Communication Topologies for Decentralized Federated Learning

Michael Dötzer
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

Yixin Mao
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

***Abstract*—In this paper, we consider implementing federated learning with different topologies in a network without a central entity. A simple method is induced to estimate the communication effort of a topology. Experimental results show that models in a decentralized network with certain topologies can achieve similar performances as models in a centralized network. The results also indicate that cluster topologies that leverage similarities of data distributions mitigate the communication effort without sacrificing performance.**

***Index Terms*—Federated learning, clustering applications, network topology**

I. Introduction

The rapid development of technology has substantially reduced the size and cost of devices while improving their capabilities. This has enabled the use of a large number of IoT devices in various fields, such as industrial artificial intelligence [1] and health monitors [2]. Billion more IoT devices are expected over the next several years.[1] These devices will collect or generate large amounts of data. A traditional centralized method of machine learning, where all data is aggregated on a central server, seems unlikely. First, the storage or bandwidth capacity of a central server could reach its limits and model training on that scale of data could be computationally intensive. Second, companies, users or devices might prefer not to share data for privacy reasons. Under these circumstances, a new important paradigm, federated learning (FL), has emerged in large-scale machine learning.

FL, proposed in [3], aims at enabling clients to contribute to a global training process while keeping their own data local. The authors also introduce the federated averaging algorithm (FedAvg). They show that FedAvg is robust in training convolutional neural networks (CNNs) on some widely used image classification datasets, for example, MNIST [4] and CIFAR-10 [5].

In a network with many devices collecting data, it is likely that some of the devices will train on data with similar distributions. It is known that models which are trained on data with similar distributions perform similar model updates [6]. Studies have been done on how to take advantage of this property. In [7], clients with similar data distributions are put into one cluster to accelerate global training. In [8] the authors focused on specialized cluster models so it becomes a multi-task problem after clustering the clients.

However, these studies are based on a network where a central server is available. It has been scarcely studied how these methods can be extended to a central-server-free network. The goal of our work is to compare the performance of federated learning in different topologies in a central-server-free network. Our main focus is on the communication overhead generated by Federated Learning. We do not address aspects of data privacy.

The contribution of our paper can be listed as follows:

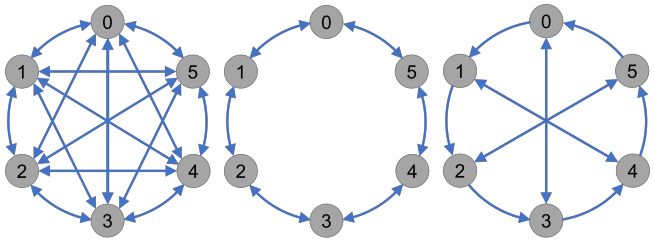
- We induce a simple method to estimate the information transfer performance of a topology and confirm the estimates empirically.
- The results show that the models in decentralized topologies can converge to a similar level of performance as the models in a centralized network.
- We show from empirical results that communication overhead can be reduced by clustering clients with training data with similar distributions without sacrificing performance.

II. Related Work

As the number of devices on the network increases, it can no longer be assumed that there is a central entity with sufficient bandwidth or computing resources to handle the volume of requests. Therefore, realistically, we cannot always deploy a central node. Against this background, various approaches of decentralized federated learning (DFL) have been proposed [9]–[12]. Such a method can also avoid a major communication bottleneck, as each client communicates with its neighbors rather than with a central server. DFL is based on decentralized optimization and decentralized stochastic gradient descent (SGD), where using a gossip-based model averaging instead of a central server. In [13], a central-server-free framework that uses a hierarchical architecture was proposed. Instead of a central server, this framework uses multiple edge servers, with each coordinating a few clients.

Clustering clients that gather data with similar distribution is an intuitive way to design a topology. There are various methods in the literature that use probability distributions to quantify the distance (or similarities) between different

[1]https://www.statista.com/statistics/1183457



(a) Fully connected (b) Ring (c) Wheel

Fig. 1: The three backbone topologies that are tested in this paper

data partitions. In [7], Hellinger distance [14] is applied to two types of data distributions: label distribution $P(y)$ and conditional distribution $P(x|y)$. A more popular metric to measure the similarity is the earth mover's distance (EMD) which is also called Wasserstein metric [15]–[17]. There are also approaches that measure the similarity of models or properties of them. For example in [18], the cosine similarity is inspected between gradient updates of clients to distinguish their hidden data distributions. In [19], the similarity of clients is measured by comparing the local updates to the joint global model.

