Demo: A Blockchain Based Protocol for Federated Learning

Qiong Zhang[1], Paparao Palacharla[1], Motoyoshi Sekiya[2], Junichi Suga[2], Toru Katagiri[2]

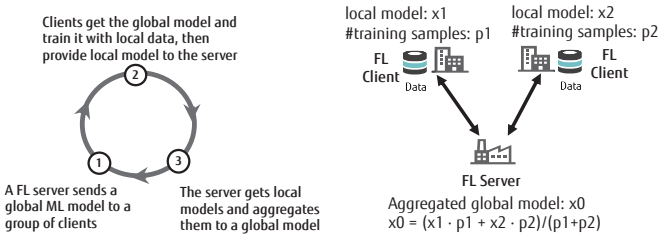
[1] Fujitsu Network Communications, Richardson, Texas, USA
[2] Fujitsu Laboratories Limited, Kawasaki, Japan

***Abstract*—In this demo, we demonstrate a novel blockchain based protocol for federated learning. We present the system architecture and describe the blockchain based protocol that seamlessly provides secure communication in federated learning with physically distributed data sets.**

I. Introduction

Federated learning (FL) [1] is a distributed Machine Learning (ML) approach which enables ML models training on decentralized private data. In FL, multiple *clients* that usually own local private data collaborate to train a ML model, under the coordination of a central *server*. A training round in FL includes three steps, as shown in Fig. 1(a). The server provides a global ML model and sends it to a group of clients at Step 1. At Step 2, clients receive the global model and train local models with local data, and send local models to the server. At Step 3, the server *aggregates* local models from clients to a new global model, which is sent to clients starting a new training round. FL can have hundreds of training rounds until converged. Fig. 1(b) shows the weighted averaging approach for server aggregation. The global model sent from the server to clients is x_0. The local model trained at Client i is x_i and the number of training samples in Client i's data set is p_i. The server computes the new global model as: $x_0 = \sum(x_i \cdot p_i) / \sum p_i$.

FL preserves data privacy by training ML models locally at clients, and also saves network bandwidth by avoiding the transfer of large amounts of training data to a central location. Instead, only ML models are transferred between a server and clients. FL can be applied in many applications. Google has applied FL to next word prediction for mobile keyboards to improve user experience. Another application is in healthcare, where data from hospitals cannot be shared due to regulations. Training ML models on the data distributed at hospitals, e.g., for predicting mortality and hospital stay time, becomes extremely important [2]. FL can also be applied in the telecom industry, where distributed network data are generated from networks belonging to different operators and domains (such as network edges and core). Usually, these network data cannot be shared. FL can be used to train robust ML models with distributed data for predicting network performance and faults.



(a) A single round of FL training (b) FL server aggregation

Fig. 1: Federated learning description

Here, we focus on cross-silo FL, where organizations act as FL server/clients and share a common incentive to train a model based on all of their data. Usually, FL server and clients are physically distributed at different organizations. One challenge in FL is the secure communication between server and clients.

Hyperledger Fabric, a private blockchain platform, can provide immutable records on a distributed ledger (also called blockchain). The blockchain is replicated across many peer nodes in a blockchain network, where the replicas are kept synchronized by a consensus process among peer nodes. A secure data exchange system [3] is built based on Hyperledger Fabric, where the description of data (called *metadata*), not the actual data, is recorded on the blockchain. The metadata can be searched on the blockchain, and if found, the actual data can be securely transferred upon requests.

In this demo, we demonstrate FL to train a neural network model on physically distributed data sets at FL clients, where the underlying communication between server and clients uses a novel blockchain based protocol on the secure data exchange system. We present the system architecture and the blockchain based protocol for FL, then describe the demo scenario.

II. System Architecture and Protrocol

A. System Architecture

Figure 2 shows a blockchain based data exchange system for FL, where each peer node in the blockchain network has a consistent copy of the blockchain. For a FL task, the server and clients can write the description of their ML models (i.e., *metadata*) to the blockchain using *metadataPublish*() function. The FL server and clients can also search the metadata on their blockchain copy by *metadataGet*(). If the metadata of a desired ML model is found, the desired ML model is then transferred directly between the server and clients by *dataGet*(). *dataGet*() configures a secure network connection to provide data transfer between server and clients. The above mentioned functions [3] are implemented as smart contracts on Hyperledger Fabric. *Note that, only the metadata of ML models are recorded in the blockchain, the actual ML models are directly transferred between server and clients via a secure network connection in our system.*

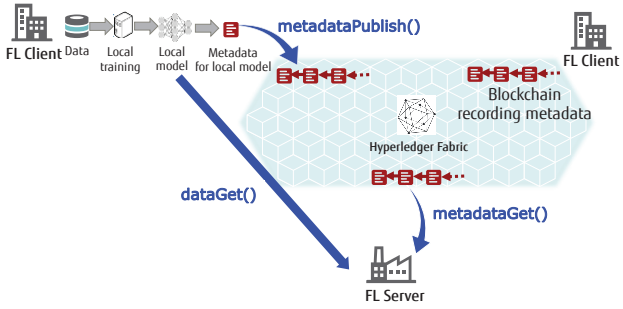


Fig. 2: Blockchain based data exchange system and functions for FL

978-1-7281-6992-7/20/$31.00 ©2020 IEEE

In addition, a new FL server and client can join FL by adding a corresponding peer node to the blockchain network using the authentication process provided on the blockchain platform.

