CoLearn: Enabling Federated Learning in MUD-compliant IoT Edge Networks

Angelo Feraudo
University of Bologna, Italy
angelo.feraudo@studio.unibo.it

Poonam Yadav
University of York, UK
poonam.yadav@york.ac.uk

Vadim Safronov
University of Cambridge, UK
vadim.safronov@cl.cam.ac.uk

Diana Andreea Popescu
University of Cambridge, UK
diana.popescu@cl.cam.ac.uk

Richard Mortier
University of Cambridge, UK
richard.mortier@cl.cam.ac.uk

Shiqiang Wang
IBM Research, USA
wangshiq@us.ibm.com

Paolo Bellavista
University of Bologna, Italy
paolo.bellavista@unibo.it

Jon Crowcroft
University of Cambridge, UK
jon.crowcroft@cl.cam.ac.uk

ABSTRACT

Edge computing and Federated Learning (FL) can work in tandem to address issues related to privacy and collaborative distributed learning in untrusted IoT environments. However, deployment of FL in resource-constrained IoT devices faces challenges including asynchronous participation of such devices in training, and the need to prevent malicious devices from participating. To address these challenges we present CoLearn, which build on the open-source Manufacturer Usage Description (MUD) implementation *osMUD* and the FL framework *PySyft*. We deploy CoLearn on resource-constrained devices in a lab environment to demonstrate (*i*) an asynchronous participation mechanism for IoT devices in machine learning model training using a publish/subscribe architecture, (*ii*) a mechanism for reducing the attack surface in FL architecture by allowing only IoT MUD-compliant devices to participate in the training phases, and (*iii*) a trade-off between communication bandwidth usage, training time and device temperature (thermal fatigue).

CCS CONCEPTS

• **Networks** → **Network algorithms**; *Network experimentation*; **Network privacy and anonymity**; • **Computer systems organization** → *Embedded and cyber-physical systems*; • **Computing methodologies** → *Neural networks*; *Machine learning*.

KEYWORDS

Distributed machine learning, edge computing, federated learning, Internet of Things (IoT), Manufacturer Usage Description, anomaly detection, privacy, security

ACM Reference Format:
Angelo Feraudo, Poonam Yadav, Vadim Safronov, Diana Andreea Popescu, Richard Mortier, Shiqiang Wang, Paolo Bellavista, and Jon Crowcroft. 2020. CoLearn: Enabling Federated Learning in MUD-compliant IoT Edge Networks. In *3rd International Workshop on Edge Systems, Analytics and Networking (EdgeSys '20), April 27, 2020, Heraklion, Greece.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3378679.3394528

EdgeSys '20, April 27, 2020, Heraklion, Greece

ACM ISBN 978-1-4503-7132-2/20/04…$15.00
https://doi.org/10.1145/3378679.3394528

1 INTRODUCTION

IoT devices are resource-constrained and highly heterogeneous in both underlying system capability and statistical network behaviour, and are widely distributed. Many applications for such distributed IoT infrastructures require user privacy protection and bounded-time (low latency) response. They could thus benefit from data being processed either locally on the device or at the edge of the network. However, applications such as anomaly detection also need information from other application instances. An affordable and practical solution for privacy-preserving collaborative learning in resource-constrained IoT is thus desirable [5, 22].

Federated Learning (FL) is a promising technique for addressing privacy issues in collaborative learning and has gained recent attention from both academia and industry [16, 18, 23, 26]. However, deploying FL in resource-constrained IoT and edge devices faces additional challenges, such as how to handle asynchronous participation of distributed battery-powered IoT devices, and how to prevent malicious devices from taking part in the training phase. To reduce the attack surface in IoT, the Internet Engineering Task Force (IETF) has ratified a standard for IoT device manufacturers to provide a Manufacturer Usage Description (MUD) with their IoT devices [14]. The MUD standard restricts and limits traffic end-points and rates in and out of IoT devices, thus limiting the attack surface and enabling identification of volumetric and man-in-the-middle attacks [11, 27]. Deployment of FL in MUD-compliant networks thus not only meets necessary security requirements but also provides a practical solution for limiting the attack surface.