The topology of the network graph plays an important role in FL. For example, an intuitive thought for a topology that ensures a sufficient interconnectivity is a ring-based topology, as shown in [20], [21]. The authors demonstrate that CNNs in this topology are able to converge, but take a long time to do so. An exchange between a node and the furthest node requires $(n-1)/2$ iterations of the federation process, where n is the number of clients. Based on this finding, information exchange is accelerated by allowing each client to communicate with another client $2^i + 1$ hops away in the ring, where $i \in [0, \log(n) - 1]$ [22].

In [21], a new topology called Ring-Star was proposed, which divides clients into different groups and selects one of them as the so-called representative. This representative is responsible not only for merging the models within his group, but also for communicating with two neighboring representatives.

In [23], experiments are done on different network graphs. The results show that a hierarchical organization of the network can increase the rate of convergence.

III. Topologies

In our work we use a two layer approach and refer to these layers as backbone topology and cluster topology. The backbone topology specifies how the clusters are interconnected. The cluster topology defines how clients within one cluster are connected to each other.

A. Backbone Topology

We use three backbone topologies in this paper, namely fully connected, ring, and wheel topology. Figure 1 shows illustrations of the backbone topologies. We use the fully connected network to show how the performance of clients would be under the best connectivity condition. Since we designed the wheel and ring topologies to have the same number of directed communication links, they are quite comparable. In the following paragraphs we show that these two topologies differ in information transfer performance. Please note, that in this paper we refer to the learned properties of a model as model parameters.

We estimate the information transfer performance by calculating the standard deviation between the client's model parameters w and the average of parameters of all models. We expect that the lower the standard deviation within a topology, the closer the models of the nodes are to consensus. If nodes have an averaged model of $(\sum_{i=1}^{n} w_n)/n$ then the standard deviation is 0 which would be the optimum for central-server-free FL (consensus model). w_i^t denotes the model parameters of the i-th client after t-th round of communication. A round of communication involves all clients sharing their model parameters along all links in the topology design.

A prerequisite for our implication on the consensus is that each model is equally important for the global and optimal model. We are able to estimate the relative performance of information transmission by comparing the standard deviation of the different topologies in this manner.

Without loss of generality, we use a simplified decentralized network with 6 nodes for demonstration. This corresponds to the topologies shown in figure 1.

After local training and update, all nodes have their own model parameters $w_0^0 ... w_5^0$ at time $t = 0$. For simplicity, it is assumed that the training is completed and only communication, or more precisely federation, continues. In this way, the model parameters w_i are static in the federation process. The model consensus is defined as $(w_0^0 + w_1^0 + w_2^0 + w_3^0 + w_4^0 + w_5^0)/6$ and not affected by new model parameters that can be brought up by further training.

As an example, the model parameters of node 3 in the ring topology (Figure 1b) are calculated in equation 1. The calculation of the others follows the same scheme because of the symmetry of the used topologies. After the first round of communication at time $t = 1$, the new model of node 3 comprises the average of model parameters of itself and its two neighbors. The subsequent steps follow the same scheme.

The comparison of these three topologies is plotted in figure 2. According to this result, the wheel topology seems to have a higher information transfer performance than the ring topology.

Please note, that our approaches on estimating the information transfer performance or the communication effort of topologies are based on the number of necessary communication interactions per round. Our estimates disregard aspects that are important in reality, such as transfer volumes and bandwidths, since these are not subject of our work.

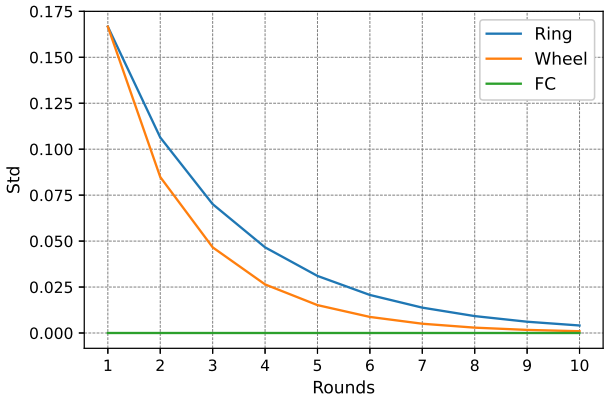


Fig. 2: Standard deviation of model parameters decreases with communication rounds

