Jon Crowcroft,
http://www.cl.cam.ac.uk/~jac22
Apr/May 2021
osMUD - concern over misbehaving devices

PySft - network worker + coordinator

Scale up for future work:-)

Secure Multiparty Computation via SPDZ & SecureNN

Threat model - device owner fear of bad publicity?
When the MUD manager receives a DHCP request from a connected device, it interacts with the MUD Server and the UPS, sending a list of valid MUD-compliant devices. The Coordinator interacts with IoT devices to help reduce the attack surface in resource-constrained devices.

3 COLEARN ARCHITECTURE

To enable FL in MUD-compliant IoT edge networks, we chose the Open Source MUD implementation. Therefore, only a few deployment scenarios and proof-of-concept (PoC) implementations currently exist.

The MUD manager uses the osMUD manager implementation. The osMUD deployment raises two implementation issues. First, the UPS must identify the network addresses of the MUD-compliant devices. To insert MUD rules into the database, the UPS provides a JavaScript program that executes insertion queries. The UPS provides a GUI for an administrator to upload new MUD rules.

Interaction between the main components of CoLearn (the MUD manager, the router, and the Coordinator) is depicted in Figure 1. The 'Thing' represents a MUD-compliant IoT device. The Coordinator interacts with IoT devices to the Coordinator helping to reduce the attack surface in resource-constrained routers and Open Source MUD implementation.

CoLearn addresses communication efficiency, systems heterogeneity, statistical heterogeneity, and privacy.

In the current version, the UPS implements the union of MUD rules inserted by the administrator. For our prototype, the UPS implemented the union of MUD rules inserted by the administrator in order to request a new MUD rule. The User Policy Server (UPS) allows network administrator to enforce new rules beyond those defined in the MUD specification. The UPS administrator can also define a new MUD rule using the devices' MAC addresses. The UPS provides a JavaScript program that executes insertion queries. The UPS provides a GUI for the administrator to upload new MUD rules.

The introduction of UPS in CoLearn raises two implementation issues. First, the UPS must identify the network addresses of the MUD-compliant devices. To insert MUD rules into the database, the UPS provides a JavaScript program that executes insertion queries. The UPS provides a GUI for the administrator to upload new MUD rules.

The UPS administrator can also define a new MUD rule using the devices' MAC addresses. The UPS provides a JavaScript program that executes insertion queries. The UPS provides a GUI for the administrator to upload new MUD rules.

The UPS administrator can also define a new MUD rule using the devices' MAC addresses.
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) keep the devices in the on-device learning activities and receives device availability information – we refer to this as the wait state of FL between IoT devices and the Coordinator, which collects the communication. At the same time, the Coordinator subscribes to the Broker, which then mediates the devices' connection requests. The Coordinator from overloading. The latter represents the wait state when the training starts. The former may make the communication dataset.

1. subscribe: "topic/state"
2. Publish in "topic/state"
3. Receive event e.g. TRAINING
4. Resources allocation, connection establishment and model serialization
5. Send the model
6a. Train the model
6b. Inference making
7a. Return model updated
7b. Return the inference result or null
We measured training losses to understand the trade-off between the number of rounds and iterations used. To give a valid comparison, results are shown in Table 1 where we found that training loss value decreases with total number of iterations but is largely negligible. From this analysis, we can conclude that the number of iterations will influence the device's CPU temperature rather than the number of rounds (interactions between the Coordinator and devices; (Figure 5) increases the average of 0.4 °C, followed by 2000 iterations with 58.4 °C, as result of the highest initial temperature. The 12 rounds case with 1000 max. iterations has 0.5 °C, the 1000 iterations with an average of 6.75 °C, and lastly the 1000 iterations with 5.875 °C, as result of the highest initial temperature.

The number of rounds will likely have a significant impact on the number of features selected from the dataset used changes – that is, we choose a new random subset of the dataset used for training. In our experiments we consider three, six, and 12 rounds; an algorithm able to collect workers’ information and manage the temporal window problem.

In our experiments, we use a stochastic gradient descent steps per device). For example, using a fixed batch size of one and vary the number of rounds and iterations used. To give a valid comparison, results are shown in Table 1 where we found that training loss value decreases with total number of iterations but is largely negligible. From this analysis, we can conclude that the number of iterations will influence the device’s CPU temperature rather than the number of rounds (interactions between the Coordinator and devices; (Figure 5) increases the average of 0.4 °C, followed by 2000 iterations with 58.4 °C, as result of the highest initial temperature. The 12 rounds case with 1000 max. iterations has 0.5 °C, the 1000 iterations with an average of 6.75 °C, and lastly the 1000 iterations with 5.875 °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 in an idle case, i.e., before training ends. For the baseline value, we analysed the temperature variation behaviour of the RPi in an idle case, i.e., before training ends. For the baseline value, we analysed the temperature variation behaviour of the RPi in an idle case, i.e., before training ends. We noticed that the temperature did not exceed 60 °C after training ends. We noticed that the temperature did not exceed 60 °C after training ends. From a trade-off perspective, considering the above temperature effects of training, it is preferred to increase the interactions rather than the number of rounds.
Evaluation

- Open Source botnet id dataset
- feed forward neural net - 2 hidden layers - see paper
Some results

![Graph showing temperature variations over time for different iterations and devices.](image)
1. The model size depends on the number of parameters involved. In this experiment we present the training time taken by the RPis.

2. 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 the steps' dimensions of the RPi outgoing bandwidth, we computed the total outgoing traffic 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

3. We have presented and evaluated CoLearn, an integrated system that includes SMPC (Secure Multi-Party Computation) and SecureNN. The SMPC algorithm relies on two crypto protocols: One for generating the parameters and the other for computing the global model using federated learning. The SecureNN implementation is then used to compute the global model using federated learning when using SMPC.

4. Figure 5: RPi temperatures for 12 rounds and 1000 iterations. Figure 6 presents outgoing traffic for different numbers of iterations and rounds.

5. Figure 7: Comparison of training time with and without SMPC.
Any Questions?

- ref: colearn
- https://doi.org/10.1145/3378679.3394528
- alt: ppfl