We start with a brief review of background and related work, including introduction of osMUD and PySyft, the two major components on which we build (§2). We then present the CoLearn architecture and how it supports MUD profile management and federated learning (§3). We evaluate CoLearn in lab conditions to understand its performance (§4). We conclude with a brief discussion of future work (§5).

2 BACKGROUND & RELATED WORK

Many distributed learning algorithms assume that the data are homogeneously distributed among the distributed nodes. However, in Federated Learning (FL), the dimensions and contents of the datasets are typically heterogeneous, and the FL coordinator server does not have full control of the distributed computational resources. Additionally, to preserve privacy, the FL coordinator server often receives only model updates, and not the local training data from participating IoT devices. Thus recent research in FL has focused on system design and scalability [3, 5] and algorithm design to address communication efficiency, systems heterogeneity, statistical heterogeneity, and privacy [12, 16, 23]. However there is limited work on deploying FL on resource-constrained IoT devices [4, 26].

The use of MUD as an isolation-based defensive mechanism to restrict traffic generated from IoT devices is still in its early phase. Therefore, only a few deployment scenarios and proof-of-concept (PoC) implementations currently exist [1, 2, 10, 11, 19, 28]. To the best of our knowledge, no work has considered the deployment of FL in MUD-compliant networks.

To enable FL in MUD-compliant IoT edge networks, we chose the Open Source MUD implementation[1] and the PySyft [24] framework. Open Source MUD (osMUD) is developed by a consortium of device manufacturing and network security companies. It is designed to be suitable for resource-constrained routers and firewalls, running the OpenWRT platform, though it can also be compiled outside of OpenWRT for most C compatible environments. It integrates with *dnsmasq* for network infrastructure services, and running it outside of OpenWRT requires a compatible firewall and a MUD-compliant DHCP server able to extract the MUD URL from the DHCP header packet. To build a federated learning system, we use the PySyft [24] framework, built on top of *PyTorch* [8] to provide transparent APIs for privacy-preserving deep learning. This allows straightforward implementation of privacy-preserving constructs, such as FL, Secure Multiparty Computation, and Differential Privacy.

3 COLEARN ARCHITECTURE

Interaction between the main components of CoLearn (the MUD manager, the User Policy Server (UPS) and the FL Coordinator) are depicted in Figure 1. The "Thing" represents a MUD-compliant IoT device that emits its MUD-URL in its DHCP request (the exact protocol used depends on MUD implementation). The MUD manager then interacts with MUD Server and UPS, sending a list of valid devices to the Coordinator helping to reduce the attack surface in the FL protocol [3] if device authentication is used by excluding compromised IoT devices. The Coordinator interacts with IoT devices using MQTT broker and WebSockets. Details of components and their interactions are presented in §3.1 and §3.2. We assume the entity that hosts the Coordinator is trusted and so all the communication between the osMUD manager and the entity hosting the UPS and the Coordinator are trusted and encrypted.

3.1 MUD Manager & User Policy Server

The MUD manager uses the osMUD manager implementation designed to be integrated with the dnsmasq and OpenWRT services. When the MUD manager receives a DHCP request from a connected

[1] https://github.com/osmud/osmud

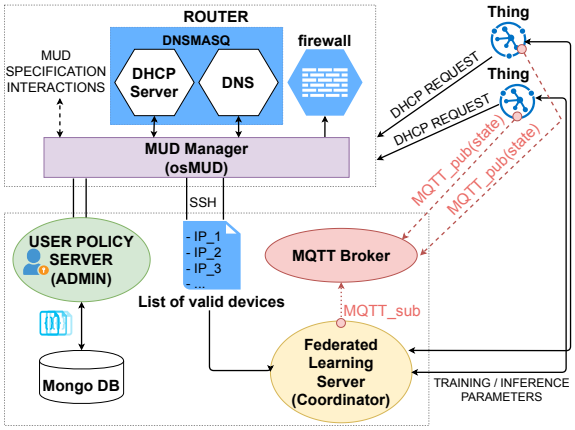

Figure 1: CoLearn architecture.

