Towards Asynchronous Peer-to-Peer Federated Learning for Heterogeneous Systems

Christos Sad
Aristotle University of Thessaloniki
Greece
csant@auth.gr

George Retsinas
National Technical University of Athens
Greece
gretsinas@central.ntua.gr

Dimitrios Soudris
National Technical University of Athens
Greece
dsoudris@microlab.ntua.gr

Kostas Siozios
Aristotle University of Thessaloniki
Greece
ksiop@auth.gr

Dimosthenis Masouros
National Technical University of Athens
Greece
dmasouros@microlab.ntua.gr

Abstract

Federated Learning (FL) enables collaborative model training across distributed, privacy-sensitive data sources. Traditional FL follows a centralized client-server architecture, relying on synchronized updates and uniform participation. However, real-world deployments face challenges such as client heterogeneity, stragglers, non-independent data distributions, and single points of failure due to server centralization. To address these limitations, we propose an asynchronous Peer-to-Peer FL scheme that enhances learning efficiency in heterogeneous environments. Our method employs a gradient-aware aggregation algorithm with a progress-based adaptive fusion weight, mitigating the impact of resource disparities among clients. Experimental results on CIFAR-10/100 datasets indicate that our scheme achieves $4.8 - 16.3\%$ and $10.9 - 37.7\%$ higher accuracy compared to FedAVG and FedSGD, considering constrained total number of exchanged updates among clients. Furthermore, it effectively handles client heterogeneity through its dynamic fusion weight adjustment.

Keywords: Federated Learning, Asynchronous Communication, Peer to Peer, Gradients, Heterogeneous Systems

ACM Reference Format:
Christos Sad, George Retsinas, Dimitrios Soudris, Kostas Siozios, and Dimosthenis Masouros. 2025. Towards Asynchronous Peer-to-Peer Federated Learning for Heterogeneous Systems. In *The 5th Workshop on Machine Learning and Systems (EuroMLSys '25), March 30–April 3, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3721146.3721952

1 Introduction

In recent years, Federated Learning(FL) [1] has emerged as a prominent distributed machine learning paradigm, enabling the training of models across decentralized data sources. By ensuring that data remain on local devices, FL addresses critical privacy and security concerns, particularly in domains such as healthcare and finance, where direct data sharing is restricted by strict regulation constraints[2].

Traditional FL employs a client-server architecture, where a central server coordinates learning across multiple clients [3]. The most widely used FL algorithms are Federated Stochastic Gradient Descent (FedSGD) and Federated Averaging (FedAVG)[1]. In FedSGD, clients compute gradients on their local data and send them to the server for aggregation, whereas in FedAVG, clients train locally for multiple iterations before sharing model updates, reducing communication overhead. While these methods are effective, they rely on a centralized server, synchronous updates, and uniform participation, which can be problematic in real-world settings that often involve client device heterogeneity, dynamic network conditions, and non-independent and identically distributed (non-i.i.d.) datasets. These non-ideal characteristics introduce significant challenges, including stragglers – where slower or low-bandwidth devices delay the entire training [4, 5] – as well as convergence instability and bias due to non-uniform data distributions [6]. On top of that, centralized server aggregation can act as a single point of failure, making the system vulnerable to collapsion.

To mitigate these limitations, various solutions have been proposed in the literature. Some efforts focus on addressing device heterogeneity while retaining the centralized nature of FL, by selectively deciding each clients' contribution on every aggregation round [7], or by incorporating techniques such as gradient compression and quantization to reduce the computational and communication burden of stragglers [8–18]. Moreover, asynchronous FL approaches have been introduced to improve scalability by allowing clients to update the global model at different intervals, rather than synchronizing updates in fixed rounds [19, 20]. To tackle the centralized nature of traditional FL, decentralized and peer-to-peer (P2P) FL approaches have been proposed. In decentralized FL [21–24], clients form connected network graphs, exchanging updates without a central server, gradually converging towards a global consensus. Yet, the effectiveness of decentralized FL depends on network topology and communication efficiency, and fully decentralized methods may suffer from slow convergence. P2P solutions [25] further extend decentralization by allowing direct client-to-client collaboration, where clients exchange and aggregate models with their neighbors.

