FlexiFed: Adaptive Resource-Based Client Topology Selection and Similarity-Based Aggregation in Decentralized Federated Learning

Vo Van Truong[1], Pham Khanh Quan[1] and Taehong Kim*[1]

[1] School of Information and Communication Engineering, Chungbuk National University, Cheongju, Korea

{truongvo, khanhquan, taehongkim}@cbnu.ac.kr

Abstract

In decentralized federated learning (DFL), optimizing system performance amid heterogeneous client resources and non-IID data remains challenging. Traditional methods like FedAvg and FedProx face limitations due to their simplistic handling of these issues. Recent approaches such as FedHP [6], CoCo [14], and YOGA [7] attempt to address these challenges using coordinators to manage client interactions. However, these coordinators can become bottlenecks due to their reliance on maintaining robust connections across all clients. This paper presents the FlexiFed algorithm, a two-stage method that dynamically adapts client topology and parameter aggregation without the need for a coordinator. In the first stage, FlexiFed adjusts client topologies based on adaptive resource evaluations, balancing computational load and communication efficiency. The second stage improves parameter aggregation by leveraging similarities between local and neighboring models, enhancing the relevance of aggregated information. Experimental results demonstrate that FlexiFed significantly outperforms traditional methods like FedAvg and FedProx in heterogeneous environments, offering a more scalable and efficient solution without the coordination overhead.

Keywords— *Deep Learning, Machine Learning, Decentralized Federated Learning, Network Topology selection, Parameter Similarity*

I. Introduction

Federated Learning (FL) is a collaborative machine or deep learning technique that involves training models across multiple decentralized devices, thereby preserving data privacy by keeping data localized on each device [9]. In a typical FL setup, each client trains its local model on its data and then shares only the model parameters with a central server. The server aggregates these parameters to create a global model, which is then redistributed to the clients. However, this centralized approach faces significant challenges such as scalability issues, security vulnerabilities, and the risk of a single point of failure as the number of participating clients increases [1, 2, 16].

Decentralized Federated Learning (DFL) addresses these limitations by removing the central server and enabling direct communication between clients [4, 16, 10]. In DFL, clients interact with their neighbors or within local clusters, thereby reducing latency and distributing computational tasks more evenly across the network. This decentralized structure enhances scalability and mitigates the risks associated with a single point of failure. Furthermore, by distributing model parameters among multiple clients instead of sending them to one central server, DFL can improve security by making it harder for attackers to compromise the system.

Despite the potential advantages of DFL, implementing it in resource-constrained edge computing environments presents unique challenges. Edge devices often have limited computational power, memory, and communication bandwidth, which can hinder the efficiency of the training process. Additionally, the data collected by edge devices is often heterogeneous non-independent and identically distributed (non-IID) [9], complicating model training and convergence. Moreover, each client sends their training parameters to every neighbor in the system, which increases the communication cost and can lead to significant network congestion. This overhead can be particularly problematic in large networks with many devices, where frequent communication can quickly saturate available bandwidth. Furthermore, the variability in device capabilities and network conditions can cause synchronization issues, where slower devices lag behind, delaying the overall training process.

In this paper, we introduce FlexiFed, a novel framework designed to address these challenges and optimize DFL in edge environments. FlexiFed eliminates the need for a central coordinator, allowing each client to autonomously manage its resources and select optimal topologies based on adaptive resource-based and similarity-based [15] parameter aggregation. Our framework enhances the robustness and scalability of DFL, making it well-suited for dynamic and heterogeneous edge scenarios. The contributions of this paper are as follows:

- **Adaptive Client Topology Selection (Stage 1):** We propose FlexiFed, a mechanism that dynamically adjusts the client topology based on resource availability. By evaluating and normalizing client resources, the algorithm ensures an optimal and fair distribution of connections among clients, which reduces communication cost and enhances the efficiency of the learning process.
- **Similarity-Based Parameter Aggregation (Stage 2):** FlexiFed also utilizes a similarity-based aggregation mechanism that weights model updates according to the similarity of local models. This approach not only improves the overall model accuracy but also maintains stability across different communication rounds, addressing the challenges posed by non-IID data distributions.
- **Heterogeneous and non-IID data evaluation:** The proposed method was rigorously evaluated using a MobileNetV2 model in an edge computing environment with varying CPU and RAM capacities. The experiments demonstrated that FlexiFed outperforms both the traditional FedAvg [12] algorithm and the FedProx [5] algorithm in terms of accuracy, communication cost, processing time, and convergence speed, particularly in scenarios with heterogeneous resources and non-IID data distributions.

