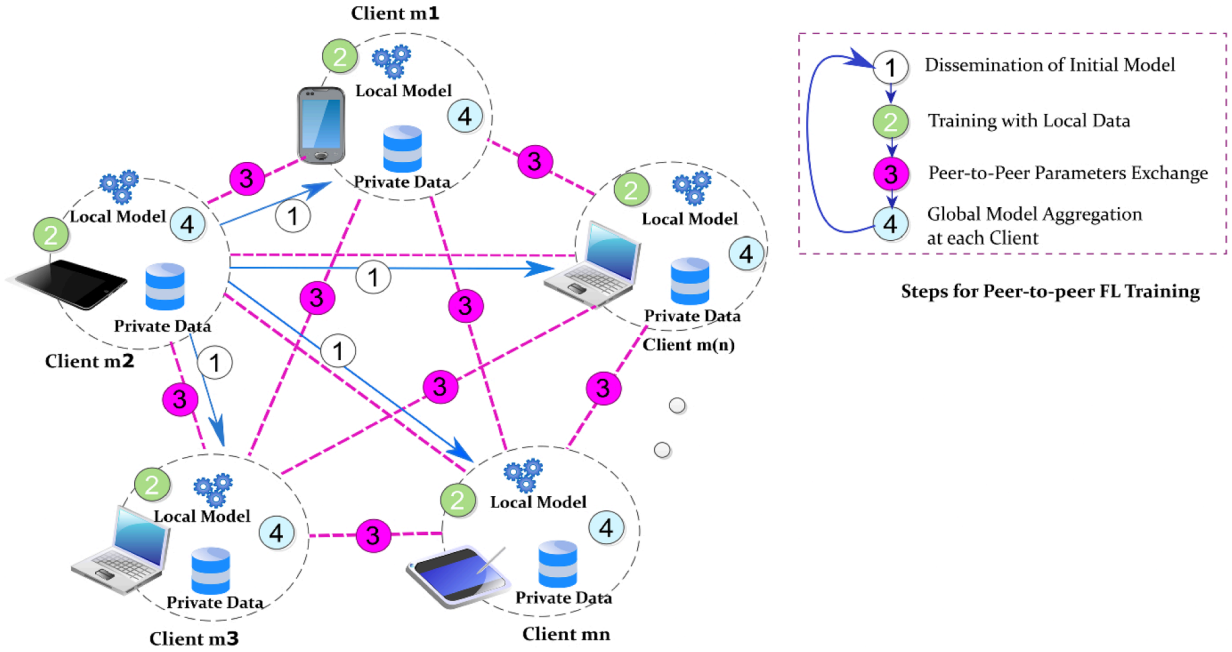


Figure 1. *The illustration is a mesh of clients each having their own data and each client sends model updates to other peer clients that have a similar amount of data so they can learn together without a centralized server.*

2. Literature Review

The development of Federated Learning (FL) has been influenced by numerous developments in distributed computing, privacy-preserving computations, and large-scale machine learning, which have taken place over the last ten years. The early foundational work and subsequent research have shaped the models, frameworks, and challenges that characterize current FL systems.

An important milestone was reached in 2017, when Google Research presented the first concrete realization of a practical federated learning solution. They illustrated how machine learning models can be trained cooperatively across millions of user-decentralized devices without exchanging the raw data. Rather than transmitting raw data, only the updated model parameters were exchanged. This breakthrough enabled the creation of the first practical, privacy-preserving, and communication-efficient learning methods and marked the beginning of the development of FL as an attractive alternative to traditional centralized training architectures.

In 2020 researchers at the University of Oxford collaborated with other industry partners to release Flower, an open source platform for developing decentralized and flexible federated learning systems. The open source platform Flower offered a modular architecture that was able to allow researchers to experiment across different types of devices, network types and different training strategies. Due to Flowers simplicity of use and ability to easily deploy a heterogeneous environment, it greatly aided in the acceleration of research within decentralized and cross-silo FL systems.

Previous work done by McMahan et al. (2018) —also from Google— identified a fundamental problem in Federated Learning: the communication bottleneck. They demonstrated through their research that federated learning has to repeatedly communicate between clients and servers, and therefore in both large-scale and bandwidth limited environments the communication cost can become prohibitive. The cost of communication in federated learning led to additional research in the areas of communication round reduction, efficient aggregation of client models and lossy compression of client updates to reduce overhead.

A second critical research contribution to federated learning were the findings from the group at EPFL (École Polytechnique Fédérale de Lausanne) in 2019 regarding the security vulnerabilities inherent to federated learning systems. Their research found that federated learning systems are susceptible to poisoning attacks where either malicious or compromised clients can purposefully submit poisoned model updates to the global model. Poisoned model updates can result in a degradation of performance and/or introduce hidden backdoors into the global model. The identification of these vulnerabilities in federated learning systems motivated a substantial amount of research in the development of secure aggregation methods, anomaly detection methods and robust optimization techniques to increase the trustworthiness of federated learning environments.

The combined efforts of all three of the aforementioned contributions form the basis of today's Federated Learning research, demonstrating not only the potential benefits of Federated Learning, but also the fundamental technological obstacles that continue to foster innovations within the Federated Learning space including scalability, communication efficiency and security.

3. Problem Statement

A significant problem exists when trying to use the centralized training method in the distributed environment of IoT and/or other similar big data applications. While there have been many successes using the centralized training approach in traditional machine learning environments, it has many problems that make it impractical for use in large, distributed systems. A major problem is the single point of failure associated with centralized architectures. Because all of the data processing and aggregation occurs at the central server, if the central server fails or is down due to hardware or network issues, the entire training workflow will stop until the central server comes back online. Even a few minutes of lost time during training can cause delays in updating models, decrease the overall reliability of the system and increase the risk of operationally impacting an organization's ability to meet business goals and customer expectations. Since the centralized architecture depends on a single component to operate successfully, it does not lend itself well to being used in real world environments that can contain thousands to millions of devices sending data to the system.

Another major issue with centralized architectures is that they quickly run into scalability challenges once there are many more nodes in the system than the number of cores available in the central server. Each node sends either the raw data or training results directly to the central server which creates a significant amount of communications traffic. Therefore, as more nodes are added to the system, the server becomes a bottleneck for communications and can have difficulty handling the large amounts of data exchange. Oftentimes, this results in longer latencies between the nodes, slower rates of model convergence, and lower efficiency, particularly in very large-scale applications such as IoT networks, mobile devices, and distributed sensor environments. Furthermore, the central server requires significantly more hardware and bandwidth to support additional nodes and therefore is costly to scale and difficult to maintain.

Beyond performance issues, centralized systems also face concerns related to privacy, energy consumption, and fairness across nodes, further showing that this approach becomes less practical as data grows more decentralized. These limitations highlight the need for alternative training methods—such as federated learning—that distribute computation, reduce server load, and eliminate single points of failure.

In addition to performance issues, centralized architectures also have serious drawbacks regarding issues of privacy, power usage, and fairness among the nodes. Therefore, as the decentralized nature of the data continues to grow, the centralized architecture continues to lose viability as a viable solution.

Overall, these issues highlight the need for new ways of training models—ways such as decentralized federated learning—that will allow for computation to be distributed away from the central server, provide for reduction of the server load, and remove the single point of failure inherent in the centralized architecture.

