Client Selection in Fault-Tolerant Federated Reinforcement Learning for IoT Networks

Semih Cal*, Xiang Sun†, and Jingjing Yao*
*Department of Computer Science, Texas Tech University, Lubbock, TX 79409, USA.
†Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131, USA.

***Abstract*—In wireless Internet of Things (IoT) networks, Federated Reinforcement Learning (FRL) has emerged as a decentralized strategy for data-driven decision-making, enabling devices to learn directly from real-time environmental interactions, sidestepping the need for labeled data. This method promises enhanced data privacy and finds practical applications in autonomous driving, smart grids, and industrial automation. However, the integrity of FRL can be compromised by malicious clients injecting false data, underlining the need for a fault-tolerant mechanism to sustain the robustness and accuracy of the learning phase. Moreover, the inherent client heterogeneity within IoT networks propels the demand for judicious client selection, optimizing computational and communication resources. This paper investigates client selection problem within a fault-tolerant FRL framework for wireless IoT networks. Our objective is to explore the tradeoff between maximizing client participation and minimizing energy consumption of IoT devices. We formulate our problem as a mixed-integer linear programming (MILP) model and design an efficient algorithm with low computational complexity to address it. Extensive simulations are conducted to demonstrate the superiority of our proposed algorithm.**

***Index Terms*—Federated reinforcement learning, internet of things, client selection, fault tolerance, energy consumption.**

I. Introduction

Internet of Things (IoT) has become indispensable in a myriad of applications and services due to its advanced embedded monitoring and data collection capabilities [1]. The collected IoT data are usually offloaded to the remote cloud or edge servers for processing by machine learning (ML) techniques. However, relying on third-party edge servers or the cloud might introduce privacy risks when those data encompass sensitive details such as an individual's location or preferences [2], [3]. Federated learning is proposed to address this challenge by allowing different IoT devices to train their ML models locally and only share the ML models in the edge servers without sharing the raw data [4].

In certain applications, such as autonomous vehicles and online gaming, the labeled training data are often absent. Instead, decisions are usually derived from real-time interactions with a continuously changing environment. This is where Federated Reinforcement Learning (FRL) proves its worth [5]. Integrating the principles of both federated learning and reinforcement learning (RL), FRL empowers multiple agents (like IoT devices) to collaboratively refine a shared RL model while upholding data privacy [6]. In this setup, every agent undergoes local RL training within its unique environment, assimilating knowledge through interactions and enhancing its local model via RL techniques. These refined models are then transmitted to a centralized server, where they are merged to construct an improved amd more robust global model [7].

FRL, despite its merits, remains susceptible to threats like Byzantine attacks [8]. In such attacks, malicious clients can manipulate authentic models and gradients, intentionally corrupting the training data. This could mislead the global model, making it assimilate inaccurate data, or even trigger system breakdowns [9]. The integration of adversarial clients into FRL can notably hinder its convergence, or in extreme cases, halt the learning process entirely. Hence, it is important to design fault-tolerant FRL systems in IoT networks [10].

In wireless FRL systems, the diverse capabilities of clients, marked by distinct computation and communication capacities, emphasize the critical importance of client selection within FRL. The choice of clients in every global iteration of the FRL training significantly influences training duration. A flawed selection can result in a straggler dilemma, leading to prolonged training time. Traditionally, the strategy behind client selection leaned towards engaging as many clients as possible within the designated time frame of each global iteration, based on the presumption that increased participation can hasten the FL training convergence [11], [12]. Yet, this approach might not always serve the best because extensive participation might intensify energy consumption during both the training and data transmission stages [13], [14]. Therefore, the tradeoff between maximizing the number of selected clients and minimizing the total energy consumption needs to be explored.

Client selection can filter out malicious participants, enhance training efficacy, and regulate energy usage, paving the way for fault-tolerant, energy-efficient, and time-optimized FRL systems. Therefore, in this paper, we investigate the client selection problem in fault-tolerant FRL framework within wireless IoT networks to explore the tradeoff between maximizing the participant client number and minimizing the system energy consumption. We design an algorithm with low computational complexity to address this problem and demonstrate its performance via extensive simulations.

