Hierarchical Federated Learning for Collaborative IDS in IoT Applications

Hassan Saadat[1], Abdulla Aboumadi[1], Amr Mohamed[1], Aiman Erbad[2], Mohsen Guizani[1]
[1]Department of Computer Science and Engineering, Qatar University, Doha, Qatar
[2]College of Science and Engineering, Hamad Bin Khalifa University, Qatar
Email:{hs1608346, aa1514129, amrm}@qu.edu.qa, and{aerbad, mguizani}@ieee.org.

***Abstract*—As the Internet-of-Things devices are being very widely adopted in all fields, such as smart houses, healthcare, and transportation, extremely huge amounts of data are being gathered, shared, and processed. This fact raises many challenges on how to make the best use of this amount of data to improve the IoT systems' security using artificial intelligence, with taking into consideration the resource limitations in IoT devices and issues regarding data privacy. Different techniques have been studied and developed throughout the years. For example, Federated Learning (FL), which is an emerging learning technique that is very well known for preserving and respecting the privacy of the collaborating clients' data during model training. Therefore, in this paper, the concepts of FL and Hierarchical Federated Learning (HFL) are evaluated and compared with respect of detection accuracy and speed of convergence, through simulating an Intrusion Detection System for Internet-of-Things applications. The imbalanced NSL-KDD dataset was used in this work. Despite its infrastructure overhead, HFL proved its superiority over FL in terms of training loss, testing accuracy, and speed of convergence in three study cases. HFL also showed its efficiency over FL in reducing the effect of the non-identically and independently (non-iid) distributed data on the collaborative learning process.**

***Index Terms*—Internet of things, Federated learning, Hierarchical federated learning, intrusion detection, Imbalanced data**

I. INTRODUCTION

As the adoption of internet and technology is drastically growing in all domains today, the concept of Internet-of-Things (IoT) is broadly existing. As per the study in [1], the world has an estimated 9 billion IoT devices deployed in 2020, and that number is projected to reach over 25 billion within the coming ten years. Generally, any device that has network connectivity is considered to be as part of the IoT world. Therefore, to define IoT, it can be thought of as the wireless inter-connected system of heterogeneous devices that gather, process, and share data with the minimum need for human beings' intervention [2]. An example of an IoT device can come in very different and unrelated forms, where it can vary from being a smart house device, such as a furnace or a fridge, through being part of a bigger system, such as a camera in a surveillance system or an electronic chip in a vehicular system, to being an electronic device, such as a cell phone or laptop, or a medical device implanted inside the body of a patient. Based on the examples given, it can be obvious that IoT devices are deployed in diverse types of environments, including residential buildings, transportation, healthcare, agriculture, and many other systems. Therefore, most IoT devices have general common features of being interoperable, mobile, and continuously connected to the network. However, that comes with the burden of those devices having constraints related to resources like power supply, memory size, and processing and computational power [3].

In general, IoT end-devices tend to share a lot of information and data with nearby devices, edge nodes, or main servers. That extensive exchange of data sometimes happens over long ranges and unsecure networks. Depending on the capabilities of the hardware components built-in inside IoT devices, some of them use 3G or 4G, which are highly energy-consuming, while others communicate using lighter communication protocols, such as ZigBee, Bluetooth, and RFID, which are more resource-efficient, but with trade-offs between performance metrics, such as energy consumption, communication range, and data rate [4]. That inconsistency of the networking protocols used in IoT networks, alongside with the weak infrastructure, and the low computational capabilities introduce severe security challenges [5]. That makes the network layer vulnerable to different types of intrusion attacks. Those attacks may compromise the confidentiality and integrity of data, and the availability of devices and services in highly sensitive environments, which can result in disastrous consequences. Therefore, adopting security countermeasures such as Intrusion Detection Systems (IDS) in IoT applications is of extreme necessity.

