Adaptive Client Selection in Federated Learning: A Network Anomaly Detection Use Case

William Marfo, Deepak K. Tosh, Shirley V. Moore

Department of Computer Science, *University of Texas at El Paso*, El Paso, USA
wmarfo@miners.utep.edu, dktosh@utep.edu, svmoore@utep.edu

***Abstract*—Federated Learning (FL) has become a ubiquitous approach for training machine learning models on decentralized data, addressing the myriad privacy concerns inherent in traditional centralized methods. However, the efficiency of FL depends on effective client selection and robust privacy preservation mechanisms. Inadequate client selection may lead to suboptimal model performance, while insufficient privacy measures risk exposing sensitive data. This paper proposes a client selection framework for FL that integrates differential privacy and fault tolerance. Our adaptive approach dynamically adjusts the number of selected clients based on model performance and system constraints, ensuring privacy through calibrated noise addition. We evaluate our method on a network anomaly detection use case using the UNSW-NB15 and ROAD datasets. Results show up to a 7% increase in accuracy and a 25% reduction in training time compared to FedL2P. Moreover, we highlight the trade-offs between privacy budgets and model performance, with higher privacy budgets reducing noise and improving accuracy. Our fault tolerance mechanism, while causing a slight performance drop, enhances robustness to client failures. Statistical validation using Mann-Whitney U tests confirms the significance of these improvements ($p < 0.05$).**
[1]

***Index Terms*—Federated learning, Client selection, Machine learning**

I. Introduction

Federated learning (FL) has emerged as a powerful paradigm in machine learning (ML), enabling models to be trained across decentralized data sources without the need to centralize sensitive data [1]. This approach is instrumental in privacy-preserving environments, where data cannot be easily shared across borders or entities [2]. The efficiency of FL, however, depends heavily on client participation during training. Client selection in FL is not a simple random process but requires careful consideration of various factors such as data heterogeneity, system performance, and computational resources [2]–[4].

The challenges of client selection in FL are multifaceted. First, preserving privacy during client selection and model updates is crucial to prevent adversaries from inferring sensitive information about individual clients' data [5]. Second, the unpredictable nature of client availability and potential failures in distributed environments can disrupt the training process, affecting model performance and convergence [5]. Additionally, there exists a fundamental trade-off between maintaining strong privacy guarantees and achieving high model accuracy, which complicates the design of effective FL systems [2].

We apply our proposed client selection method to the domain of network anomaly detection, which poses unique challenges due to the complexity and scale of modern networks. Detecting anomalies across distributed and diverse networks, such as the Internet of Things (IoT) systems or automotive communication networks, requires FL techniques that can manage computational efficiency while preserving data privacy. Traditional centralized ML approaches exacerbate privacy concerns, especially when dealing with sensitive network traffic data. To evaluate our method, we chose two prominent datasets: the UNSW-NB15 dataset [6], which captures a range of network traffic patterns and attacks, and the ROAD dataset [7], which focuses on automotive cybersecurity and includes complex, stealthy masquerade attacks. These datasets allow us to test the generalizability of our approach in both typical and specialized network environments, demonstrating its effectiveness in enhancing anomaly detection.

Motivated by these challenges and the limitations of existing approaches, we propose a client selection algorithm that integrates differential privacy (DP) and fault tolerance mechanisms. Our approach combines adaptive client selection with privacy-preserving techniques by applying DP to model updates rather than utility scores. Specifically, we perturb the gradients with Gaussian noise to ensure (ϵ, δ)-differential privacy, protecting sensitive client data during training [5]. The method also includes a robust checkpointing mechanism for fault tolerance, allowing efficient recovery from client dropouts and ensuring continuity in real-world applications. By separating client selection from the privacy-preserving update process, we balance the need for strategic client selection with the necessity of privacy guarantees aligned with FL standards.

The key contributions of this paper are as follows:

- We propose a client selection framework that integrates differential privacy through Gaussian noise added to model updates, ensuring privacy preservation without compromising model performance.
- We incorporate a fault tolerance mechanism via checkpointing, enhancing system resilience to client failures.

[1]This material is based upon work supported by the United States Department of Energy's (DOE) Office of Fossil Energy (FE) Award DE-FE0031744.

- We demonstrate the effectiveness of our approach through experiments on network anomaly detection, achieving significant improvements in accuracy and training efficiency over state-of-the-art baselines.