B. Blockchain based Protocol for FL

Figure 3 shows the blockchain based protocol for FL. Steps listed at the server and clients are for a single FL training round. The metadata that describes a global/local ML model includes:

task_id: the id of a FL task (e.g., training a given ML model)
round_id: the id of a training round
node_id: the id of a server/client (e.g., the organization id)
num_samples: # of training samples in a local data set
training_accuracy: training accuracy of a global/local model
test_accuracy: testing accuracy of a global/local model
weights_file: the file name at a node storing global/local model
min_clients: min # of clients required to join each round

Metadata can also include other information, such as server/client learning rate, batch size and the number of epochs at clients, for customized ML algorithms. We can see that the metadata of ML models written to the blockchain can also provide immutable records of the FL training process.

In the beginning, the server defines an initial global model and writes its metadata to the blockchain (Step 1). The global model can be an arbitrary neural network (NN) model (i.e., NN, CNN, or RNN). The server and clients periodically search a desired model from the blockchain (Steps 2 and 7). If the desired model shows up on the blockchain (available), a client (server) proceeds to directly transfer models from the server (client) to the client (server) (Steps 4 and 9). Clients can determine to join a training round or not, and the server can also select clients to join a training round, based on the metadata. Local models that do not meet the requirement (e.g., *num_samples* is small) will not be requested to transfer to the server. After a client finishes local training or the server aggregates a new global model, the client/server publishes the metadata of the model *prior to* transferring the model (Steps 6 and 11).

At the end of each training round, *round_id* is incremented by 1 at the server when writing the metadata of a new global model to the blockchain. Clients update their local training *round_id* based on *round_id* from the metadata. Training/testing accuracy and loss of an aggregated global model are also obtained by weighted averaging over selected local models. FL training stops when the number of training rounds or the model accuracy reaches a threshold.

There are several advantages to use blockchain technologies for FL. First, the immutable records on the blockchain not only ensure that data usage is compliant, but also tracks FL training steps for auditing purposes. Second, the consensus made among

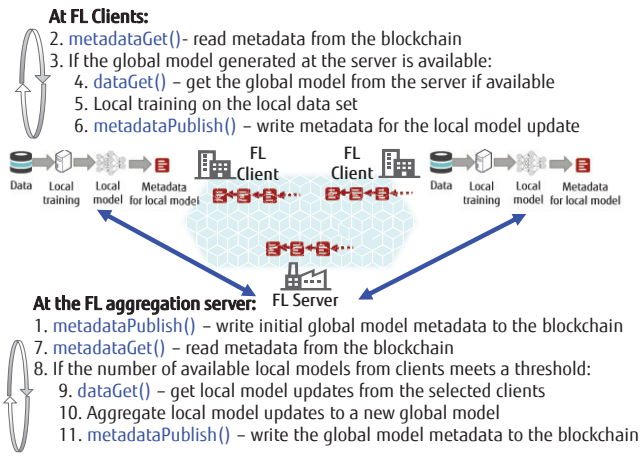


Fig. 3. Blockchain based protocol for FL

FL server and clients on blockchain can indicate the availability and the quality of ML models, enabling client selection at the server without transferring unnecessary local models to the server. Third, the security features provided on the blockchain platform can simplify the underlying network configurations for FL. For example, the authentication of FL server and clients can take advantage of the authentication process on the blockchain platform. Also, secure exchange [3] of models between FL server and clients includes hiding the actual location of models, as well as establishing HTTPS connections between server and clients, so that attackers cannot access the transferred models. Applying blockchain technologies to mitigate FL-specific attacks, e.g., honest-but-curious adversaries, will be published in the future.

Our proposed protocol is unique in that the information of desired models can be searched on the blockchain, and then the actual models can be transferred upon requests, avoiding unnecessary transfer of local models as in [4].

III. Demo Scenario

For demonstration purposes, we create one FL server and two clients for FL, each of which is running as a Docker container in a PC. Each client has a physically decentralized local data set. A blockchain network is constructed at a different HP server, in which a single channel with three node peers is created. The FL server and clients are connected to their corresponding node peers in the blockchain network. We train a NN model based on the public data set MNIST, where each client has an even partition of the data set. The initial global NN model and the number of training rounds can be defined at the server in the beginning.