Due to the discussed resource limitations found in IoT devices, implementing an IDS on each IoT device is a challenging task. IDS implementation techniques that use machine and deep learning algorithms and take into consideration the issues related to energy consumption, data privacy, and non-iid (non-identically and non-independently distributed) data are required. Three of the main techniques are:

A. Centralized Learning (CL)

Centralized learning, Fig. 1.a), requires having a main cloud that contains a learning model, where the data fed to that model are the ones gathered from the connected IoT devices. Although this type of learning ensures higher learning accuracy than other models due to the large amount of data that it trains itself on, it suffers from several disadvantages [6]. One of its major drawbacks is the lack of data privacy, where end-clients must send their private data, such as personal, financial, medical, and other types of sensitive information to a main server.

B. Federated Learning (FL)

In a simple FL scenario, Fig. 1.b), the principle is to make the IoT devices collaborate in the learning process. That is achieved

978-0-7381-3361-4/21/$31.00 ©2021 IEEE

2021 10th Mediterranean Conference on Embedded Computing (MECO) | 978-1-6654-3912-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/MECO52532.2021.9460304

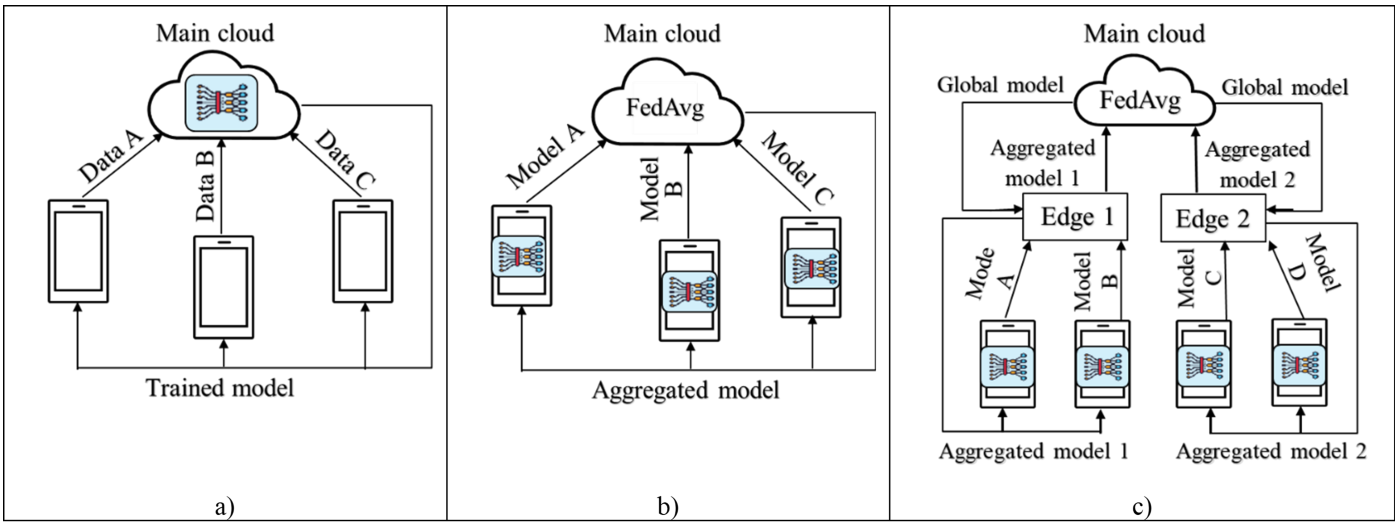

Fig. 1. High-level view of a) Centralized Learning, b) Federated Learning, and c) Hierarchical Federated Learning

by the server sending initial model weights to the clients. Then, the clients train their local learning models using the initial weights received from the server on their respective data. After that, the server aggregates the trained weights received from the clients and sends a global model back to the clients, and the process repeats. Theoretically, FL should be able to ensure the privacy of the local data of the clients. This is because the server and clients only communicate model weights instead of their private data [6]. Also, the speed of convergence of the learning process can be controlled by tuning a hyperparameter 'K', that determines how many local learning epochs the clients should do before sending their weights to the server. FL is also thought to be very efficient when the data on the clients are non-iid, where each client will benefit from previously unseen other clients' knowledge. However, that comes with the burden of reducing the speed of convergence of the aggregated model's loss and accuracy.