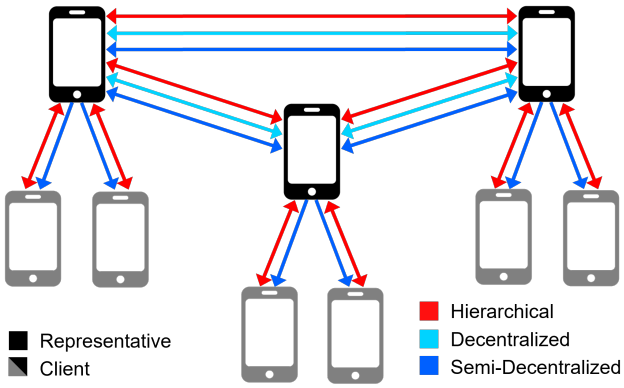


Fig. 3: Cluster topologies with fully connected backbone topology

$$
\begin{aligned}
&\textit{Node } 3 \\
&\textit{Round } 1: \quad w_3^1 = \frac{w_2^0 + w_3^0 + w_4^0}{3} \\
&\textit{Round } 2: \quad w_3^2 = \frac{w_0^0 + 2w_1^0 + 3w_2^0 + 2w_3^0 + 1w_4^0}{9} \\
&\textit{Round } 3: \quad w_3^3 = \frac{2w_0^0 + 3w_1^0 + 6w_2^0 + 7w_3^0 + 6w_4^0 + 3w_5^0}{27}
\end{aligned} \tag{1}
$$

B. Cluster Topology

In our work we implement three cluster topologies: hierarchical topology where all clients within a cluster contribute to the cluster model, semi-decentralized topology where only representatives of the clusters do the contribution, and a decentralized topology. Figure 3 illustrates the topologies. The black devices are the representatives of a cluster. They usually have the most data and/or highest cross-linking degree within the cluster. Grey devices represent normal clients.

The basis for hierarchical and semi-decentralized FL topologies is clustering. In figure 3 the red lines form the structure of a hierarchical topology. There are two types of communication in the hierarchical topology: cluster (intra-cluster) and backbone (inter-cluster) communication.

In cluster communication in the hierarchical topology, all members of a cluster perform an federation process within the cluster. The representative of a cluster behaves similar to the central server in a conventional centralized architecture: It collects the model parameters from all members, computes a synthesis from them, and sends it back to all other clients.

In backbone communication the representatives of each cluster communicate with each other in a certain backbone topology. They aggregate the shared cluster models and send the model parameters to the members of their cluster. All members of a cluster fully adopt the model from their representative by substituting all model parameters.

The structure and procedure of the semi-decentralized topology is similar to the hierarchical topology. The difference is that only representatives contribute to the global model. In figure 3, only the blue lines form the structure of a semi-decentralized framework. All representatives perform local updates and share their model parameters according to the backbone topology. Subsequently they share their model with all members of their cluster. All members of a cluster fully adopt the model from their representative. This approach is based on the assumption that the amount of training data of the representatives is sufficient and their data distribution is resembling the data distribution of its cluster. It is a way of making use of data similarity to reduce communication overhead.

Another advantage of the semi-decentralized topology is that it also reduces the computational overhead of the cluster members, since they do not perform any training. This is beneficial for clients with limited communication and computing capacity, such as IoT devices.

The cyan lines in figure 3 illustrate the structure of a decentralized topology. Each client can be considered as a node in the backbone topology. This topology does not use a second (cluster) layer.

In our experiments, we test cluster and backbone topologies in combination rather than separately. Note that models in the combination of decentralized and fully connected topology perform the same as models in FL setting with a central server because each client is connected to all the others. This setting can be considered as a reference.

To limit the scope, we confine our paper to the basic topologies just presented. We selected them as they have an intuitive structure and are frequently used as network topologies. Other topologies and combinations as well as additional layers would be possible.

IV. Experiments

A. Experimental Setup

For our empirical tests, we utilize a widely used machine learning problem, namely image classification. We use datasets, models and methods that we consider to be realistic.

We perform the experiments on three datasets that are commonly used in image classification, i.e. MNIST [4], Fashion-MNIST [24] and CIFAR-10 [5]. We used two partitioning methods to generate non-IID data from all three datasets. The first method uses Dirichlet distribution to partition the original dataset. This approach has been widely used to generate a low degree of non-IIDness [25]–[28].

