A Communication-efficient Federated Learning Approach for Intrusion Detection in IoT

Oumaima ENNASRI, Manal ABBASSI, Yann BEN MAISSA, Rachid EL MOKADEM

National Institute of Posts and Telecommunications (INPT), STRS Laboratory, Rabat, Morocco
oumaimaennasri@gmail.com, abbassimanal.official@gmail.com, benmaissa@inpt.ac.ma, r.elmokadem@gmail.com

***Abstract*—With the rapid growth of the Internet of Things, Intrusion Detection Systems (IDSs) have become critical to protect sensitive data. Conventional approaches collect all data on a central server, which poses privacy and security issues. Federated Learning (FL) provides a decentralized alternative by enabling collaborative model training without sharing raw data. Despite this, standard FL methods such as FedAvg still suffer from high communication overhead due to the transmission of full model weights at each iteration, which is problematic in resource-constrained IoT environments. To address this problem, we build upon our previously proposed communication-efficient federated learning algorithm, eXtreme Federated Learning (XFL), and extend it to IoT IDSs by incorporating attack classification capabilities within the edge nodes. This work extends the original XFL with an IoT-oriented adaptation, attack detection, and hardware testbed evaluation. At each round, XFL optimizes and sends a *single* neural network layer per IDS client using the block coordinate gradient descent algorithm, thus lowering communication costs, a key factor in IoT applications. We evaluate both XFL and FedAvg on a hardware testbed (10 Raspberry Pi devices) using a real-world traffic dataset (UNSW-NB15), showing that XFL achieves a 60% reduction in communication overhead with a comparable detection accuracy. Our implementation is publicly available on GitHub.**

***Index Terms*—Federated Learning, Communication Efficiency, Internet of Things (IoT), Intrusion Detection Systems (IDS), Edge Computing**

I. Introduction

Context. The rapid growth of the Internet of Things has led to the generation of massive amounts of data in various network environments. Although this connectivity and data diversity enable transformative applications, they also introduce significant security challenges, particularly in protecting sensitive information against cyber threats. Intrusion Detection Systems play a pivotal role in securing IoT networks by detecting and mitigating potential attacks.

Motivation. Traditional Intrusion Detection Systems (IDSs) typically rely on centralized data gathering and processing, which raises significant privacy concerns and increases vulnerability to attack. Federated learning has emerged recently [1] as a decentralized machine learning paradigm that allows collaborative model training without revealing data, thus mitigating privacy concerns. However, standard Federated Learning methods such as Federated Averaging (FedAvg) also come with significant communication overhead via the transmission of full model weights at each iteration. This high overhead is particularly challenging in Internet of Things environments, where devices are limited in resources and networks are often bandwidth-constrained.

Contribution. To address these challenges, we propose a communication-efficient federated learning approach to intrusion detection in IoT networks. Building upon our previously introduced eXtreme Federated Learning (XFL) algorithm [2], developed by our team (El Mokadem, Ben Maissa, et al.), we extend it by incorporating attack classification capabilities directly on edge devices (distributed IDS). XFL transmits a single neural network layer from each device during updates enhancing communication efficiency in resource-constrained IoT environments. It is inspired by *Block Coordinate* gradient descent optimization [3], where only a subset of variables—here, a neural network layer—is optimized in each training iteration while the others remain fixed. We validate our method on a hardware testbed of 10 Raspberry Pi devices using the real-world UNSW-NB15 [4] dataset. The code is publicly available on GitHub [5]. Experimental results demonstrate a 60% reduction in communication overhead versus the standard FedAvg algorithm while maintaining comparable intrusion detection accuracy, which shows XFL is a promising solution for privacy-preserving IoT IDSs.

Contents. In Section II we present the Related Works. Section III introduces our approach, including the underlying XFL mathematical foundations, the deep learning models used, and the type of IDS. Section IV outlines our experimental setup, including the dataset and the client-server configuration. In Section V, we detail the results of our experiments and discuss them. Section VI concludes the paper.

II. Related Work

Extensive research has been conducted on FL-based IDSs, with several works looking into ways to improve scalability, privacy, and speed.

Mármol Campos et al. [6] examined FL-based IDS in the IoT domain, focusing on client participation, communication rounds, and comparing aggregation methods like FedAvg and Fed+ under varying client data distributions.