C. Hierarchical Federated Learning (HFL)

In HFL, at least one layer of edges is added between the clients and the main cloud. An HFL system consists of small FL instances, where each edge and its directly attached clients can be thought of as one instance of FL. To illustrate, in Fig. 1.c), each of Edge 1 and its clients A and B, Edge 2 and its clients C and D, and the main cloud and its clients Edge 1 and Edge 2, form one instance of FL. Just like FL, the clients under each edge collaborate in the learning process and their weights get aggregated at the edge level, and after a specific number of client-edge communication rounds, the weights present on the edge level get aggregated on the main cloud to form a global model, and the process repeats. It is clear that, just like FL, the privacy of the data of the clients is preserved.

D. Power Consumption and Practical Issues

CL, FL, and HFL were discussed in the previous subsections in terms of the learning steps and their effect on preserving the local data privacy. However, practical issues, such as the power consumption and communication efficiency of the different techniques should be highlighted. In CL, high energy and network bandwidth are consumed by IoT devices due to the large amount of data that they have to share with the cloud. Also, a high delay in delivering the data is expected due to the very far distance between the cloud and its clients [6]. Despite their identical infrastructure, FL tends to have less communication cost than CL, which is true because the clients and the cloud only communicate model weights rather than large amounts of raw data. Power consumption of IoT devices can be further controlled in FL and HFL by tuning the communication frequency parameter 'K'. The work of [7] proved that a high value of K can reduce the energy exerted by the less local computations performed, however, that increases the energy consumed by the frequent model weights exchange, therefore, a balanced trade-off is required. In addition, despite its infrastructure overhead and complexity of assigning clients to edges arbitrarily, one of the main advantages of HFL over FL is that the distance between the edge and its clients is always shorter than the distance between the main cloud and the clients, which results in better communication efficiency, unlike FL, where the probabilities of delay and communication disconnection are higher [7].

In this paper, based on the background given above, the objectives are as follows:

- Investigate the differences between the performances of FL and HFL in implementing an IDS using NSL-KDD dataset [8].
- Study the effect of adding edge layers to the FL scheme on the training loss, testing accuracy, and their speed of convergence.
- Observe how changing client-edge assignment in HFL can reduce the effect of the non-iid data distribution over the clients.

II. LITERATURE REVIEW

Many researchers have attempted implementing IDS using machine and deep learning techniques. A comparison between

the performance of K-Means and fuzzy C-Mean clustering on NSL-KDD dataset was done in [9]. A comprehensive study on the implementation of IDS over different imbalanced datasets, such as KDDCup 99, NSL-KDD, UNSW-NB15, and WSN-DS was conducted by [10]. A variety of other machine and deep learning techniques were applied for IDS on NSL-KDD and other datasets [11]–[13]. Most of the previously mentioned approaches tackled the problem from a single node's point of view that sees the whole data, not as FL or HFL. DeepFed [14] implemented an FL approach for IDS for Cyber-Physical Systems (CPS) using industrial CPS datasets. In [15], a comparison between Centralized, Self, and Federated Learning on NSL-KDD using real implementation on raspberry pi was conducted. An enhanced FL algorithm for intrusion detection was developed by [16], and it was tested on a dataset that is a combination of KDDCUP 99, CICIDS2017, and WSN-DS datasets.

FL was modified by adding a hierarchical clustering step, where clients with similar local updates get clustered together and trained independently [17]. Their work proved its efficiency on the graphical MNIST and FEMNIST datasets. A comparison between FL, edge-based FL, and HFL was conducted on MNIST and CIFAR-10 datasets [7]. A study by [18] implemented HFL with optimal client-edge assignment. Their work showed positive results on MNIST, FashionMNIST, and CIFAR-10 datasets.

In this paper, a comparison, in terms of training loss, testing accuracy, and their speed of convergence at different network topology constraints, between FL and HFL on the NSL-KDD dataset will be conducted.

III. LOSS FUNCTION ESTIMATION

The widely used cross entropy loss function is used in this paper. Since the NSL-KDD dataset consists of 5 classes, therefore, the loss function is calculated as a categorical cross entropy loss function.