IoT device, *dnsmasq* invokes a script that processes the MUD request, fetching the MUD files from the MUD server. The User Policy Server (UPS) allows network administrator to enforce new rules beyond those defined by the manufacturer. The manufacturer thus does not need insight into internal network behaviour – the UPS, in addition to making all devices MUD-compliant, allows definition of categories of rules specific to the network in which MUD is deployed and so hard to be determined by the manufacturer, while preserving the YANG structure that is typical for MUD files [15].

The introduction of UPS in CoLearn raises two implementation challenges that need to be addressed. First, the osMUD manager has to be modified in order to request a new MUD file for each MUD-compliant device. Second, the UPS must identify the network administrators and keep a separate session for each of them.

To insert MUD files into the database, the UPS provides a JavaScript program that executes insertion queries. The UPS provides a GUI with which it is possible to have a view of all the MUD files that are hosted by it. By analysing the osMUD deployment we configured how to implement a new feature that provides the ability for an administrator to upload new MUD files. Using UPS, an administrator can also define a new MUD file for MUD-compliant devices, which results in the production of new rules different from those defined by the manufacturer. Thus, the UPS enables us to both *make non-MUD compliant device MUD-compliant* and *insert administrator-defined rules*.

The files inserted by the administrator represent MUD files (UPS files) that must be retrieved by the MUD manager. In our setup, the MUD-compliant devices do not provide an extra MUD-URL to identify the UPS location, the MUD manager must know in advance the UPS location and locate the MUD file using the devices' MAC addresses. At the UPS side, the *administrator authentication* and *validation of the UPS file name* are required. For our prototype, we implemented a *simple authentication form with username and password*, where the user data is stored in MongoDB. The form allows a separate environment for each end-user, so that they can insert and remove only their files. The UPS file names are validated using a regular expression matching schema.

In the current version, the UPS implements the union of MUD rules, which can result in redundant rules. Additional problems we

see in the current solution are resolution of conflict rules, i.e. the administrator rules that can either remove or reverse the manufacturer rules, requires thoughtful considerations. Even though the current incremental-rule enforcement mechanism using UPS needs further work, we believe that the introduction of UPS provides a secure way to consider different deployment scenarios which can improve the security and reliability of a MUD deployment.

3.2 CoLearn: Federated Learning

To deploy an automated federated learning mechanism in our MUD-compliant network we chose *PySyft* framework as it provides the *Network Worker* structure that enables the remote communication of the model and uses the Web Socket protocol to lower overhead, and facilitates real-time data transfer from and to the Server.

The FL architecture needs a central entity that coordinates all the on-device learning activities and receives device availability information – we refer to this as the *Coordinator*. The Coordinator must be able to recognise when devices are ready to start a training phase, and when they are ready to receive the model in order to do inference on their data.

To automate the above process in light of the heterogeneity in IoT devices, we opted for a lightweight publish/subscribe architecture implemented using Message Queue Telemetry Transport (MQTT) [21] protocol. In our setup, the Coordinator becomes a subscriber and the devices, which want to communicate their state, become the publishers. We define three states of FL systems in IoT networks: (*i*) a device being ready for *training* a model; (*ii*) a device needing to perform *inference* on its local data; and (*iii*) a device could be in *not ready* state when the training starts.

A challenge due to heterogeneity and unavailability of devices in FL is that, even if a device declares its training intention, it may not be available anymore when the training starts. To address this problem, Bonawitz et al. [5] proposed a solution where the Coordinator waits until the number of devices is enough to obtain improvements to the model. We therefore include a *wait state* at the Coordinator. We model *wait state* as a *temporal window* in which the Coordinator waits and collects *training requests*. During this *wait state*, the devices can remove or drop themselves from the Coordinator's devices list. The *temporal window* starts after the first training request has been received by the Coordinator, whereas the end of the window depends on the architecture design (e.g., how many devices the model needs in order to have some improvements).

Due to the asynchronous pattern of training requests, it is possible that some devices declare their training intentions when the training has already begun. In this case, the Coordinator chooses one of the two actions: (*i*) *discard* these devices' requests or (*ii*) *keep* them until a new temporal window starts. The former may make sense when either a further cycle of model training is not needed, because the model has been trained sufficiently or to prevent the Coordinator from overloading. The latter represents the *default case* especially in the early stages of training.