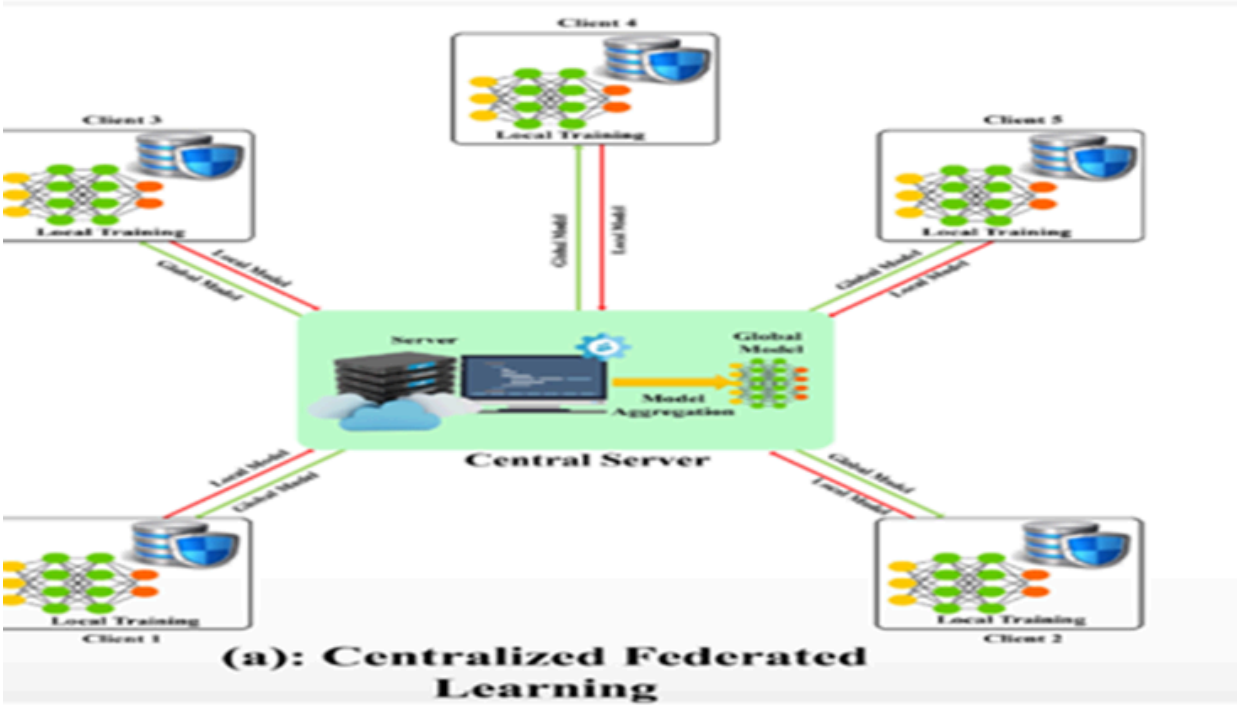


Figure 2. *Illustrates how in the classic federated learning topology there is one main or central server and many clients. Each client trains on their own data and sends it back to the central server to be aggregated. The central server then uses these updated models to train its own model which is then sent out to all the clients again.*

3. Objectives

This project aims to develop a scalable P2P communication network that eliminates the central bottleneck. Specifically, the objectives of this project include:

- Develop a communication system where clients can communicate directly via gRPC.
- Implement decentralized model aggregation and gossip-based propagation.
- Measure scalability and convergence rate relative to baselines of centralized approaches.

4. Functional and Non-Functional Requirements

4.1 Functional requirements

- **Communication of Client:**
  Each client must be able to dynamically find and establish a connection to other peers.
  Topology of System Support for various types of topologies (Mesh, Hybrid, etc.).

- **Training & Update Models:**
  Each client trains independently a local version of the model.
  Secure Exchange and Aggregation of Updates

  Updates for the model should be exchanged securely and aggregated.

- **Fault Tolerant Synchronization:**
  The system must be able to manage node failures and inconsistencies in model updates.

4.2 Non-Functional Requirements

- **Scalability**: Support a large number of clients.
- **Performance**: Minimize latency in exchange of models.
- **Reliability**: Ability to tolerate partial failures.
- **Compliance**: Ensure compliance with privacy regulations (e.g., GDPR).

5.System Architecture and Design

5.1 High-Level Architecture

As opposed to the star topology found in most centralized FL architectures, we propose the use of a mesh-like P2P topology. Each client functions as both a sender and a receiver of model parameters and communicates directly with other peers.

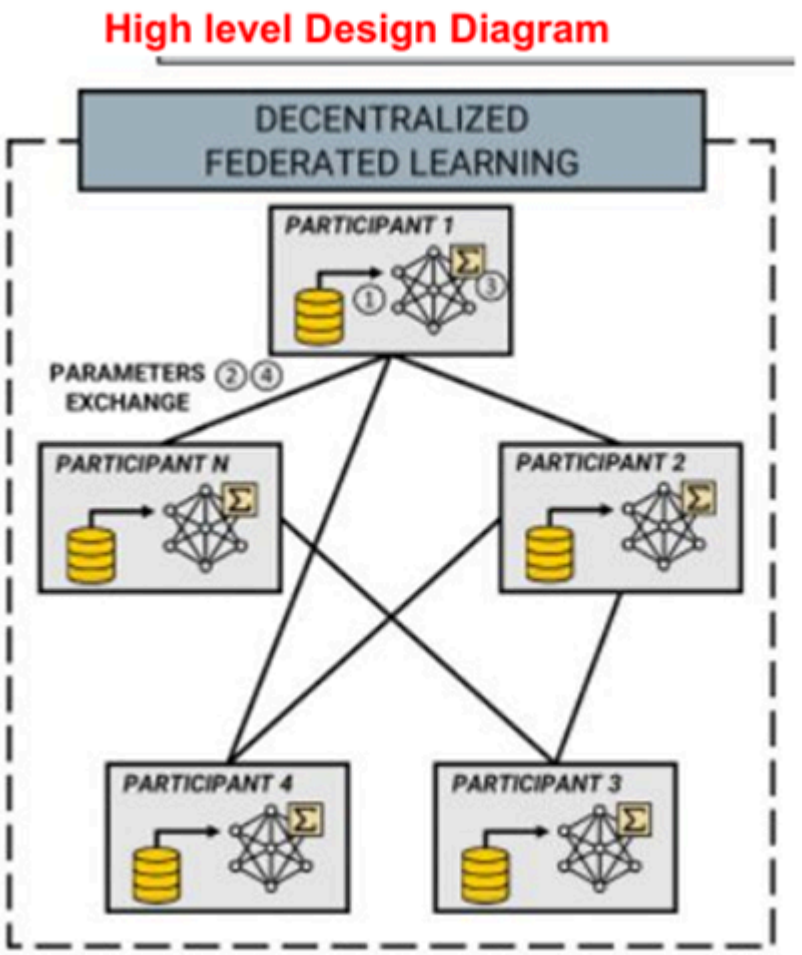


Figure 3. *This is an overview of how decentralized federated learning works. It shows participants in a mesh of peer to peer connections, exchanging parameters.*

The illustration shows a Decentralized Federated Learning system consisting of multiple smartphones acting as nodes in a mesh network. Each smartphone maintains and trains its own "local model" (the brain icon), using private data. Unlike centralized FL architectures, these smartphones do not depend on a central server; instead, they communicate with one another through a web of blue lines (W^2) enabling them to exchange gradient updates and model weights directly. With the collaboration possible throughout the network, the network can work together to build shared intelligence and achieve agreement, all without the transmission of raw data.