A. Centralized Learning

In this case, the whole data is gathered and the learning is happening at the main cloud. Therefore, the loss function is simply calculated as [19]:

$$f_{CL}(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w) \tag{1}$$

$$f_i(w) = -\sum_{j=1}^{C} y_j \cdot \log\left(\hat{y_j}\right) + (1-y_j)\cdot\log\left(1-\hat{y_j}\right) \tag{2}$$

Where w is the model weights, n is the total number of samples, C is the number of classes, y_j is the true probability of class j for sample i (0 or 1), $\hat{y_j}$ is the predicted probability of class j for sample i, and $f_i(w)$ is the loss at data sample i.

B. Federated Learning

In the case of FL, (1) is repeated for each client, and they are summed and weight-averaged at the main cloud. Therefore, the loss function of FL is calculated as [19]:
At the cloud:

$$f_{FL}(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \tag{3}$$

Where at client k:

$$F_k(w) = \frac{1}{n_k}\sum_{i\in P_k} f_i(w)$$

Where K is the number of clients, n_k is the number of samples given to client k, P_k is the set of the indexes of the samples given to client k, and f_i is the same as (2).

C. Hierarchical Federated Learning

In the case of HFL with one layer of edges, to calculate the loss function of the main cloud, a weighted average is implemented on the losses of the edges:

$$f_{HFL}(w) = \sum_{e=1}^{E} \frac{n_e}{n} F_e(w)$$

Where E is the number of edge nodes, n_e is the number of samples given to edge e, and $F_e(w)$ is the loss calculated at edge e in exactly the same way as (3).

IV. SYSTEM SETTINGS

In this section, the characteristics of the dataset and neural network used are presented.

A. NSL-KDD [8]

The NSL-KDD has been very widely used in IDS researches previously [11]. That is due to its sufficiency in terms of number of data records and attack types diversity. It contains 125,973 training samples and 22,544 testing samples. Each of the 148,517 samples consist of 41 features. The dataset contains normal and abnormal classes. The training dataset contains 39 attacks categorized to Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L) and Probe attacks. The testing set, besides the training set attacks, contains 17 extra attacks to assess the flexibility of the model in detecting unseen-before attacks. In order to understand the upcoming performance evaluation study cases, it is important to highlight the percentage distribution of the classes in the training and testing sets, which are shown in Fig. 2.

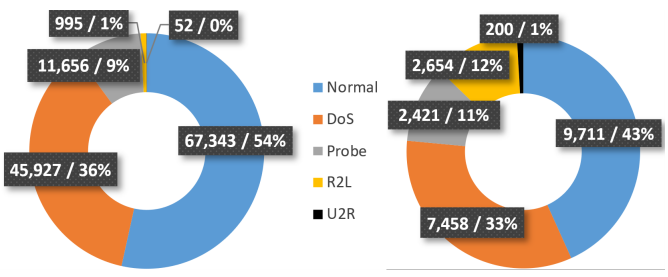

Fig. 2. Class distribution in NSL-KDD training set (left) and testing set (right)

B. Dataset Preprocessing and Neural Network (NN)

Since the neural network is a mathematical model, it can only be fed with numbers. Therefore, one-hot encoding [20] is used to convert the non-numeric features, which are protocol type, service, and flags, into numeric ones. The one-hot encoding results in each data sample having 122 features. Then, the values of features are normalized to be between 0 and 1 to avoid any inconsistency. Based on the number of features, our NN, Fig. 3, is composed of 122-neuron input layer, and two hidden layers of 80 and 40 neurons, respectively, and 5-neuron output layer. It also performs neuron-dropout of 30% after each of the hidden layers, and it uses ReLU as an activation function.

V. PERFORMANCE EVALUATION AND DISCUSSION

In this section, the FL and HFL systems will be evaluated. The number of learning epochs per communication round 'K' is constant and set to 5 for the client-cloud layer in FL and edge-cloud layer in HFL. However, in the client-edge layer in HFL, FedSGD [19] is implemented, which is equivalent to 1 learning epoch per communication round, i.e., the clients send their local models to the edge following every single training epoch. Different study cases of training data distribution will be evaluated as follows:

A. IID Client-Edge Assignment

In this case, the FL scenario consists of 8 clients, where every 2 clients share one attack class samples in half, i.e., each one of client 1 and client 2 has one half of the DoS attack samples, each one of client 3 and client 4 has one half of the Probe attack samples, and so on. The normal samples are divided among the clients based on the portion of attack samples that they have, e.g., client 1 has one half of the DoS attack samples, which is around 39% of all the attack samples, therefore it gets 39% of the normal samples. This strategy of the normal samples' distribution applies to cases B. and C.

