Clustered Hierarchical Distributed Federated Learning

Yan Gou, Ruiyu Wang, Zongyao Li, Muhammad Ali Imran, Lei Zhang

School of Engineering, University of Glasgow, Glasgow, G12 8QQ, UK
Email: {2289306G}@student.gla.ac.uk, {r.wang.1, l.zongyao.1}@research.gla.ac.uk,
{Muhammad.Imran, Lei.Zhang}@glasgow.ac.uk

***Abstract*—In recent years, due to the increasing concern about data privacy security, federated learning, whose clients only synchronize the model rather than the personal data, has developed rapidly. However, the traditional federated learning system still has a high dependence on the central server, an unguaranteed enthusiasm of clients and reliability of the central server, and extremely high consumption of communication resources. Therefore, we propose Clustered Hierarchical Distributed Federated Learning to solve the above problems. We motivate the participation of clients by clustering and solve the dependence on the central server through distributed architecture. We apply a hierarchical segmented gossip protocol and feedback mechanism for in-cluster model exchange and gossip protocol for communication between clusters to make full use of bandwidth and have good training convergence. Experimental results demonstrate that our method has better performance with less communication resource consumption.**

***Index Terms*—Distributed Federated Learning, Clustered, Hierarchical System**

I. Introduction

In recent years, the development of deep learning (DL) growing rapidly in computer vision, natural language processing, and voice recognition areas due to the increase of data processing demand [1]–[7]. However, to improve the performance of DL, a large amount of data is generated from different user equipment (UE) and trained centrally. In this case, the data security of UE can not be guaranteed. Therefore, the UE privacy leakage problem becomes significant. Without privacy protection, sensitive data will be exposed to leaks, attacks, and cyber risks [8]. Therefore, Federated Learning (FL) is motivated to improve privacy protection, which allows UE to share the locally-trained models rather than their original data.

All clients in the FL system store their personal data locally and can train their own local models. Hence, the data collection and processing functions of clients are fully used. Instead of sending data, either model weights or gradients from clients are sent to a central server to build a global model, which is shared with client equipment so that the clients can exchange knowledge with each other. In this way, the local models are trained and built distributedly on client equipment instead of central processing units, which not only protects the personal privacy of clients but also reduces the huge overhead of data collection, the data processing pressure of the central processing unit and the single point of failure to a certain extent.

However, there are some challenges to traditional FL. For example, traditional FL overly relies on a central server to periodically aggregate local model parameters and synchronize the models. So, any malfunction of the central server can bring down the entire network. Besides, training of some models involves the participation of confidential data so that the model cannot be published, and the central server is difficult to be fully trusted. Moreover, reliable and high-bandwidth communication links are required to establish between the servers and the clients for the transmission of potentially large amounts of data. Also, the bandwidth consumption required for model transmission is expensive. Therefore, how to reduce communication consumption while ensuring good training results is the key issue to be discussed. Furthermore, in practice, the amount of data owned by nodes participating in FL is different, which makes the enthusiasm of clients with more data to participate in FL very low. As a result, how to mobilize the enthusiasm of clients is a problem worth being considered.

Inspired by the existing distributed segmented gossip-based FL algorithm [9], we propose Clustered Hierarchical Distributed Federated Learning method to solve the problems mentioned above. Firstly, we group clients to accelerate convergence by communicating and aggregating models among different clusters. Secondly, in segmented gossip distributed FL [9], in each round, each client collects model segments and performs segmented model aggregation to update its own model, which makes the number of communication resources and computing resources consumed in each round very huge. To solve this, we add a hierarchical system and feedback mechanism. In a hierarchical system, we set up a leader-follower relationship to collect model segments and obtain the partial models. The feedback mechanism will help the dissemination of the global model to speed up the optimization of the local model in the next round. In this way, not only the accelerated convergence are retained but also the bandwidth resources advantages of segmented gossip, communication and computing resources are saved. In addition, the communication between clusters is realized through the communication between leaders of each round. After each leader obtains the partial model, the leaders will communicate with each other through gossip protocol to obtain a unified global model and realize the optimization function of the central server of the