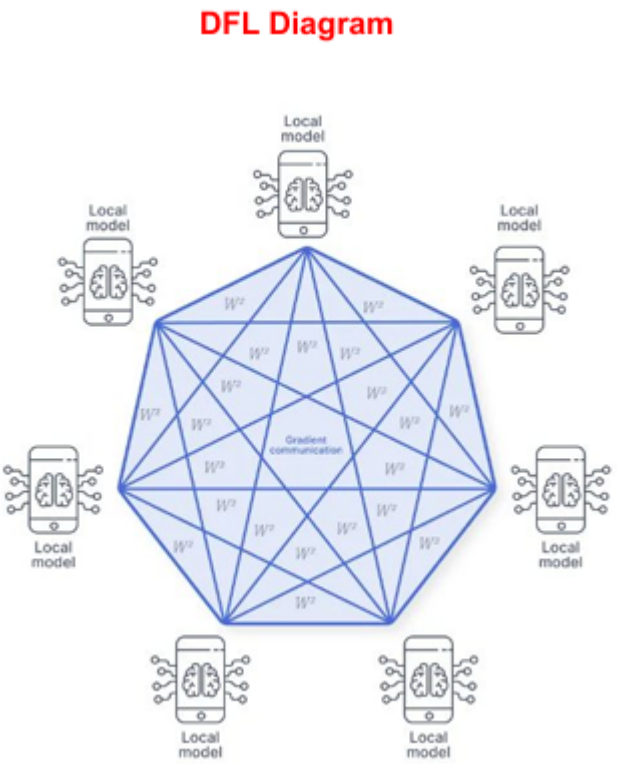


Figure 4. *Shows smartphones connected to a decentralized mesh network exchanging parameters from phone to phone.*

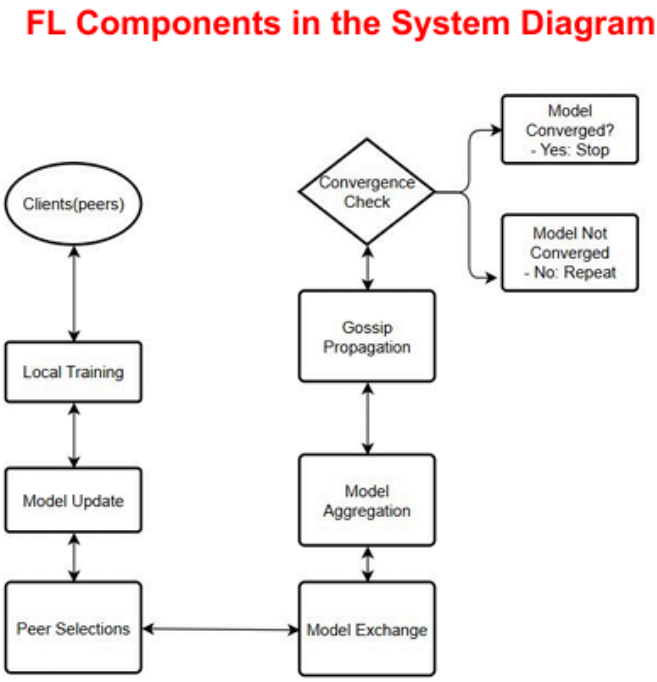


Figure 5. *Is a process diagram illustrating the flow of the federated learning algorithm. This includes local training and model update, peer selection, model exchange and aggregation.*

5.2 Data Flow

The training process is conducted in a cycle:

- **Local Training:** Clients perform local training using their private data.
- **Selection of Peers:** Clients dynamically select peers based on network conditions.
- **Exchange of Models:** Updates are exchanged securely via gRPC.
- **Aggregation:** Local aggregation (FedAvg) and gossip-based propagation occur at each node.

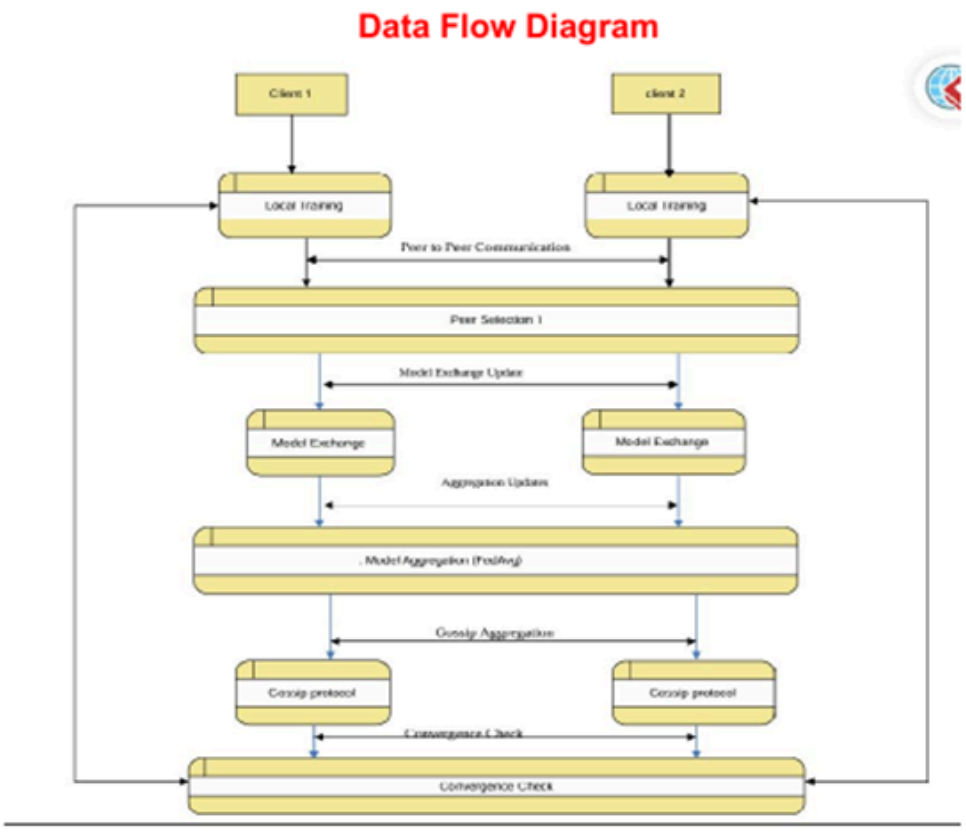


Figure 6. *This diagram illustrates the step by step process of how two clients perform decentralized learning, from the beginning of the process until the end when the clients confirm that the learning has converged.*

6. Methodology & Implementation

6.1 Methodology Used

Agile development was used to implement the project. The agile methodology uses iterative cycles of Requirements Analysis, Design, Development, Testing. Each cycle refines the communication protocols and coordinates the decentralized process within the Flower framework. This has ensured that the communication between all the nodes is reliable and stable.

6.2 Technology Stack

- **Framework**: The modified version of the Flower Framework was designed to enable Peer-To-Peer Aggregation.
- **Protocol**: Low Latency Language Neutral Messaging via gRPC and Protocol Buffers (.proto).
- **Model**: Convolutional Neural Networks (CNN) were trained on the MNIST Dataset.
- **Language**: Python was used to leverage the Machine Learning Ecosystem and Rapid Prototyping capabilities of gRPC.