In our second partitioning method, we assign only samples from two classes to each client. This is to represent an extreme case with a high degree of non-IIDness. We tried several methods for measuring the distances of data distributions using the MNIST dataset. The Hellinger distance [14] turned out to be the best interpretable method and is therefore used throughout this paper.

We selected convolutional neural network (CNN) models for the experiments based on high accuracy and fast rate of convergence with the respective dataset while maintaining a manageable computational load.

For the MNIST dataset, we utilize a CNN model with two convolutional layers followed by a max pooling layer, a dropout layer [29] and two fully connected layers.

For the Fashion-MNIST dataset, we use a CNN model with two convolutional layers followed by a batch normalization layer, a max pooling layer and one fully connected layer.

For the Cifar10 dataset, we use VGG-11 [30]. It is a standard CNN architecture that contains eight convolutional layers, five max pooling layers and three fully-connected layers. A detailed model architecture of all three models is shown in the appendix.

We add momentum to the SGD optimizer to prevent it from getting stuck in a local minimum or saddle point. Based on empirical findings, we set the momentum to 0.5, the learning rate to 0.01 and the batch size to 30. The number of clients is 50, which is large enough to show the benefits of clustering clients and small enough to keep the computational effort reasonable.

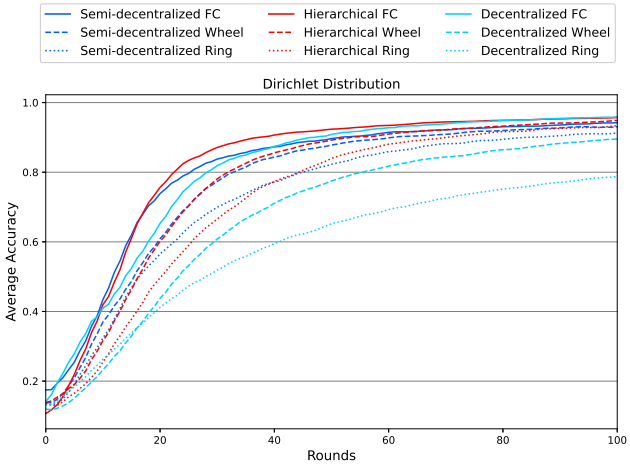
We assume that clients with data with similar distributions are also close to each other in the physical network. Therefore, we expect that exchanging model parameters among them will improve their performance insignificantly.

We build clusters by using the density-based spatial clustering of applications with noise (DBSCAN) algorithm [31]. We split all clients into ten clusters. We thus create the same number of clusters as classes with the intention to ensure similar training data distribution within each cluster.

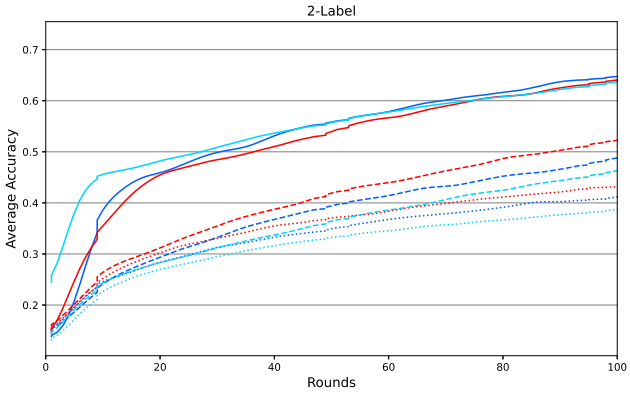
Clusters in all topologies or clients in the decentralized topology are randomly positioned. The aim is to minimize the influence of the position on the performance.

We define τ_1 as the frequency of cluster communication and τ_2 as the frequency of backbone communication. In the decentralized topology τ_2 is irrelevant, as there is only backbone communication.

For the MNIST dataset, we use $\tau_1 = \tau_2 = 2$. For the FashionMNIST and the Cifar10 dataset, we use $\tau_1 = \tau_2 = 1$ as the training tasks of the two datasets are more complex.



(a) Dirichlet distribution



(b) 2_Label

Fig. 4: Average accuracy plotted against round

Every experiment runs for 100 rounds. Each round consists of local training with five batches of data, one iteration of the federation process and a test of the models.

In the following sections, we mainly refer to the results of the experiments done on the MNIST dataset. Results from using the FashionMNIST and Cifar10 dataset show similar trends and can be found in table I. Each experiment is run twice with different model initialization. As both runs yield almost identical results, the average of them is used throughout this paper. For better readability, we smoothed all plots using Gaussian kernel smoother methods.