The remainder of this paper is organized as follows. The related work is summarized in Section II. Section III provides an analysis of our system architecture, security model, latency

This work was supported by the National Science Foundation under Award under grant no. CNS-2323050 and CNS-2148178, where CNS-2148178 is supported in part by funds from federal agency and industry partners as specified in the Resilient & Intelligent NextG Systems (RINGS) program.

ICC 2024 - IEEE International Conference on Communications | 978-1-7281-9054-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICC51166.2024.10622515

model, and energy consumption model. The problem is formulated in Section IV. Section V outlines our proposed algorithm. In Section VI, simulation results are presented and analyzed. Finally, Section VII concludes this paper.

II. Related Work

The idea of FRL has grown significantly in acceptance and has been the focus of multiple research. Wu *et al.* [15] introduced CAFR, a novel approach that combines asynchronous federated learning and deep reinforcement learning for cooperative caching in vehicular edge computing (VEC). Zhang *et al.* [16] proposed a FRL-based algorithm for task offloading and resource allocation in future connected automated vehicle (CAV) networks, addressing low-latency data sharing for cooperative automated driving. The algorithm optimizes task execution delay while considering communication and computing constraints, demonstrating improved system performance in simulations and hardware tests. Nguyen *et al.* [7] proposed DeepMonitor, a traffic monitoring framework for SDN-based IoT networks that enhances flow-table capacity and improves intrusion detection performance compared to existing solutions like FlowStat, as demonstrated through extensive emulations.

Multiple research has explored the development of secure and fault-tolerant FRL systems. Tang *et al.* [17] investigated a node security problem and an improved practical Byzantine fault tolerance (EPBFT) algorithm to safeguard the traffic offloading procedure. They formulated the traffic offloading challenge as a Markov decision problem (MDP) and applied the Blockchain-based Federated Asynchronous Advantage Actor-Critic (BFA3C) algorithm for its resolution. Islam *et al.* [18] investigated backdoor attacks and proposed versatile defense strategies to protect against backdoor attacks in multi-UAV scenarios within the context of federated deep reinforcement learning. Fan *et al.* [19] investigated a FRL framework that guarantees convergence and robustness to system failures and attacks, without the need to share raw trajectories. The framework is shown to improve sample efficiency with the number of agents and is empirically validated on RL benchmark tasks. However, none of the above work investigates the client selection problem in the FRL framework.

Client selection in federated learning framework has been investigated in multiple research. Zhang *et al.* [20] proposed a mobile edge computing (MEC) system and an adaptive client selection algorithm based on reinforcement learning to minimize energy consumption and training delay in the federated learning framework. Qiao *et al.* [21] proposed a federated learning-based content caching system for low-latency 5G edge networks. It employs deep reinforcement learning to optimize client selection and local iteration frequency, significantly improving cache efficiency. Yu *et al.* [22] designed the ELASTIC algorithm, which dynamically balances client selection and energy consumption, optimizing model accuracy in IoT networks. However, none of the above works consider the client selection problem in fault-tolerant FRL systems.

To the best of our knowledge, no prior work has addressed the client selection problem within a fault-tolerant FRL framework in wireless IoT networks. In this paper, we seek to bridge this research gap by delving into this problem.

III. System Model

In our proposed system model, we incorporate an FRL framework within wireless IoT networks, depicted in Fig. 1. The FRL mechanism functions in an iterative manner, including both global and local training stages. Initially, each client accesses the global policy and conducts local policy update on its dataset using reinforcement learning techniques, such as the policy gradient algorithm [23]. After local training concludes, clients relay their updated parameters to a centralized server located at the base station (BS). The server subsequently aggregates these parameters to enhance the global policy. This iterative process continues, consistently refining the policy based on individual client experiences, until either a specified accuracy threshold or a predefined number of rounds is attained.