In the HFL scenario, two edges are put in the middle, and each edge gets four clients that have completely non-iid data.

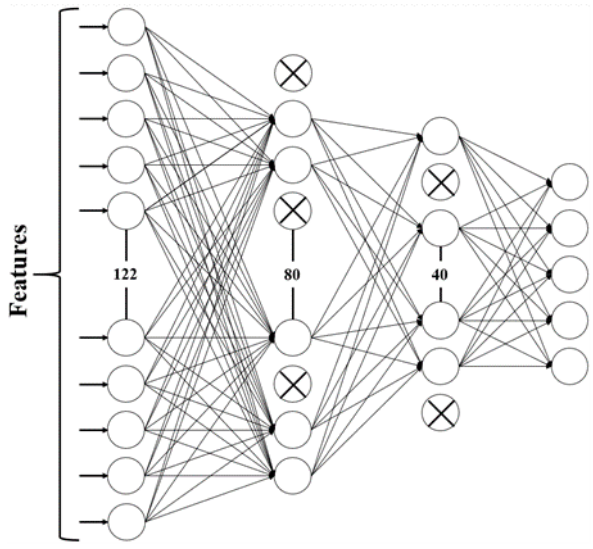

Fig. 3. Our neural network diagram

For example, edge1 gets client 1 (Dos attack), client 3 (Probe attack), client 5 (R2L attack), and client 7 (U2R attack), and edge 2 gets the other four clients. As a result, the two edges have iid data and can see all types of attacks.

The FL scenario suffers from the non-iid distribution and the inconsistency in clients' training data. That fact, as seen in Fig. 4, affected the training loss, testing accuracy, and the speed of convergence. Whereas in the HFL scenario, the iid client-edge assignment enhanced the performance in terms of all of the three metrices. This is highlighted through showing the loss which is almost negligible, the detection accuracy which is better than FL with 3-4% difference, and the convergence state which is reached around 19 rounds faster than FL as illustrated by the black line.

B. IID Client-Edge Assignment with Equal Number of Attacks' Samples

In this case, the FL scenario also consists of 8 clients. Each one of client 1 and client 2 has 500 randomly chosen DoS samples and 500 randomly chosen normal samples (each client has its own distinct set of samples). Each one of client 3 and client 4 has 500 randomly chosen Probe samples and 500 randomly chosen normal samples, and so on for the other two attacks and four clients. Of course, as seen in Fig. 2 , U2R has only 52 samples, therefore, its samples were Naively duplicated until they reached a total of 1000 samples. Although this approach is not efficient, but it was done only to facilitate the FL comparison in this study case.

For the HFL scenario, same as study case A., two edges are put in the middle, and each edge gets four clients that have completely non-iid data. As a result, each edge aggregates the model for all classes, which helps speed up the convergence.

Fig. 5 shows that the FL scenario performed poorly in terms of maximum testing accuracy with 6% drop compared to HFL, and in terms of the speed of convergence which is around 14 rounds slower than HFL as illustrated by the black line. On the other hand, the HFL scenario started with high testing accuracy and reached near the convergence state from the first communication round. One thing to notice is that the loss is almost equal in both scenarios.

C. Non-IID Client-Edge Assignment

In this study case, the FL scenario consists of 4 clients, each having all the samples corresponding to one specific attack and the corresponding portion of normal samples. For example, client 1 has all the 45,927 samples of the DoS attack (78% of all the attacks' samples), therefore it has 78% of the normal samples.

Regarding the HFL scenario, two edges are added in the middle between the clients and the cloud. The HFL scenario is repeated three times: i) where edge 1 has DoS + Probe attacks and edge 2 has the others, ii) where edge 1 has DoS + R2L attacks and edge 2 has the others, and iii) where edge 1 has DoS + U2R attacks and edge 2 has the other classes of attacks.