Figure 2 shows the workflow of FL between IoT devices and the Coordinator. In Steps ①, ② and ③, the Coordinator and the devices establish a connection with the Broker, which then mediates the communication. At the same time, the Coordinator subscribes

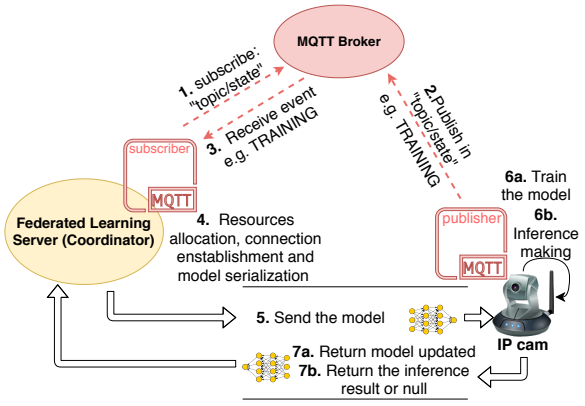

Figure 2: CoLearn Distributed architecture for Federated Learning.

itself to a particular *topic* (*topic/state* in the figure), so that it can receive the status updates from the devices (publishers). Steps ④ and ⑤ involves sending the model to devices. The Coordinator allocates and prepares all the parameters that are useful for the communication phase (resource allocation, model serialisation, etc.) before this step. In Steps ⑥ and ⑦, the participating devices start the training or inference depending on their states.

In case of training, Figure 2 also defines a *round*, which includes: (*i*) selection of clients that demonstrate their training intention; (*ii*) training the received model and computation of model updates; (*iii*) sending the model updates; (*iv*) aggregation of the model updates by the Coordinator to construct an improved global model. The round is typically repeated more than once to reach acceptable levels of accuracy due to the limited capabilities of the devices. For efficient use of the limited amount of available resources (communication and computation), it is necessary to find the best trade-off between the number of rounds and the number of iterations (device performance) [26], or apply model compression techniques [12].

4 EXPERIMENTAL EVALUATION

We seek to check the feasibility of running such a system in home routers or small business environments using a setup depicted in Figure 3. We use an OpenWRT Netgear router (model WNDR 3700v2) to host the MUD manager, a MacBook Pro (Intel Core i5 and 8 GB of RAM) to host both the UPS and the FL Coordinator, and two Raspberry Pi 3 B+ (hereafter, RPi) as edge devices running FL clients and supporting the Python environment needed for PySyft.

All communications from the IoT devices thus pass through the associated RPi, which collects the data and performs training or inference as required. This allows us to make all the devices MUD compliant. As a consequence of interfacing the IoT devices to the network through a RPi, it is possible to modify the DHCP client configurations to forge MUD-compliant DHCP requests.

To evaluate CoLearn federated learning, we use the open-source IoT BoTnet identification dataset [13] and a feed forward neural network model composed of two hidden layers, one with 50 neurons and the other with 30 neurons, an input size of 10 (the number

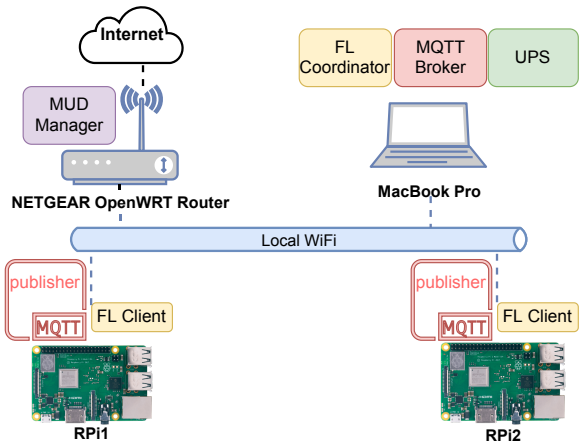

Figure 3: Experimental setup.