Ficco et al. [7] proposed a technique combining FL and TL to enable on-device training of machine learning models on IoT devices. Their approach showed enhanced model performance while protecting the privacy of the data. The potential of FL and TL to improve model performance on devices with limited resources is demonstrated in this work.

979-8-3503-9281-4/25/$31.00 ©2025 IEEE

Tanyildiz et al. [8] showed how successful FL is at predicting attacks using the UNSW-NB15 dataset. Their model achieved an accuracy of 99.93% and a sensitivity of 100%, showcasing the potential of FL in enhancing network security while preserving data confidentiality.

Sarhan et al. [9] proposed a Cyber Threat Intelligence Sharing Scheme based on Federated Learning for Network Intrusion Detection and adopted the NF-UNSW-NB15-v2 with NF-BoT-IoT-v2 datasets for their experiment to validate performance. Another study by Chunduru et al. [10] examined privacy-preserving intrusion detection through a comparative study between federated learning and traditional ML techniques, identifying the best approach to improve NIDS systems using the UNSW-NB15 dataset.

Marfo et al. [11] proposed a network anomaly detection system using federated learning. They tested a neural network model on the UNSW-NB15 dataset, achieving an accuracy of 97.21%, reducing training time compared to traditional centralized models.

While prior work has extensively explored intrusion detection in IoT environments, no study proposes a major reduction in communication overhead, and much of it has focused on simulations or assumed ideal hardware conditions. To address this gap, we target IDSs for IoT using edge devices such as Raspberry Pis, communication-efficient federated learning [2], and lightweight neural network classification adapted to edge/IoT constraints.

III. Proposed Approach

Our approach consists in building upon our eXtreme Federated Learning (XFL -subsection III-A) algorithm for collaborative training of lightweight neural networks (subsection III-C) on Raspberry Pi edge devices. XFL is inspired by a type of gradient descent called *block coordinate* descent optimization, where a subset of variables — here, a neural network layer — is optimized in each training iteration while others remain fixed. Thus, we transmit one layer from each device in each round, enabling the edge nodes to act as distributed IDSs (subsection III-B), while minimizing communication.

A. XFL Framework and Mathematical Formulation

eXtreme Federated Learning (XFL) takes part of its inspiration in the *Block Coordinate Descent* (BCD) optimization technique [12], which updates only a subset of variables (a block) at each iteration while keeping others fixed.

Given an objective function $f(x_1, \ldots, x_n)$, the BCD update for block x_i at iteration k is:

$$x_i^{k+1} = \arg\min_y f(x_1^k, \ldots, x_{i-1}^k, y, x_{i+1}^k, \ldots, x_n^k)$$

where x_i represents a block of parameters (e.g., a network layer), and f is the global loss function to minimize.

Our idea was to treat each neural network layer as a block. For a client k and layer i, the local update becomes:

$$W_{k,i}^{t+1} = W_{k,i}^t - \eta \nabla_i f_k(W_k^t)$$

where $W_{k,i}^t$ is the weight of layer i on client k at round t, η is the learning rate, and $\nabla_i f_k(W_k^t)$ is the gradient of the local objective with respect to layer i.

In the federated setting, only the selected layer l is transmitted by each client, and the server performs the aggregation as:

$$W_{\text{global},l}^{t+1} = W_{\text{global},l}^t - \eta \cdot \frac{1}{n_l} \sum_{k \in S_t} \Delta W_{k,l}$$

where $\Delta W_{k,l}$ is the update sent by client k, S_t is the set of clients selected in round t, and n_l is the number of updates received for layer l.

The overall XFL process consists of the following steps.

1. Global Initialization: The server initializes a global model W_{global} with L layers and sends it to all clients.

2. Local Training: Each client minimizes its local loss function

$$f_k(w) = \frac{1}{|D_k|} \sum_{i \in D_k} \ell(x_i, y_i, w)$$

using stochastic gradient descent (SGD), where D_k is the local dataset and ℓ is the loss on each data point.

3. Layer Selection and Update: In each round, clients update and transmit one selected layer $\Delta W_{k,l}$, typically chosen cyclically to ensure full model coverage over time.

4. Aggregation: The server aggregates the received updates for the selected layer using the formula above.

5. Redistribution: The updated global model is sent back to all clients. This process is repeated for $R \geq L$ rounds to ensure that all layers are updated at least once.