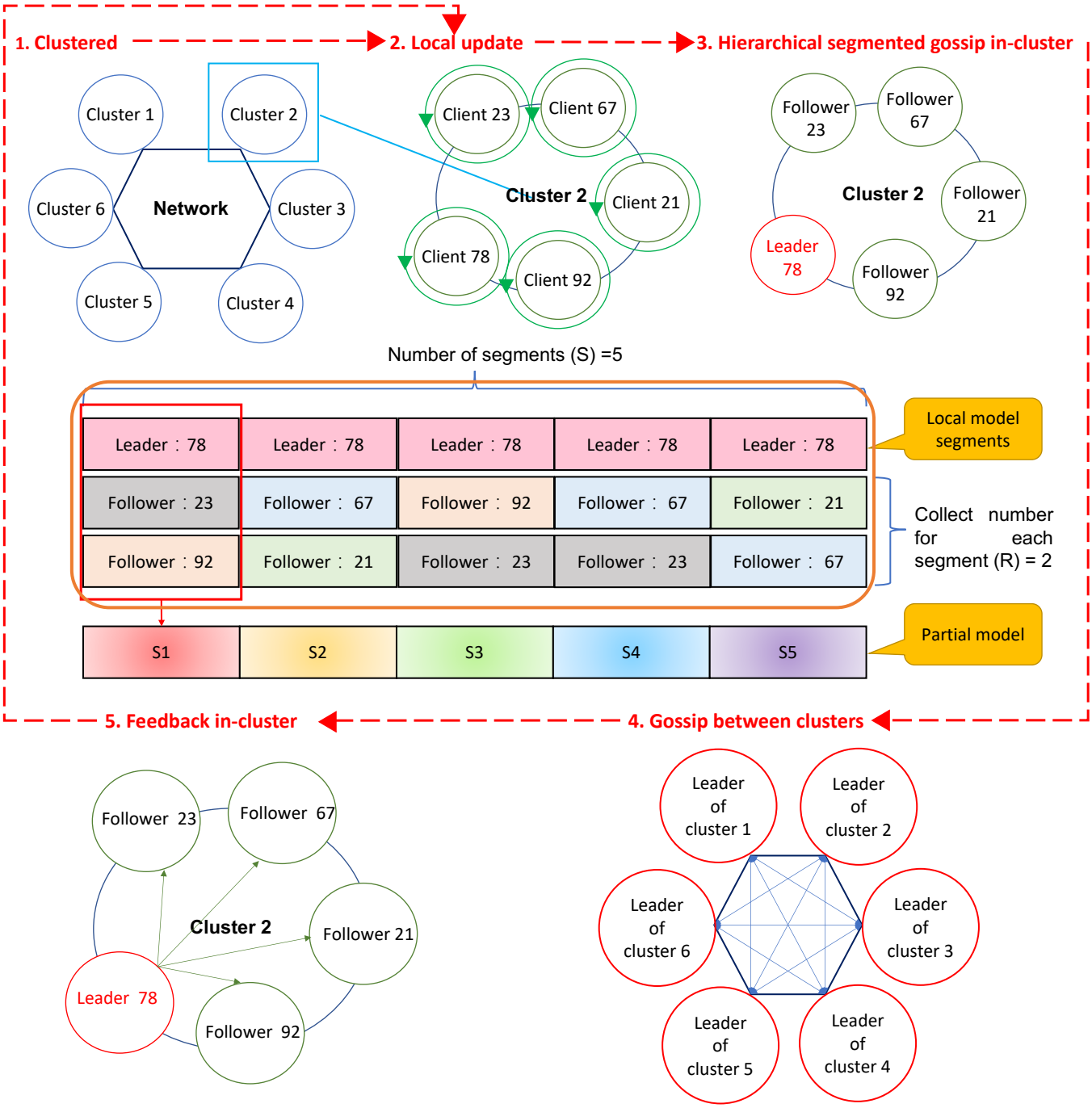

Fig. 1. Architecture of Clustered Hierarchical Distributed Federated Learning.

original centralized FL. At the same time, the high randomness of the leader also improves the privacy security of the system.