Intuitively, P2P FL is a highly promising paradigm as it eliminates the reliance on a central server, enabling scalable and resilient learning. From a machine learning perspective, P2P introduces stochasticity in model aggregation, by enabling clients to aggregate updates in an iterative, diffusion-like manner, leading to robust, well-regularized convergence. From a systems perspective, P2P FL avoids single points of failure while also simplifying client coordination and reducing communication costs by distributing aggregation and enabling decentralized peer interactions.

In this paper, we propose a novel asynchronous P2P FL scheme. It incorporates an aggregation algorithm that exploits gradient information and employs a progress-based adaptive fusion weight to effectively mitigate the impact of heterogeneous client resources, enhancing overall performance. Our scheme introduces an aggregation decision buffer and maker to regulate client interactions and address asynchronous training in P2P systems. It leverages inter-client gradients over simple averaging and employs progress-based adaptive fusion for improved performance in heterogeneous environments. Experimental results show that we achieve $4.8\% - 16.3\%$ and $10.9\% - 37.7\%$ higher accuracy than FedAVG and FedSGD on CIFAR-10/100, under constrained communications. Additionally, our scheme reduces the impact of stragglers on other clients, cutting relevant accuracy drops by 18%.

2 Background on Federated Learning

Federated Learning (FL) is a machine learning paradigm that enables the training of high-quality models across distributed clients or devices while preserving data privacy [3].

2.1 Traditional Federated Learning

Traditional FL follows a training approach involving clients and a server, as illustrated in Figure 1a. Clients perform local training and periodically send updates to a global server after a predefined number of local iterations, referred to as a *local round*. The server collects these updates, verifies compliance with aggregation constraints (e.g., minimum client participation), and performs a *global model update* using an *aggregation algorithm*. The updated model is then redistributed to the clients for the next training round. In vanilla FL schemes [1], such as FedAvg and FedSGD, aggregation is performed by computing a weighted average of the model parameters (or gradients) received from participating clients.

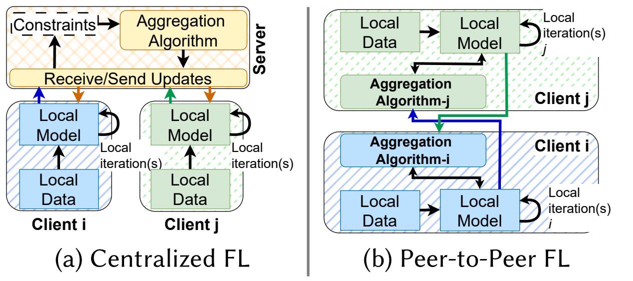

Figure 1. Overview of (a) Centralized and (b) Peer-to-Peer Federated Learning schemes.

Despite its advantages, this approach has several limitations. First, its *centralized architecture* creates a **single point of failure**, i.e., if the server becomes unavailable, the entire learning process is disrupted [26]. Second, vanilla FL operates in a *synchronous* manner, requiring the server to wait for updates from all selected clients before aggregation. This makes FL susceptible to **stragglers**, i.e., clients with limited computational or network resources, causing delays in the training process [26]. Last, it also struggles with *statistical heterogeneity*, particularly when client datasets are not independently and identically distributed (non-iid). In such cases, the naive aggregation of client updates does not consider skewed or imbalanced datasets, leading to **biased global updates** and **slower convergence** [27].

2.2 Peer-to-peer Federated Learning

Peer-to-peer (P2P) FL eliminates the need for a central server by allowing clients to collaborate directly, as illustrated in Figure 1b. Similar to traditional FL, the training process consists of local model updates, where each client trains on its private dataset for an arbitrary number of local iterations, followed by *peer-to-peer synchronization*, where models are shared and aggregated among connected peers. This decentralized approach tackles both device heterogeneity and single points of failure, by allowing clients to update their models at their own pace [4, 26]. Moreover, the aggregation protocol can vary per client, enabling personalized update strategies based on data similarity or statistical divergence [25].

2.2.1 P2P challenges.

While P2P federated learning mitigates traditional FL challenges, it introduces new complexities, such as determining which (and how many) clients should participate in each synchronization round and designing effective aggregation protocols. Novel communication topologies have been proposed to reduce communications while preserving client accuracy [23], facing challenges arising from dataset heterogeneity and varying client capabilities. Additionally, similarity-based neighbor matching mechanisms improve client pairing within peer neighborhoods, considering non-i.i.d. data distribution but overlooking the asynchronous nature of real-world applications [25]. In some cases, commitment mechanisms with an elected committee system manage the FL process [24], yet client heterogeneity and asynchronous FL challenges remain unaddressed.

