TUNE-FL: adapTive semi-synchronoUs semi-deceNtralizEd Federated Learning

Houssem Jmal, Kandaraj Piamrat, Ons Aouedi

TUNE-FL: adapTive semi-synchronoUs semi-deceNtralizEd Federated Learning

Houssem Jmal†, Kandaraj Piamrat†, Ons Aouedi‡
† Nantes University, École Centrale Nantes, IMT Atlantique, CNRS, INRIA, LS2N, UMR 6004, Nantes, France
‡ SnT, SIGCOM, University of Luxembourg, Luxembourg

***Abstract*—Today, Federated Learning (FL) stands out as the solution to addressing the challenges of distributed computing and empowering a wide range of edge devices with artificial intelligence capabilities. One variant of FL called semi-decentralized FL (SDFL) enables multiple server units to coordinate the learning task instead of relying on only one central server, hence preventing single-point failures. However, SDFL requires careful consideration regarding the coordination between server nodes, and dealing with the heterogeneous computing resources and data distributions across end devices (FL clients). Therefore, we propose TUNE-FL, an adapTive semi-synchronoUs semi-deceNtralizEd Federated Learning that addresses the clients' heterogeneity challenges. TUNE-FL alleviates these challenges by (i) ensuring consensus regardless of the network topology, and (ii) deploying an adaptive semi-synchronous mechanism for coordinating the learning process across all nodes while taking into consideration the heterogeneity presented by end devices. We evaluated TUNE-FL for the intrusion detection system (IDS) datasets and compared it with the three most representative baseline models. The experimental results demonstrate that TUNE-FL outperforms the baselines in accuracy while greatly reducing the duration of FL training by approximately 97 times.**

***Index Terms*—Semi-Decentralized federated learning, edge device heterogeneity, adaptive synchronization.**

I. Introduction

The rapid proliferation of sensory data generated by ubiquitous end devices, coupled with the emergence of cloud computing and the widespread availability of wireless networks, has catalyzed the rise of Federated Learning (FL) [1]. The goal of FL is mainly to maintain the privacy of the data that can contain sensitive and confidential information and to reduce communication costs by transferring only model updates. In this context, decentralized Federated Learning (DFL), where edge servers are connected in a peer-to-peer manner without a central server, further improves the limitations of having a single point of failure, trust dependencies, and bottlenecks at the central server node [2]. DFL has been widely employed in various real-life scenarios. For instance, the Industrial Internet of Things (IIoT) can leverage the scalability of DFL to seamlessly adapt to geographically dispersed production sites [3]. To further prevent communication bottlenecks and address the challenge of heterogeneity in data, semi-decentralized FL (SDFL) has been used as an alternative [4]. In SDFL, multiple edge servers, connected to each other, orchestrate the learning process and clients are divided into clusters where each cluster is directly connected to one corresponding edge server.

Generally, in FL, there are two communication modes. First, *synchronous* mode requires edge servers to wait for all clients to finish training; however, it can be impractical in scenarios with varying clients' computational speeds, leading to delays. Second, *asynchronous* mode updates the global model immediately after receiving local models, causing model divergence due to outdated updates. In addition, [5] presents a semi-decentralized federated edge learning system (SD-FEEL). Although the authors considered providing the synchronous and asynchronous versions of their approach, the latter still suffers from different aspects, such as static training time or redundant model exchange between edge servers.

Finally, to address these issues, a hybrid semi-asynchronous/synchronous FL has been introduced as an alternative. It is a mechanism that merges both synchronous and asynchronous strategies. In this context, we present `TUNE-FL`, an adapTive semi-synchronoUs semi-deceNtralizEd Federated Learning. Our work addresses the challenges of adaptiveness to heterogeneous clients with varying resource availability, as well as generalizability to different and complex network topologies. Overall, our research contributions are as follows.

- We develop a novel semi-synchronous mechanism, called `TUNE-FL` that reduces the FL training process duration by relying on clients' time estimations to reach convergence in the next FL round while taking into consideration their dynamic behavior.
- We present an optimized method for information exchange between edge servers that avoids redundancy and communication overhead.
- We evaluate our approach for intrusion detection systems against three baselines. Our results show that our method outperforms the baselines while significantly reducing the training time duration.