The rest of the paper is organized as follows. The related work is stated in Section II. The architecture and description of our algorithm are presented in Section III. In Section IV, simulation results and discussion is presented. The paper concludes in Section V.

II. Related Work

To avoid bottleneck and single point failure, decentralized architecture is a great solution. Patarasuk and Yuan [10] adopt an all-to-all scheme. In that system, each client broadcasts the local model updates to all other clients. Although it achieves the same synchronization effect as a centralized system, it consumes much more bandwidth resources between clients. Daily *et al.* [11] put forward a gossip-based model synchronization. The clients send local updates to only one

or a group of selected clients, which reduces the transmission cost. Thorsten Wittkopp and Alexander Acker [12] provide a multi-cluster decentralized FL method. In each cluster, models are trained on the same objective with their locally training data. Besides, they introduced the teacher-student concept for the training process. Every model can adopt the role of a student or teacher. Models adopt the role of teachers to generate the knowledge representations and scores, which indicates how well the model performs on this task. After communication between clusters and comparing the scores, the received knowledge representations are selectively accepted and adopted by models in the student role.

In order to reduce the communication costs, some studies focus on utilizing the characteristics of different topologies like ring, tree and graph topology. Some researchers reduce communication overhead by segmenting the model. Pappas *et al.* [13] introduced a decentralized FL framework, named Interplanetary Learning System (IPLS). They divide the model into layers and assign them to different clients. After local studying, clients who are in charge of the storage of the same partition will exchange the newly calculated values to calculate the new global parameters. Hu *et al.* [9] put forward segmented gossip FL. All models are divided into the same number of segments. In the aggregation stage, this method allows the client to pull different parts of the model parameters from different clients. Every client chooses a peer client for each segment and pulls the corresponding segment from it. After the client fetches all the model segments back, a new mixed model can be rebuilt from the segments. In this step, the total transmission size is equal to one model, which is the same as the gossip-based schemes, but the traffic is dissolved among not one but multiple links. Then, for each segment, every client aggregates all the corresponding parts with their own segment to obtain the aggregated segment. Then each client collects all the aggregated segments to rebuild the final aggregated model. With the carefully forming dynamical synchronization gossiping groups, the bandwidth can be fully used so that this method has good training convergence.

III. Algorithm Description

A. Clustered

The procedure of clustered hierarchical distributed federated learning is shown in Fig. 1. The first step of our method is clustered. A vital part of FL is to mobilize the enthusiasm of clients. To make the final training result better, it is necessary to attract more clients and contribute their models actively. However, since the amount of data of each client is uncertain in practice, if all agents are in the same status, the enthusiasm of agents with large amounts of data to participate in the system and contribute their own models will be reduced. The proper introduction of clustering algorithms can accelerate the learning progress and increase the enthusiasm of participating clients. So, the clients in our method are clustered according to the amount of local data. The clients with a similar data size are divided into one cluster so that the exchange of models for each client in the cluster will be relatively fair. In addition, the number of clients in each cluster is not less than 5 to ensure the diversity of local data in each cluster, which makes the partial model not overly biased.

Consider there are N agents in the distributed system and divided into m clusters without overlapping. The set of clients in each cluster is $C_i = \{c_{ij} | 1 \leq i \leq m, 1 \leq j \leq n_i,\ i, j \in N^*\}$, where N^* is the set of positive integer, n_i represents the number of clients in the i-th cluster. Different clusters may have a different number of clients.