Per round, each client's forward and backward pass on a layer requires $O(m \cdot n)$ operations, while server aggregation over k clients requires $O(k \cdot m)$, where m and n are the layer dimensions.

B. Intrusion Detection Systems for the IoT

An IDS detects and responds to attacks within a computer system by identifying patterns that indicate unauthorized or malicious behavior [13]. In this paper, we are interested in network-based anomaly detection, a subset that identifies outliers from normal behavior within a system or network traffic [14]. Unlike other IDSs, network-based anomaly detection can identify new or unknown attacks, which makes it particularly appropriate for dynamic and changing environments like the IoT. The distributed configuration is especially powerful: each node (device) has a local model analyzing traffic and contributes to the shared global model.

Deploying such IDSs in Internet of Things (IoT) environments poses challenges due to the limited processing power of devices and strict data privacy requirements. This is why we employ XFL.

C. Neural Network Models and Training Details

1) Model Architecture

For the purpose of the IoT/edge device training, we use two lightweight neural network models to implement XFL [2] and FedML, which are MLP and RNN. Table II shows the layer configurations and the number of parameters for each model.

TABLE I
EXPERIMENTAL PARAMETERS

Parameter	Range/Values	Description
l (Layer updates)	1,All	Number of neural network layers for which updates are sent
E (Local epochs)	10	Number of training iterations per client before aggregation
B (Batch size)	64	Number of samples per gradient update
Lr (Learning rate)	0.001	Step size for gradient descent optimization
N (Nodes)	8, 9	Number of participating clients in FL training
c (Classes)	9 (iid), 9 (non-iid)	Number of target classes in the training data

TABLE II
MODEL ARCHITECTURES AND PARAMETER SIZES

Model	Layer Configuration	Number of Parameters	Approx. Size
MLP	2 Dense + Output Dense (Softmax)	∼53,641	∼210 KB
RNN	3 SimpleRNN + Output Dense (Softmax)	∼139,625	∼540 KB

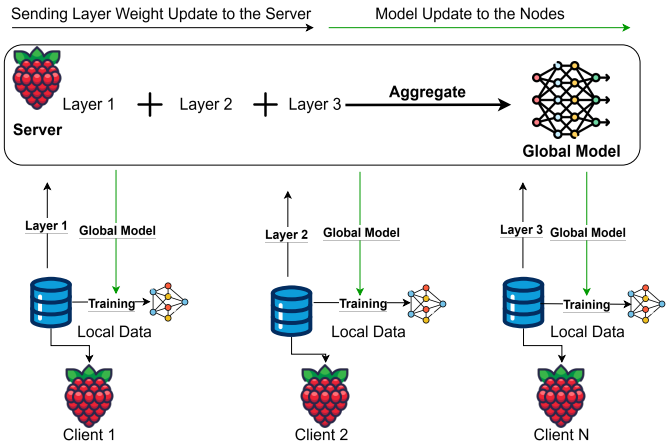

Fig. 1. Architecture of the IDS based on XFL

The MLP consists of two fully connected layers with 256 and 128 neurons respectively, and one output layer with 9 neurons to simulate the classes or attacks that we need to classify. The RNN model has three SimpleRNN layers with 256, 128, and 64 neurons respectively, and one output layer with a softmax activation function. These models are tested on both IID and non-IID data distributions using the XFL and FedML architectures.

2) Training Details

The parameters used are summarized in Table I. For the learning rate, we applied grid search and found that a value of 0.001 is the best, giving the highest accuracy. For the batch size, after empirically testing different values, we set it to 64. For the number of epochs, we chose 10 based on the trade-off between model performance and training time. The architectural details and parameter sizes for each model are given in Table II.

IV. EXPERIMENTAL AND SYSTEM SETUP

A. Hardware testbed and architecture

For the experiments, we used a total of 10 Raspberry Pi 4 devices in our testbed setup, which are widely available in IoT applications. In this paper, we implement two testbeds (see Fig. 2): one for running the MLP model using 9 Raspberry Pis - 1 as a server and 8 as clients — and another for the RNN model using all 10 devices — 1 as a server and 9 as clients. The detailed specifications of the Raspberry Pi 4 Model B are shown in Table III.