The rest of the paper is organized as follows. In Sec. II, we review the literature. Sec. III presents our assumptions and problem formulation. Sec. IV describes our work, and Sec. V evaluates the proposed method. Finally, Sec. VI concludes our paper and provides possible research direction.

II. Related Work

The performance of Decentralized and semi-decentralized FL (DFL/SDFL) is greatly influenced by FL clients' heterogeneity, which can be divided into two types, *System Heterogeneity* and *Data Heterogeneity*, refereed to the stragglers' capacity constraints and non-IID data, respectively.

First, the heterogeneous nature of devices entails varying computational resources, such as processing speed, and memory usage. To address this challenge, many works proposed using selection mechanisms, staleness-aware procedures, and cache-based approaches [6]. Specifically, [7] leverages an adaptive asynchronous protocol in DFL. In this protocol, workers or nodes dynamically determine the number of neighboring workers to communicate with, prioritizing only the fastest ones. Each iteration begins with identifying these neighbors and concludes once local gradient computations are completed. Also, in [8], edge devices form local cluster topologies based on their communication capability. The authors account for the local topologies between edge devices, aiming to optimize the trade-off between energy consumption, delay, and model accuracy by introducing an aperiodic consensus procedure of models within local device clusters and aperiodic global aggregations by a server. Moreover, [9] proposes a semi-synchronous FL protocol for the Internet of Vehicles. In this work, during each FL round, clients are selected based on their computing capacity, network capacity, and the learning value of their training samples. Lately, The server waiting time is then calculated as the maximum of the sum of both computing and communication times for all selected clients.

Furthermore, numerous works have used the network topology to optimize the FL process. In particular, [10] presents a new multigraph topology to reduce training time in cross-silo FL. This is achieved by decomposing the multigraph into simpler graphs. The approach identifies nodes capable of performing model aggregation independently of others in each simple graph, based on the delay introduced by each node's links, which significantly reduces FL training time. In addition, the authors in [11] introduce MATCHA, which breaks down the set of possible communications into pairs of clients and tunes the frequency of inter-node communication during each round. Finally, the work in [12] addresses the slow convergence challenge in complex graphs by incorporating additional edges. The approach consists of finding the set of most beneficial edges serving as the path for faster convergence.

In summary, our work differs from existing methods by addressing the dynamic client resource availability in DFL or SDFL, regardless of the network topology between edge servers. In particular, we develop an adaptive synchronization strategy for clients, taking into consideration their heterogeneous and dynamic behavior.

III. Problem Formulation

In this section, we present the context, the assumptions, and finally the problem formulation for SDFL and the semi-synchronous model.

Our environment consists of two main components: *edge servers* and *clients*, as depicted in Fig. 1. We consider an FL setting composed of powerful edge servers with stable and rapid connectivity, acting as the main computational hub for aggregation and information sharing. On the other hand, clients are distinguished from each other by their diverse computational power, dynamic behavior, and data distributions. We

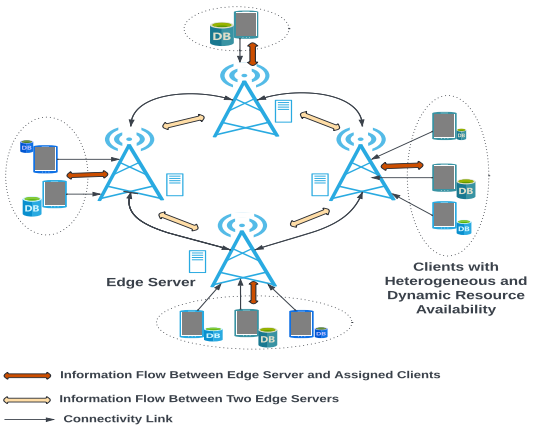


Fig. 1: TUNE-FL environment.

adopt the client categorization from [13], considering only edge devices capable of training ML models. Hence, our clients are categorized as follows: $Slow$, $Medium$, and $Fast$.