3 Proposed Asynchronous P2P-FL Scheme

To address the aforementioned challenges, we propose a novel P2P FL scheme. Unlike traditional synchronous approaches requiring global synchronization, our scheme operates asynchronously, allowing clients to update and communicate at their own pace, reducing the impact of stragglers in heterogeneous FL environments. It also addresses asynchronous convergence progression (§2.2.1) and non-iid data distributions with an advanced aggregation algorithm that uses inter-client gradients and an adaptive fusion weight to normalize each client's impact, instead of relying on simple averaging.

3.1 Proposed FL Scheme Processes

Figure 2 depicts our methodology, comprising the aggregation algorithm (❽) and the communication protocol/topology. Each client incorporates an Aggregation Decision Maker (ADM) (❻) to determine whether to initiate communication and an Update Transmitter (❼) to prepare and send update messages. Additionally, it includes an aggregation algorithm (❽); in our approach, all clients use the same aggregation algorithm (§3.5). Finally, the scheme features a common Aggregation Decision Buffer (❷), that stores all pending communication requests and a Buffer Explorer (❶), responsible for searching for pending clients within the buffer. The common Aggregation decision buffer is stored distributed in a random subset of clients for resilient execution (§3.3).

3.2 Proposed FL scheme life cycle

Each client trains locally (❹) on its dataset (❸) for a set number of iterations. Upon completion, the ADM (❻) is triggered (⓬) to decide on aggregation. This decision is sent (❾) to the Buffer Explorer (❶), which accesses the Aggregation Decision Buffer (❷) containing pending aggregation requests. If a client skips aggregation, it continues local training. Otherwise, the Buffer Explorer searches for an available peer. If found, the buffer provides (❿) both clients with peer information (❾), including identifiers, which are then transferred (⓯) to the Update Transmitter (❼). The two clients exchange (⓫) model weights, prepared (⓭) into an update message containing metadata such as iteration count. Finally, each client applies the Aggregation Algorithm (❽) to compute the inter-client gradient (§3.5) and updates (⓮) its model

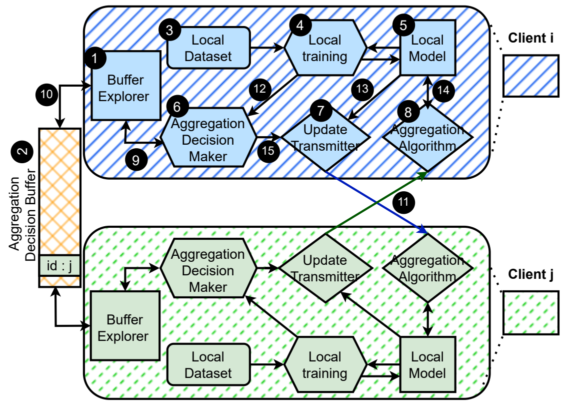

Figure 2. Proposed Asynchronous Peer-to-Peer Federated Learning Scheme

(❺). If no peer is available, the client's request remains in the buffer until an aggregation request arrives.

3.3 Communication protocol

Our communication protocol combines asynchronous FL with P2P communication to minimize overhead. In this asynchronous P2P scheme, each aggregation occurs between two clients, significantly reducing communication costs. Choosing more than two clients per communication round would require a mesh topology, leading to exponential communication overhead, being impractical for large deployments.

Aggregation Decision Buffer and Explorer: a common Aggregation Decision Buffer (❷) is introduced solely to maintain the latest client pending for communication. It interacts with each client's Buffer Explorer to either store a communication request or return a pending request to another initiating client. It achieves this by maintaining the IDs of pending clients, if any. This design preserves the **asynchronous** nature of our FL scheme, as clients can access information about available peers and establish P2P communication independently. The Aggregation Decision Buffer is stored inside all clients to ensure resilient execution in case of failures. For the synchronization of the Aggregation Decision Buffer, whenever a client needs to store a pending request, it sends a message containing only its identifier (clients id) to all other clients. Conversely, when a client initiates communication with a pending client, it notifies all other clients through an update that includes the negation of the pending client's identifier (– pending client's id). This ensures that all clients remove the pending client's identifier from their Aggregation Decision Buffer. It is important to note that, in the Aggregation Decision Buffer, typically at most one client is pending for communication at a time. This is because the Buffer Explorer first searches for pending requests before storing any new ones. Thus, the buffer contains minimal information and can be accommodated by all clients regardless of their available resources. Each client has also, its own Buffer Explorer (❶) that searches the buffer.