6.3 Implementation

The standard Flower Architecture uses a Tightly Coupled Client-Server Model, where all communication goes through a Central Coordinator. To enable Decentralized Communication, we have embedded a Lightweight gRPC Server and gRPC Client inside every participating Node. The modifications allow each node to be able to send and receive model parameters directly to/from other neighboring peers without needing to rely solely on the Central Server. The gRPC Layer will manage the Peer Discovery, Request Handling and Secure Parameter Transmission to allow Nodes to independently initiate or respond to updates exchanges. The Design enables each client to be a Semi-Autonomous Communication Entity and allows them to participate in both Federated Learning Rounds and Peer-To-Peer Synchronization. By allowing Direct Client-To-Client Interaction, the System will reduce Server Load, Improve Resilience, Support Scalable Decentralized Learning Workflow, while Maintaining Compatibility with Flower’s Main Orchestration Loop.

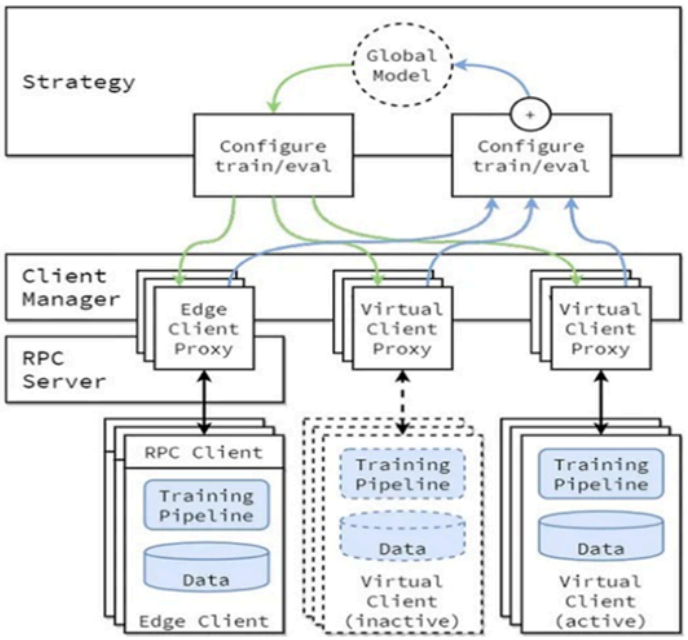


Figure 7. *Flower core framework architecture with both Edge Client Engine and Virtual Client Engine. Edge clients live on real edge devices and communicate with the server over RPC. Virtual clients on the other hand consume close to zero resources when inactive and only load model and data into memory when the client is being selected for training or evaluation.*

6.4 gRPC Protocol

gRPC (gRPC Remote Procedure Calls), is an Open Source High Performance Framework Developed by Google for Enabling Remote Procedure Calls Across Distributed Systems. Applications are enabled to communicate with one another regardless of Programming Language or Platform by Defining Service Interfaces and Message Types in a Language Neutral Way Using Protocol Buffers (.proto). Due to this gRPC is Ideal for Building Scalable Microservices, Client Server Applications and Peer-To-Peer Networks.

gRPC Core Concept is to Enable a Client Application to Call Methods on a Remote Server as if the Methods were Local Functions. gRPC Automatically Handles Network Communication, Message Serialization, Deserialization, Low-Level Details such as Connection Management, Error Handling, and Retries. Abstraction Reduces Complexity of Development and Ensures Consistent Patterns of Communication Across Distributed Components.

gRPC Supports Four Types of Communication Patterns; Unary RPC, Server Streaming, Client Streaming and Bidirectional Streaming. Unary RPC is the Simplest Type Where One Single Request From the Client Receives One Single Response from the Server. Server Streaming Enables the Server to Send Multiple Responses to a Single Client Request. Client Streaming Enables the Client to Send a Sequence of Messages to the Server. Bidirectional Streaming Allows Both Client and Server to Exchange Multiple Messages Asynchronously in Real Time Which is Particularly Useful for Decentralized Federated Learning Applications.

One of the Major Advantages of gRPC is its Use of HTTP/2 as the Transport Protocol. HTTP/2 Provides Multiplexed Connections, Header Compression and Low-Latency Communication Making gRPC Faster and More Efficient Than Traditional REST Over HTTP/1.1. Additionally, gRPC Integrates Well With Modern Security Standards Offering Support For TLS Encryption, Authentication and Identity Verification to Ensure Safe Communication Between Distributed Networks.

In Federated Learning Systems, gRPC Is Ideal for Allowing Client-Server and Peer-To-Peer Communication. It Enables Nodes to Send Model Parameters, Gradients and Control Signals Efficiently While Maintaining Low Latency and High Throughput. When Combining Protocol Buffers with gRPC, it Enables Compact, Structured, and Strongly Typed Messages Which Reduces Network Overhead and Enhances Reliability When Training Models Across Multiple Nodes.

Overall, gRPC Provides a Robust, Scalable and Efficient Framework for Communication in Distributed Machine Learning Applications. Combination of Language Neutrality, HTTP/2 Transport and Streaming Support Makes gRPC a Powerful Tool for Implementing Decentralized Learning Architectures, Ensuring Seamless Interaction Between Nodes While Maintaining Data Security, Consistency and Performance.

7. Experiments & Results

7.1 First Experiment: Setting Up the Client-Server (Flower Code-Based) System

Validation of the system took place in an environment containing 1 Server (used primarily for the purposes of monitoring), and 3 Clients for 3 Rounds of Training.

7.2 Evaluation Metrics for the First Experiment

Success in the decentralized training process was demonstrated through the fact that the model converged.

Accuracy: Accuracy of the model increased gradually from 14% in Round 1 to 33% by Round 3.

Loss: Decrease in Training Loss was demonstrated by a decrease in Training Loss from 359.96 to 296.17, which indicates that the model learned effectively.

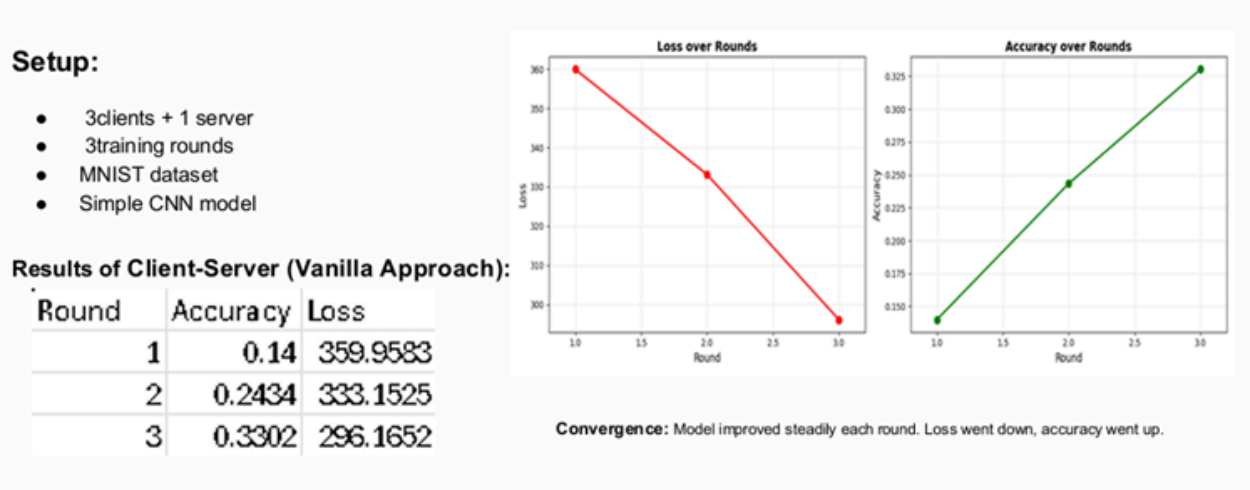


Figure 8. *These two graphs represent the loss and accuracy of a trained model over time while using centralized federated learning.*

7.3 Confirmation of Decentralized Operation

Client Logs confirmed that client nodes (Node 1, Node 2, Node 3) received parameter values from one another and performed local aggregation. Specifically, Node 3 successfully aggregated the two models sent to it by Nodes 1 and 2, without centralized interference.