From Fig. 6, it can be seen that even if the edges end up aggregating models from clients with non-iid data, they can still perform better than FL with non-iid data, whether in terms

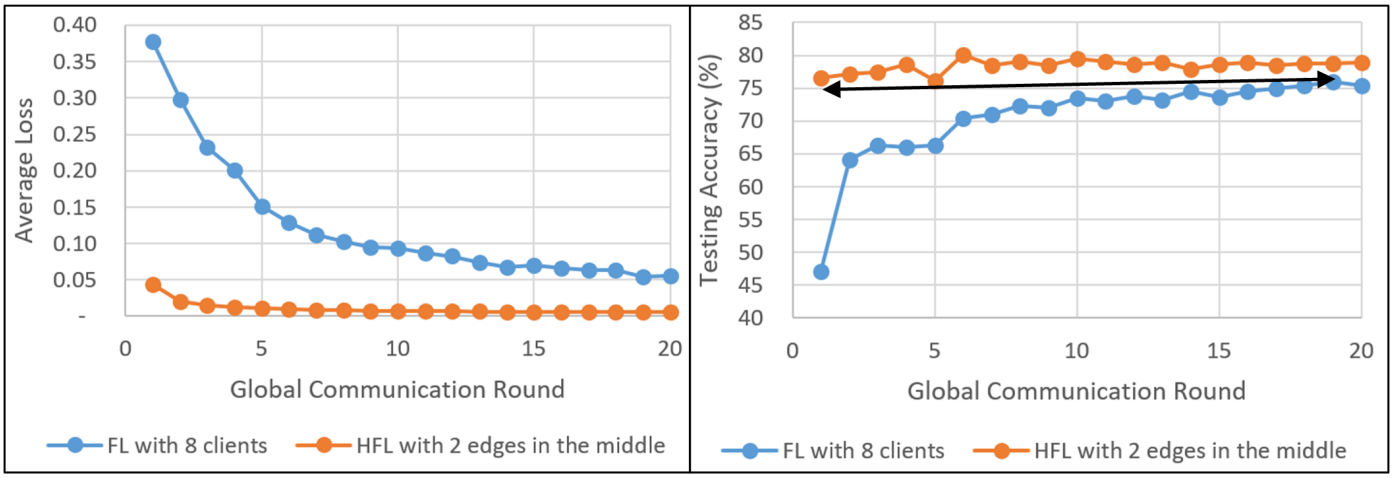

Fig. 4. Average loss (left) and testing accuracy (right) for study case A.