First, we represent a given network as an undirected connectivity graph $G(V, E, C)$, where communication between any two nodes is bidirectional. V implies the set of vertices representing edge servers, E is the set of edges reflecting connections between servers, and C is the set of available clients. We assume the graph is strongly connected, meaning there is at least one path between any two edge servers.

We also denote the set of clients assigned to the edge server node v_i as C_{v_i}. Each client has a set of private local training data denoted as $\mathcal{D}_k = \{\boldsymbol{x}_j, y_j\}_{j=1}^{|\mathcal{D}_k|}, k \in C$ where x_j is the training sample and y_j its corresponding label. Let $f(x_j, y_j; W)$ be the loss of the data point (x_j, y_j) based on the model parameter W. The goal of FL is to generate one global model W by minimizing the global loss across all clients as follows:

$$\min_{W} F(W) = \sum_{k \in C} p_k F_k(W) \tag{1}$$

where $F_k(W) = \frac{1}{|\mathcal{D}_k|} \sum_{(x_j, y_j) \in \mathcal{D}_k} f(x_j, y_j; W)$ and p_k refers to the normalized participation weight of the k-th device.

The objective of FL is to iteratively refine the parameters of the global model W, following a predefined communication mechanism. We define a *semi-synchronous FL* model as a hybrid approach where there is a synchronization mechanism that forces all participating clients to synchronize their models, but it does not necessarily wait for all clients to complete their training. We aim to determine an *adaptive synchronization time* T^{r+1} for each FL round $r + 1$. Please note that we only consider the computation time, which is dependent on the client category and the size of the data set $|\mathcal{D}_k|$.

To determine the necessary number of communication rounds N between edge servers, we consider the message from node i at round r as m_i, where $m_i = (\{W_i^r\}, \{S_k^{r+1}\}_{k \in C_{v_i}})$ and W_i^r is the sub-global model of node i. Our objective is to enable each node to possess all messages $\{m_1, \cdots, m_{|V|}\}$. At each message passing round n, each node v_i diffuses the set $\Delta M_i(n-1)$, which are new messages received from the previous communication round, to its neighboring nodes

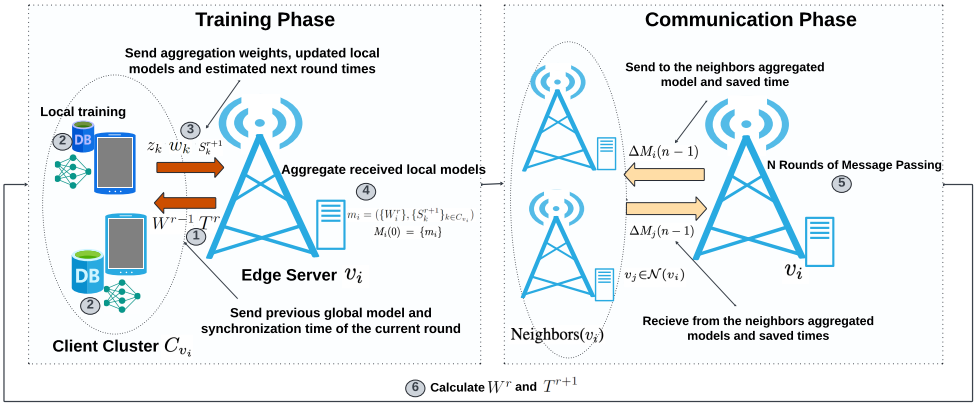


Fig. 2: TUNE-FL Process Workflow.

$\mathcal{N}(v_i)$ to optimize communication and avoid redundancy while sharing information. Hence, the set of messages at nodes v_i is updated as follows:

$$M_i(n) = M_i(n-1) \cup \left(\bigcup_{v_j \in \mathcal{N}(v_i)} \Delta M_j(n-1) \right) \quad (2)$$

where $M_i(0) = \{m_i\}$, and $\Delta M_i(n-1) = M_i(n) \setminus M_i(n-1)$, $\forall v_i \in V$. Therefore, our objective is to identify the minimal number of message-passing rounds irrespective of the network topology as follows:

$$\min_N M_i(N) = \{m_1, \cdots, m_{|V|}\}, \forall v_i \in V. \quad (3)$$

IV. Proposed Solution: TUNE-FL

In this section, we provide a detailed description of our methodology. This includes determining the essential number of message passing rounds and describing the various steps during the training and communication phases.