II. Related works

This section examines practical DFL implementations, reviewing relevant studies that evaluate these systems. DFL presents a promising solution for training models on distributed datasets. However, as highlighted by surveys from Martínez *et al.* [8] and Yuan*et al.* [16], DFL faces significant challenges. These include limited resources (CPU, RAM, bandwidth) on edge devices, non-IID data distribution [17] across clients, and client heterogeneity. Such as edge devices have varying capabilities and data distributions, leading to slow convergence and reduced model accuracy.

To address these challenges, Liao *et al.* [6] proposed FedHP, a framework that adaptively controls both local updating frequency (how often devices update their models) and network topology (which devices communicate). They aim to balance convergence speed and accuracy by dynamically determining updating frequencies and constructing the topology using an optimization algorithm. However, FedHP relies on a central coordinator for global information and topology construction, it assumes full connectivity among clients, which is a significant limitation in real-world edge computing environments.

Tang *et al.* [13] presented GossipFL, a communication-efficient decentralized federated learning framework that incorporates sparsified communication and adaptive peer selection. GossipFL significantly reduces communication traffic by allowing clients to exchange only a sparse subset of their model parameters. It also utilizes a bandwidth-aware gossip matrix to dynamically choose communication partners. While GossipFL addresses communication efficiency, it still relies on a coordinator for managing the communication topology, which introduces a single point of failure and limits scalability. In contrast, our proposed framework, FlexiFed, is fully decentralized, eliminating the need for a coordinator and enhancing robustness. Each client dynamically selects its neighbors based on available resources, allowing for adaptation to changing network conditions.

Wang *et al.* [14] proposed CoCo (Collaborative Consensus) to address the challenges of non-IID data in decentralized federated learning. CoCo prioritizes selecting peers with significant divergence in data distribution, enabling clients to learn from more diverse datasets and improve model accuracy. However, CoCo relies on a central coordinator for managing peer selection, which can limit scalability and introduce a single point of failure. Our proposed framework, FlexiFed, overcomes this limitation by enabling clients to dynamically choose their neighbors based on available resources, eliminating the need for a coordinator. Furthermore, our approach incorporates similarity-based model aggregation, making it more robust against non-IID data than CoCo, which only leverages data distribution for peer selection.

YOGA [7] proposed a decentralized federated learning framework that employs layer-wise model aggregation to reduce communication costs. It dynamically determines which layers of a model each client should share with other clients, enhancing communication efficiency. YOGA uses a coordinator to manage this process. While this approach achieves decentralization without a parameter server, it still relies on a coordinator for global communication.

To the best of our knowledge, this is the first paper to propose FlexiFed, a hybrid DFL framework that utilizes dynamic neighbor selection based on resource availability, combined with similarity-based model aggregation. This is not only enhances robustness against non-IID data but also improves scalability and resilience to single points of failure. By allowing each client to independently manage its communication strategy, FlexiFed adapts seamlessly to varying edge conditions, offering a flexible and efficient approach to FL.

III. Methodologies

Figure 1 illustrates a novel two-stage architecture for adaptive parameter aggregation in a distributed learning system. The proposed approach aims to optimize the performance and efficiency of the learning process by dynam-