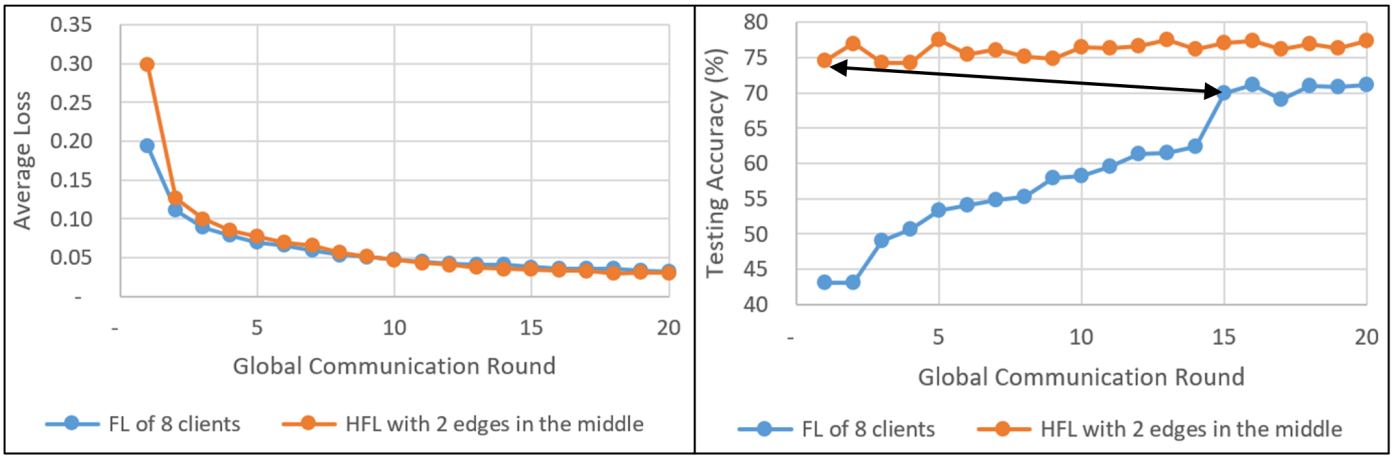

Fig. 5. Average loss (left) and testing accuracy (right) for study case B.

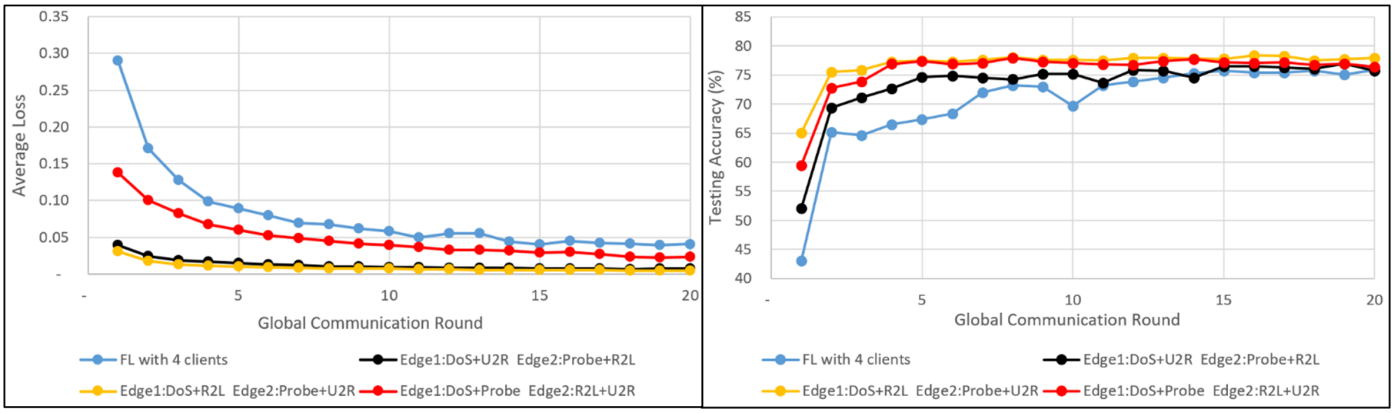

Fig. 6. Average loss (left) and testing accuracy (right) for study case C.

of training loss, testing accuracy (slightly better), or speed of convergence. It seems like the edge layer tends to absorb some of the effect of the non-iid distribution before it sends the aggregated model to the main cloud.

VI. CONCLUSION AND FUTURE WORK

In this paper, the challenges and issues growing with the huge deployment of IoT systems are discussed. The necessity of paying attention to and proactively taking countermeasures towards the security vulnerabilities in IoT environments is pointed to. Also, an IDS on NSL-KDD using FL and HFL is simulated. A comparison based on performance evaluation between FL and HFL in terms of training loss, testing accuracy, and speed of convergence is conducted. HFL proved its efficiency over FL

in two iid client-edge assignment study cases and one non-iid client-edge study case. Since FL and, especially, HFL are still new trending topics, they have many challenges that are still to be tackled and explored. The work done in this paper has many aspects that can be enhanced, which are left for future work. For example, applying Generative adversarial networks (GANs) to balance the NSL-KDD dataset rather than naively duplicating some of the classes' samples. Also, applying more comprehensive studies, such as considering the communication efficiency and energy consumption.

VII. ACKNOWLEDGEMENT

This work was made possible by NPRP grant NPRP12S-0305-190231 from the Qatar National Research Fund (a member of Qatar Foundation). The findings achieved herein are solely the responsibility of the authors.

REFERENCES

[1] A. Holst, "Number of Internet of Things (IoT) connected devices worldwide from 2019 to 2030," 2021. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (accessed Feb. 26, 2021).