The paper is organized as follows: Section II reviews related work, Section III formulates the problem, Section IV presents the client selection method, Section V outlines the use case and results, and Section VI concludes with future work.

II. Related Work

Trindade et al. [2] proposed a two-step client selection method for Hierarchical FL (HFL), achieving notable reductions in resource usage while maintaining model accuracy on datasets like MNIST and CIFAR-10. Despite these improvements, they acknowledge the difficulty of comparing client selection methods due to varied evaluation scenarios and metrics, and their work is based on emulated environments that may not capture real-world complexities. Ruan et al. [4] examined the trade-offs in FL client recruitment, showing that more clients do not always enhance model performance, particularly in resource-constrained settings. They proposed an optimal solution for client recruitment and highlighted the need to integrate recruitment with existing selection methods. Li et al. [3] introduced AdaFL, an adaptive strategy that dynamically adjusts client numbers during training and evaluates contributions based on current and historical performance. AdaFL improved test accuracy and reduced training time. However, the study noted that many strategies still rely on a fixed number of clients, which may not optimally balance training efficiency and model performance throughout the learning process. Huang et al. [8] proposed ACFL, an active client selection framework for Clustered FL (CFL) with non-IID data, using metrics like uncertainty sampling to select informative clients. Their experiments showed that traditional FL methods, like FedAvg, struggle with class-imbalanced datasets, whereas ACFL significantly improved accuracy and reduced communication overhead. However, they emphasized the need for more refined strategies to effectively handle non-IID data in clustered environments.

Compared to previous studies, this work uniquely integrates adaptive client selection with differential privacy and fault tolerance in FL for network anomaly detection. We introduce an adaptive client selection approach and a checkpointing mechanism using Weibull distribution modeling [9], enhancing both performance and fault tolerance. Evaluated on UNSW-NB15 and ROAD datasets, our approach demonstrates improved accuracy and efficiency over baselines like ACFL and FedL2P. This comprehensive solution balances privacy, performance, and fault tolerance, addressing scalability challenges in distributed environments not fully explored in existing literature.

III. Problem Formulation

We formulate the client selection problem in FL as follows:

Consider a FL system with N clients, denoted as $C = \{c_1, c_2, \ldots, c_N\}$, where each client c_i has a local dataset D_i. The objective is to train a global model w by aggregating updates from a subset of clients $S_t \subseteq C$ in each round t, where $|S_t| = K$. The client selection problem is to find the optimal subset S_t^* that maximizes performance while satisfying system constraints, including privacy preservation and fault tolerance.

Key assumptions and constraints include non-IID data distribution across clients, privacy preservation through DP (applied to model gradients), limited communication bandwidth, heterogeneous client computational capabilities, and dynamic client availability. We define our objective function as $F(S_t) = \alpha \cdot \text{Accuracy}(S_t) - \gamma \cdot \text{Cost}(S_t)$, where S_t is the subset of selected clients in round t, and α and γ are weighting factors balancing the importance of accuracy and cost, respectively [3]. Here, $\text{Accuracy}(S_t)$ represents the contribution to model performance from selected clients, which could be measured using metrics such as AUC-ROC, as discussed later in the paper. The cost function $\text{Cost}(S_t) = \sum_{i \in S_t} (\text{Comm}_i + \text{Comp}_i)$ accounts for both communication cost Comm_i and computation cost Comp_i for client i. The optimization problem is then to find $S_t^* = \arg\max_{S_t} F(S_t)$, subject to $|S_t| = K$ and $S_t \subseteq \text{Available_Clients}_t$. The set $\text{Available_Clients}_t$ represents the clients that are online and have sufficient resources to participate in the current round, as determined by the GetAvailableClients() function in our algorithm. This formulation incorporates considerations for DP (applied to gradients) and fault tolerance (via checkpointing), which are detailed later in the paper. These mechanisms ensure privacy preservation and system robustness while maintaining the efficiency of the client selection process.

A. Federated Learning Architecture for Adaptive Client Selection

Our FL architecture consists of two main components: *clients* and a *global server*. Fig. 1 illustrates the FL architecture, highlighting the coordination between the global server and local client training, including checkpointing and fault tolerance mechanisms.