3.4 Aggregation Decision Maker (ADM)

The proposed framework also relies on the Aggregation Decision Maker (ADM) (❻), a component that each client must include to participate in our proposed FL scheme. The ADM contains the mechanism responsible for deciding whether the client should initiate communication. Each client can employ its own ADM, which may range from a simple (e.g., random decision mechanism) to a more complex (e.g., decisions on prior actions or current states) implementation.

In our implementation, we use a simple ADM based on Bernouli distribution. Specifically, the procedure follows the conventional approach described in § 2.1, where each client performs its local training iterations. At the end of each local round, the ADM determines whether to initiate communication with a constant probability, ensuring equal participation in communications across clients. Equation 1 describes the probability that the ADM decides for a client to initiate communication, where $Pr\{X = communicate\}$ is the probability to communicate and $const$ is a constant number. In our experiments, we set the parameter const to be equal to the half of total number of participating clients.

$$Pr\{X = communicate\} = \frac{100}{const}\% \tag{1}$$

At the end of each local round, the probability that a client initiates communication is $\frac{2}{K} \times 100\%$, where K is the total number of clients. Consequently, for K clients with similar capabilities starting training simultaneously, an average of $K \times \frac{2}{K} = 2$ clients initiate communication per local round, forming **the communication pair for the round**

3.5 Aggregation Algorithm

Our proposed aggregation algorithm (❽) is built on two key concepts: the influence of peers' model differences on the update and the dynamic adjustment of a fusion weight. The peers' gradient is a vector that captures the difference between the model weights of two clients, analogous to a gradient update in optimization. In essence, this gradient indicates the direction in which we should adjust the model weights to make one more like the other. However, rather than completely transforming one model into the other, we take a step in that direction to begin incorporating its information. The key issue here is determining precisely how large that step should be.

Fusion protocol: Our fusion protocol involves each client computing a gradient-like vector that represents the difference between its own weights and those of its peer during communication. The client then updates its local weights by adjusting them in the direction of the peer's gradient, normalized by a weight referred to as fusion weight (wf). This weight wf_i takes values from 0 (completely disregard the second model) to 1 (completely disregard the current client's model).

$$\begin{aligned} g_{ij} &= W_i - W_j \\ W_{i,agg} &= W_{ij} = W_i - wf_i \times g_{ij} \end{aligned} \tag{2}$$

Equations 2 present the equations for the fusion protocol. W_i and W_j represent the weights of the peers before aggregation, $W_{i,\text{agg}}$ denotes the updated weights of client i after aggregation, g_{ij} is the gradient computed by client i based on the weights of client j, and wf_i is the fusion weight . It is evident that the updated weights of client$_i$ are significantly influenced by the peers' gradient g_{ij}, while the parameter wf_i plays a crucial role in modulating the overall process.

Dynamic adjustment of fusion weight (wf) : To ensure client independence in asynchronous FL, we design our aggregation algorithm to accommodate heterogeneous clients with varying computational and communication capabilities, which may cause latency during local training or model updates. Additionally, some clients may communicate more frequently. To address these challenges, we dynamically adjust the fusion weight based on the training progress of each peer. Each client includes its progress in the update message. The peer receiving this update message incorporates the client's progress when computing its own updated weights. Thus, clients with low progress exert a minimal influence on clients with high progress, preventing the degradation of their weights, while clients with high progress can provide valuable updates to clients with lower progress.

$$\begin{aligned} p_i &= \frac{current\_iteration_i}{target\_iterations_i}, wf_i = wf_0 \times \frac{p_j}{p_i + p_j} \\ p_j &= \frac{current\_iteration_j}{target\_iterations_j}, wf_j = wf_0 \times \frac{p_i}{p_i + p_j} \end{aligned} \tag{3}$$