[2] A. S. Gillis, "What is IoT (Internet of Things) and How Does it Work?" https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT (accessed Feb. 26, 2020).

[3] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," J. Electr. Comput. Eng., vol. 2017, 2017, doi: 10.1155/2017/9324035.

[4] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi, "Internet of Things (IoT) Communication Protocols: Review," in 2017 8th International Conference on Information Technology (ICIT), 2017, pp. 685–690, doi: 10.1109/ICITECH.2017.8079928.

[5] F. Meneghello, M. Calore, D. Zucchetto, M. Polese, and A. Zanella, "IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices," IEEE Internet Things J., vol. 6, no. 5, pp. 8182–8201, 2019, doi: 10.1109/JIOT.2019.2935189.

[6] S. Abdul Rahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A Survey on Federated Learning: The Journey from Centralized to Distributed On-Site Learning and Beyond," IEEE Internet Things J., 2020, doi: 10.1109/jiot.2020.3030072.

[7] L. Liu, J. Zhang, S. H. Song, and K. B. Letaief, "Client-Edge-Cloud Hierarchical Federated Learning," 2020, doi: 10.1109/ICC40277.2020.9148862.

[8] "NSL-KDD dataset." https://www.unb.ca/cic/datasets/nsl.html (accessed Feb. 26, 2021).

[9] P. S. Bhattacharjee, A. K. M. Fujail, and S. A. Begum, "A Comparison of Intrusion Detection by K-Means and Fuzzy C-Means Clustering Algorithm Over the NSL-KDD Dataset," 2017, doi: 10.1109/ICCIC.2017.8524401.

[10] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman,"Deep Learning Approach for Intelligent Intrusion Detection System," IEEE Access, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[11] R. Thomas and D. Pavithran, "A Survey of Intrusion Detection Models based on NSL-KDD Data Set," in 2018 Fifth HCT Information Technology Trends (ITT), 2018, pp. 286–291, doi: 10.1109/CTIT.2018.8649498.

[12] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security," IEEE Commun. Surv. Tutorials, vol. 22, no. 3, pp. 1646–1685, 2020, doi: 10.1109/COMST.2020.2988293.

[13] A. M. Mahfouz, D. Venugopal, and S. G. Shiva, "Comparative Analysis of ML Classifiers for Network Intrusion Detection," in Fourth International Congress on Information and Communication Technology. Advances in Intelligent Systems and Computing, 2020, vol. 1027, doi: 10.1007/978-981-32-9343-4_16.

[14] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber-Physical Systems," IEEE Trans. Ind. Informatics, 2020, doi: 10.1109/tii.2020.3023430.

[15] S. Abdul Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of Things Intrusion Detection: Centralized, On-Device, or Federated Learning?," IEEE Netw., pp. 1–8, 2020, doi: 10.1109/MNET.011.2000286.

[16] Z. Chen, N. Lv, P. Liu, Y. Fang, K. Chen, and W. Pan, "Intrusion Detection for Wireless Edge Networks Based on Federated Learning," IEEE Access, vol. 8, pp. 217463–217472, 2020, doi: 10.1109/ACCESS.2020.3041793.

[17] C. Briggs, Z. Fan, and P. Andras, "Federated learning with hierarchical clustering of local updates to improve training on non-IID data," 2020.

[18] N. Mhaisen, A. Awad, A. Mohamed, A. Erbad, and M. Guizani, "Optimal User-Edge Assignment in Hierarchical Federated Learning based on Statistical Properties and Network Topology Constraints," IEEE Trans. Netw. Sci. Eng., 2021, doi: 10.1109/tnse.2021.3053588.

[19] H. B. Mcmahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Areas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in Proceedings of the 20 th International Conference on Artificial Intelligence and Statistics (AISTATS), 2017, vol. 54.

[20] D. Yadav, "Categorical encoding using Label-Encoding and One-Hot-Encoder," 2019. https://towardsdatascience.com/categorical-encoding-using-label-encoding-and-one-hot-encoder-911ef77fb5bd (accessed Feb. 26, 2021).