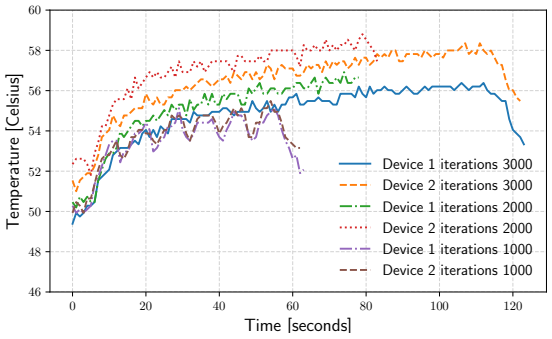

Figure 4: RPi temperature variations with 1000, 2000, 3000 iterations per round and six rounds. Peaks correspond to the end of a round in all cases.

of features selected from the dataset) and one output neuron for anomaly detection [9].

In our experiments, we use a fixed batch size of one and vary the number of rounds (interactions between the Coordinator and the edge devices) and maximum number of iterations (number of stochastic gradient descent steps per device). For example, using 1000 iterations indicates that random batches of data are selected to perform training for up to a maximum of 1000 times. Given a batch size of one this means that 1000 random rows of the dataset are selected to perform training. At each new round the subset of the dataset used changes – that is, we choose a new random subset from the dataset. In our experiments we consider three, six, and 12 rounds, and 1000, 2000, 3000 iterations.

Overall, we see that increasing iterations reduces the relative bandwidth overheads and produces more accurate results. However, doing so can cause problems with device performance as a large number of iterations will influence the device's CPU temperature and may induce thermal throttling. We validated the following components useful for a remote case implementation: (*i*) a MQTT subscriber able to receive and process state events from unknown devices; (*ii*) a parser able to process and verify the event syntax correctness; and (*iii*) an algorithm able to collect workers' information and manage the temporal window problem.

4.1 Training Loss

We measured training losses to understand the trade-off between the number of rounds and iterations used. To give a valid comparison we initialize the model with random weights in each test, then train the model for the given number of iterations and rounds. Tests are performed with 1000, 2000, 3000 iterations and three, six and 12 rounds. Results are shown in Table 1 where we found that training loss value decreases with total number of iterations but is largely unaffected by the number of rounds for a fixed number of total iterations. This is because, with our current experimentation setup, the data at devices are independent and identically distributed (IID). The number of rounds will likely have a significant impact on the training loss if devices have non-IID data [26].

4.2 Temperature Monitoring

The temperature monitoring experiments show the effects of training and learning on the temperature rise in the RPis, a potentially important parameter for use in the decision logic of FL clients. In this experiment, the temperature measurements are repeated three times to obtain an average value and are considered in a time window that starts 5 seconds before training starts and ends 5 seconds after training ends. For the baseline value, we analysed the temperature variation behaviour of the RPi in an idle case, i.e., before the training phase. We noticed that the temperature did not exceed the 50°C threshold (stayed between 48 and 50°C) in a one-hour monitoring window. This value is used as a baseline for comparison to understand if the temperature rise during training can cause slowdown problems due to thermal throttling around 85°C. Overall, Figure 4 demonstrates that the temperature does not exceed 60°C. Rather, the maximum temperature reached is the case with 2000 iterations with 58.4°C, as result of the highest initial temperature.

It is noticeable that for both RPis the worst case is represented by the case with up to 3000 iterations, where the temperature increases with an average of 6.75°C, followed by 2000 iterations with around 6.18°C, and lastly the 1000 iterations with an average of 5.875°C (Figure 4). The 12 rounds case with 1000 max. iterations (Figure 5) increases the average of 0.4°C which is considered negligible. From this analysis, we can conclude that the number of iterations influences the temperature of the components involved. From a trade-off perspective, considering the above temperature analysis, it is preferred to increase the interactions rather than the data analysed locally in one round. Obviously, as the next analysis confirms, the reduction of data analysed is made at the expense of other parameters.