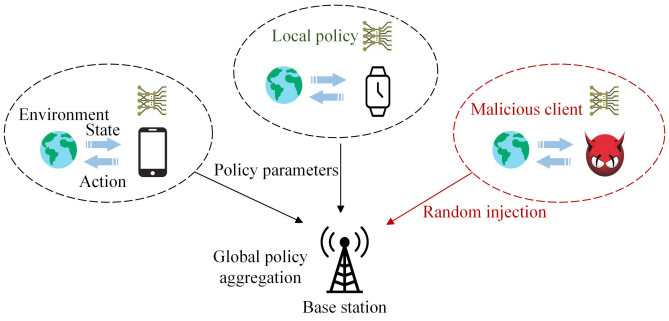

Fig. 1. Federated reinforcement learning in wireless IoT networks.

Within the BS's coverage area, assume that there are K clients (i.e., IoT devices) and we designate the set of client indices as $\mathcal{K} = 1, 2, 3, ..., K$. Each client is represented by a binary variable x_k, indicating its participation status in the ongoing FRL global iteration. If client k is selected, $x_k = 1$; otherwise, $x_k = 0$. During each global iteration, the BS disseminates the global policy parameter θ to every client. Subsequently, individual clients refine their policies, denoted as $\pi_\theta(a|s)$, through interactions with their local environments. Here, $\pi_\theta(a|s)$ determines the probability of an agent selecting action a in state s. Each client, based on its current state s, generates an action a and receives a corresponding reward $R(s, a)$. The underlying goal of the RL algorithm is to maximize the cumulative discounted reward, expressed as $J(\theta) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$, where γ denotes the discount factor and $R(s_t, a_t)$ symbolizes the time-slot-specific reward. State-action-reward trajectories (i.e., data samples) are accumulated in the replay memory. From this memory, batches of trajectories are drawn and utilized to adjust the policy, following the update rule: $\theta = \theta + \eta \nabla J(\theta)$, with η representing the learning rate [24].

A. Fault Tolerant Model

In order to establish a fault-tolerant FRL system, each client forwards its gradient, denoted as μ_k, to the BS during

every global round. However, malicious clients might transmit arbitrary vectors. To counteract this, the BS processes these gradients to pinpoint and filter out potential malicious clients. Consequently, the aggregation phase only includes gradients from clients presumed to be trustworthy [19].

We adopt the filtering model to identify the malicious clients [19], [25]. The filtering model assumes that all good clients are clustered in a small region. The maximum distance between any two good agents is $\mathfrak{T}_\mu = 2\sigma\sqrt{2\log(2K/\delta)/B}$, where σ is the variance bound, $\delta \in (0,1)$ is a constant, and B is the batch size [25]. A set of vector medians $\mathcal{S}$ can be constructed, where each client is chosen if it is close to more than $K/2$ clients, i.e., $\mathcal{S} = \{\mu_k\}$, where μ_k should satisfy $|\{\mu_{k'} : \|\mu_{k'} - \mu_k\| \leq \mathfrak{T}_\mu\}| > \frac{K}{2}$. Here, $|\cdot|$ indicates the size of a set. Next, we find a mean of median vector μ_k^{mom} from $\mathcal{S}$, which is defined as the client closest to the mean of all vectors in $\mathcal{S}$, i.e., $\mu_k^{\text{mom}} = \operatorname{argmin} \left\|\mu_k - \bar{\mathcal{S}}\right\|$. Lastly, the set of reliable clients can be calculated as those lying within a distance to μ_k^{mom} that is smaller than or equal to the maximum distance $\mathfrak{T}_\mu$ [19], i.e.,

$$\mathcal{R}_c = \{k \in \mathcal{K} : \|\mu_k - \mu_k^{\text{mom}}\| \leq \mathfrak{T}_\mu\}. \tag{1}$$

B. Latency Model

In a given global round, a client's latency comprises both its local computation duration and the time taken to upload the policy parameter to the BS. It's worth noting that the time for downloading the global model from the BS is typically much smaller in comparison to the upload duration; hence, we disregard it in our analysis [22]. Within each global round, the local computation duration for client k can be calculated as [26]

$$t_k^{\text{comp}} = I_k \frac{C_k D_k}{f_k}, \tag{2}$$

where I_k is the number of local iterations, C_k is the number of CPU cycles in one local iteration required to train a sample, D_k is the number of samples, and f_k is the CPU frequency.