Equations 3 compute the progress p_i and p_j for client$_i$ and client$_j$ as the ratio of the current iteration to the total target iterations. They also describe the fusion weights wf_i and wf_j used by clients i and j, respectively, with wf_0 representing the initial fusion weight. The fusion weights of the two clients are **complementary** and sum to wf_0. When both clients have similar progress, the fusion weight is approximately half of wf_0. It decreases toward zero when peer client has minimal progress and approaches wf_0 when the peer client's progress is much higher. This allows the peer client to influence aggregation based on its progress. If more than one peer (e.g., K) communicates simultaneously with the client, the fusion weight for each peer(wf_{ij}) is computed based on the sum of the progress of all peers ($\Sigma_{k=1}^{k=K} p_k$) and the client's progress(p_i). The updated weights are then computed by combining all the peers' inter-client gradients(g_{ij}), as described in equation 4.

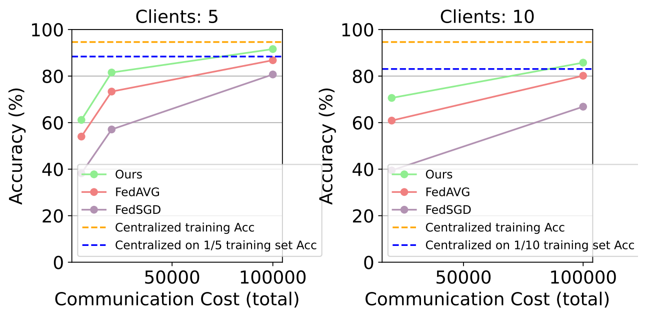

Figure 3. Pareto optimal analysis of accuracy and communication cost

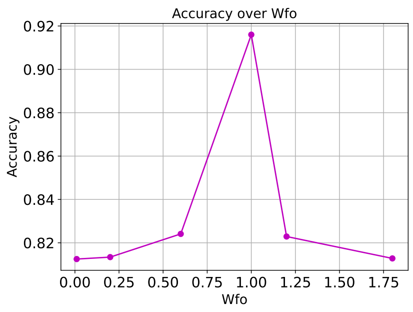

Figure 4. Analysis of accuracy over wf_0 values

$$wf_{ij} = wf_0 \times \frac{p_j}{p_i + \Sigma_{k=1}^{k=K} p_k}$$
$$W_{i,agg} = W_i - \Sigma_{j=1}^{j=K} wf_{ij} \times g_{ij} \tag{4}$$

4 Evaluation

We evaluate our methodology on widely used FL settings with CIFAR-10 and CIFAR-100 [28], using the ResNet-18 architecture [29]. The dataset is partitioned into the standard training, testing, and validation subsets, with the training set evenly divided into K partitions, where K is the number of clients. Accuracy is computed on a common, unseen testing set for consistency across clients. We compare our algorithm with FedAVG and FedSGD [1].

4.1 Parameter fine-tuning

Local training configuration : For local training, we use the SGD optimizer with momentum = 0.9, weight_decay = 5×10^{-4}, and an initial learning rate of 0.01 with a MultiStepLR scheduler. For CIFAR-10, milestones occur at 50% and 75% of iterations, with $\gamma = 0.1$; for CIFAR-100, milestones are at 30%, 60%, and 80%, with $\gamma = 0.2$. A batch size of 32 is used for all three FL algorithms.

Federated Learning parameters tuning : We define the target total communication cost via Pareto-optimal analysis and parameter tuning for each algorithm on CIFAR-10. The results are shown in Figure 3, where total communication cost represents the number of update messages exchanged during federated training. Additionally, we perform centralized training on both the total and partitioned datasets, using $\frac{1}{5}$ and $\frac{1}{10}$ of the dataset for 5 and 10 clients, respectively. Average accuracy across splits is calculated, simulating independent client training without inter-client communication.

The analysis shows that higher communication costs enhance accuracy by enabling more frequent updates and improved knowledge sharing. A communication cost limit of $100,000$ is selected, where our scheme outperformed centralized training on split datasets reaching 91.6% compared to 88.23% of centralized training on $\frac{1}{5}$ split and 85.75% compared to 83.45% of $\frac{1}{10}$ split. Notably, FedAVG (86.8% and 80.14%) and FedSGD (80.69% and 66.84%) achieve lower accuracy and do not surpass the centralized training results for corresponding splits in the tested range of total communications($\leq 100,000$).