4.3 Bandwidth Monitoring

In this experiment, we cumulatively measure bandwidth at the network interfaces of both RPis and FL Coordinator using 1000 iterations per round for training. We send the same FL model in all scenarios, and so we present here only the outgoing traffic monitoring. We conducted experiments for three, six, and 12 rounds, starting measurement three seconds before training start time and

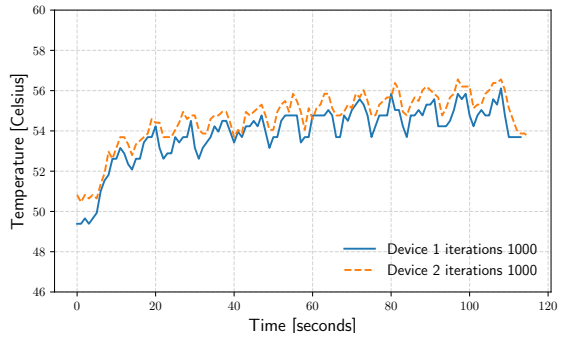

Figure 5: RPi temperatures for 12 rounds and 1000 iterations per round.

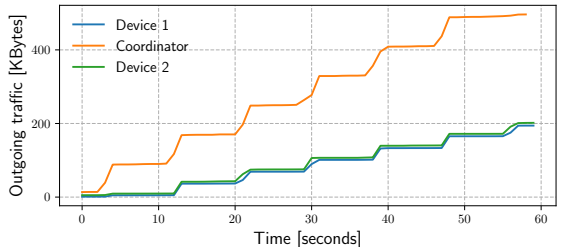

Figure 6: Outgoing traffic in six rounds with 1000 iterations per round.

finishing three seconds after training ends. We repeat each experiment three times.

Figure 6 presents outgoing traffic measurements (aggregated over time) for six rounds and 1000 iterations per round. The stair-shaped step represents the round in which the model is sent and so the step value is an indication of the approximate size of the model. The model size depends on different factors, such as the number of hidden layers and weights. The bottom part of the first step for the coordinator is around 13 kB, while the top part is approximately 85 kB. By using these two values, it is possible to find the approximate dimension of the TrainConfig object (around 36 kB), the object used in PySyft to transfer a model in training phases. By looking at the steps' dimensions of the RPi outgoing bandwidth, we computed an approximate average model size of 30.61 kB. However, we also take into consideration that the WebSocket protocol requires the exchange of other values to perform the initial handshake, which implies we need to subtract this overhead of around 1 kB while calculating the bandwidth used by model parameters. We can see that the total outgoing traffic, as expected, is strictly correlated to the model dimension, number of rounds, and the number of devices involved.

4.4 Training Time

In this experiment we present the training time taken by the RPis. This depends on the device's capabilities and allows us to understand how long the Coordinator waits for a response from the RPis on average. We present the total training time in Table 1 for different numbers of iterations and rounds.

Table 1: Average total training time and training losses.

	Per-round Iterations	Number of Rounds	Total Iterations	Training Time [s]	Training Loss
1	1000	3	3000	26.868	0.001814
2	1000	6	6000	53.148	0.001068
3	1000	12	12000	105.921	0.000863
4	2000	3	6000	38.378	0.00107
5	2000	6	12000	76.139	0.000877
6	3000	3	9000	56.467	0.000957
7	3000	6	18000	112.247	0.000852

As expected, the total training time increases with total number of iterations. However, comparing rows 2 (iterations: 1000, rounds: 6) and 4 (iterations: 2000, rounds: 3), both having 6000 iterations, we see that training time is less for row 4 (38.378 s) than row 2 (53.148 s); similarly for rows 3 and 5. That is, apart from the total number of iterations, increasing the number of rounds also increases total training time due to additional communication time and waiting/processing delay at FL clients and Coordinator.

4.5 Impact of Privacy Preservation via SMPC

NB. Results for this experiment were performed in an emulated configuration with the FL Coordinator and FL Clients as Virtual Workers on the MacBook Pro as we unfortunately lost access to our lab-based RPi configuration due to the COVID-19 outbreak triggering closure of University buildings. Nonetheless our results indicate the relative overheads of introducing SMPC as a privacy preserving measure.