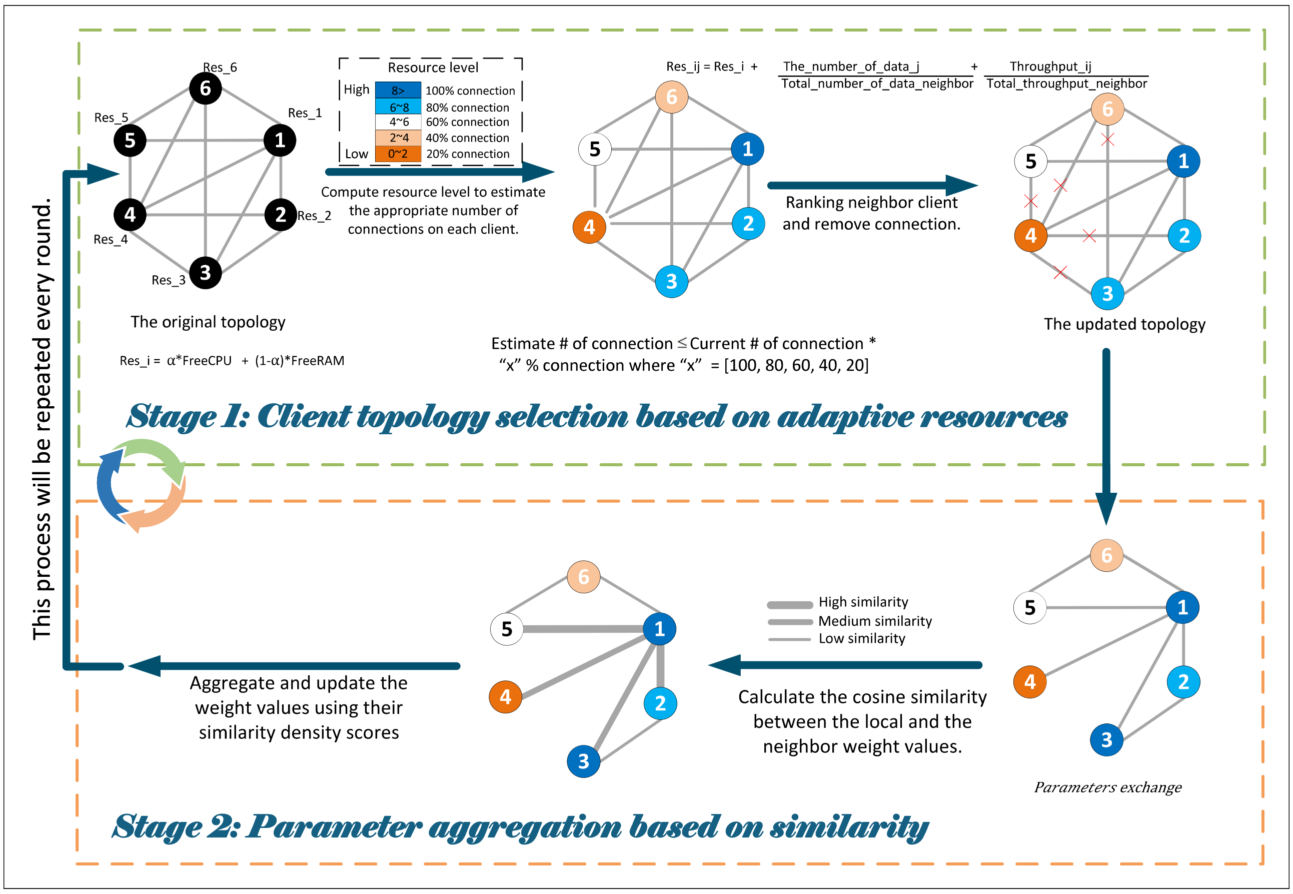
Res_6
Res_5
Res_1
Res_4
Res_2
Res_3
6
5
1
4
2
3
The original topology
Res_i = α*FreeCPU + (1-α)*FreeRAM
Resource level
High
Low
8> 100% connection
6~8 80% connection
4~6 60% connection
2~4 40% connection
0~2 20% connection
Compute resource level to estimate the appropriate number of connections on each client.
Res_ij = Res_i + The_number_of_data_j / Total_number_of_data_neighbor + Throughput_ij / Total_throughput_neighbor
Ranking neighbor client and remove connection.
The updated topology
Estimate # of connection ≤ Current # of connection * "x" % connection where "x" = [100, 80, 60, 40, 20]
Stage 1: Client topology selection based on adaptive resources
This process will be repeated every round.
High similarity
Medium similarity
Low similarity
Calculate the cosine similarity between the local and the neighbor weight values.
Aggregate and update the weight values using their similarity density scores
Parameters exchange
Stage 2: Parameter aggregation based on similarity

Fig. 1. Architecture.

ically adjusting the client topology and aggregating parameters based on client resource availability and weight similarity.

Stage 1 focuses on selecting the client topology dynamically based on the adaptive resources available to each client in the network. This stage involves several steps to evaluate the resources of each client, estimate the appropriate number of connections, and update the topology accordingly. In the original topology, there are six clients (nodes) interconnected in a completed graph, each client has specific hardware resources in terms of CPU cores and RAM. Then a resource index Res_i for each client i is calculated by combining the available free CPU cores and RAM as follows:

$$Res_i = \alpha FreeCPU + (1-\alpha)FreeRAM \tag{1}$$

The resource level and the estimation of the number of connections are closely related in DFL. The resource level of each client determines how many connections it can handle effectively, ensuring optimal performance and resource utilization. Higher resource levels indicate that a client has more available CPU cores and RAM, allowing it to manage more connections without experiencing performance degradation. Conversely, lower resource levels suggest limited computational and memory resources, necessitating fewer connections to prevent overloading and ensure efficient operation. This dynamic adjustment of connections based on resource levels is crucial for maintaining a balanced load across the network, enhancing the overall efficiency and robustness of the FL process. Based on categorizing the resource level, the appropriate number of connections for each client is estimated based on its current number and the percentage of connections.