B. Local Update

From Fig. 1, the second step of our method is the local update. At the beginning of the learning process, every client updates the model with their local dataset. In particular, they take the model results of the last iteration as the input model and update it by stochastic gradient descent (SGD) with the local dataset. In order to reduce the communication cost, there will be several SGD training rounds before the communication with other workers.

C. Hierarchical Segmented Gossip In-cluster

After the local update, there will be model communication within the cluster which adopts a hierarchical segmented gossip protocol. We introduce leader-follower relationships in each cluster for hierarchical management so that every client can play the role of leader or follower in each round. The main procedure of hierarchical segmented gossip is shown in Fig. 2.

The first step is leader selection. There is one selected leader in each cluster, and the rest of the clients automatically become followers. For example, Fig. 2 shows that in k-th cluster, there are five clients $C_k = \{c_{k1}, c_{k2}, c_{k3}, c_{k4}, c_{k5}\}$. After one of them becomes the leader represented by L, the remaining four clients become followers represented by $F1$ to $F4$. For example, $C_k = \{F_1, F_2, L, F_3, F_4\}$. Then, in the model segmentation step, all local models required by the leader in its cluster are divided into S_k segments without overlapping. Let P denote the local model parameters, which are the training results after the local update defined as:

$$P_{c_{kj}} = (P_{c_{kj}}(1), P_{c_{kj}}(2), P_{c_{kj}}(3), ..., P_{c_{kj}}(S_k)). \quad (1)$$

In Fig. 2, for example, $S_k = 5$, $P_{F2}(3)$ represents the third segmented local model of follower $F2$. In the participant selection step, leader chooses R different followers for each segment to upload their corresponding segment, where $R_k \leq n_k$. Also, leader records follower list for each segment. Let F_k represent the follower list of the k-th cluster.

$$F_k = (F_k(1), F_k(2), F_k(3), \ldots, F_k(S_k)). \quad (2)$$

In Fig. 2, $R_k = 2$, $F_k(1)$ the record of followers chosen by leader L to upload their first segment to the leader. The next step is the segment collection. In this step, after receiving the segment number S and the corresponding segment request from the leader, the follower will divide their local model into S segments according to the requirements and send the required model segments to the leader. After receiving all