Client k's wireless data rate for uploading policy parameters can be calculated according to the Shannon equation, i.e., $r_k = W_k \log_2\left(1 + \frac{p_k G_k}{N_0 W_k}\right)$, where W_k is the allocated bandwidth, p_k transmission power, N_0 is the noise power spectrum density, and G_k is the channel gain between client k and the BS. Then, client k's uploading latency becomes [22]

$$t_k^{\text{up}} = \frac{s_k}{r_k} = \frac{s_k}{W_k \log_2\left(1 + \frac{p_k G_k}{N_0 W_k}\right)}, \tag{3}$$

where s_k is the size of the policy parameters.

C. Energy Consumption Model

The energy consumed by a client encompasses the energy expended during local computation and the energy used during the policy parameter upload [22]. Typically, the energy utilized per CPU cycle is denoted by $r f_k^2$, with r representing the switching capacitance and f_k indicating the CPU frequency [27]. Hence, the energy required for local computation is given by:

$$E_k^{\text{comp}} = I_k C_k D_k r f_k^2, \tag{4}$$

where the product $I_k C_k D_k$ means the total CPU cycles for client k in each global round.

The energy expended to upload the policy parameter from client k to the BS is determined by multiplying the transmission power with the transmission duration, expressed as:

$$E_k^{\text{up}} = p_k \times t_k^{\text{up}} = \frac{p_k s_k}{W_k \log_2\left(1 + \frac{p_k G_k}{N_0 W_k}\right)}. \tag{5}$$

IV. Problem Formulation

In this section, we formulate our client selection problem in fault-tolerant FRL system for wireless IoT networks as follows:

$$\textbf{P0:} \quad \max_{x_k, \tau} \ \lambda \sum_{k \in \mathcal{K}} x_k - (1-\lambda) \sum_{k \in \mathcal{K}} x_k \left(E_k^{\text{comp}} + E_k^{\text{up}}\right) \tag{6}$$

$$s.t. \quad \|\mu_k - \mu_k^{\text{mom}}\| \, x_k \leq \mathfrak{T}_\mu, \ \forall k \in \mathcal{K}, \tag{7}$$

$$\left(t_k^{\text{comp}} + t_k^{\text{up}}\right) x_k \leq \tau, \ \forall k \in \mathcal{K}, \tag{8}$$

$$x_k \in \{0,1\}, \ \forall k \in \mathcal{K}. \tag{9}$$

The objective of problem **P0** in Eq. (6) is to optimize the tradeoff between maximizing the number of selected clients and minimizing the energy consumption of clients, where the weight parameter $\lambda \in [0,1]$ is a constant to adjust the weights of the two objectives. Eq. (7) identifies the malicious clients and ensures only the trustworthy clients are selected. Eq. (8) imposes each selected client's latency not to exceed the global iteration time τ. Eq. (9) indicates that x_k is a binary variable.

Note that problem **P0** is a mixed integer linear programming (MILP) problem, making it computationally intensive to achieve an efficient solution. We design an algorithm to address this problem with low computational complexity in the next section.

V. Proposed Algorithm

The complexity of problem **P0** is rooted in the global iteration time, τ, which intertwines the selection x_k of each client, as depicted in Eq. (8). In other words, if Eq. (8) is excluded, problem **P0** can be divided into K independent subproblems, where each determines the selection status of an individual client. As a result, the core strategy of our proposed algorithm aims to sever this linkage. We achieve this by enumerating τ based on any client's latency $t_k^{\text{comp}} + t_k^{\text{up}}$ (i.e., enumerating the possible slowest client). Then, we select the τ that achieves the maximum object value.