We show the traces of the blockchain based protocol during FL training, in which the training/testing accuracy of models at each round are listed, as the shown in Fig. 4. We also show the blockchain containing immutable metadata records. With the even partition of data sets among clients, the accuracy of the resulting model trained by FL is very close to centralized training. Our demonstration shows that the proposed blockchain based protocol can seamlessly provide secure communication between server and clients in federated learning.

Traces at the server (*node_id* = 2):

```
read from blockchain
At round 1, clientsReceivedSet = {0, 1}, clients = {0, 1}
{'roundId': '1', 'nodeId': '0', 'numSamples': '27500', 'trainAcc': '0.71770906', 'testAcc': '0.7263', 'weightsFile': 'train_
weight_name_W_round_1_client_0.npy', 'biasFile': 'train_weight_name_B_round_1_client_0.npy', 'numClientsRequired': 'NA'}
{'roundId': '1', 'nodeId': '1', 'numSamples': '27500', 'trainAcc': '0.72705454', 'testAcc': '0.7263', 'weightsFile': 'train_
weight_name_W_round_1_client_1.npy', 'biasFile': 'train_weight_name_B_round_1_client_1.npy', 'numClientsRequired': 'NA'}
At round 1, training accuracy = 0.7223818, testing accuracy = 0.7263

write to blockchain - {"jsonrpc": "2.0", "method": "writeTransaction", "params": {"key": "55ac7414-be77-4d6c-99f2-bcba7b39f
92a", "id": "FED00020000", "data": "{\"roundId\": \"2\", \"nodeId\": \"2\", \"numSamples\": \"NA\", \"trainAcc\": \"NA\",
\"testAcc\": \"NA\", \"weightsFile\": \"train_weight_name_W_round_2.npy\", \"biasFile\": \"train_weight_name_B_round_2.npy
\", \"numClientsRequired\": \"2\"}"}, "id": "0"}
{"jsonrpc":"2.0","result":{"message":"Write Successful"},"id":"0"}

read from blockchain
At round 2, clientsReceivedSet = {0}, clients = {0, 1}

read from blockchain
At round 2, clientsReceivedSet = {0, 1}, clients = {0, 1}
{'roundId': '2', 'nodeId': '0', 'numSamples': '27500', 'trainAcc': '0.81578183', 'testAcc': '0.8258', 'weightsFile': 'train_
weight_name_W_round_2_client_0.npy', 'biasFile': 'train_weight_name_B_round_2_client_0.npy', 'numClientsRequired': 'NA'}
{'roundId': '2', 'nodeId': '1', 'numSamples': '27500', 'trainAcc': '0.8214545', 'testAcc': '0.8258', 'weightsFile': 'train_w
eight_name_W_round_2_client_1.npy', 'biasFile': 'train_weight_name_B_round_2_client_1.npy', 'numClientsRequired': 'NA'}
At round 2, training accuracy = 0.818618165, testing accuracy = 0.8258
```

Traces at a client (*node_id* = 0):

```
read from blockchain
{'roundId': '2', 'nodeId': '2', 'numSamples': 'NA', 'trainAcc': 'NA', 'testAcc': 'NA', 'weightsFile': 'train_weight_name_W_r
ound_2.npy', 'biasFile': 'train_weight_name_B_round_2.npy', 'numClientsRequired': '2'}
prio training: training accuracy: 0.8164          testing accuracy: 0.8258
after training: training accuracy: 0.8509         testing accuracy: 0.8554

write to blockchain - {"jsonrpc": "2.0", "method": "writeTransaction", "params": {"key": "55ac7414-be77-4d6c-99f2-bcba7b39f3
92", "id": "FED00020000", "data": "{\"roundId\": \"2\", \"nodeId\": \"0\", \"numSamples\": \"27500\", \"trainAcc\": \"0.8163
6363\", \"testAcc\": \"0.8258\", \"weightsFile\": \"train_weight_name_W_round_2_client_0.npy\", \"biasFile\": \"train_weight
_name_B_round_2_client_0.npy\", \"numClientsRequired\": \"NA\"}"}, "id": "0"}
{"jsonrpc":"2.0","result":{"message":"Write Successful"},"id":"0"}
```

Fig. 4. Partial traces at the server and at a client

[1] P. Kairouz, et. al., "Advances and Open Problems in Federated Learning," https://arxiv.org/abs/1912.04977

[2] L. Huang, et. al., "Patient Clustering Improves Efficiency of Federated Machine Learning to Predict Mortality and Hospital Stay Time using Distributed Electronic Medical Records," *Journal of Biomedical Informatics*, vol. 99, Nov. 2019.

[3] J. Suga and Q. Zhang, "Cross-Organizational Secure Data Exchange with Access Control using Blockchain," presented at Hyperledger Global Forum, March 2020.

[4] J. Passerat-Palmbach, et. al., "A Blockchain-Orchestrated Federated Learning Architecture for Healthcare Consortia," https://arxiv.org/abs/1910.12603