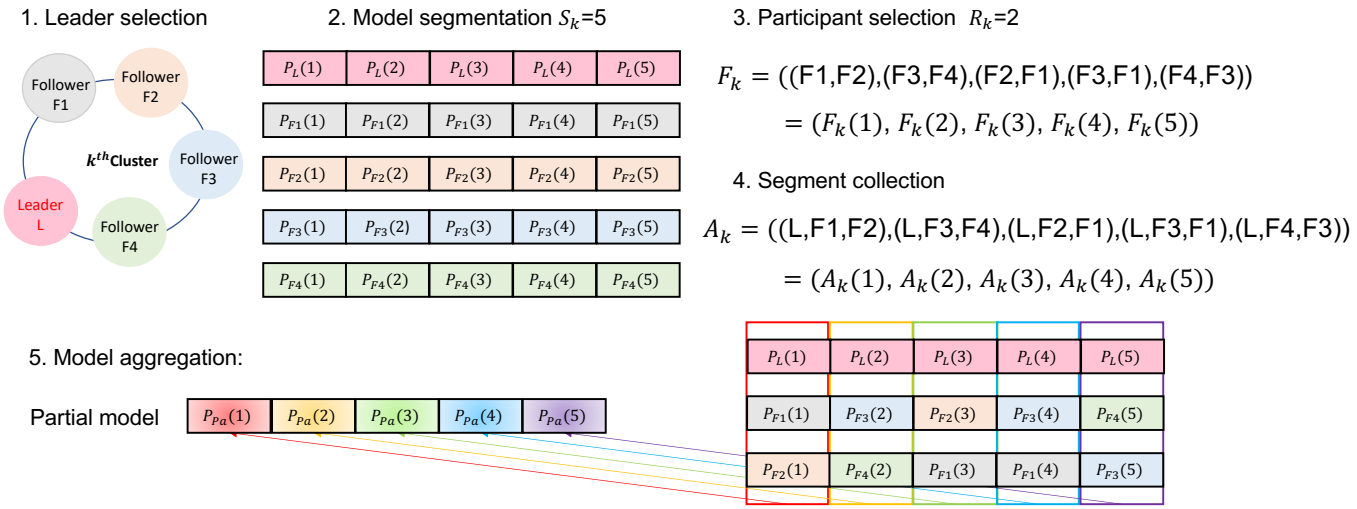

Fig. 2. Hierarchical Segmented Gossip In-cluster.

segments from followers, in the model aggregation step, the leader performs a weighted average of the segments at the same position to obtain the partial model segment. Let A represent each client participating in the model aggregation, which contains the list of followers and the leader. Specifically, in k-th cluster, for l-th segment ($l \leq S_k$), $A_k(l) = F_k(l) \cup L$. As for model aggregation, the weight of each segment W is proportional to the training data amount D of the client. For every client a belongs to $A(l)$, the weight of each segment is:

$$W_a = \frac{D_a}{\sum_{a \in A(l)} D_a}, \tag{3}$$

where D_a is the training data amount of client a. The l-th segment of the partial model segment is:

$$P_{pa}(l) = \sum_{a \in A(l)} W_a \times P_a(l). \tag{4}$$

Then, according to the original segment order, the leader will arrange the resulting partial model segments in order and join them to obtain the partial model:

$$P_{pa} = (P_{pa}(1), P_{pa}(2), ..., P_{pa}(S_k)). \tag{5}$$

D. Gossip Between Clusters

As shown in Fig. 1, after the leaders of all clusters generate their partial models, they will exchange partial models following the gossip protocol [14]. Then, the leaders perform a weighted average according to the percentage of the total number of data owned by the clients in each cluster to obtain a unified global model, which is:

$$W_{C_i} = \frac{\sum_{c_{ij} \in C_i} D_{c_{ij}}}{D_{all}}, \tag{6}$$

$$P_{gl} = \sum W_{C_i} \times P_{pa,i}, \tag{7}$$

where $1 \leq i \leq m$, $D_{c_{i,j}}$ is the training data amount of client, who is the j-th client in the i-th cluster, D_{all} is the total number of training data in the system, P_{gl} is the global model.

E. Feedback In-cluster

After getting a unified global model, we add a feedback mechanism. By recording the list of followers who already contributes to the leader with model fragment in the current round, the leader will feedback the global model to the follower. Then the model is updated by both the leaders and followers before the next round of model training. After model updating, the system gets back to the second step and starts the next iteration.

IV. Performance Evaluation

A. Setup

Data setting: We adopt the standard MNIST dataset[1] which consists of 60000 images for training and 10000 for validation. Each image is a 28×28 sized handwritten digits grayscale image from zero to nine.

We set the number of clients in this system to be 100. The data of each label is randomly divided into 20 groups of varying numbers without overlapping. Then 200 data groups are randomly assigned to 100 clients; each client has two groups of data as a local dataset so that the data for each client has only one or two types of labels. The reason why we distribute the data in this way is that although the independent and identically distributed (IID) sampling of the training data is essential to ensure that the stochastic gradient is an unbiased estimate of the full gradient [15]–[17], in practice, the data of each agent has different personal preferences. It is impossible that the local data on each edge device is always IID [18]. Non-IID data will be closer to the data distribution in practice.

Neural network setting: Our neural network includes 2 hidden layers with 100 units using ReLu activations (89610 parameters). The models are trained on each client by the SGD algorithm with the same hyper-parameters. The learning rate is 0.1, and the batch size is 32.