A. Methodology

In order to address the problem defined previously, we propose TUNE-FL as an adaptive semi-synchronous FL solution, as illustrated in Fig. 2. Before starting the FL process, we first determine the required number of message passing rounds, denoted as N based on the connectivity established between edge servers (Sec. IV-B). Each FL global round includes multiple steps. At the beginning of each global round r, edge servers trigger the training phase (Sec. IV-C) by dispatching the prevailing global model W^{r-1} along with the synchronization time T^r to their assigned clients ①. Then each device k starts its local training using its private dataset $\mathcal{D}_k$. They continue to train until the deadline T^r is met, or they have converged before it ②. After the deadline, each client k estimates, the time required to achieve convergence for the subsequent global round, then communicates its participation weight z_k, local model update w_k and estimated next round time S_k^{r+1} to their respective edge servers v_i ③, who will collect this information, aggregate the local updates received and saves the new sub-global model $W_{v_i}^r$ together with the set of estimated next round time for each client $\{S_k^{r+1}\}_{k \in C_{v_i}}$ ④. Following this, edge servers trigger the communication phase ⑤ (Sec. IV-D), they exchange information through message-passing to finally compute a unified global model W^r and a next-round synchronization point T^{r+1}⑥, reflecting the insights of all participating devices.

B. Determining Number of Message Passing Rounds

In graph theory, message passing refers to the process of exchanging data between nodes. Our goal is to determine the number of message-passing rounds N that satisfies Equation 3, regardless of the network topology formed by the FL systems. Thus, we propose the following solution. Given the connectivity graph G between edge servers, let $A = (a_{ij})_{1 \le i,j \le |V|}$ be the adjacency matrix of the graph G. A represents the nodes visited after 1 step in the graph, specifically $a_{ij} = 0$ indicates that node i cannot reach node j and the same is true for node j due to the symmetry of A (undirected graph). Therefore, A^n gives us the visited nodes after the n steps. By summing the adjacency matrices $A^1 + A^2 + \cdots + A^N$, we accumulate the visited nodes at each step, capturing the complete traversal history within N hops. Let $H_N = \sum_{n=1}^{N} A^k$, we need to find N such that $H_N = (h_{ij})_{1 \le i,j \le |V|}$ satisfies the following condition, i.e., $\forall 1 \le i,j \le |V|$, $h_{ij} \neq 0$. A zero in H_N indicates that the corresponding node is not reachable after N hops on the graph. To calculate H_N, we use the following equations:

$$(A - I) \times H_N = A^{N+1} - I \quad (4)$$

If $A - I$ is invertible where I is the identity matrice, then by multiplying by $[A - I]^{-1}$

$$H_N = [A - I]^{-1} \times \left[A^{N+1} - I\right] \quad (5)$$

Since null elements in H_N decreases as N increases, we employ a binary search algorithm to efficiently determine the required number of message-passing rounds N. Fig. 3 illustrates the intuition behind our method for determining N.

C. Training Phase

At the beginning of round r, each edge server v_i disseminates the global model W^{r-1} and the synchronization time T^r to their assigned clients C_{v_i}. Subsequently, clients initiate their local training using their private dataset D_k. In the j-th epoch, each client k performs a model update with his local

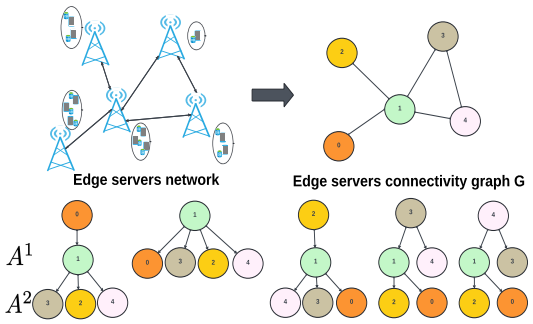


Fig. 3: Relation between adjacency matrices A and N.