7.4 Second Experiment: A Peer-to-Peer Decentralized Implementation

A peer-to-peer environment consisting of 3 connected clients (with no actively functioning central server) was used to test the decentralized system. Each client node had equal responsibility in exchanging and aggregating model updates during 3 Rounds of Training.

7.5 Performance Matrix for the Second Experiment (Peer-to-Peer Decentralized Implementation):

Stable decentralized training behavior was demonstrated throughout the three rounds of training among all participating client nodes.

Training Phase:
Each round of training was successful in all client nodes; local training was executed; peers exchanged updates; and aggregation of updates was completed without failure in any of the client nodes.

Evaluation Phase:
All evaluations were performed in a completely decentralized manner, and there were no reports of aggregation inconsistencies or communication errors between client nodes.

Accuracy:
Accuracy of the distributed model improved in each round of training from 0.089 in Round 1 to 0.213 in Round 3.

Loss:
Decrease in Training Loss was demonstrated by a steady decrease in Training Loss over the three rounds of training:

- **Round 1**: **0.1621**
- **Round 2**: **0.1154**
- **Round 3**: **0.0836**

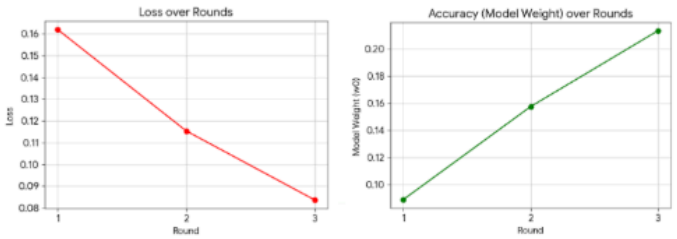
-

Setup:

- **4 Peers** (Decentralized Mesh Network)
- **3 Training rounds**
- **Neighbor-based Aggregation** (Gossip Protocol)
- **Metric Logged:** Loss (Cross-peer average)

Results of Peer-to-Peer (Decentralized Approach):

Round	Average Loss (All Peers)	Improvement Status
1	0.1621	Initial aggregation
2	0.1154	Significant drop
3	0.0836	Best performance



Convergence: Model improved steadily each round. Loss went down, accuracy went up.

Figure 9. *These two graphs demonstrate the improvement in loss and accuracy over time while using decentralized federated learning.*

Convergence: The model improved in each subsequent round of training. Loss decreased, and accuracy increased.

7.6 Workflow

This workflow enables multiple computers to cooperatively develop a shared artificial intelligence model privately. The computers connect, confirm the existence of connections, perform local training, exchange knowledge (parameters), combine this knowledge, and continue to do so until the model has sufficient accuracy.

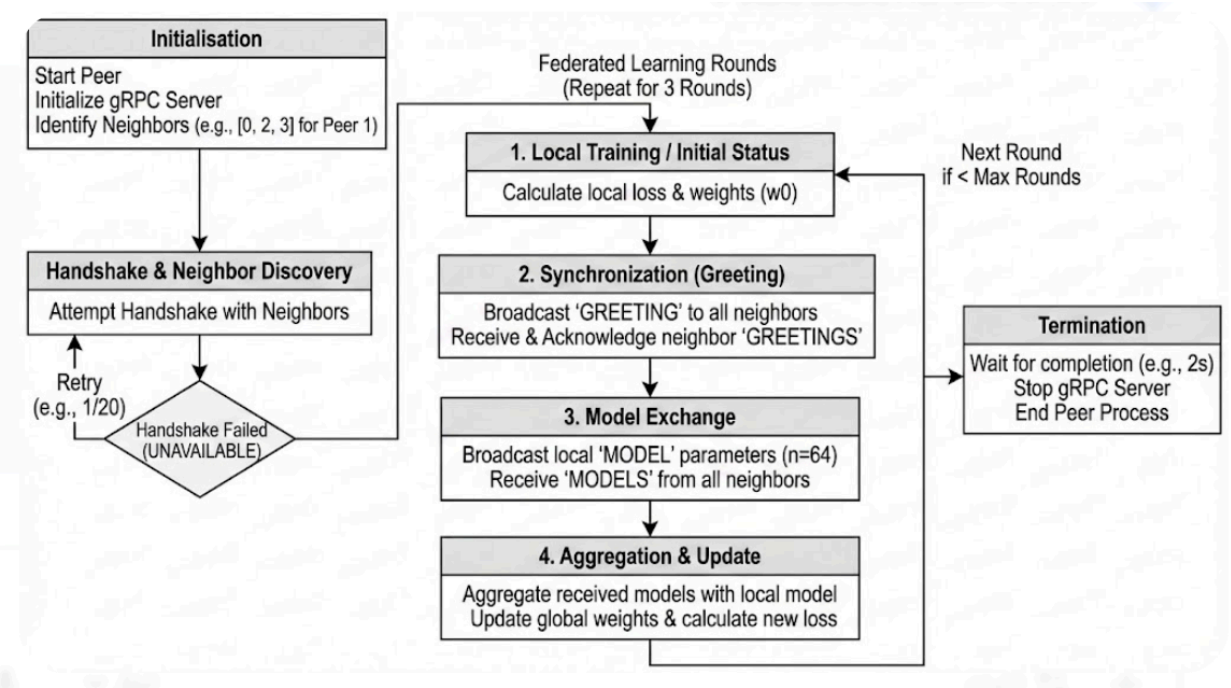


Figure 10. *This diagram illustrates the steps involved in the decentralized federated learning workflow. The workflow consists of several steps such as initialization, handshake, synchronization, model exchange, aggregation, and termination.*

Explain the P2P Training Process:

1. Initialization: Peer initiates operation, configures communication server (gRPC), and identifies which neighboring peers it will need to communicate with.

2. Connection: The peer establishes a connection with its neighbors, retries until they become available.

3. Local Training: The peer develops the model based on private data to compute local weights and loss.

4. Exchange & Aggregation: The peer exchanges model parameters with its neighbors and aggregates the received models to update the global weights of the model.

5. Loop & Termination: The training loop continues for a specified number of rounds, and once complete, the server terminates and the process is complete.