We assume that client i is the slowest one and so the global iteration time $\tau_i = t_i^{\text{comp}} + t_i^{\text{up}}$. The objective function in Eq. (6) is equivalent to $\sum_{k \in \mathcal{K}} [\lambda - (1-\lambda)(E_k^{\text{comp}} + E_k^{\text{up}})] x_k$. Then, problem **P0** can be separated into the following subproblems for each client k:

$$\textbf{P1:} \quad \max_{x_k} \ [\lambda - (1-\lambda)(E_k^{\text{comp}} + E_k^{\text{up}})] x_k \tag{10}$$

$$s.t. \quad \|\mu_k - \mu_k^{\text{mom}}\| \, x_k \leq \mathfrak{T}_\mu, \tag{11}$$

$$\left(t_k^{\text{comp}} + t_k^{\text{up}}\right) x_k \leq \tau_i, \tag{12}$$

$$x_k \in \{0, 1\}. \tag{13}$$

The candidate set $\mathcal{V}$ of problem **P1** is the clients that satisfy Eq. (11) and Eq. (12), i.e.,

$$\mathcal{V} = \{k \in \mathcal{K} : \|\mu_k - \mu_k^{\text{mom}}\| \leq \mathfrak{T}_\mu, \; t_k^{\text{comp}} + t_k^{\text{up}} \leq \tau_i\}. \tag{14}$$

To optimize the objective of problem **P1** as presented in Eq. (10), we examine the coefficient $\lambda - (1-\lambda)(E_k^{\text{comp}} + E_k^{\text{up}})$. If this coefficient is positive, we set $x_k = 1$; conversely, if it is negative, we assign $x_k = 0$. Therefore, the solution of problem **P1** can be described as:

$$x_k = \begin{cases} 1, & \text{if } \lambda - (1-\lambda)(E_k^{\text{comp}} + E_k^{\text{up}}) \geq 0, \; k \in \mathcal{V}, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

Our proposed algorithm is delineated in detail in Algorithm 1. From Lines 2-10, we iterate through each potential slowest client. During Lines 4-7, the client selection decision is determined for each client. In Lines 8-9, we compute the objective value associated with these client selection decisions. In Lines 11-12, the optimum solution is selected after comparing all objective values. Note that this algorithm functions within a single global iteration of the FRL training process, and the algorithm is repeatedly executed for client selection until the FRL process terminates.

Algorithm 1: Proposed Algorithm

Input : K, λ, E_k^{comp}, E_k^{up}, t_k^{comp}, t_k^{up}, μ_k, μ_k^{mom}, and $\mathfrak{T}_\mu$.
Output: x_k

1 Possible objective value vector $P = \emptyset$;
2 **for** $i = 1, ..., K$ **do**
3 Assume $\tau_i = t_i^{\text{comp}} + t_i^{\text{up}}$;
4 **for** $k = 1, ..., K$ **do**
5 Calculate the candidate clients $\mathcal{V}$ according to Eq. (14);
6 Calculate x_k according to Eq. (15);
7 **end**
8 Calculate the objective value p according to Eq. (6);
9 $P[i] = p$;
10 **end**
11 Choose i that achieves the largest $P[i]$;
12 Choose τ_i and corresponding x_k as the optimum solution;

VI. Performance Evaluation

In this section, we present simulations to assess the performance of our proposed algorithm, which we label as *Propose*. Our simulations are executed on a Dell towerstation equipped with an Intel Xeon(R) W-2245 CPU @ 3.90GHz (16 CPUs), 128GB RAM, and an NVIDIA Quadro RTX 6000/8000 GPU. For a comparative analysis, we incorporate three benchmark algorithms: *Time-Average*, *Attack-Aware*, and *Random*. The *Time-Average* method, inspired by [28], determines the global iteration time based on the average latency of all clients and then excludes those whose latencies exceed the global iteration time. Meanwhile, the *Attack-Aware* approach, inspired by [19], selectively excludes unreliable clients within the FRL system by the filtering model. Lastly, *Random* chooses clients at random during each global iteration.