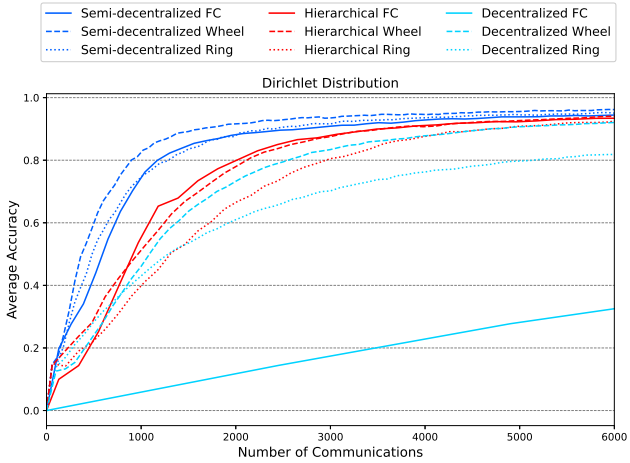
B. Dirichlet Distribution

Figure 4a displays the average accuracy of models in the aforementioned topologies against the rounds. It shows that all combinations except the decentralized ring topology converge to a high accuracy, but with different rates of convergence. The models of clients in a fully connected topology converge the fastest, followed by the models in wheel and ring topologies. We expect this phenomenon, since it corresponds to the overall frequency of exchange of information.

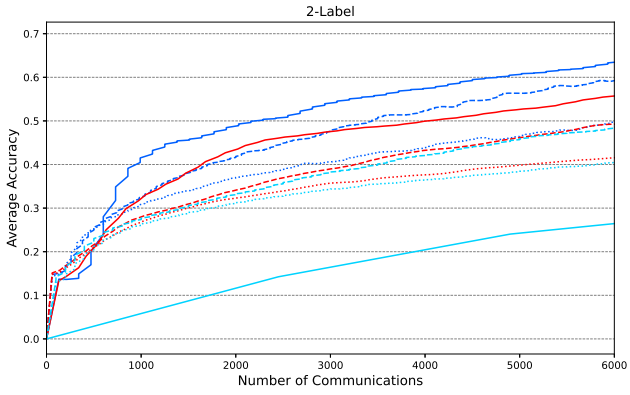
As shown in section III-A, we estimate that the wheel topology has a higher information transfer effort than the ring

TABLE I: The performances of topologies tested on three datasets. Numbers in the parentheses indicate the number of communication

Cluster topology	Backbone topology	Dirichlet_Distribution			2Label		
		MNIST	FashionMNIST	Cifar10	MNIST	FashionMNIST	Cifar10
Decentralized	FC	95.89%(122500)	84.79%(245000)	72.84%(245000)	63.72%(122500)	71.23%(245000)	55.56%(245000)
	Wheel	89.53%(5000)	81.67%(10000)	64.93%(10000)	46.30%(5000)	42.78%(10000)	24.19%(10000)
	Ring	78.93%(5000)	76.56%(10000)	55.84%(10000)	38.71%(5000)	29.01%(10000)	24.03%(10000)
Hierarchical	FC	95.76%(10500)	84.34%(17000)	72.10%(17000)	64.05%(10500)	70.26%(17000)	57.26%(17000)
	Wheel	95.00%(7000)	83.16%(10000)	67.00%(10000)	52.24%(7000)	48.18%(10000)	32.70%(10000)
	Ring	93.43%(7000)	81.17%(10000)	64.58%(10000)	43.15%(7000)	34.07%(10000)	30.74%(10000)
Semi-Decentralized	FC	94.23%(6500)	82.38%(13000)	71.71%(13000)	64.73%(6500)	70.93%(13000)	54.90%(13000)
	Wheel	93.18%(3000)	82.47%(6000)	66.78%(6000)	48.78%(3000)	43.46%(6000)	34.83%(6000)
	Ring	91.42%(3000)	79.41%(6000)	63.11%(6000)	41.16%(3000)	35.52%(6000)	34.46%(6000)



(a) Dirichlet distribution



(b) 2_Label

Fig. 5: Average accuracy plotted against number of communications

topology. Consequently, the models of the clients in the wheel topology should reach consensus faster. This effect can be seen in figure 4a: the models in the decentralized wheel topology converge to notably higher accuracy than the models in the ring topology.