1. Clients: In FL, each client owns local data, which remains on the device to preserve privacy. Each client also maintains a local instance of the model that it trains using its local dataset. Let there be N clients, denoted as c_i where $i \in \{1, \ldots, N\}$. Each client has its dataset X_i, where $X_i \in \mathbf{R}^{m_i \times d}$ with m_i representing the number of samples for client i and d representing the number of features in each sample. Each client also has a local model, with parameters denoted by w_i. The local model is trained on the dataset X_i for a specified number of epochs. After training, the updated parameters w_i are sent to the global server for aggregation. Specifically, if f_i represents the function (or architecture) of the local model and w_i represents the parameters, the training process updates w_i based on the client's data X_i. The updated parameters w_i are then sent to the global server for aggregation in the global model.

2. Global Server: The global server manages the global model by aggregating the updated parameters received from all selected clients. For each round, the global server collects

the parameters w_i from the selected clients and updates the global model parameters w_g by averaging the received updates. Specifically, if N clients are selected in round t, the global model parameters are updated as $w_g = \frac{1}{N}\sum_{i=1}^{N} w_i$. This iterative process allows the global model to learn from diverse client environments without centralizing sensitive data. After updating the global model, the server broadcasts the updated parameters w_a to all participating clients for the next round.

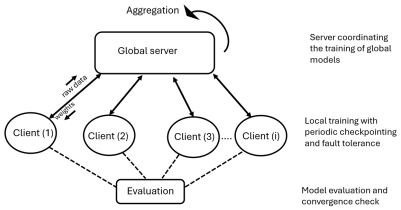

Fig. 1: FL architecture showing global server coordination and local client training with checkpointing and fault tolerance.

IV. Proposed Client Selection Method

In this section, we outline our proposed client selection method for FL, which is designed to balance accuracy, privacy, and fault tolerance. We discuss (1) the client selection algorithm, which adaptively selects clients based on their potential contribution to the global model; (2) the implementation of checkpointing strategies, which ensure robust system performance by allowing recovery from client failures; and (3) the integration of DP, which adds noise to client model updates (gradients) to protect against inference attacks.

A. Client Selection Algorithm

Our selection strategy adapts client participation dynamically based on model performance and system constraints, building upon the approaches in [3], [5]. We compute utility scores for all clients, considering factors like data quality and computational capacity. These scores are solely used for client selection and do not impact user privacy [2], [8]. In each communication round, clients are selected based on these utility scores, striking a balance between performance optimization and client diversity. Once selected, they train their local models, and DP is applied by adding Gaussian noise to their model updates (gradients) before sending them to the global server, protecting sensitive data. To ensure fault tolerance, clients periodically checkpoint their progress, enabling recovery in case of failure. After training for a set number of epochs, each client's local model parameters w_{f_i} are sent to the global server, where they are aggregated to update the global model. Algorithm 1 summarizes the overall process, incorporating both DP and fault tolerance mechanisms.

Termination condition/epochs in Algorithm 1: The algorithm proceeds iteratively over multiple communication rounds, terminating when the global model converges or when the maximum number of rounds is reached. During each round, clients train their local models for a set number of epochs. To ensure fault tolerance, checkpoints are saved at intervals t_c^*, allowing clients to recover from failures without losing progress. If a failure occurs, the client recovers from the last saved checkpoint. This mechanism ensures smooth continuation of the training process while maintaining efficient updates and aggregation at the global server.

Algorithm 1 Client selection with differential privacy and fault tolerance

Require: Set of clients C, clients to select K, privacy budget ϵ, checkpoint interval t_c^*, failure probability p_f
Ensure: Selected clients S_t per communication round t

Initialize utility scores U_i for all clients $i \in C$
for each communication round t **do**
  $A_t \leftarrow$ GetAvailableClients(C)
  $S_t \leftarrow$ SelectTopK(A_t, K, ComputeUtility(U_i))
  for each client $i \in S_t$ in parallel **do**
    last_checkpoint $\leftarrow$ current_time()
    while not converged **do**
      noisy_grad$_i \leftarrow$ grad$_i + \mathcal{N}(0, \sigma^2)$
      SendToServer(noisy_grad$_i$)
      if current_time() - last_checkpoint $\geq t_c^*$ **then**
        SaveCheckpoint(i)
      end if
      if RandomFailure(p_f) **then**
        RecoverFromCheckpoint(i)
      end if
    end while
  end for
  AggregateUpdates(S_t) and UpdateGlobalModel()
  if GlobalModelConverged() **then**
    break
  end if
end for
return S_t