dataset using the minibatch SGD algorithm, as shown in the following expression:

$$w_j \leftarrow W^{r-1} - \eta f_k^j(\xi_k; W^{r-1}) \tag{6}$$

where W^{r-1} is the downloaded global model from the $r-1$-th round, $f_k(\xi_k^{(r)}; W^{r-1})$ is gradient calculated based on a randomly selected batch of local training data $\xi_k^{(r)}$ from client k, while η represents the learning rate. After each epoch, we update the minimal loss $f_k^{min} = min(f_k^{min}, f_k^j)$, and compare it to the current j-th loss value. We define the convergence state of client k as follows:

$$|f_k^j - f_k^{min}| \leq \epsilon,\ \epsilon > 0 \tag{7}$$

Throughout the training process, clients continuously update their local models until the deadline T^r is met or their local model has converged before the deadline. We denote t_k^r, the elapsed time took client k if he has converged before T^r. Our goal is to enable all clients to adaptively estimate the necessary time needed S_k^{r+1} for the next round to satisfy Equation 7. Initially, in the first FL round, each edge server must wait for all clients to reach their convergence state, thus $S_k^1 = t_k^1$, where t_k^1 is the actual time it took for the client k to converge. Lately, based on its individual training progress, each client estimates the required time to achieve convergence for the next round using the following expression, $\forall r \geq 1$:

$$S_k^{r+1} = \begin{cases} \beta t_k^r + (1-\beta) S_k^r & if\ t_k^r \leq T^r \\ (1-\beta) T^r + \beta S_k^r & otherwise. \end{cases} \tag{8}$$

where $0 \leq \beta \leq 1$ is a parameter for controlling the weight of the most indicative convergence time point.

Clients can show different behaviors during their training; some may not be able to converge before the deadline, which results in incomplete training, and can seriously hinder the convergence of the global model. Rather than discarding the contributions of these clients, we adjust their influence in the aggregation process by considering the ratio of the synchronization point to their estimated convergence time at round r, as described below:

$$z_k = \begin{cases} |\mathcal{D}_k|, & if\ S_k^r \leq T^r \\ \frac{T^r}{S_k^r} \cdot |\mathcal{D}_k| & otherwise \end{cases} \tag{9}$$

z_k is computed locally by the client and denotes the participation weight of client k and $|\mathcal{D}_k|$ his data size. After receiving the participation weights, we update the sub-global model $W_{v_i}^r$ of v_i with the following equation:

$$W_{v_i}^r = \sum_{k \in C_{v_i}} \frac{z_k}{\sum_{j \in C_{v_i}} z_j} w_k \tag{10}$$

D. Communication Phase

Each edge node possesses a "Mailbox" dedicated to saving incoming messages. This can be implemented in traffic control messages either in mobile standardization or wireless standard via protocols such as the Optimized Link State Routing Protocol (OLSR). Initially, each mailbox is empty. Once the training phase is completed, the communication phase begins. During this phase, each server gathers the necessary information to construct $M_i(0)$, exchanging information as detailed in Sec. III. This process aims to generate a unified global model and a synchronization point for the subsequent round, benefiting from the contribution of all clients.

After all rounds of message passing are completed, each edge server has received and stored all sub-global models $W_{v_i}^r$ from other edge nodes and all next-round estimated times of all clients S_k^{r+1} as described in equation 3. Each edge node aggregates all the sub-global models received from other servers, resulting in a unified global model that encapsulates contributions from all clients as follows:

$$W^r = \frac{1}{|V|} \sum_{v_i \in V} W_{v_i}^r \tag{11}$$

Given the heterogeneous nature of clients, as discussed in Sec. III, discrepancies in computation times for the next round can arise, especially with outliers, i.e., slow and fast clients. To address this, we used the interquartile mean (IQM) for its robustness against outliers. Thus, T^{r+1} is calculated using the following expression:

$$T^{r+1} = IQM(S_1^{r+1}, \cdots, S_{|C|}^{r+1}) \tag{12}$$

V. Evaluation and Results

The evaluation process is twofold: (1) analyzing the influence of the hyper-parameter β on our approach (Sec. V-C), and (2) comparing the performance and training speed between our approach and the baselines (Sec.V-D).