The FL Coordinator is a subscriber of a particular *topic* on which all the device states are published. The state event follows the *(device name, state)* syntax. The device name is ordinarily represented by the IP address of the device generating the event. The state, commonly indicating the device's intention, specifies the behaviour assumed by the Virtual Worker, and is drawn from these three states: Training, Inference, and Not Ready. At the end of each training round, the model's updates are gathered in a dictionary, which is then used to compute the global model using federated averaging [17, 20].

We repeated the model training using Secure Multi-Party Computation (SMPC). The SMPC algorithm relies on two crypto protocols SPDZ [7] and SecureNN [25]. We compare the training times in Figure 7. Both experiments were repeated five times, using only one round to compute the training time and varying batch sizes (100, 200, 800, 1000). Figure 7 clearly shows the extra time spent training when using SMPC.

5 DISCUSSIONS & FUTURE WORK

We have presented and evaluated CoLearn, an integrated system which we built on top of the state-of-the-art open-source MUD implementation *osMUD* [6] and open-source FL framework *PySyft* [24]. By deploying CoLearn on resource-constrained devices in a laboratory setting, we demonstrated a mechanism for reducing the attack

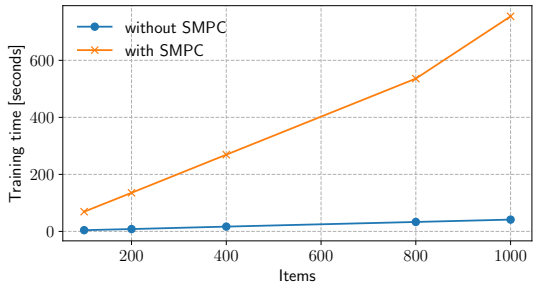

Figure 7: Comparison of training time with and without SMPC.

surface in FL architecture by allowing only IoT MUD-compliant devices to participate in the distributed learning phases using MQTT based publish/subscriber architecture.

We demonstrated a trade-off between communication bandwidth usage and training time and effects on device temperature (thermal fatigue) by using an anomaly detection dataset. We presented insights derived from system design and experimental results, which can help to design efficient, adaptive and secure FL logic clients, taking into consideration both device heterogeneity and the applications' communication-efficiency and performance requirements.

In the current CoLearn deployment, we assumed that edge devices do not fail in the training phases and during their activity of traffic eavesdropping. This assumption necessitates further considerations and improvements. Additionally, we did not focus on IoT device identification and authentication, which is vital for both MUD-compliant networks and FL architecture, and can be solved by using IoT device manufacturer provisioned X.509 certificate. Another valuable improvement for our architecture is to extend the YANG-based MUD file by adding a field containing structure and weights of a model. Thus, the manufacturers can define a model that describes normal behaviours for each device category, and is able to identify and flag abnormal behaviour. Furthermore, auto-adaptive temporal window sizing can improve the model training process and Coordinator performance.

ACKNOWLEDGEMENTS

Mortier is supported in part by funding provided through the Government's modern industrial strategy by Innovate UK, part of UK Research and Innovation, and so this research forms part of the Centre for Digital Built Britain's work within the Construction Innovation Hub. Mortier and Popescu are supported in part by EPSRC grants EP/N028260/2 and EP/R03351X/1. Yadav is supported in part by EPSRC grant EP/R045178/1. Safronov is supported by Huawei.

REFERENCES

[1] Yehuda Afek, Anat Bremler-Barr, David Hay, Ran Goldschmidt, Lior Shafir, Gafnit Abraham, and Avraham Shalev. 2019. NFV-based IoT Security for Home Networks using MUD. (2019). arXiv:1911.00253

[2] Vafa Andalibi, DongInn Kim, and L. Jean Camp. 2019. Throwing MUD into the FOG: Defending IoT and Fog by expanding MUD to Fog network. In *2nd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 19)*. USENIX Association, Renton, WA. https://www.usenix.org/conference/hotedge19/presentation/andalibi

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2018. How to backdoor federated learning. *arXiv:1807.00459* (2018).