Handling training failures in decentralized ML:

To handle client dropouts or disconnections in distributed environments, our framework uses a checkpointing mechanism, enabling smooth recovery and ensuring training continuity [10].

a) Recovery protocol without checkpointing

In the absence of checkpointing, recovery can involve either restarting the training from scratch or reinitializing the failed client's model using the latest global weights. We prefer the latter as it minimizes interruptions and allows training to continue with minimal disruption, though it may introduce temporary inconsistencies.

b) Recovery protocol with checkpointing

In the case of checkpointing, clients periodically store their model state as binary files. When a failure happens, the system restores the client's state from the most recent checkpoint, allowing training to continue without restarting. If a failure occurs during aggregation, the global server can either pause for recovery or reassign the client's workload to other active clients, ensuring uninterrupted training.

c) Optimal checkpointing interval

We estimate the likelihood of client failure using a Weibull distribution, as shown in [9], which is well-suited for distributed systems. The failure probability within a checkpoint-

ing interval t_c is modeled as $p_f(t_c) = 1 - \exp\left(-\left(\frac{t_c}{\lambda}\right)^k\right)$, where λ and k represent the scale and shape parameters. The total cost function, balancing checkpointing overhead and recovery, is given by $C(t_c) = \frac{t_c}{T} + p_f(t_c) \cdot \frac{t_r}{T}$, where T is the total computation time and t_r is the recovery time. The optimal checkpointing interval t_c^* is found by solving $\frac{dC}{dt_c} = 0$ numerically, using the estimated values of λ and k derived from historical failure data.

Incorporating differential privacy in FL:

We implement DP in our client selection process to protect client data during communication with the global server. Rather than applying noise to the utility scores, we add Gaussian noise directly to the gradients (model updates) after local training is completed on each client. This ensures that sensitive client data is protected during aggregation at the server. For each selected client c_i, the local gradient ∇w_i is perturbed using Gaussian noise to ensure (ϵ, δ)-differential privacy. The perturbed gradient $\nabla\tilde{w}_i$ is given by $\nabla\tilde{w}_i = \nabla w_i + \mathcal{N}(0, \sigma^2)$, where $\mathcal{N}(0, \sigma^2)$ represents Gaussian noise with variance σ^2, calibrated to the privacy budget ϵ. The noise is added in proportion to the sensitivity of the gradients, ensuring that individual client data remains private. Note that this approach guarantees that, even if an adversary gains access to the global model or intercepts the communication channels, the privacy of individual client data remains protected. We evaluate the impact of different privacy budgets, with lower ϵ values offering stronger privacy but potentially reducing model performance due to increased noise. This trade-off between privacy and utility is crucial in FL, as stronger privacy often comes at the cost of reduced accuracy (see results in Section V-C-2).

We now demonstrate the practical application of our client selection method in network anomaly detection, addressing key challenges such privacy preservation, and system robustness.

V. Use Case: Network Anomaly Detection

Network anomaly detection serves as an ideal test case for our client selection framework, given the distributed and privacy-sensitive nature of network traffic data [1]. Our approach combines adaptive client selection for diverse network behavior capture, DP for data protection, and fault tolerance for continuous learning despite connectivity issues. We evaluate this framework through experiments (Section V-A), compare against baselines (Section V-B), and present statistical validation (Section V-C).

A. Experimental Setup

We evaluated our method using the UNSW-NB15 dataset [6] for network intrusion detection and the ROAD dataset [7] for controller area network traffic, focusing on correlated masquerade attack. Experiments were run on a system with a 12th Gen Intel® Core™ i9-12900HK CPU, NVIDIA RTX 3080 Ti GPU, and 32GB RAM, using `Python 3.8.18`, `TensorFlow 2.6.0`, and `PyTorch 0.5.0`. Key hyperparameters ($\epsilon \in [0.1, 10.0]$, checkpoint interval t_c^*, client fraction K) were optimized through grid search over 10 repeated trials with different random seeds, using 200 communication rounds with 5 local epochs per round. Results reported are means across all trials.

B. Baseline Methods and Evaluation Metrics

We compared our client selection method to two baselines: (1) **ACFL** [5], which selects informative clients using active learning, and (2) **FedL2P** [11], which applies meta-learning for personalized fine-tuning. We used accuracy and AUC-ROC, standard metrics in anomaly detection, to evaluate performance.