A. Experimental Settings

We choose to evaluate `TUNE-FL` on intrusion detection system (IDS) datasets. In this regard, we employ two commonly used datasets: UNSW-NB15 [14] and CIC-IDS2017 [15]. Both datasets contain a comprehensive set of network traffic features collected from a realistic environment, including normal and attack traffic. We use a fully connected Neural Network with three fully connected layers for the binary classification task (benign/attacks).

To validate the effectiveness of our approach, we compare it against three FL baselines:

- FedAvg-SDFL [16], where all edge servers must wait for their clients to finish their training, we set 10 local epochs for all clients.

- SemiSync-SDFL [17] which uses a periodic synchronization determined based on the maximum time required by any learner to complete a single epoch. Clients will continue training until the deadline is met.
- SD-FEEL [5], which is a synchronous SDFL approach that consists of edge servers that will perform inter-cluster aggregation when the j-th epoch is a multiple integer of τ_1, but not $\tau_1\tau_2$, and they will perform intra-cluster aggregation when j is a multiple integer of $\tau_1\tau_2$ (FL round). Here $\tau_1 = 1$ and $\tau_2 = 3$.

B. Implementation and Simulation

We implement the simulations using PyTorch 1.13.1, and NetworkX 3.1 to model the behavior of edge servers and clients. The simulation environment consists of 100 clients (60% fast clients, 20% medium, and 20% slow [18]) randomly assigned to 10 edge servers. We also randomly generate a strongly connected graph for the network topology. To simulate the heterogeneity and dynamic behavior of our clients, we define a client class threshold TH where $\forall k \in C$:

$$TH(k) = \begin{cases} 0.3, & if\ k\ is\ slow \\ 0.6 & if\ k\ is\ medium \\ 0.9 & otherwise \end{cases} \tag{13}$$

TH indicates the robustness of client k to continue his training. Hence, after each iteration, we generate a random value $b \sim \mathcal{U}(0,1)$. If $b \geq TH(k)$, client k pauses for 0.02s to simulate a training interruption.

For all experiments, we set a batch size of 15, 0.01 as the learning rate and $\epsilon = 0.001$. To simulate the non-IID distribution, we partition the datasets among clients using the Dirichlet distribution $Dir_{100}(0.5)$, where for each client k a proportion of $q_{y,k}$ sampled from $q_y \sim Dir_{100}(0.5)$ is assigned to k. Also, we conduct all experiments using a MacBook Pro equipped with an Apple M1 Pro processor and 32 GB RAM.

C. Hyper-parameter Assessment

The goal of this experiment is to determine β that maximizes the test accuracy for both datasets.

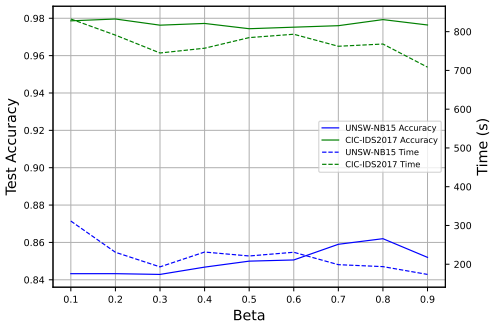
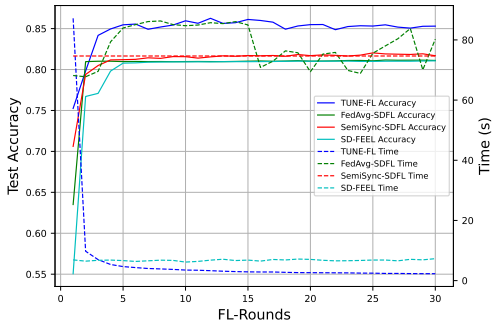


Fig. 4: Influence of β on FL training duration (30 FL rounds) and test accuracy.