In our approach, the total number of clients that communicate at the end of each round is $K \times \frac{2}{K} = 2$ (§3.4), with P2P (direct) communication, resulting in $2 \times 1 = 2$ exchanged updates per round. In contrast, FedAVG requires all K clients to exchange updates with the server at each round, and similarly in FedSGD. For both, the exchanged updates per round are $2 \times K$. The number of rounds is given by $\frac{total\_iterations}{local\_iterations}$.

Centralized training consists of 250,000 total iterations, a configuration replicated in both our approach and FedAVG. To control the communication cost, the number of local iterations per round was adjusted for both methods. In contrast, FedSGD, with fixed local iterations of 1 [1], regulates communication cost by adjusting total iterations. To meet the $100,000$ communication constraint, we set 5 and 25 local iterations for our approach and FedAVG with $K = 5$ clients, and 5 and 50 for $K = 10$. For FedSGD, we use $10,000$ and $5,000$ total iterations for 5 and 10 clients, respectively.

We also, tested several initial values for the fusion weight (wf_0), ranging from 0 (no peer contribution) to 2 (total model transformation when $p_i = p_j$), including 1 (averaging if $p_i = p_j$). Figure 4 shows experiments on CIFAR10 with $K = 5$ clients for different wf_0 values. The optimal value is $wf_0 = 1$, as expected with i.i.d. distributions, and was used in our experiments. For more complex distributions, this parameter can be fine-tuned through Pareto-optimal analysis.

4.2 Comparison with traditional FL approaches

In this section, we compare our scheme with FedAVG and FedSGD. To ensure fairness, identical configurations are used for local training across all algorithms. A communication cost limit of $100,000$ is imposed for all experiments. Additionally, we use **similar clients that started training simultaneously**, though our scheme is capable of operating in heterogeneous conditions.

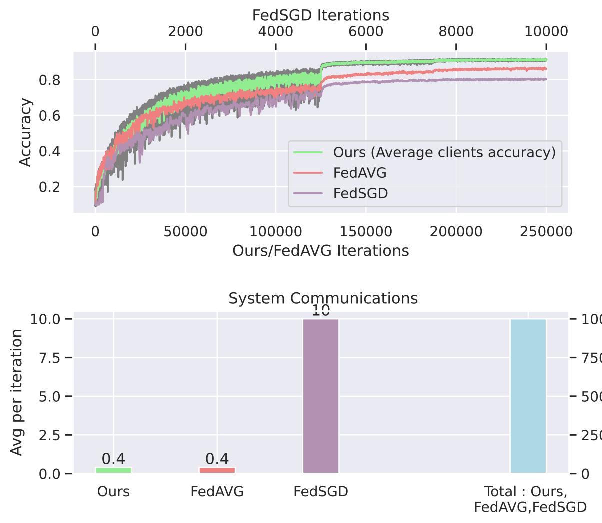

Figure 5. Accuracy and Communication cost : 5 clients - Cifar10.

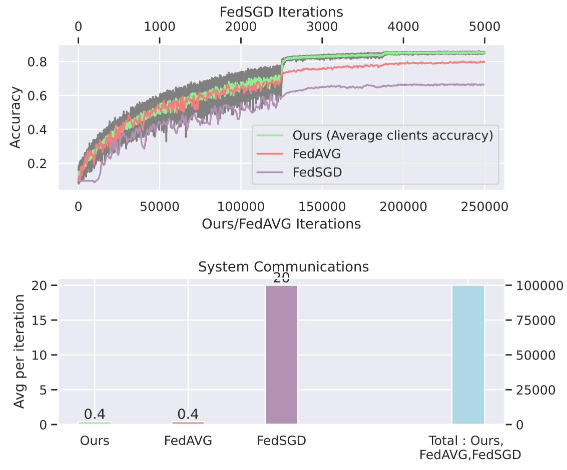

Figure 6. Accuracy and Communication cost : 10 clients - Cifar10.

Figures 5 and 6 show experimental results on CIFAR-10 for 5 and 10 clients, respectively. In both cases, ours average(green line) and individual client accuracies (gray lines) outperform FedAVG(red line) and FedSGD(purple line). In the 5-client scenario, our scheme achieves 91.6%, surpassing FedAVG (86.8%) and FedSGD (80.7%). In the 10-client scenario, ours reaches 85.8%, while FedAVG and FedSGD score 80.1% and 66.8%, respectively. These results highlight our scheme's superior communication efficiency.