7.7 Result

The logs show the successful collaboration of 4 peers (0-3) in a decentralized training session over 3 rounds. Following the initiation of gRPC servers, the peers successfully established connections with their respective neighbors after overcoming initial connection failures through the use of the retry mechanism (i.e., "retry 1/20"). During the rounds, the nodes synchronized using "GREETING" messages and exchanged model parameters (n=64) to aggregate their knowledge. This collaborative process was successful, as evident by the significant reduction in the loss metrics of the peers (e.g., Peer 1's loss decreased from 0.1949 in Round 1 to 0.0806 in Round 3). Ultimately, once the third round was complete, all nodes shut down their servers, and the script terminated with the message "All peers finished".

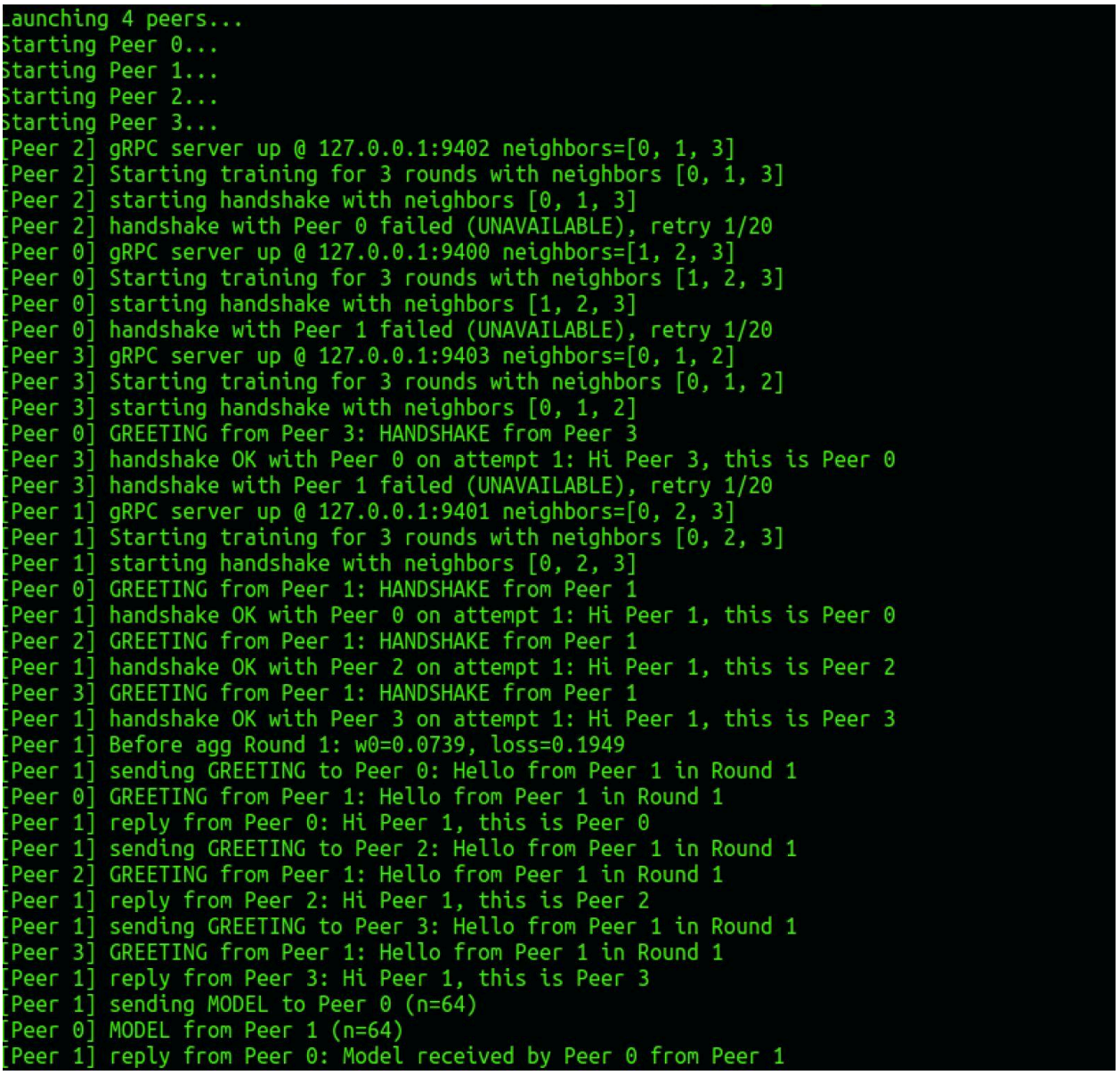


Figure 11. *Shows peers retrying failed handshakes, establishing connections, and exchanging the first set of models in Round 1.*

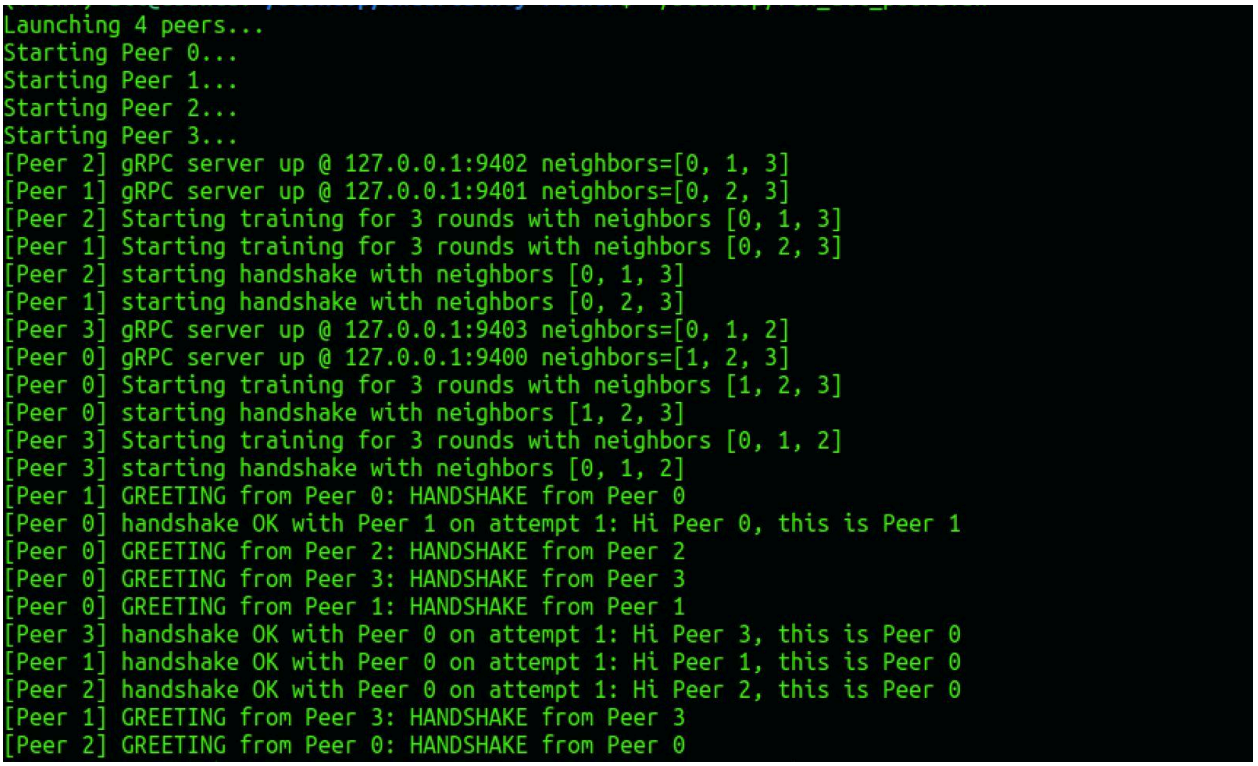


Figure 11.1 *Shows peers launching, starting gRPC servers, identifying neighbors, and exchanging initial GREETING messages.*