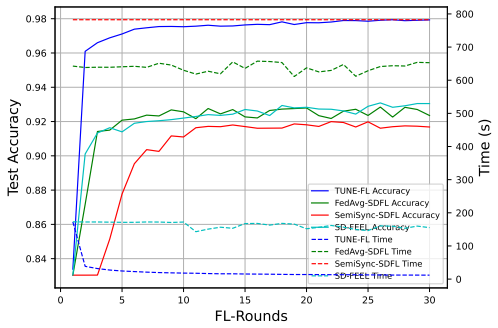
We start by varying β from 0.1 to 0.9 for both datasets, recording the resulting test accuracy and the total duration of training after 30 FL rounds, as shown in Fig. 4. Overall, we observe a general trend in test accuracy as β changes. We notice that the accuracy for both datasets steadily increases, reaching its maximum at $\beta = 0.8$. This behavior is attributed to the fact that TUNE−FL provides sufficient time for training in each global round at $\beta = 0.8$. Each client estimates the time required to reach its convergence state in each FL round by prioritizing the most recent convergence duration, as described in Equation 8. However, we notice a decrease in accuracy after $\beta = 0.8$, which is due to insufficient training duration as we also observe a drop in the FL training time.

D. Performance Evaluation

We continue our evaluation process with $\beta = 0.8$ since it yields the highest accuracy.



(a) UNSW-NB15



(b) CIC-IDS2017

Fig. 5: Comparison between TUNE-FL and baselines in test accuracy-synchronization point T^r evolution per FL round.

- Time Cost Comparison. We show the test accuracy evolution along with each FL round training duration over 30 global rounds in Fig. 5. First, as TUNE−FL waits for all clients to achieve convergence in the first FL round, we report a higher synchronization point compared to the baselines. Lately, We observed that our approach rapidly converges in terms of test accuracy and training duration, exhibiting an increase in test accuracy and an exponential decrease in training duration. This shows that TUNE−FL not only handles heterogeneous clients effectively but also becomes aware of each client's convergence state, thereby avoiding unnecessary training. On the other hand, it is expected for FedAvg-SDFL and SemiSync-SDFL to convey large training durations with the presence of $Slow$ clients. We notice fluctuations in the synchronization time per FL round for FedAvg-SDFL, which is attributed to the dynamic behavior of the clients. In contrast, SemiSync-SDFL

TABLE I: Performance comparison across both datasets.

Dataset	UNSW-NB15				CIC-IDS2017			
Approach	**TUNE-FL**	FedAvg-SDFL	SemiSync-SDFL	SD-FEEL	**TUNE-FL**	FedAvg-SDFL	SemiSync-SDFL	SD-FEEL
Accuracy	**0.862**	0.811	0.851	0.810	**0.979**	0.928	0.919	0.930
Precision	**0.815**	0.745	0.755	0.744	**0.955**	0.999	1	0.999
Recall	**0.940**	0.997	0.986	0.999	**0.921**	0.547	0.383	0.552
F1 Score	**0.948**	0.853	0.857	0.853	**0.937**	0.707	0.554	0.711

maintains a static synchronization period from the beginning, resulting in a prolonged training duration. We also observe that TUNE−FL consistently records shorter synchronization points compared to SD-FEEL. We note that SD-FEEL has been proposed as a solution for achieving faster convergence, and it highly abuses the communication between clients and edge servers. Overall, our approach reduces the training duration by approximately 92×, 91× and 7× for UNSW-NB15 and 96×, 97× and 86× for CIC-IDS2017 compared to FedAvg-SDFL, SemiSync-SDFL and SD-FEEL, respectively.

- Classification Comparison. We compare the performance of TUNE−FL to the baselines in Table I. We notice that our approach outperforms the baselines on both datasets. In particular, our method demonstrates more balanced metrics with 0.948 and 0.937 F1-Score for UNSW-NB15 and CIC-IDS2017, respectively, which results in fewer false positives and a higher true positive detection rate. We draw attention to the consensus achieved with the N rounds of message passing that not only makes TUNE−FL capable of adaptively estimating the synchronization point but also benefits from the contribution of all clients' local models, improving the performance of the IDS. Note that a more advanced model can be used to further improve the accuracy of the classification.