In our simulation, there are $K = 10$ IoT devices uniformly distributed in a 2 *km* $\times$ 2 *km* area. The path loss between each IoT device and the BS follows $128.1 + 37.6 \lg d_k$, where d_k is the distance between client k and the BS. The transmission power $p_k = 1$ *W*, allocated bandwidth $W_k = 1$ *MHz*, and the noise power spectrum density $N_0 = -90$ *dBm/Hz*. The weight coefficient $\lambda = 0.65$. For simulating the environment of the IoT devices, we adopt the CartPole reinforcement learning setting [29]. In each global iteration, each client trains its local policy over $D_k = 500$ samples and the batch size $B = 16$. Each client runs 1 epoch for local training, i.e., $I_k = 1$. The required CPU cycles for training a sample C_k is randomly chosen from the uniform distribution $C_k \sim \mathcal{U}(3, 5) \times 10^5$. The CPU frequency f_k is randomly selected from 10 to 20 *GHz*, and the switch capacitance coefficient $r = 10^{-28}$. The parameters in the fault-tolerant model σ and δ are set as 0.06 and 0.6, respectively. The above parameters are consist with [19], [22]. Note that all the above parameters are default values and may be adjusted when we analyze their impact on the performance.

Fig. 2 illustrates the performance of our proposed algorithm and three benchmark algorithms with different numbers of clients ranging from 10 to 30. It can be observed that *Propose* achieves better performance than the other three benchmark algorithms, which is attributed to the efficient management of the trade-off between the number of chosen clients and energy consumption. This is because *Time-Average* method chooses the average latency of all clients as the global iteration time, discarding any client whose latency surpasses this average, which does not explore the optimum client selection. *Attack-Aware* filters out the unreliable clients but does not optimize the energy consumption. *Random* selects clients randomly and so does not achieve optimum. Besides, the performance of *Propose* increases as the number of clients grows because more clients means the number of selected clients can be higher, leading to a larger objective value.

Fig. 3 shows the return reward of our proposed algorithm and three benchmark algorithms within the context of the CartPole environment-based FRL system in different rounds (i.e., global iterations). We adopt the random noise attack

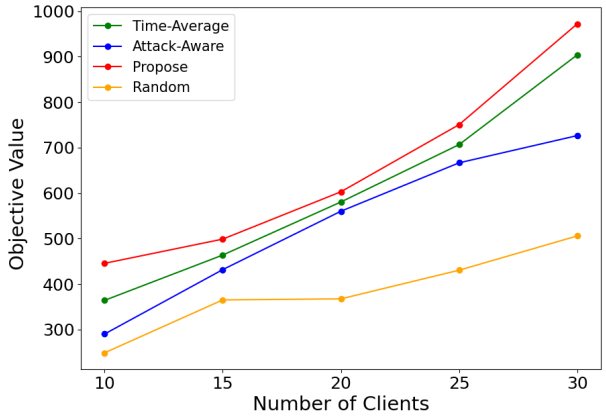

Fig. 2. Objective value vs number of clients.

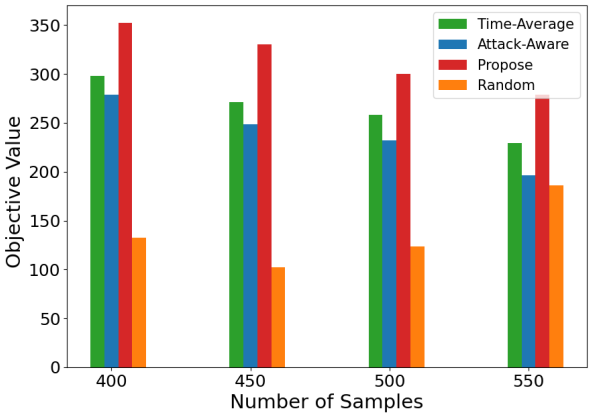

Fig. 4. Objective value vs number of samples.

where malicious clients intentionally transmit random noise data to the BS. *Time-Average* does not filter out malicious clients, causing a significant degradation in learning quality as these clients send random vectors to the BS. Similarly, *Random* selects clients randomly and offers no resistance to the noise, resulting in diminished rewards compared to *Propose* and *Attack-Aware*. *Propose* achieves a similar reward with *Attack-Aware* while it performs better in addressing the client selection problem, as shown in Fig. 2.