After estimating the appropriate number of connections for each client based on their hardware resources, the next step involves calculating the resource level between clients by considering the number of training samples and throughput information among neighboring clients. The resource level between any two clients i and j (Res_{ij}) is calculated using the formula below:

$$Res_{ij} = Res_j + \frac{Data_j}{TotalData} + \frac{Throughput_{ij}}{TotalThroughput} \tag{2}$$

This formula incorporated the normalized resource level of the neighboring client j (Res_j). It also included the ratio of the data ($\frac{Data_j}{TotalData}$) handled by the neighboring client j to the total number of training samples across all neighboring clients connected to client i. Additionally, it factored in the throughput ratio ($\frac{Throughput_{ij}}{TotalThroughput}$) using the throughput information from client i to j and the total throughput across total neighboring connections. Using the resource level between client data, clients rank their neighbors. This ranking is crucial as it identifies which connections are less efficient and should be removed to enhance the overall network performance. For instance, connections involving Client 4 and Client 6, which have lower throughput and resource levels, are marked for removal. The updated topology reflects these adjustments, showing a network where inefficient connections have been pruned. By integrating these components, the formula aimed to ensure a balanced resource distribution, enhance data handling efficiency, and optimize the communication throughput between clients, thereby improving the overall performance and fairness of the network.

Stage 1 ensures that only the most efficient and high-throughput connections are maintained, leading to an optimized and streamlined network. The iterative nature of this process ensures continuous optimization, as the topology is reassessed and adjusted in each round based on the latest available hardware resources and throughput data.

Stage 2 of the proposed architecture focuses on optimizing the aggregation of parameters in a decentralized federated learning system by calculating the similarity between the local and neighboring weight values. This stage ensures that the aggregated parameters reflect the most relevant and accurate data, enhancing the overall learning performance.

The process begins with calculating the similarity between the local client's weight values and those of its neighbors. This similarity is determined using similarity density scores, which quantify how closely the weight values of one client match with those of another. The similarity between a local client and a neighboring client is represented visually using different line thicknesses between nodes in the graph: high similarity is depicted with thick grey lines, medium similarity with medium grey lines, and low similarity with thin grey lines. For instance, in the updated topology, client 1 has high similarity with client 2 and client 5, as indicated by the thick lines connecting these nodes. On the other hand, client 1 has medium similarity with clients 3 and 4, indicated by medium grey lines, and low similarity with client 6, represented by thin grey lines.

Once the similarity scores are calculated, each client aggregates its weight values by considering the weight values of its neighbors, weighted by their similarity scores. This weighted aggregation ensures that clients with higher similarity have a more significant influence on the aggregated weight. After calculating the aggregated weight, the client's weight is updated iteratively by combining the current aggregated weight with the previous weight. This update is controlled by a mixing parameter that balances the influence of the new aggregated weight and the previous weight. This two-stage approach not only optimizes resource usage and bandwidth communication efficiency but also ensures continuous improvement in dynamic and resource-constrained environments.

IV. Algorithm

Algorithm 1 proposes FlexiFed, a novel resource-adaptive approach for hybrid client topology selection and similarity-based parameter aggregation in DFL. As already mentioned in section iii., the algorithm is structured into two primary stages: client topology selection based on adaptive resources and parameter aggregation based on similarity. This section provides a detailed analysis of each stage, highlighting the key operations and their impact on the overall efficiency and performance of the DFL process.