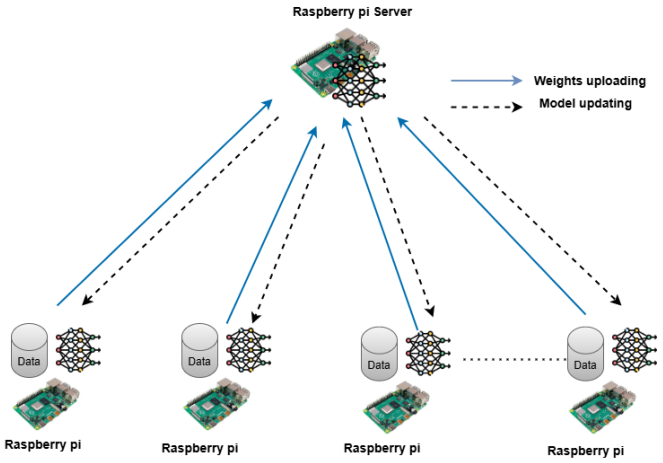

Fig. 2. IoT Testbed Architecture

TABLE III
LIST OF HARDWARE COMPONENTS

Component	Model / Description
Microcontroller	Raspberry Pi 4 Model B (1 GB RAM)
microSD Card	32 GB, Class 10 (Raspberry Pi OS)
Power Supply	5V / 3A USB-C (official adapter)
Networking	Wi-Fi

B. Dataset and preprocessing

We use the UNSW-NB15 dataset [4] for the training and testing. Released in 2015, the dataset contains a total of 2,540,044 records to detect and evaluate network intrusion. We believe this specific dataset is comprehensive; it contains

49 features per record, which fall into five data types: binary, timestamp, float, integer, and nominal. Basic features, content features, flow features, time features, labeled features, and additional generated features were the six categories into which the acquired features were separated.

After preprocessing our data, balancing classes, and selecting features while considering IoT device resources, it still contains 20 features, as shown in Table IV. And as the target labels of our model ($attack\_cat$), there are different categories for the classification of network traffic, as follows.

- **Normal**: Legitimate network traffic with no malicious activity.
- **Generic**: Attacks exploiting cryptographic weaknesses across multiple protocols.
- **Exploits**: Attacks leveraging system vulnerabilities for unauthorized access.
- **DoS (Denial of Service)**: Overloading a system or network to disrupt services.
- **Fuzzers:** Sending malformed inputs to uncover security flaws.
- **Reconnaissance**: Gathering network information for future attacks.
- **Analysis**: Inspecting network traffic to extract sensitive data.
- **Worms**: Self-replicating malware spreading via vulnerabilities.
- **Backdoor**: Hidden access points bypassing authentication.

C. Frameworks and Software Stack

The frameworks used in this research were TensorFlow and Flower [15]. The TensorFlow framework was used to define and train DL models, which are neural networks designed for intrusion detection.

Flower is a more recent framework designed to create a client-server architecture for federated learning training and communication between distributed clients. It uniquely supports heterogeneous clients across different ML frameworks (including TensorFlow and PyTorch) and also includes an RPC client, a federated learning loop, and support for libtorch C++ for high-performance deployment. The full implementation of our system is publicly available at [5].

D. Client and Server Configuration

Each FL client is deployed on a Raspberry Pi 4, simulating real-world edge devices with constrained resources. Clients are responsible for local data handling, training, and communication with the server. Each client receives a specific subset of the dataset, and the number of samples assigned varies depending on the model used, simulating different workload distributions. Clients perform local training using TensorFlow for a predefined number of epochs before sending updates to the server. We vary several hyperparameters, such as batch size, learning rate, and number of local epochs (Table I). Additionally, clients may send only partial updates of the model layers depending on the configuration (i.e., updating just the last layer or the full model).

All communication between clients and the server takes place over a local network, providing a controlled environment to measure latency, model accuracy, and communication efficiency.

Regarding the federated learning server configuration, we simply use another Raspberry Pi 4 for this role.

V. Simulation Results and Discussion

In this section, we present the results of each model on a specific number of nodes (Raspberry Pi), while also discussing the implications. Each model has specific parameters (Table I) that must be respected in order to adapt it to both approaches, XFL and FedAvg.