C. Results and Analysis

1) Detection performance evaluation

Table I compares our method to ACFL and FedL2P across 40 clients and 200 communication rounds with 5 local epochs per round, following the neural network model in [1]. Our method consistently outperforms the baselines in accuracy, AUC-ROC, and training time due to adaptive client selection, fault tolerance, and differential privacy. On UNSW-NB15, we achieved 94.8% accuracy in 570 seconds, while on ROAD, we reached 90.3% accuracy and 0.88 AUC-ROC in 680 seconds.

Fig. 2 illustrates a performance comparison, showing up to 7% higher accuracy and 25% faster training compared to baselines. These gains are attributed to the selection of informative clients and balancing privacy with performance, alongside robust fault tolerance, which ensures learning continuity during client failures.

TABLE I: Performance comparison of ACFL, FedL2P, and Proposed method.

Method	Accuracy (%)	AUC-ROC	Time (s)
UNSW-NB15			
ACFL	87.8	0.86	760
FedL2P	92.1	0.91	600
Proposed	**94.8**	**0.93**	**570**
ROAD			
ACFL	83.3	0.81	905
FedL2P	88.7	0.86	710
Proposed	**90.3**	**0.88**	**680**

2) Impact of differential privacy and fault tolerance

We evaluate the impact of privacy budgets (ϵ) and fault tolerance on performance. Fig. 3 shows that on UNSW-NB15, accuracy improved from 86% ($\epsilon = 10$) to 89% ($\epsilon = 100$), with a loss reduction from 3 to 2.5. ROAD showed a greater improvement, with accuracy rising from 73% to 82% and loss decreasing from 10 to 9. Table II summarizes the impact of fault tolerance. While accuracy and AUC-ROC slightly decrease (e.g., 94.8% to 92.1% on UNSW-NB15), fault tolerance enhances robustness, ensuring continuous learning during client failures—essential for real-world scenarios.

Note that trade-offs between communication costs and training time are common in FL [1]. Though our approach improves performance, future work will explore the bandwidth-accuracy trade-off in more detail.

3) Statistical significance testing

To validate the performance differences between our proposed method and the baselines (ACFL and FedL2P), we employed

TABLE II: Impact of fault tolerance on model performance.

Configuration	Accuracy (%)	AUC-ROC	Time (s)
UNSW-NB15			
Without fault tolerance	**94.8**	**0.93**	570
With fault tolerance	92.1	0.91	**600**
ROAD			
Without fault tolerance	**90.3**	**0.88**	680
With fault tolerance	88.7	0.86	**710**

the Mann-Whitney U test [12]. We compared AUC-ROC distributions across the UNSW-NB15 and ROAD datasets. Our null hypothesis (H_0) is that there is no significant difference in performance, while the alternative hypothesis (H_1) suggests our method's superiority. Table III presents the results, showing consistently low p-values ($p < 0.05$) for all comparisons. Consequently, we reject H_0, confirming that our proposed method significantly outperforms the baselines in terms of AUC-ROC across both datasets.

TABLE III: Mann-Whitney U test results comparing the proposed method against baseline methods.

Comparison	U Statistic	P-value
UNSW-NB15		
Proposed vs. ACFL	10234.0	6.72e-10
Proposed vs. FedL2P	10898.0	4.23e-12
ROAD		
Proposed vs. ACFL	9853.0	8.45e-08
Proposed vs. FedL2P	10623.0	3.97e-11

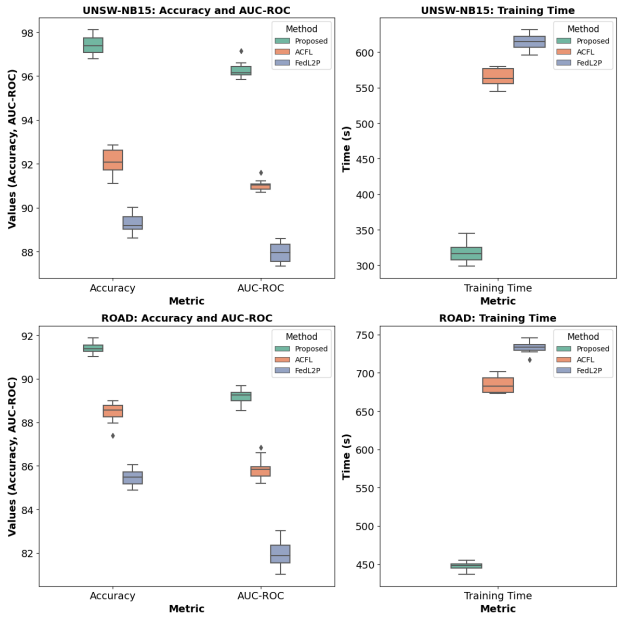

Fig. 2: Performance comparison of the proposed method, ACFL, and FedL2P across accuracy, AUC-ROC, and training time metrics for the UNSW-NB15 and ROAD datasets.