At the bottom of each figure, a bar plot shows communications for each method. While total communications are equal across methods (100, 000), the average communications

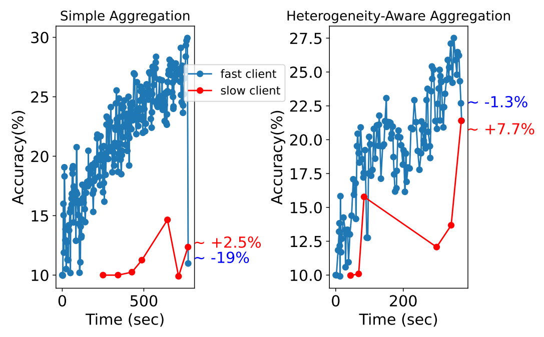

Figure 7. Learning curves of heterogeneous clients

per iteration (ACPI), calculated as $\frac{total\_communications}{total\_iterations}$, is the same for our approach and FedAVG (0.2 and 0.4 for 5/10 clients) but differs for FedSGD (10, 20 respectively).

Table 1. Comparison of Federated Learning Approaches

Dataset	Clients	Com. Cost	Ours(%)	FedAVG(%)	FedSGD(%)
Cifar10	5	100, 000	**91.6**	86.8	80.69
Cifar10	5	20, 000	**81.53**	73.34	57.06
Cifar10	5	5, 000	**61.15**	54	38.2
Cifar10	10	100, 000	**85.76**	80.14	66.84
Cifar10	10	20, 000	**70.6**	60.87	39.45
Cifar100	5	100, 000	**68.05**	58.05	41.53
Cifar100	5	20, 000	**45.4**	30.88	*15.6*
Cifar100	10	100, 000	**57.05**	40.77	*19.34*
Cifar100	10	20, 000	**28.96**	*15.9*	*7.3*

Table 1 presents results for both Cifar10/100 with 5/10 clients across communication levels from 100,000 (**extensive communication**) to 5,000 (**limited**). Ours consistently outperforms others, achieving 4.8 – 16.3% higher accuracy than FedAVG and 10.9 – 37.7% higher than FedSGD. As communication constraints tighten, only ours mitigates accuracy drops by distributing communication more efficiently with the ADM ❻. On CIFAR-100, FedAVG and FedSGD sometimes drop to ≤ 20% accuracy. While they perform well on CIFAR-10/100, constrained communications hinder knowledge sharing. With an **unlimited communication budget**, these algorithms could achieve higher accuracy.

4.3 Heterogeneous Clients Environment

Finally, we conduct a controlled experiment with two clients: a *fast client* which performs frequent local updates due to higher computational power and a *slow client*, which updates less frequently and remains in an earlier convergence phase. We implement the experiment on our algorithm **with a fixed value (1) on the fusion weight** wf, reffed as **naive** aggregation. Also, the experiment is conducted on our proposed algorithm **with the progress based adaptive fusion weight**. Figure 7 presents the accuracy over time for both clients. On the left plot, naive aggregation causes the slow client's lower accuracy to negatively impact the fast client, resulting in an accuracy drop of nearly 19%, while the slow client improves by only 2.5%. However, employing our proposed heterogeneity-aware aggregation strategy (§ 3.5) significantly mitigates this issue (right plot), reducing the fast client's accuracy loss to only 1.3%, while the slow client benefits from a 7.7% improvement.

5 Conclusion

We introduce an asynchronous P2P FL scheme with a novel aggregation protocol. This protocol leverages inter-client gradients and dynamically adjusts a fusion weight based on client progress, addressing communication and computation heterogeneity. Experiments on CIFAR-10/100 show that our scheme outperforms FedAvg and FedSGD in accuracy while effectively handles heterogeneous clients.

Acknowledgements

The authors would like to thank the National Infrastructure for Research and Technology Network (GRNET) for funding the cloud infrastructure used to evaluate the results of this paper. This work has been partially funded by the PRIVATEER project. PRIVATEER has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101096110. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or SNS JU. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[2] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *arXiv preprint arXiv:1905.06731*, 2019.