A. Simulation Results and Analysis

We provide in Fig. 3 the evolution of accuracy per aggregation round for XFL and FedAvg under both IID and non-IID data using MLP and RNN models. We observe that, in the IID scenario where each client has the same number of samples for all classes, FedAvg and XFL achieve approximately the same accuracy (Table V) with a slight difference. FedAvg always performs well, but it requires transmitting the entire model in each round, leading to communication overhead. Despite this, XFL achieves competitive accuracy by transmitting only one layer in each round. This highlights the communication efficiency of the algorithm. For the non-IID scenario, where each client has samples for only some classes and the data is not balanced, we observe that FedAvg achieves better accuracy than XFL. This can be attributed to the greater sensitivity of XFL to data heterogeneity, as the partial updates exchanged may not capture the full model dynamics under highly skewed distributions.

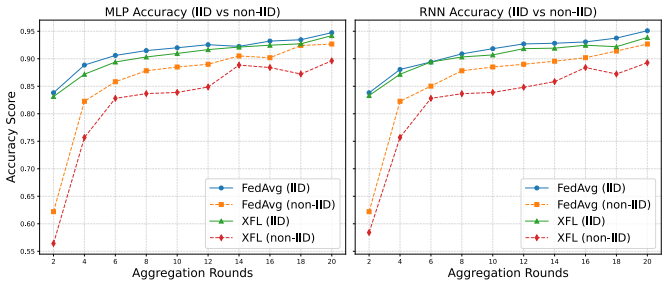

Fig. 3. Accuracy score comparison for RNN and MLP

Fig. 4 compares the performance of FedAvg and XFL. It illustrates the relationship between global accuracy and cumulative bandwidth consumption for the RNN model under both IID and non-IID scenarios. We observe a consistent increase in data exchanged over communication rounds.

XFL is more efficient in terms of communications in both IID and non-IID scenarios. As shown in Figure 4, it achieves approximately 93.3% global accuracy by transmitting only 516.7 MB, compared to FedAvg, which scores 94.4% with 1241 MB. In the non-IID case, although FedAvg achieves higher accuracy (91.8% vs. 87.1% for XFL), our approach remains competitive while significantly reducing bandwidth usage.

TABLE IV
DESCRIPTION OF INPUT FEATURES.

Feature	Description	Type
dur	Connection duration	Numeric
service	Type of service	Categorical
sbytes	Bytes sent from source	Numeric
dbytes	Bytes received at destination	Numeric
rate	Packet transfer rate	Numeric
dload	Download load	Numeric
dinpkt	Packets received at destination	Numeric
djit	Jitter at destination	Numeric
stcpb	First TCP sequence number sent	Numeric
dtcpb	First TCP sequence number received	Numeric
dwin	TCP window size at destination	Numeric
ackdat	Acknowledgment data in TCP	Binary
smean	Mean source packet size	Numeric
dmean	Mean destination packet size	Numeric
trans_depth	Transaction depth	Numeric
response_body_len	HTTP response body size	Numeric
ct_state_ttl	Count of same state and TTL connections	Numeric
ct_dst_ltm	Count of connections to same destination	Numeric
ct_ftp_cmd	Count of FTP commands	Numeric
ct_flw_http_mthd	Count of HTTP methods used	Numeric
attack_cat	Attack category (if any)	Categorical

TABLE V
EXPERIMENTAL RESULTS FOR DIFFERENT MODELS ON THE UNSW-NB15 DATASET

Approach	Model	Dataset	Clients	Layers	Epochs	Accuracy	Data Sent/Round (MB)	Local Training Time (s)	CPU Usage (%)	RAM Usage (MB)
FedML	MLP	UNSW-NB15 (IID)	10	2	10	94.7	73.00	46.9	19.81	56.43
	MLP	UNSW-NB15 (non-IID)	10	2	10	92.54	25.05	46.66	23.59	53.11
XFL	MLP	UNSW-NB15 (IID)	10	2	10	94.2	11.00	39.34	20.10	58.02
	MLP	UNSW-NB15 (non-IID)	10	2	10	89.62	3.77	39.29	23.53	51.35
FedML	RNN	UNSW-NB15 (IID)	8	4	10	95.09	62.06	113.22	46.09	62.53
	RNN	UNSW-NB15 (non-IID)	8	4	10	92.63	12.71	112.59	32.67	59.40
XFL	RNN	UNSW-NB15 (IID)	8	4	10	93.9	25.84	101.51	51.55	60.19
	RNN	UNSW-NB15 (non-IID)	8	4	10	89.27	9.97	101.53	33.22	60.00