The first stage of the algorithm focuses on dynamically adjusting the client topology based on the available resources of each client. The following steps outline the process:

- **Resource Level Calculation:** Each client i computes its resource level Res_i by considering both the free CPU and RAM resources. This computation uses a weighted sum where α balances the contributions of CPU and RAM resources.
- **Connection Estimation:** Based on the resource levels Res_i, the algorithm estimates the appropriate number of connections for each client. This estimation helps determine the network load each client can handle efficiently.
- **Resource Level Calculation for Connections:** For each client connection, a resource level Res_{ij} is calculated. This metric considers the normalized resource level of the client, the ratio of data samples D_j to the total data of neighbors, and the throughput T_j to the total throughput of neighbors.
- **Ranking and Topology Update:** The connections are ranked based on the calculated resource levels Res_{ij}. Clients then update their topologies by selecting the top connections, ensuring that the network is optimized according to the adaptive resource levels.

Once the topology is updated, the algorithm proceeds to the second stage, which focuses on aggregating parameters based on the similarity of local models:

- **Model Exchange:** Clients exchange their local model parameters with their selected neighbors.
- **Similarity Calculation:** We calculate the cosine similarity between the local model w_i and each neighbor's model w_j. Cosine similarity measures the angular difference between two vectors, representing the models' feature spaces. This similarity metric helps identify the most relevant neighbors for parameter aggregation, ensuring that models from similar clients contribute more to the aggregated model.

Algorithm 1 FlexiFed Algorithm

```
    Input: $w$: local model parameters for each client;
    $R$: Number of communication rounds;
    $D$: Number of training samples on each client;
    $L$: List of connected neighboring clients;
    $T$: Throughput exchanged for each connection.
    Output: Updated local model parameters $w$ for each client.
 1: procedure
 2:     Initialize each client with its local model $w$ and training sample $D$.
 3:     Each client selects its own neighbors as a list $L$ based on the original topology.
 4:     for $r$ = 1 to $R$ do
 5:         Train the local model parameter $w^r$ on local dataset $D$ for each client $i$.
 6:         Stage 1: Client Topology Selection based on Adaptive Resources
 7:         Compute the resource level on each client $i$:
                $Res_i = \alpha FreeCPU + (1-\alpha) FreeRAM$
 8:         Compute the resource level for each $ij$ client connection (Eq. 2):
                $Res_{ij} = Res_j + \frac{D_j}{TotalData} + \frac{Throughput_{ij}}{TotalThroughput}$
 9:         Rank all the $Res_{ij}$ connections and update the list of connected neighboring client L (Update the topology).
10:         Stage 2: Parameter aggregation based on similarity
11:         Exchange the model parameters with neighbors in list $L$.
12:         Compute similarity scores between the local model $w_i^r$ and received neighbor model $w_j^r$, with $j \in L$ :
                $s_{ij} = Cosine\_similarity(w_i^r, w_j^r)$
13:         Normalize the similarity score and aggregate the neighbor parameters with similarity score:
                $w_{\text{neighbors}} = \sum_{j \in L} s_{ij} w_j^r$ where $\sum_{j \in L} s_{ij} = 1$
14:         Update the new model parameter $w_i$, weighted by their similarity density scores:
                $w_i^{r+1} \leftarrow \beta w_i^r + (1-\beta) w_{neighbors}^r$
15:     end for
16: end procedure
```

- **Normalization:** The similarity scores are normalized to ensure that they sum to one, providing a weighted influence for each neighbor's model in the aggregation process.
- **Neighbor Model Aggregation:** An aggregate neighbor model $w_{neighbors}$ is computed using the normalized similarity scores as weights.
- **Local Model Update:** The local model w_i is updated by combining it with the aggregate neighbor model $w_{neighbors}$, which is incorporated by the β. This factor is critical in balancing the contributions from the local and aggregated neighbor models.

The FlexiFed algorithm effectively combines adaptive resource-based topology selection with similarity-based parameter aggregation to enhance the performance of the DFL system. By dynamically adjusting client connections and leveraging model similarities, FlexiFed ensures efficient utilization of resources and minimizes communication overhead. This approach addresses the challenges of heterogeneous resources and data distribution, making it suitable for a wide range of DFL applications.

V. Results and Discussion

A. Experimental Setup

In this experiment, we deployed a MobileNetV2 [11] model on a simulated edge computing environment with 10 clients. Each edge client was configured with varying CPU and RAM capacities. For instance, the CPU capacities ranged from 1 to 32 units, while the RAM capacities ranged from 1 to 32 units as well. The data distribution among the clients was non-IID, following a Dirichlet distribution with α set to 0.1, using the CIFAR-10 [3] dataset. This setup was designed to evaluate the performance and stability of the MobileNetV2 model under heterogeneous and resource-constrained edge environments.