For the ring and wheel topologies, the models in a decentralized topology converge slower than models in the other two cluster topologies. In the fully connected topology, models achieve similar high performances to models in the other two cluster topologies. However, it incurs the highest communication overhead, which can be seen in 5a.

This shows that the decentralized topology is outperformed by the other two cluster topologies in our experiment. Models in all three combinations with a fully connected topology reach similar performance. The also means that our reference, the combination of decentralized and fully connected topology is not superior in terms of performance. Instead the decentralized topology needs the highest number of communications and is often outperformed by the other two cluster topologies in our experiment.

The results also show the significance of making use of data similarity. First, clustering clients based on data similarity accelerates training. It indicates that combining the same backbone topology with the semi-decentralized and hierarchical cluster topologies leads to faster convergences than the combination with the decentralized topology.

Second, models do not improve much if mainly clients with training data with similar distributions communicate. The slower convergence in decentralized topologies reflects this. The clients in the decentralized topology are more likely to be affected by this, as clients in the semi-decentralized and hierarchical topologies are clustered before communication.

Lower connectivity can worsen this situation, as it may take several rounds for a client to receive useful information from other clients. On the other hand, higher connectivity may mitigate the effect, since the rate of convergence of the decentralized fully connected topology is as good as that of the other two cluster topologies combined with the fully connected topology.

In addition to average accuracy relative to rounds, we also consider average accuracy relative to the number of communication interactions required. We use the number of communication interactions as an estimate of the communication overhead caused by the federation process.

We expect the communication overhead of the semi-decentralized topology to be lower than that of the hierarchical topology. We estimate that the lack of a return channel from clients to the representative reduces communication costs without much reduction in performance.

In figure 5a, we plot average test accuracy relative to number of communication interactions. We find that in our experiment the semi-decentralized topology is the most communication-efficient regardless of the backbone topology. It reaches a high accuracy with fewer communication interactions. The decentralized, fully connected topology is the least communication-efficient due to complete interconnection and the associated high communication effort in each round. As the number of clients increases, its efficiency of communication becomes even lower.

C. 2-Label

Figure 4b, where we plot average test accuracy relative to rounds, shows the results of the extreme case in which each client has training data from only 2 classes. The performances of all combinations drop compared to the case where the data is split by the Dirichlet distribution. This can be explained by the higher degree of non-IIDness among the data of the clients. However, the proportions of the performances of the topologies stay similar: the fully connected topologies converge fastest followed by the wheel topology and the ring topology. The hierarchical topology reaches the highest performance in the end and also converges faster than the other two cluster topologies, except when combined with the fully connected topology.

Figure 5b shows average accuracy plotted against number of communication interactions. Unlike in the setup splitting data using Dirichlet distribution, where the semi-decentralized topology is the most communication-efficient cluster topology regardless of the backbone topology, the trend is different now. The best performing combinations are semi-decentralized FC, semi-decentralized wheel, and hierarchical FC. The topologies with the semi-decentralized topology are still better than the same backbone topologies with the other two cluster topologies. Looking at the backbone topologies, the fully connected topology in combination with hierarchical and semi-decentralized performs better than the other two in this case.

This indicates that backbone topologies with low connectivity are more likely to be affected by a high degree of non-IIDness of data: the more uneven the data distribution of clients, the worse ring and wheel topologies perform. This infers how we can handle extreme cases where non-IIDness is high: either we increase the connectivity of the network or we let clients communicate more frequently.

V. Conclusion

In this work, we studied the performances of central-server-free topologies for federated learning. The results show that using similarities of data distributions can speed up training while barely affecting performance. Moreover, if the data distribution of one client is able to represent the data distributions of all other clients in its cluster, then a federation progress within the cluster improves the performance limitedly. In this situation, a federation process only among the clusters performs as well as the same process in combination with a federation within the cluster. This indicates benefits of a hierarchical structure in the topologies. By calculating the standard deviation of model parameters, we show that there can be differences in the effort of information transfer in different topologies even if the number of communications is the same. This can also be seen in differences of rate of convergence and performances in our experiments.

References

[1] M. Hao, H. Li, X. Luo *et al.*, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, 2020.

[2] Q. Wu, X. Chen, Z. Zhou *et al.*, "Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring," *IEEE Transactions on Mobile Computing*, 2022.

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics*, 2016.

[4] Y. Lecun, L. Bottou, Y. Bengio *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[5] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.