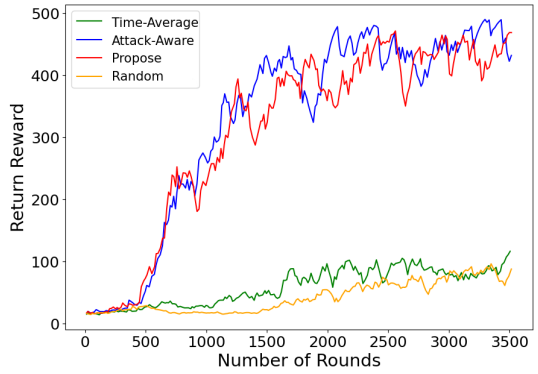

Fig. 3. Reward return vs number of rounds.

Fig. 4 compares the objective value of our proposed algorithm and three benchmark algorithms with different numbers of samples ranging from 400 to 550. A larger sample size leads to longer parameter uploading latency and higher energy consumption. Hence, fewer clients will be selected to satisfy the global iteration time constraint, leading to a smaller objective value. Therefore, the object value of all algorithms decreases as the number of samples increases, except for *Random*. Besides, our proposed algorithm *Propose* achieves a higher object value than other benchmark algorithms.

In Fig. 5, the impact of an increasing number of attacks on the objective value across all algorithms is depicted. As the number of attacks swells, both *Propose* and *Attack-Aware* see their objective values diminish. This is because they exclude a higher number of malicious clients when the number of attacks increases. Consequently, the number of selected clients declines, leading to a reduction in the objective value. Contrarily, the objective values of *Time-Average* and *Random* are not affected by the number of attacks, given their indifference to the presence of malicious clients. Similar to Fig. 2, *Propose* outperforms all benchmark algorithms.

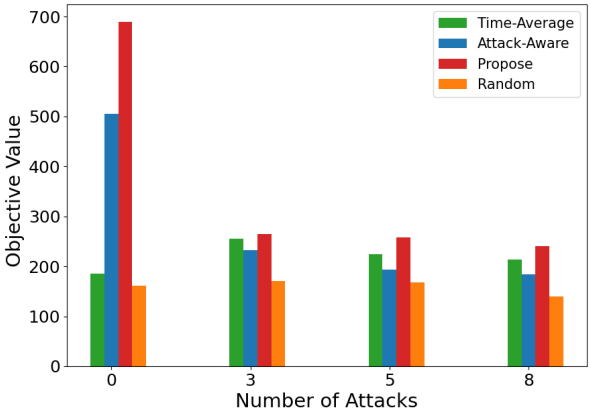

Fig. 5. Objective value vs number of attacks.

VII. Conclusion

In this study, we have investigated the client selection problem in the fault-tolerant FRL framework for wireless IoT networks to address challenges arising from client heterogeneity and potential malicious activities. We have formulated the client selection problem as an MILP model. An efficient algorithm with low computational complexity has been designed to address our problem. Through extensive simulations, we have demonstrated the robustness and superiority of our proposed solution.

REFERENCES

[1] J. Yao and N. Ansari, "Joint content placement and storage allocation in C-RANs for IoT sensing service," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1060–1067, 2019.

[2] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. Vincent Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622–1658, 2021.

[3] X. Sun and N. Ansari, "Edgeiot: Mobile edge computing for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, 2016.

[4] J. Yao and N. Ansari, "Enhancing federated learning in fog-aided IoT by CPU frequency and wireless power control," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3438–3445, 2020.

[5] S. Lee and D.-H. Choi, "Federated reinforcement learning for energy management of multiple smart homes with distributed energy resources," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 1, pp. 488–497, 2022.

[6] A. Jarwan and M. Ibnkahla, "Edge-based federated deep reinforcement learning for IoT traffic management," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 3799–3813, 2023.

[7] T. G. Nguyen, T. V. Phan, D. T. Hoang, T. N. Nguyen, and C. So-In, "Federated deep reinforcement learning for traffic monitoring in SDN-based IoT networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1048–1065, 2021.

[8] X. Ma, X. Sun, Y. Wu, Z. Liu, X. Chen, and C. Dong, "Differentially private byzantine-robust federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 3690–3701, 2022.