[3] P. Qi, D. Chiaro, A. Guzzo, M. Ianni, G. Fortino, and F. Piccialli, "Model aggregation techniques in federated learning: A comprehensive survey," *Future Generation Computer Systems*, vol. 150, pp. 272–293, 2024.

[4] M. Ye, X. Fang, B. Du, P. C. Yuen, and D. Tao, "Heterogeneous federated learning: State-of-the-art and research challenges," *ACM Comput. Surv.*, vol. 56, Oct. 2023.

[5] E. Diao, J. Ding, and V. Tarokh, "Hetero{fl}: Computation and communication efficient federated learning for heterogeneous clients," in *International Conference on Learning Representations*, 2021.

[6] D. A. E. Acar, Y. Zhao, R. M. Navarro, M. Mattina, P. N. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," *arXiv preprint arXiv:2111.04263*, 2021.

[7] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[8] Y. Xu, Y. Liao, H. Xu, Z. Ma, L. Wang, and J. Liu, "Adaptive control of local updating and model compression for efficient federated learning," *IEEE Transactions on Mobile Computing*, vol. 22, no. 10, pp. 5675–5689, 2022.

[9] M. K. Nori, S. Yun, and I.-M. Kim, "Fast federated learning by balancing communication trade-offs," *IEEE Transactions on Communications*, vol. 69, no. 8, pp. 5168–5182, 2021.

[10] T. Parcollet, J. Fernandez-Marques, P. P. Gusmao, Y. Gao, and N. D. Lane, "Zerofl: Efficient on-device training for federated learning with local sparsity," in *International Conference on Learning Representations (ICLR)*, 2022.

[11] F. Ilhan, G. Su, and L. Liu, "Scalefl: Resource-adaptive federated learning with heterogeneous clients," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24532–24541, 2023.

[12] S. Horvath, S. Laskaridis, M. Almeida, I. Leontiadis, S. Venieris, and N. Lane, "Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12876–12889, 2021.

[13] H. Huang, W. Zhuang, C. Chen, and L. Lyu, "Fedmef: Towards memory-efficient federated dynamic pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27548–27557, 2024.

[14] F. Ilhan, G. Su, and L. Liu, "Scalefl: Resource-adaptive federated learning with heterogeneous clients," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24532–24541, 2023.

[15] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 9, pp. 3400–3413, 2019.

[16] S. Chen, C. Shen, L. Zhang, and Y. Tang, "Dynamic aggregation for heterogeneous quantization in federated learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 10, pp. 6804–6819, 2021.

[17] H. Chen and H. Vikalo, "Mixed-precision quantization for federated learning on resource-constrained heterogeneous devices," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6138–6148, 2024.

[18] D. Jhunjhunwala, A. Gadhikar, G. Joshi, and Y. C. Eldar, "Adaptive quantization of model updates for communication-efficient federated learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3110–3114, IEEE, 2021.

[19] T. Zhang, L. Gao, S. Lee, M. Zhang, and S. Avestimehr, "Timelyfl: Heterogeneity-aware asynchronous federated learning with adaptive partial training," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5064–5073, 2023.

[20] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," in *International Conference on Artificial Intelligence and Statistics*, pp. 3581–3607, PMLR, 2022.

[21] L. Yuan, Z. Wang, L. Sun, S. Y. Philip, and C. G. Brinton, "Decentralized federated learning: A survey and perspective," *IEEE Internet of Things Journal*, 2024.

[22] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[23] M. De Vos, S. Farhadkhani, R. Guerraoui, A.-M. Kermarrec, R. Pires, and R. Sharma, "Epidemic learning: Boosting decentralized learning with randomized communication," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[24] C. Che, X. Li, C. Chen, X. He, and Z. Zheng, "A decentralized federated learning framework via committee mechanism with convergence guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4783–4800, 2022.

[25] Z. Li, J. Lu, S. Luo, D. Zhu, Y. Shao, Y. Li, Z. Zhang, Y. Wang, and C. Wu, "Towards effective clustered federated learning: A peer-to-peer framework with adaptive neighbor matching," *IEEE Transactions on Big Data*, 2022.

[26] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, *et al.*, "Advances and open problems in federated learning," *Foundations and trends® in machine learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[27] J.-H. Duan, W. Li, D. Zou, R. Li, and S. Lu, "Federated learning with data-agnostic distribution fusion," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8074–8083, 2023.

[28] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.