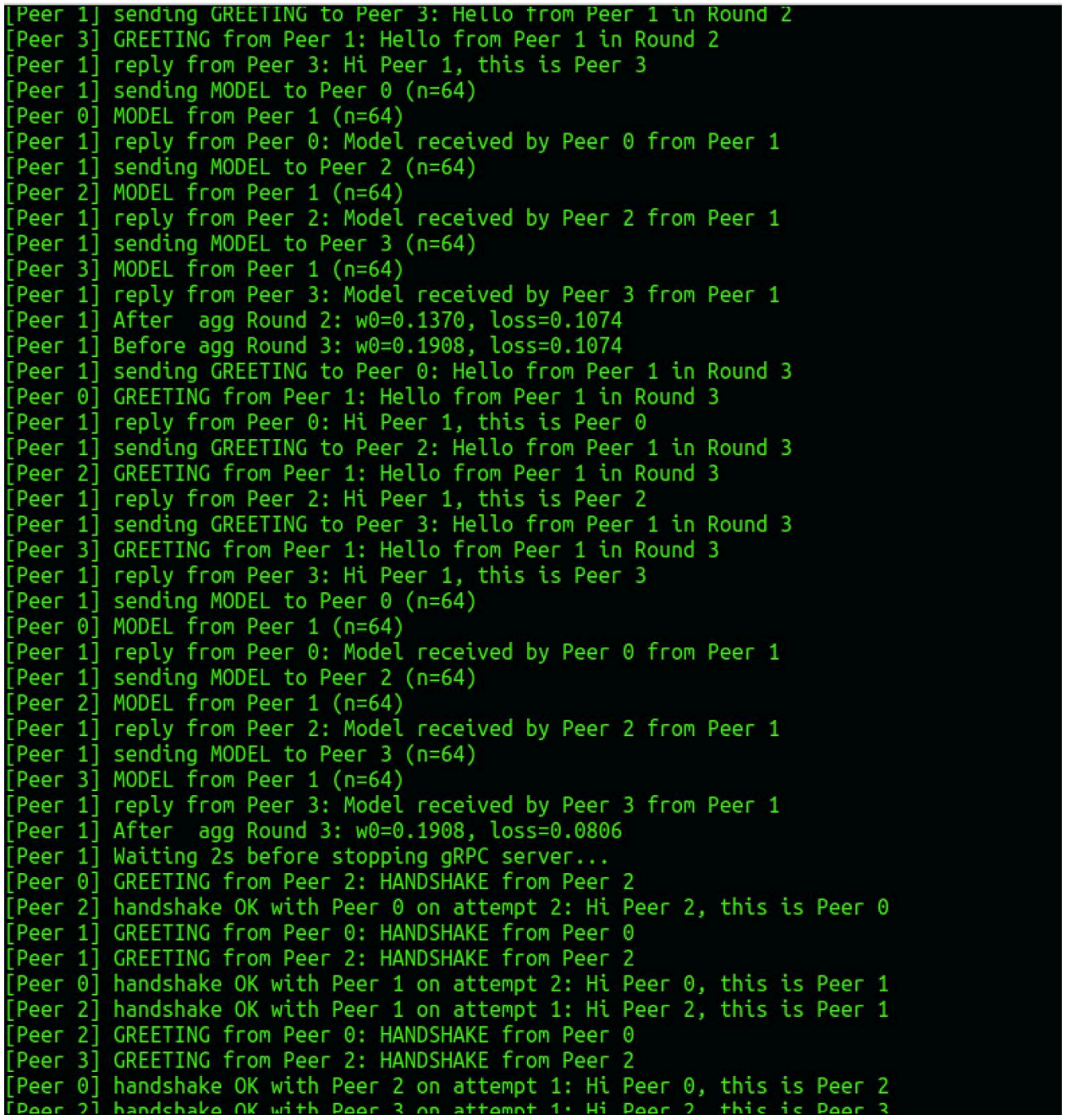


Figure 11.2 *Displays peer logs showing GREETING messages, model exchanges, and aggregation results during Round 3.*