VI. Conclusion and Future Work

To address the challenges presented by the clients' heterogeneity, this work presents an adaptive semi-synchronous mechanism in semi-decentralized networks. We demonstrate the ability of TUNE−FL to handle resource-constrained clients and their dynamic behaviors regardless of the network topology while reducing the duration of the training and achieving promising results. We also highlight the need for an adaptive synchronization mechanism throughout the FL process to ensure training efficiency. The results demonstrate the ability of TUNE−FL to handle three main challenges in SDFL: (i) a general approach for any network topology, (ii) the non-IID distribution, and (iii) the heterogeneous and dynamic computing capabilities in edge devices.

Our future work includes extending TUNE−FL capabilities to estimate communication time in case of a lossy channel, as delays can be introduced during the communication phase between edge servers, altering the synchronization time.

Acknowledgement

This work was supported by the ANR CHIST-ERA project Di4SPDS-Distributed Intelligence for Enhancing Security and Privacy of Decentralised and Distributed Systems.

References

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," 2019.

[2] L. Yuan, Z. Wang, L. Sun, P. S. Yu, and C. G. Brinton, "Decentralized federated learning: A survey and perspective," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[3] M. Du, H. Zheng, X. Feng, Y. Chen, and T. Zhao, "Decentralized federated learning with markov chain based consensus for industrial iot networks," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 4, pp. 6006–6015, 2023.

[4] J. Wu, S. Drew, F. Dong, Z. Zhu, and J. Zhou, "Topology-aware federated learning in edge computing: A comprehensive survey," 2023.

[5] Y. Sun, J. Shao, Y. Mao, J. H. Wang, and J. Zhang, "Semi-decentralized federated edge learning with data and device heterogeneity," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1487–1501, 2023.

[6] C. Xu, Y. Qu, Y. Xiang, and L. Gao, "Asynchronous federated learning on heterogeneous devices: A survey," 2023. [Online]. Available: https://arxiv.org/abs/2109.04269

[7] G. Xiong, G. Yan, S. Wang, and J. Li, "Straggler-resilient decentralized learning via adaptive asynchronous updates," 2024. [Online]. Available: https://arxiv.org/abs/2306.06559

[8] F. P.-C. Lin, S. Hosseinalipour, S. S. Azam, C. G. Brinton, and N. Michelusi, "Semi-decentralized federated learning with cooperative d2d local model aggregations," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 12, pp. 3851–3869, 2021.

[9] F. Liang, Q. Yang, R. Liu, J. Wang, K. Sato, and J. Guo, "Semi-synchronous federated learning protocol with dynamic aggregation in internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4677–4691, 2022.

[10] T. Do, B. X. Nguyen, V. Pham, T. Tran, E. Tjiputra, Q. D. Tran, and A. Nguyen, "Reducing training time in cross-silo federated learning using multigraph topology," 2023.

[11] J. Wang, A. K. Sahu, Z. Yang, G. Joshi, and S. Kar, "Matcha: Speeding up decentralized sgd via matching decomposition sampling," 2019.

[12] M. Zhou, G. Liu, K. Lu, R. Mao, and H. Liao, "Accelerating the decentralized federated learning via manipulating edges," in *Proceedings of the ACM on Web Conference 2024*, ser. WWW '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 2945–2954. [Online]. Available: https://doi.org/10.1145/3589334.3645509

[13] A. Rocha Neto, B. Soares, F. Barbalho, L. Santos, T. Batista, F. Delicato, and P. Pires, "Classifying smart iot devices for running machine learning algorithms," 07 2018.

[14] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6.

[15] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *International Conference on Information Systems Security and Privacy*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:4707749

[16] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," 2023.

[17] D. Stripelis, P. M. Thompson, and J. L. Ambite, "Semi-synchronous federated learning for energy-efficient training and accelerated convergence in cross-silo settings," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 5, jun 2022. [Online]. Available: https://doi.org/10.1145/3524885

[18] J. Sun, A. Li, L. Duan, S. Alam, X. Deng, X. Guo, H. Wang, M. Gorlatova, M. Zhang, H. Li, and Y. Chen, "Fedsea: A semi-asynchronous federated learning framework for extremely heterogeneous devices," in *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '22. New York, NY, USA: Association for Computing Machinery, 2023, p. 106–119. [Online]. Available: https://doi.org/10.1145/3560905.3568538