B. *Experimental Result*

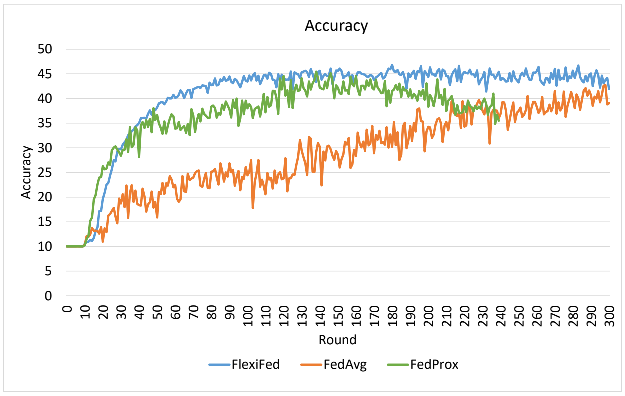

Fig. 2. Global accuracy

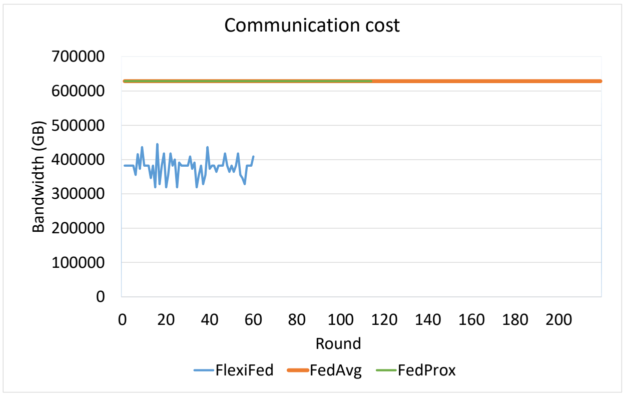

Fig. 3. Communication cost

Figure 2 compares the global accuracy of the proposed FlexiFed algorithm against the traditional FedAvg and the FedProx approach for 10 clients over 300 communication rounds. FlexiFed consistently outperforms FedAvg and FedProx, achieving higher accuracy with improved stability throughout the training process. In the initial stages (rounds 1 to 30), FlexiFed and FedProx demonstrate a more rapid increase in accuracy compared to FedAvg and FedProx exhibiting a significant margin over FedAvg. This quick rise can be attributed to the dynamic client topology adjustments and resource-aware aggregation mechanisms inherent in FlexiFed, which enable efficient utilization of client resources and faster convergence. As training progresses (rounds 51 to 300), FlexiFed maintains a higher and more stable accuracy compared to FedAvg and FedProx, which exhibit greater fluctuations and lower overall accuracy. The stability of FlexiFed highlights the effectiveness of its similarity-based parameter aggregation in maintaining consistent model performance across heterogeneous clients and non-IID data distribution.

Figure 3 shows a clear contrast in the communication cost between FedAvg, FedProx, and FlexiFed across 200 communication rounds. FedAvg and FedProx consistently incur a higher communication cost, maintaining a steady bandwidth consumption of approximately 0.6 GB throughout all rounds. In contrast, FlexiFed shows a significantly lower and more variable communication cost, fluctuating around 0.4 GB. This variability in FlexiFed's communication cost can be attributed to its adaptive topology mechanism, which allows clients to dynamically select their communication clients based on available resources on each round, thereby optimizing bandwidth usage. The lower average communication cost of FlexiFed indicates its efficiency in managing communication overhead, highlighting its advantage over both Fedavg and FedProx in reducing bandwidth consumption while maintaining DFL performance.