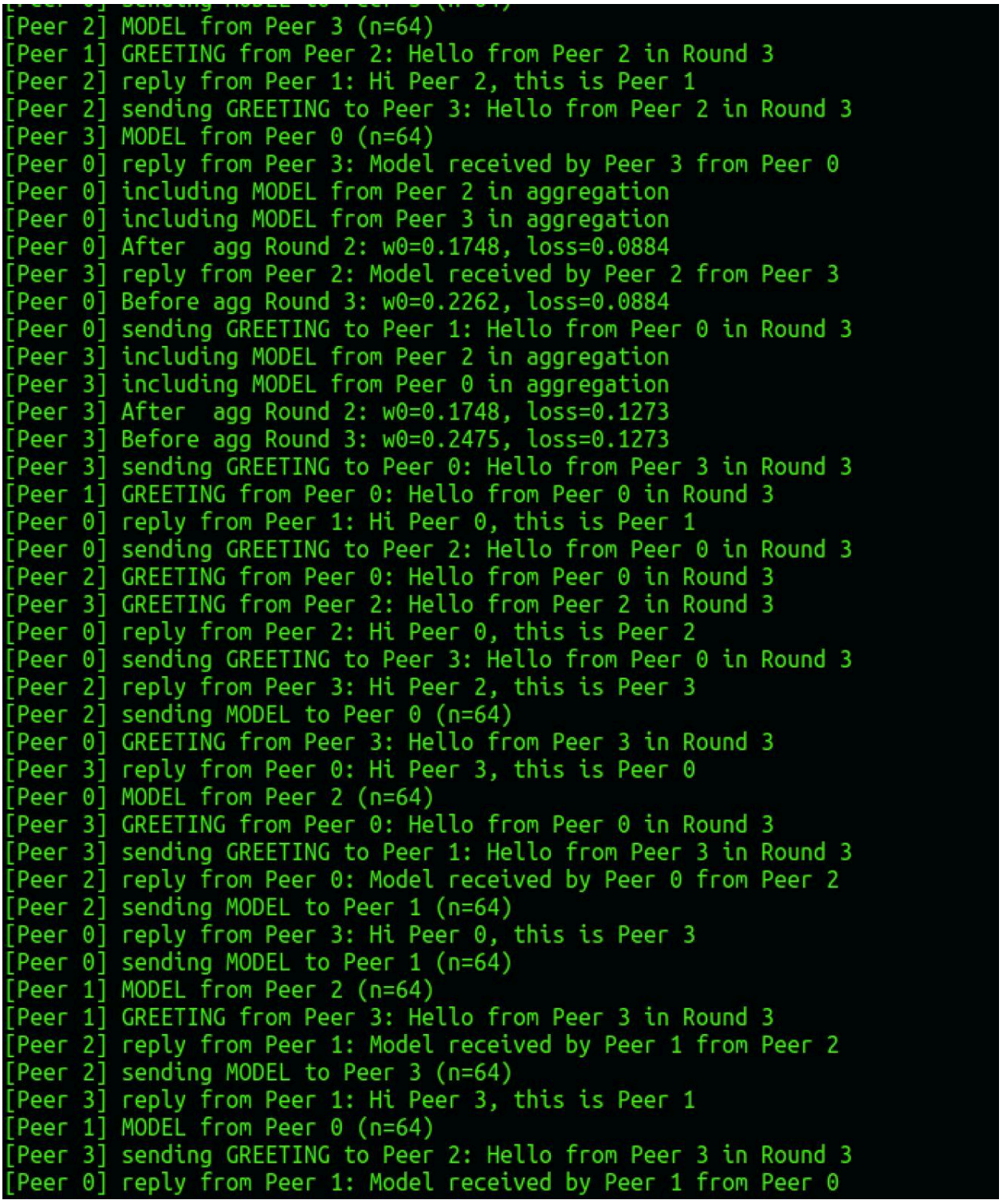


Figure 11.3 *Shows model exchanges, aggregation losses, and the system shutting down after completing the final training round.*

Figure 11.4 *Displays the last exchanges of model updates, final loss values, shutdown of gRPC servers, and confirmation that all peers finished.*

8. Challenges and Discussion

The implementation of a totally peer-to-peer (P2P) federated learning architecture in the Flower framework — originally created to support a centralized orchestrator — has raised many technical barriers related to the architecture, algorithms, and communication. We present here the main problems encountered and the design choices made to tackle them.

8.1 Framework Coupling and Architectural Adaptation

One of the biggest obstacles was due to Flower's fundamental dependency on a Server class that is central to coordinating the entire training process: each round of training, selection of clients for each round, synchronization between all clients, and global aggregation. When the Server class is removed from the framework, the Strategy class no longer has the capability to execute functions such as configure_fit and aggregate_fit, thus preventing the entire training process. To solve this problem, we have developed a Virtual Client Proxy Layer that enables each node to be able to act at once as an agent (worker) and a lightweight aggregator. Each node includes both a gRPC server and client inside of it. Therefore, we were able to eliminate the necessity for a ClientManager in the Flower's framework and create a fully decentralized communication layer using custom-defined Protocol Buffers definitions. With this change to the architecture, nodes are now capable of exchanging and coordinating parameters without having to rely on a central coordinator.

8.2 Communication Overhead and Network Scalability

Although decentralization removes the central bottleneck, it brings about new scaling issues based on communication. The communication complexity of a completely connected P2P network grows quadratically, i.e., $O(N^2)$. It increases bandwidth usage as the number of nodes increases. Even though our gRPC implementation was efficient and provided low latency for exchanging parameters for small groups of agents, it will become necessary to implement optimized topologies (e.g., k-nearest neighbor, ring-based gossiping, structured overlay) to prevent bandwidth saturation and minimize redundant messages when scaling to large networks.

8.3 Data Consistency and Asynchronous Model Drift

Due to the lack of a global coordinator enforcing strict round boundaries, maintaining model consistency is harder. Gossip-based exchanges allow updates to propagate across nodes. Thus, some peers may have models that temporarily differ from others. This creates an opportunity for model drift: the accuracy may vary because peers are training on outdated or partially updated parameters. To mitigate this, we implemented a local Convergence Check mechanism that continuously monitors performance stability and prevents premature round completion. This guarantees that decentralized aggregation converges towards a stable model approximation even if there is no strict synchronization.

9. Conclusion & Future work

9.1 Conclusion

This project has shown that a decentralized federated learning (DFL) architecture, which eliminates reliance on a central server by allowing direct peer-to-peer coordination among peers, is feasible. By adding lightweight gRPC servers and clients into each node and therefore extending the Flower framework, we have enabled peers to train, exchange and aggregate their model parameters independently. The system performed well over multiple training rounds with all nodes performing correctly, keeping stable connections, aggregating parameters in a distributed manner, and finishing training without errors.

Both experimental results from the baseline client-server setup and the P2P setup clearly show convergence. Training loss in the P2P implementation continuously decreased during the rounds and accuracy increased from 0.089 to 0.213, even without a central aggregator. These results demonstrate that decentralized learning can lead to meaningful convergence, even in small, distributed networks. Additionally, this project revealed three important challenges of decentralized learning: communication overhead in a fully connected topology, asynchronous model drift, and the tight coupling of Flower's framework with the server-orientated logic which revealed the practical limits of decentralization in the deployment of real applications.

In summary, the project shows that decentralized federated learning is a valid alternative to the traditional centralized federated learning. Decentralized FL offers higher resilience, greater scalability and less dependence on a single point of failure than centralized FL.

9.2 Future Work

To improve this project further we will concentrate on:

Complete Integration: Complete the P2P implementation directly into the Flower core framework..

Scaling Test: Expand the testbed to include a greater number of nodes in order to perform a load test on the gossip protocol.

Scientific Paper: Describe the results obtained in this project in a scientific publication for peer-review.

References

Google Research, "Federated Learning: Collaborative Machine Learning Without Centralized Training .Data," Google AI Blog, 2017

Flower Team, "Flower: A Friendly Federated Learning Research Framework," arXiv preprint .arXiv:2007.14390, 2020

H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient .Learning of Deep Networks from Decentralized Data," in Proc. AISTATS, 2017

A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Peer-to-Peer Learning with Heterogeneous Data," .arXiv preprint arXiv:1901.11150, 2019

X. Hu, Y. Zhao, L. Zhang, T. Chen, and Q. Yang, "Decentralized Federated Learning: A Survey and .Perspective," IEEE Trans. Artif. Intell., 2023

King Khalid University, Department of Computer Science, "Decentralized Federated Learning: Graduate .Project Slides," 2025

P. Kairouz et al., "Advances and Open Problems in Federated Learning," arXiv preprint .arXiv:1912.04977, 2021

M. Abadi et al., "Deep Learning With Differential Privacy," in Proc. ACM Conf. Computer and .Communications Security (CCS), 2016

P. Blot, M. Cord, and N. Thome, "Gossip Training for Deep Learning," arXiv preprint arXiv:1803.05880, .2020

Y. Shi, L. Shen, K. Wei, Y. Sun, B. Yuan, X. Wang, and D. Tao, "Improving the Model Consistency of .Decentralized Federated Learning," arXiv preprint arXiv:2302.04083, 2023

B2DFL: Bringing Butterfly to Decentralized Federated Learning Assisted with Blockchain," Journal of" Parallel and Distributed Computing, 2024. Available:
https://www.sciencedirect.com/science/article/abs/pii/S0743731524001424

A. Lalitha et al., "Peer-to-Peer Learning with Heterogeneous Data," arXiv preprint arXiv:1901.11150, 2019.

X. Hu et al., "Decentralized Federated Learning: A Survey and Perspective," IEEE Transactions on .Artificial Intelligence, 2023

P. Kairouz et al., "Advances and Open Problems in Federated Learning," arXiv preprint .arXiv:1912.04977, 2021

H. B. McMahan et al., "Communication-Efficient Learning of Deep Networks from Decentralized Data," .arXiv preprint, 2017

M. Abadi et al., "Deep Learning with Differential Privacy," ACM Conference on Computer and Communications Security (CCS), 2016