VI. Conclusion

This paper introduced a client selection method for FL with integrated DP and fault tolerance, demonstrating improved performance over FedL2P in network anomaly detection. Our method achieved up to 7% higher accuracy and 25% faster training. We identified trade-offs between privacy budgets and model performance, with fault tolerance enhancing robustness

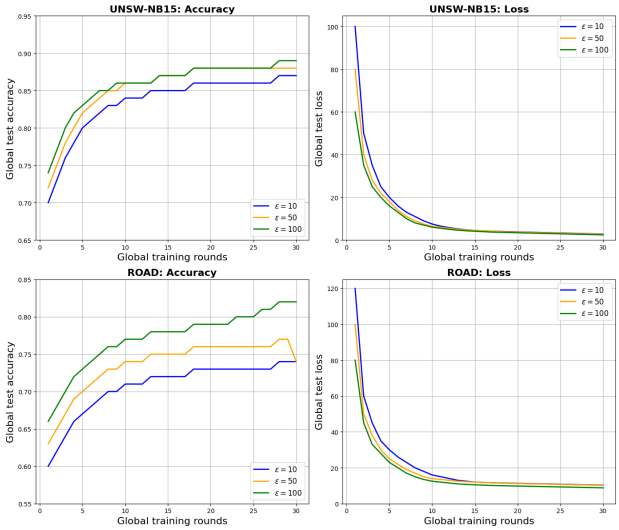

Fig. 3: Impact of privacy budgets (ϵ values) on global test accuracy and loss for UNSW-NB15 and ROAD datasets.

at a slight accuracy cost. Limitations include the lack of hyperparameter tuning. Future work will explore adaptive hyperparameter optimization and comparisons with cryptographic techniques, extending applicability to broader domains.

References

[1] W. Marfo, D. K. Tosh, and S. V. Moore, "Network anomaly detection using federated learning," in *MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM)*.

[2] S. Trindade and N. L. da Fonseca, "Client selection in hierarchical federated learning," *IEEE Internet of Things Journal*, 2024.

[3] Q. Li, X. Li, L. Zhou, and X. Yan, "Adafl: Adaptive client selection and dynamic contribution evaluation for efficient federated learning," in *ICASSP 2024-2024 IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*.

[4] Y. Ruan, X. Zhang, and C. Joe-Wong, "How valuable is your data? optimizing client recruitment in federated learning," *IEEE/ACM Transactions on Networking*, 2024.

[5] G. Yan, H. Wang, X. Yuan, and J. Li, "Criticalfl: A critical learning periods augmented client selection framework for efficient federated learning," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

[6] 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)*. IEEE, 2015.

[7] M. E. Verma, R. A. Bridges, M. D. Iannacone, S. C. Hollifield, P. Moriano, S. C. Hespeler, B. Kay, and F. L. Combs, "A comprehensive guide to can ids data and introduction of the road dataset," *PLoS one*, 2024.

[8] H. Huang, W. Shi, Y. Feng, C. Niu, G. Cheng, J. Huang, and Z. Liu, "Active client selection for clustered federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[9] A. S. S. Vardhan, A. Verma, J. Ogale, R. Saket, and S. Galloway, "Modern aspects of probabilistic distributions for reliability evaluation of engineering systems," *Reliability Analysis of Modern Power Systems*, pp. 217–245, 2024.

[10] A. Benoit, L. Perotin, Y. Robert, and F. Vivien, "Checkpointing strategies to tolerate non-memoryless failures on hpc platforms," *ACM Transactions on Parallel Computing*, 2024.

[11] R. Lee, M. Kim, D. Li, X. Qiu, T. Hospedales, F. Huszár, and N. Lane, "Fedl2p: Federated learning to personalize," *Advances in Neural Information Processing Systems*, 2024.

[12] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, 1947.