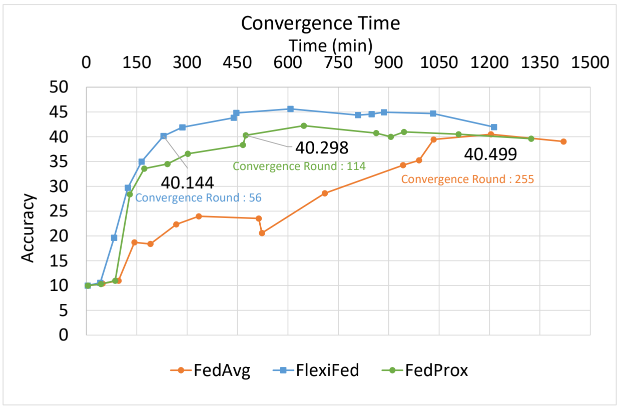

Fig. 4. Convergence time

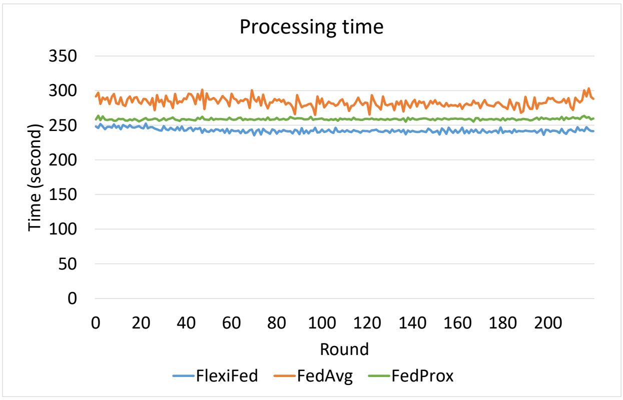

Fig. 5. Processing time

Figure 4 provides a clear comparison between FedAvg, FedProx, and FlexiFed algorithms in terms of their convergence speed. FlexiFed demonstrates a significantly faster convergence rate, reaching its convergence round 56 at around 229 minutes. In contrast, FedProx, while exhibiting rapid initial accuracy growth, converges at round 114 after approximately 470 minutes, taking twice as long as FlexiFed. FedAvg, on the other hand, demonstrates the slowest convergence, reaching a stable state at round 255 after approximately 1204 minutes, more than five times slower than FlexiFed. This remarkable difference in convergence time highlights the efficiency and effectiveness of FlexiFed's approach, which combines adaptive resource-based topology selection and similarity-driven parameter aggregation. The ability of FlexiFed to leverage model similarities and dynamically adapt to heterogeneous resource constraints across clients enables it to converge much more rapidly, making it an effective choice for DFL applications where fast convergence is crucial, such as real-time systems or resource-constrained environments.

Figure 5 illustrates the processing time per round for FedAvg, FedProx, and FlexiFed algorithms. We can observe that FlexiFed frequently exhibits lower processing times than both FedAvg and FedProx. For instance, FlexiFed's processing time averages around 250 seconds per round, which is noticeably lower than FedProx's average of 260 seconds. FedAvg consistently requires approximately 300 seconds per round during that period. This demonstrates FlexiFed's ability to efficiently utilize available resources and leverage model similarities, resulting in reduced computational overhead in certain scenarios. This highlights one of the key strengths of FlexiFed's approach, which is its adaptability to optimize resource usage and processing efficiency, ultimately contributing to its overall faster convergence compared to FedAvg and FedProx.

VI. Conclusion

FlexiFed provides an innovative solution to the challenges of DFL by dynamically adjusting client topologies and refining parameter aggregation based on resource availability and model similarity. Our experimental results show that FlexiFed reduces the communication cost while enhancing convergence time, and accuracy of the learning process in heterogeneous environments, offering significant improvements over traditional DFL approaches. The introduction of adaptive resource-based topology selection and similarity-based aggregation makes FlexiFed a robust framework for diverse and real-world applications, paving the way for more effective and efficient distributed learning systems. Future research will focus on further refining adaptive parameters and scaling the algorithm for larger and more complex networks.

VII. Acknowledgement

This research was partly supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2022R1I1A3072355, 50%) and Innovative Human Resource Development for Local Intellectualization program through the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea government(MSIT) (IITP-2024-2020-0-01462, 50%).

References

[1] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecný, Stefano Mazzocchi, H. B. McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. *ArXiv*, abs/1902.01046, 2019.

[2] Peter Kairouz, H. B. McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary B. Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Salim Y. El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Oluwasanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, R. Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Xiaodong Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14:1–210, 2019.

[3] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[4] S. Shekhar Lalitha, T. Javidi, and F. Koushanfar. Fully decentralized federated learning. In *Third workshop on Bayesian Deep Learning (NeurIPS)*, 2018.

[5] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020.

[6] Yunming Liao, Yang Xu, Hongli Xu, Lun Wang, Chen Qian, and Chunming Qiao. Decentralized federated learning with adaptive configuration for heterogeneous participants. *IEEE Transactions on Mobile Computing*, 23(6):7453–7469, 2024.

[7] Jun Liu, Jianchun Liu, Hongli Xu, Yunming Liao, Zhiyuan Wang, and Qianpiao Ma. Yoga: Adaptive layer-wise model aggregation for decentralized federated learning. *IEEE/ACM Transactions on Networking*, 32(2):1768–1780, 2024.