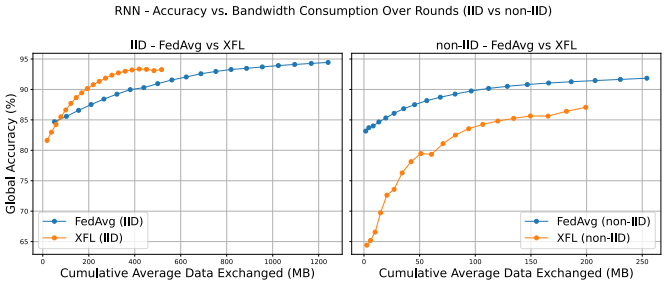

Fig. 4. Bandwidth consumption over rounds for RNN

Fig. 5 illustrates the same comparison for the MLP model, and similar patterns emerge. While XFL achieves acceptable accuracy levels more quickly and with considerably lower bandwidth usage, FedAvg leads to significantly greater data exchange. In the IID scenario, XFL reaches approximately 93.5% global accuracy by exchanging only 220 MB, compared to 1460 MB for FedAvg to reach 94.2%. In the non-IID case, although FedAvg ultimately achieves slightly higher accuracy (91.8% vs. 89.2% for our approach), we maintain a competitive performance while consuming much less bandwidth (75.3 MB versus 501 MB).

FedAvg consistently incurs higher bandwidth consumption than XFL in both IID and non-IID scenarios. This can be explained by the significant communication overhead caused by the fact that FedAvg transmits all model parameters in each cycle.

While XFL does not reach the same accuracy as FedAvg, it achieves reasonably high accuracy using substantially less bandwidth. This efficiency is particularly notable in the non-IID setting, where XFL shows practical performance with only a fraction of the data exchanged.

These findings reinforce the communication efficiency of XFL, which supports high precision when utilizing much less bandwidth compared to FedAvg, especially in scenarios involving non-IID data distributions. This balance makes the approach particularly suitable for bandwidth-constrained or limited-resource environments.

B. Discussion and insights

Several insights into the performance trade-offs between the proposed XFL approach and FedAvg are provided by the experimental results presented in Table V and Figures 4 and 5.

First, we observe that FedAvg achieves a slightly higher global accuracy but with greater communication overhead cost. By sending all model layers in each iteration, FedAvg leads to bandwidth usage up to six times higher than XFL (e.g., MLP model under IID, 1460MB vs. 220MB). By transmitting only one layer per round, XFL reduces the communication footprint while preserving competitive accuracy levels (e.g., 93.5% vs. 94.2% for MLP).

Second, in IID scenarios where data distributions among clients are balanced, XFL consistently approximates FedAvg's performance with minimal accuracy loss (less than a 1% gap). This shows that the key dynamics of the global model may be captured by sharing only partial updates for well-balanced data without significant degradation in performance. As a result, XFL promises to be a solution for environments with constrained communication resources, such as IoT edge networks.

Third, FedAvg typically performs better than XFL in non-IID data distribution, where data heterogeneity creates difficulties for model convergence. For instance, FedAvg achieves 91.8% accuracy in the RNN model, while XFL reaches 87.1%. This gap is due to the partial updates, which capture only a few patterns in non-IID data. However, XFL still performs well in reducing bandwidth, cutting total data transmission by almost ten times (e.g., 75.3MB vs. 501MB for MLP).

This highlights how XFL can perform well in IoT environments, preserving communication and energy efficiency.