[4] Colby R. Banbury, Vijay Janapa Reddi, Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang, Robert Hurtado, David Kanter, Anton Lokhmotov, David Patterson, Danilo Pau, Jae sun Seo, Jeff Sieracki, Urmish Thakker, Marian Verhelst, and Poonam Yadav. 2020. Benchmarking TinyML Systems: Challenges and Direction. In *Proceedings of the 3rd MLSys Conference* (Austin, TX, USA) *(MLSys'20)*. arXiv:2003.04821

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konecny, Stefano Mazzocchi, H Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. (2019). arXiv:1902.01046

[6] A consortium of network security companies. 2018. Open Source Manufacture Usage Specification. https://osmud.org.

[7] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. 2012. Multiparty computation from somewhat homomorphic encryption. In *Annual Cryptology Conference*. Springer, 643–662.

[8] Community driven project. 2016. PyTorch. https://pytorch.org.

[9] Angelo Feraudo et al. 2020. CoLearn: Source code. https://github.com/aferaudo/CoLearn_Federated_Learning.

[10] Angelo Feraudo, Poonam Yadav, Richard Mortier, Paolo Bellavista, and Jon Crowcroft. 2020. SoK: Beyond IoT MUD Deployments - Challenges and Future Directions. (2020). arXiv:2004.08003

[11] Ayyoob Hamza, Hassan Habibi Gharakheili, Theophilus A. Benson, and Vijay Sivaraman. 2019. Detecting Volumetric Attacks on loT Devices via SDN-Based Monitoring of MUD Activity. In *Proceedings of the 2019 ACM Symposium on SDN Research* (San Jose, CA, USA) *(SOSR'19)*. ACM, New York, NY, USA, 36–48.

[12] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. (2016). arXiv:1610.05492

[13] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. 2019. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems* 100 (2019), 779–796.

[14] E. Lear, R. Droms, and D. Romascanu. 2019. *Manufacturer Usage Description Specification*. RFC 8520. https://rfc-editor.org/rfc/rfc8520.txt

[15] L. Lhotka, Ralph Droms, and Dan Romascanu. 2016. JSON Encoding of Data Modeled with YANG. RFC 7951. https://doi.org/10.17487/RFC7951

[16] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2019. Federated Learning: Challenges, Methods, and Future Directions. (2019). arXiv:1908.07873

[17] Xiang Li, Kai xuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the Convergence of FedAvg on Non-IID Data. (2019). arXiv:1907.02189

[18] Xiang Li, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. Communication-Efficient Local Decentralized SGD Methods. arXiv:1910.09126

[19] Sara N. Matheu, Alberto Robles Enciso, Alejandro Molina Zarca, Dan Garcia-Carrillo, Jose Luis Hernandez-Ramos, Jorge Bernal Bernabe, and Antonio F. Skarmeta. 2020. Security Architecture for Defining and Enforcing Security Profiles in DLT/SDN-Based IoT Systems. *Sensors* 20, 7 (March 2020).

[20] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated Learning of Deep Networks using Model Averaging. (2016). arXiv:1602.05629

[21] MQTT. 1999. Message Queue Telemetry Transport. http://mqtt.org.

[22] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N Asokan, and Ahmad-Reza Sadeghi. 2019. DÏoT: A federated self-learning anomaly detection system for IoT. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 756–767.

[23] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. 2019. FedPAQ: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization. (2019). arXiv:1909.13014

[24] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. 2018. A generic framework for privacy preserving deep learning. (2018). arXiv:1811.04017

[25] Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2019. Securenn: 3-party secure computation for neural network training. *Proceedings on Privacy Enhancing Technologies* 2019, 3 (2019), 26–49.

[26] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. 2019. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1205–1221.

[27] Poonam Yadav, Qi Li, Richard Mortier, and Anthony Brown. 2019. Network Service Dependencies in Commodity Internet-of-things Devices. In *Proceedings of the International Conference on Internet of Things Design and Implementation* (Montreal, Quebec, Canada) *(IoTDI '19)*. ACM, New York, NY, USA, 202–212.

[28] Poonam Yadav, Vadim Safronov, and Richard Mortier. 2019. Poster Abstract: Enforcing accountability in Smart built-in IoT environment using MUD. In *The 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys 2019)*. ACM, New York.