[8] Enrique Tomás Martínez Beltrán, Mario Quiles Pérez, Pedro Miguel Sánchez Sánchez, Sergio López Bernal, Gérôme Bovet, Manuel Gil Pérez, Gregorio Martínez Pérez, and Alberto Huertas Celdrán. Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges. *IEEE Communications Surveys amp; Tutorials*, 25(4):2983–3013, 2023.

[9] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2023.

[10] Kai Ouyang, Jianping Yu, Xiaojun Cao, and Zhuopeng Liao. Towards reliable federated learning using blockchain-based reverse auctions and reputation incentives. *Symmetry*, 15:2179, 2023.

[11] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.

[12] Tao Sun, Dongsheng Li, and Bao Wang. Decentralized federated averaging, 2021.

[13] Zhenheng Tang, Shaohuai Shi, Bo Li, and Xiaowen Chu. Gossipfl: A decentralized federated learning framework with sparsified and adaptive communication. *IEEE Transactions on Parallel and Distributed Systems*, 34(3):909–922, 2023.

[14] Lun Wang, Yang Xu, Hongli Xu, Min Chen, and Liusheng Huang. Accelerating decentralized federated learning in heterogeneous edge computing. *IEEE Transactions on Mobile Computing*, 22(9):5001–5016, 2023.

[15] Jiajun Wu, Steve Drew, and Jiayu Zhou. Fedle: Federated learning client selection with lifespan extension for edge iot networks, 2023.

[16] Liangqi Yuan, Ziran Wang, Lichao Sun, Philip S. Yu, and Christopher G. Brinton. Decentralized federated learning: A survey and perspective. *IEEE Internet of Things Journal*, pages 1–1, 2024.

[17] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-iid data: A survey, 2021.

Summary of this paper

A. Problem Setup

Decentralized federated learning (DFL) holds immense promise for training models on decentralized datasets, particularly in edge computing environments. However, DFL faces challenges in dynamic and resource-constrained edge settings. Clients often have limited CPU, RAM, and bandwidth, leading to communication bottlenecks and slow training. Furthermore, non-IID data across clients can significantly degrade model accuracy. Existing DFL methods often rely on a central coordinator to manage communication and resource allocation, creating a single point of failure and limiting scalability. This research addresses these limitations by introducing a novel, fully decentralized framework that empowers clients to manage their resources and communication partners, enhancing robustness and scalability in dynamic edge environments

B. Novelty

This introduces a novel DFL framework, FlexiFed, designed for resource-constrained edge environments. Our framework operates entirely without a coordinator, empowering clients to manage their resources, select neighbors, and aggregate models autonomously. Clients dynamically adapt their connections based on resource availability, data similarity, and bandwidth, creating a flexible and efficient topology. We also introduce similarity-based model aggregation, which weights the contributions of neighbor models based on their similarity to the local model, enhancing robustness against non-IID data. This fully decentralized approach, combined with resource-adaptive communication and similarity-based aggregation, offers a practical and efficient solution for federated learning in edge settings.

C. Algorithms

The core innovation of FlexiFed lies in its resource-adaptive topology and similarity-based model aggregation. Clients dynamically assess their available resources (CPU, RAM, and bandwidth) and select the most suitable neighbors based on a combination of their own and their neighbors' resources, data distribution, and throughput. This dynamic topology ensures efficient communication and minimizes waiting times for slower clients. Additionally, our framework incorporates similarity-based model aggregation, which leverages cosine similarity to measure the alignment between feature spaces of local models. This approach provides a robust measure of similarity, even when models have different magnitudes, ensuring that models from similar clients contribute more to the aggregated model. This similarity-based aggregation improves robustness against non-IID data.

D. Experiments

The results showcase FlexiFed's consistent outperformance over FedAvg and FedProx in terms of global accuracy, communication cost, convergence time, and processing time. FlexiFed demonstrates rapid accuracy improvements and higher stability through dynamic topology adjustments and similarity-based parameter aggregation. It achieves significantly faster convergence with fewer communication rounds while optimizing bandwidth usage via adaptive topology management. FlexiFed frequently exhibits lower processing times per round, indicating superior resource utilization and reduced computational overhead. Overall, FlexiFed's adaptive topology management and similarity-based aggregation contribute to its superior performance in DFL, making it well-suited for real-time and resource-constrained scenarios.