[1]http://yann.lecun.com/exdb/mnist/

B. Leader Selection Mode

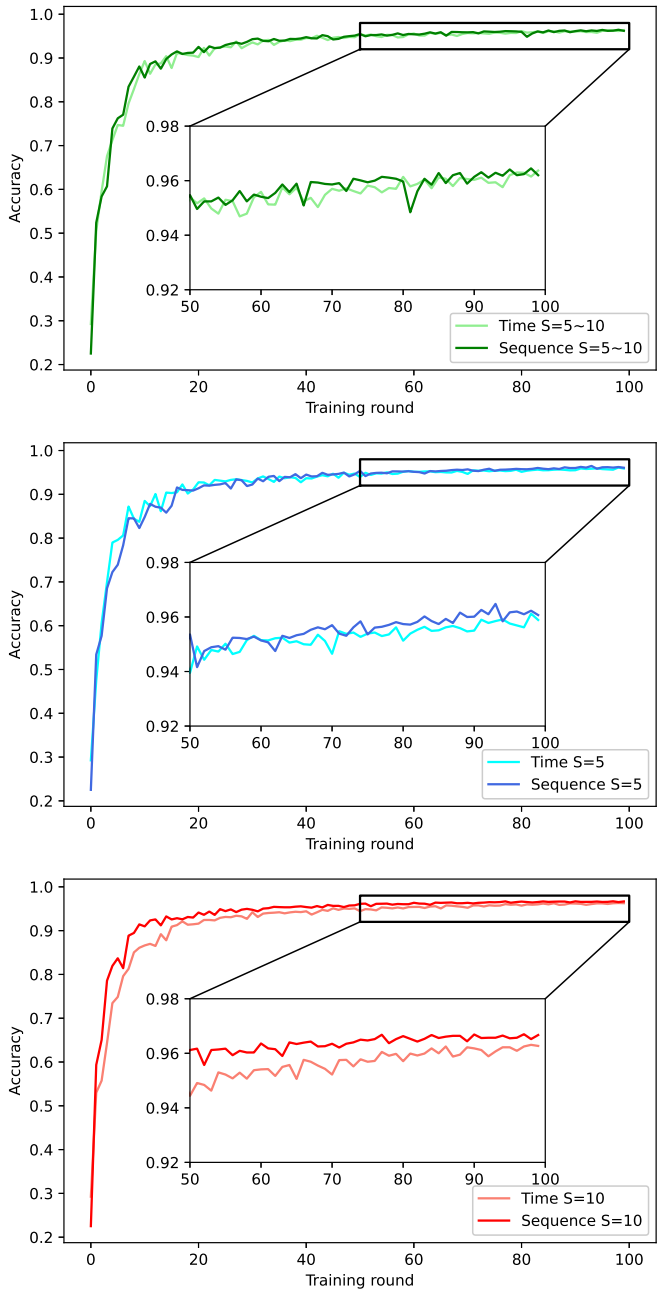

Fig. 3. Global model accuracy comparison under two leader selection mode.

Since the leader model accounts for the largest proportion of the partial model, it is important to choose an appropriate leader selection mode selecting better nodes as the leaders to accelerate the process. We define the process from step B to step E as a training round. Therefore, we compare the accuracy of the global model in each training round and the convergence speed of the system under two different leader selection methods.

In Fig. 3, we compare the two ways of leader selection. One is the client takes turns to be the leader in the cluster. The other is the client who spends the less round to complete the first round of self-learning in each cluster to become the leader. There are three simulations with different number of segments S in each cluster in each round: $S \in [5, 6, 7, 8, 9, 10]$, $S = 5$, $S = 10$. To visualize the results better, we zoom in on the performance from round 50 to 100. The dark-coloured lines in each graph are the system performance of the alternate selection method, and the light-coloured lines are the system performance of the leader selection method according to the global training time. From Fig. 3, it can be seen that the performance of both the convergence rates and model accuracy of the dark-coloured lines are better than the light-coloured lines. The reason is that the client with the shortest time to complete a round of self-learning is selected as the leader, and the probability of each client becoming the leader is inversely proportional to the number of data possessed and proportional to the calculated ability. Therefore, a client with a smaller number of data or better computing capability is more likely to become the leader and may always be the leader in its cluster, which is not conducive to model optimization. To conclude, the alternate selection method is better than the learning-time-based leader selection mode.