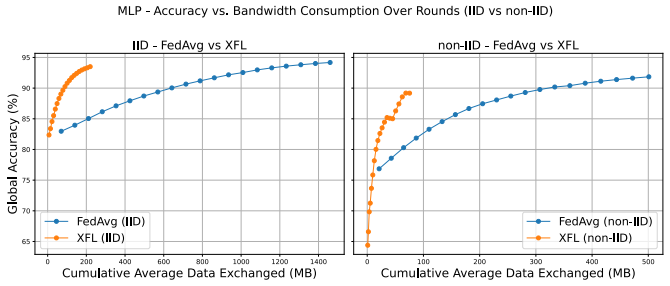

Fig. 5. Bandwidth consumption over rounds for MLP

VI. Conclusion

In this paper, we address the challenge of securing IoT environments using decentralized learning techniques to detect anomalous network behavior, with a focus on communication efficiency and energy consumption. We propose an approach for intrusion detection in the IoT, building upon our previous algorithm, eXtreme Federated Learning (XFL) [2]. We deploy deep learning-based IDS models directly on a testbed of 10 edge devices (Raspberry Pi) using the real-world UNSW-NB15 dataset. Inspired by block coordinate gradient descent, XFL transmits only a single model layer at each iteration, significantly reducing communication overhead while maintaining comparable detection accuracy.

Experimental results show a 60% reduction in communication costs compared to standard FedAvg. These findings underscore the importance of using communication-efficient federated learning in resource-constrained IoT environments to optimize resource and energy usage.

Future work will explore more complex IoT scenarios and further decentralization strategies to enhance both privacy and performance.

References

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

[2] R. El Mokadem, Y. Ben Maissa, and Z. El Akkaoui, "extreme federated learning (xfl): a layer-wise approach," *Cluster Computing*, vol. 27, no. 5, pp. 5741–5754, 2024.

[3] E. G. Birgin and J. M. Martínez, "Block coordinate descent for smooth nonconvex constrained minimization," *Computational Optimization and Applications*, vol. 83, no. 1, pp. 1–27, 2022.

[4] 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, pp. 1–6.

[5] O. Ennasri, M. Abbassi, Y. Ben Maissa, and R. El Mokadem, "XFL_IDS: Federated intrusion detection system implementation," https://github.com/yannbms/XFL_IDS_INPT, 2025, accessed: 2025-05-31.

[6] E. M. Campos, P. F. Saura, A. González-Vidal, J. L. Hernández-Ramos, J. B. Bernabe, G. Baldini, and A. Skarmeta, "Evaluating federated learning for intrusion detection in internet of things: Review and challenges," *Computer Networks*, vol. 203, p. 108661, 2022.

[7] M. Ficco, A. Guerriero, E. Milite, F. Palmieri, R. Pietrantuono, and S. Russo, "Federated learning for iot devices: Enhancing tinyml with on-board training," *Information Fusion*, vol. 104, p. 102189, 2024.

[8] H. Tanyıldız, C. B. Şahin, and Ö. B. Dinler, "Federated learning for attack prediction on unsw-nb15 training data," *NATURENGS*, vol. 5, no. 2, pp. 1–7, 2024.

[9] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "A cyber threat intelligence sharing scheme based on federated learning for network intrusion detection," *Journal of Network and Systems Management*, vol. 31, no. 3, 2022.

[10] S. Chunduru *et al.*, "Privacy-preserving network intrusion detection using federated learning," *IEEE Access*, vol. 11, pp. 1–13, 2023.

[11] W. Marfo, D. K. Tosh, and S. V. Moore, "Network anomaly detection using federated learning," in *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*. Rockville, MD, USA: IEEE, 2022, pp. 1–13.

[12] S. J. Wright, "Coordinate descent algorithms," *Mathematical programming*, vol. 151, no. 1, pp. 3–34, 2015.

[13] A. Heidari and M. A. Jabraeil Jamali, "Internet of things intrusion detection systems: a comprehensive review and future directions," *Cluster Computing*, vol. 26, no. 6, pp. 3753–3780, 2023.

[14] NordVPN, "Anomaly-based detection," 2025, accessed: 2025-05-18. [Online]. Available: https://nordvpn.com/cybersecurity/glossary/anomaly-based-detection/

[15] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão *et al.*, "Flower: A friendly federated learning research framework," *arXiv preprint arXiv:2007.14390*, 2020.

[16] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705121000381

[17] Helixstorm, "Types of intrusion detection systems: Nids vs hids," 2020, accessed: 2025-05-10. [Online]. Available: https://www.helixstorm.com/blog/types-of-intrusion-detection-systems

[18] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated learning for intrusion detection system: Concepts, challenges and future directions," *Computer Communications*, vol. 195, pp. 346–361, 2022.

[19] E. M. Maseno, Z. Wang, and H. Xing, "A systematic review on hybrid intrusion detection system," *Security and Communication Networks*, vol. 2022, no. 1, p. 9663052, 2022.