[6] Q. Li, Y. Diao, Q. Chen *et al.*, "Federated learning on non-iid data silos: An experimental study," *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 2021.

[7] Y. Wang, J. Wolfrath, N. Sreekumar *et al.*, "Accelerated training via device similarity in federated learning," in *Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking*, 2021.

[8] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multi-task optimization under privacy constraints," *CoRR*, 2019.

[9] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *CoRR*, 2019.

[10] A. Bellet, R. Guerraoui, M. Taziki *et al.*, "Personalized and private peer-to-peer machine learning," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, 2018.

[11] P. Vanhaesebrouck, A. Bellet, and M. Tommasi, "Decentralized collaborative learning of personalized models over networks," *CoRR*, 2016.

[12] W. Liu, L. Chen, and W. Zhang, "Decentralized federated learning: Balancing communication and computing costs," *IEEE Transactions on Signal and Information Processing over Networks*, 2022.

[13] Y. Sun, J. Shao, Y. Mao *et al.*, "Semi-decentralized federated edge learning for fast convergence on non-iid data," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022.

[14] E. Hellinger, "Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen." *Journal für die reine und angewandte Mathematik*, 1909.

[15] Y. Zhao, M. Li, L. Lai *et al.*, "Federated learning with non-iid data," *CoRR*, 2018.

[16] A. Chen, Y. Fu, Z. Sha *et al.*, "An EMD-based adaptive client selection algorithm for federated learning in heterogeneous data scenarios," *Frontiers in Plant Science*, 2022.

[17] Y. Lu, Z. Liu, and Y. Huang, "Parameters compressed mechanism in federated learning for edge computing," in *2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)*, 2021.

[18] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[19] C. Briggs, Z. Fan, and P. András, "Federated learning with hierarchical clustering of local updates to improve training on non-iid data," *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020.

[20] X. Lian, C. Zhang, H. Zhang *et al.*, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[21] M. Jameel, J. Grabocka, M. ul Islam Arif *et al.*, "Ring-star: A sparse topology for faster model averaging in decentralized parallel SGD," in *Machine Learning and Knowledge Discovery in Databases*. Springer International Publishing, 2020.

[22] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *Proceedings of the 35th International Conference on Machine Learning*, 2018.
[23] H. Kavalionak, E. Carlini, P. Dazzi, L. Ferrucci, M. Mordacchini, and M. Coppola, "Impact of network topology on the convergence of decentralized federated learning systems," in *2021 IEEE Symposium on Computers and Communications (ISCC)*, 2021.
[24] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, 2017.
[25] M. Yurochkin, M. Agarwal, S. Ghosh *et al.*, "Bayesian nonparametric federated learning of neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.
[26] Q. Li, B. He, and D. X. Song, "Practical one-shot federated learning for cross-silo setting," in *International Joint Conference on Artificial Intelligence*, 2020.
[27] H. Wang, M. Yurochkin, Y. Sun *et al.*, "Federated learning with matched averaging," in *International Conference on Learning Representations*, 2020.
[28] J. Wang, Q. Liu, H. Liang *et al.*, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in Neural Information Processing Systems*, 2020.
[29] N. Srivastava, G. E. Hinton, A. Krizhevsky *et al.*, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, 2014.
[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[31] M. Ester, H.-P. Kriegel, J. Sander *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.

APPENDIX

Layer	Details
1	Conv2D(1, 10, 5) ReLU, MaxPool2D(2, 2)
2	Conv2D(10, 20, 5) Dropout, ReLU, MaxPool2D(2, 2)
3	FC(320, 50) ReLU, Dropout
4	FC(50, 10)

TABLE II: The model for MNIST: for convolutional layer (Conv2D), parameters with the sequence of input and output dimensions, kernel size are listed. For the max pooling layer (MaxPool2D), kernel size and stride are listed. For the fully connected layer (FC), the input and output dimensions are listed

Layer	Details
1	Conv2D(1, 16, 5) BN(16), ReLU, MaxPool2D(2)
2	Conv2D(16, 32) BN(32), ReLU, MaxPool2D(2)
3	FC(1568, 10)

TABLE III: The model for Fashion-MNIST: for convolutional layer (Conv2D), parameters with the sequence of input and output dimensions, kernel size are listed. For the batch normalization layer (BN), the number of features is listed. For the max pooling layer (MaxPool2D), kernel size is listed. For the fully connected layer (FC), the input and output dimensions are listed