C. Performance Comparison

To verify the effectiveness of our algorithm, we compare our algorithm with the algorithm in [9] in terms of learning performance and communication overhead.

Learning performance: We present the training process over epochs to evaluate the convergence speed performance of our algorithm and algorithm in [9]. As illustrated in Fig. 4, the global model of our algorithm requires less training epochs to reach the same accuracy.

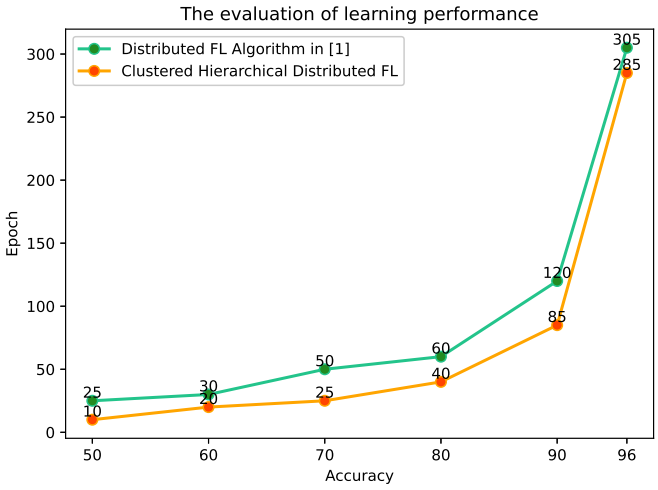

Fig. 4. The evaluation of learning performance.

According to Fig. 4, the system converges after the accuracy rate reaches 90%. Our algorithm showing in orange line converges after 85 training epochs, while algorithm in [9] showing in green line converges after 120 epochs. To conclude, our algorithm converges after only 71% percent of the learning epochs compared with the algorithm in [9].

Communication cost: Whether it is centralized FL or decentralized FL, it is inseparable from the model exchange between clients, which inevitably has a huge amount of communication overhead. In the distributed FL in [9], the communication cost of each round is the sum of the model segments collected by each node. The communication cost of our algorithm is the sum of the cost of in-cluster segmented gossip communication and gossip communication between clusters. Let Z represent the size of the model, and V represent the number of followers who participate in this round. The communication cost Q in each round is:

$$Q = \sum_{i=1}^{m} Z \times (R_i + V_i). \tag{8}$$

Compared with the algorithm in [9], the clustered hierarchical distributed FL proposed in this paper can save the consumption of communication resources to a greater extent.

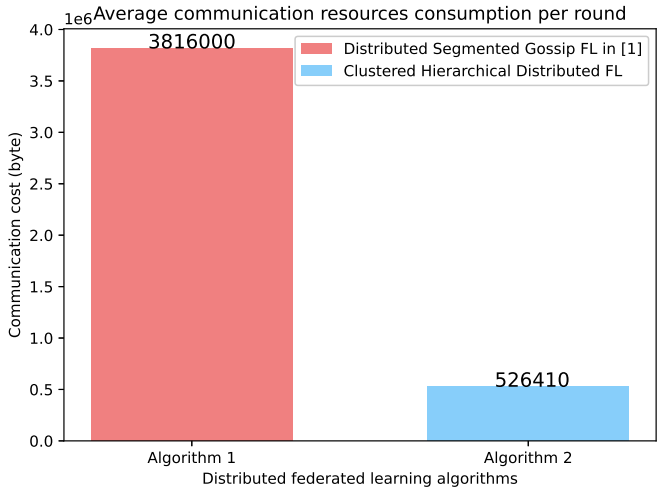

Fig. 5. Communication cost comparison.