[9] M. Li, W. Wan, J. Lu, S. Hu, J. Shi, L. Y. Zhang, M. Zhou, and Y. Zheng, "Shielding federated learning: Mitigating byzantine attacks with less constraints," in *2022 18th International Conference on Mobility, Sensing and Networking (MSN)*, 2022, pp. 178–185.

[10] J.-H. Chen, M.-R. Chen, G.-Q. Zeng, and J.-S. Weng, "BDFL: A byzantine-fault-tolerance decentralized federated learning method for autonomous vehicle," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 8639–8652, 2021.

[11] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[12] S. Zhai, X. Jin, L. Wei, H. Luo, and M. Cao, "Dynamic federated learning for GMEC with time-varying wireless link," *IEEE Access*, vol. 9, pp. 10 400–10 412, 2021.

[13] S. Wang, M. Chen, W. Saad, and C. Yin, "Federated learning for energy-efficient task computing in wireless networks," in *2020 IEEE International Conference on Communications (ICC)*, 2020, pp. 1–6.

[14] J. Yao and X. Sun, "Energy-efficient federated learning in internet of drones networks," in *2023 IEEE 24th International Conference on High Performance Switching and Routing (HPSR)*, 2023, pp. 185–190.

[15] Q. Wu, Y. Zhao, Q. Fan, P. Fan, J. Wang, and C. Zhang, "Mobility-aware cooperative caching in vehicular edge computing based on asynchronous federated and deep reinforcement learning," *IEEE Journal of Selected Topics in Signal Processing*, vol. 17, no. 1, pp. 66–81, 2023.

[16] Q. Zhang, H. Wen, Y. Liu, S. Chang, and Z. Han, "Federated-reinforcement-learning-enabled joint communication, sensing, and computing resources allocation in connected automated vehicles networks," *IEEE Internet of Things Journal*, vol. 9, no. 22, pp. 23 224–23 240, 2022.

[17] F. Tang, C. Wen, L. Luo, M. Zhao, and N. Kato, "Blockchain-based trusted traffic offloading in space-air-ground integrated networks (sagin): A federated reinforcement learning approach," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 12, pp. 3501–3516, 2022.

[18] S. Islam, S. Badsha, I. Khalil, M. Atiquzzaman, and C. Konstantinou, "A triggerless backdoor attack and defense mechanism for intelligent task offloading in multi-uav systems," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 5719–5732, 2023.

[19] X. Fan, Y. Ma, Z. Dai, W. Jing, C. Tan, and B. K. H. Low, "Fault-tolerant federated reinforcement learning with theoretical guarantee," *Advances in Neural Information Processing Systems*, vol. 34, pp. 1007–1021, 2021.

[20] H. Zhang, Z. Xie, R. Zarei, T. Wu, and K. Chen, "Adaptive client selection in resource constrained federated learning systems: A deep reinforcement learning approach," *IEEE Access*, vol. 9, pp. 98 423–98 432, 2021.

[21] D. Qiao, S. Guo, D. Liu, S. Long, P. Zhou, and Z. Li, "Adaptive federated deep reinforcement learning for proactive content caching in edge computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 4767–4782, 2022.

[22] L. Yu, R. Albelaihi, X. Sun, N. Ansari, and M. Devetsikiotis, "Jointly optimizing client selection and resource management in wireless federated learning for internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4385–4395, 2021.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[24] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[25] D. Alistarh, Z. Allen-Zhu, and J. Li, "Byzantine stochastic gradient descent," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.

[26] J. Yao and N. Ansari, "Secure federated learning by power control for internet of drones," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1021–1031, 2021.

[27] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power cmos digital design," *IEICE Transactions on Electronics*, vol. 75, no. 4, pp. 371–382, 1992.

[28] F. Lai, X. Zhu, H. V. Madhyastha, and M. Chowdhury, "Oort: Efficient federated learning via guided participant selection," in *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2021, pp. 19–35.

[29] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE transactions on systems, man, and cybernetics*, no. 5, pp. 834–846, 1983.