Figure 5 shows the average communication cost of two algorithms in 100 rounds. The column of Algorithm 1 shows the number of resources consumed by the distributed federated learning algorithm based on segmented gossip in [9]. When every local model is divided into 5 segments, and all clients fetch 3 mixed model for aggregation, each round consumes a huge amount of communication, which is 3.8×10^6 bytes. The column in blue shows the communication overhead of the clustered hierarchical distributed federated learning algorithm proposed in this paper, with $S = 5$, $R_i = \frac{n_i}{2}$. Our algorithm only costs 5.3×10^5 bytes per round, which can save 86% of the communication consumption compared with the algorithm of [9].

V. Conclusion

This paper proposes the clustered hierarchical distributed FL method, which improves the fairness of model exchange and mobilizes the enthusiasm of client participation by clustering the client with a similar amount of data. The leader-follower relationship and feedback mechanism are introduced through the hierarchical mechanism to make the communication resources usage more efficient. The distributed architecture solves the issue of single-point obstacles and privacy leakage of the central server. The segmented gossip model aggregation inside the cluster makes full use of bandwidth resources and accelerates convergence. Through the use of gossip between clusters, model optimization is accelerated. The simulation results show that the clustered hierarchical distributed FL method has a better performance compared with that in [9].

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 436–444, 2015.

[2] A. Karpathy and L. Fei-Fei, "Deep visual-semantic alignments for generating image descriptions," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. 3128–3137, 2015.

[3] S. Bosse, D. Maniry, K.-R. Müller, T. Wiegand, and W. Samek, "Deep neural networks for no-reference and full-reference image quality assessment," *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 206–219, 2018.

[4] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. 1725–1732, 2014.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, no. 3104–3112, 2014.

[6] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *ITU Journal: ICT Discoveries - Special Issue 1 - The Impact of Artificial Intelligence (AI) on Communication Networks and Services*, vol. 1, no. 39–48, 2018.

[7] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3400–3413, 2020.

[8] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5476–5497, 2021.

[9] C. Hu, J. Jiang, and Z. Wang, "Decentralized federated learning: A segmented gossip approach," *CoRR*, vol. abs/1908.07782, 2019. [Online]. Available: http://arxiv.org/abs/1908.07782

[10] P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," *Journal of Parallel and Distributed Computing*, vol. 69, no. 117–124, 2009.

[11] J. Daily, A. Vishnu, C. Siegel, T. Warfel, and V. Amatya, "Gossipgrad: Scalable deep learning using gossip communication based asynchronous gradient descent," *CoRR*, vol. abs/1803.05880, 2018. [Online]. Available: http://arxiv.org/abs/1803.05880

[12] T. Wittkopp and A. Acker, "Decentralized federated learning preserves model and data privacy," *CoRR*, vol. abs/2102.00880, 2021. [Online]. Available: https://arxiv.org/abs/2102.00880

[13] C. Pappas, D. Chatzopoulos, S. Lalis, and M. Vavalis, "Ipls: A framework for decentralized federated learning," in *2021 IFIP Networking Conference (IFIP Networking)*, 2021, pp. 1–6.

[14] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Gossip algorithms: design, analysis and applications," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3, 2005, pp. 1653–1664 vol. 3.

[15] L. Bottou, "Large-scale machine learning with stochastic gradient descent," *Proceedings of COMPSTAT'2010*, pp. 177–186, 2010.

[16] O. Shamir, "Making gradient descent optimal for strongly convex stochastic optimization," *CoRR*, vol. abs/1109.5647, 2011. [Online]. Available: http://arxiv.org/abs/1109.5647

[17] S. Ghadimi and G. Lan. (2013) Stochastic first- and zeroth-order methods for nonconvex stochastic programming. [Online]. Available: https://arxiv.org/abs/1309.5549

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