FLaS6G: Federated Learning as a Service in 6G Using Distributed Data Management Architecture

Wenxuan Ye*, Xueli An†, Xueqiang Yan‡, Mohammad Hamad*, Sebastian Steinhorst*
* TUM School of Computation, Information and Technology, Technical University of Munich
† Advanced Wireless Technology Laboratory, Munich Research Center, Huawei Technologies Duesseldorf GmbH
‡ Wireless Technology Lab, 2012 Laboratories, Huawei Technologies Co., Ltd
Email: {wenxuan.ye, xueli.an, yanxueqiang1}@huawei.com, {mohammad.hamad, sebastian.steinhorst}@tum.de

***Abstract*—Native AI support is envisioned as one of the fundamental goals driving network architecture innovation. The privacy-preserving capabilities of Federated learning (FL) make it promising in vertical applications; however, the central server-based system and lack of trusted data management limit its widespread use. In order to provide FL as a service in 6G, this work proposes a transparent and traceable data management architecture based on Distributed Ledger Technology (DLT), and enables distributed and off-chain data storage by adopting a Distributed Data Storage Entity (DDSE). We decentralize the FL central server by smart contract selecting an aggregator from a set of aggregator candidates to perform client selection and model aggregation. In addition to the system architecture, this work gives a specific implementation of FL service lifecycle management, task execution, and data upload and download procedures, which are used to realize the exchanges of model parameters between the selected clients and aggregators. The simulation result shows that the impact of introducing trusted mechanisms on the overall system time spent is related to the setting of FL, with an overhead of 15% in the worst case.**

***Index Terms*—Federated Learning (FL), Distributed Ledger Technology (DLT), Network Architecture Design, 6G**

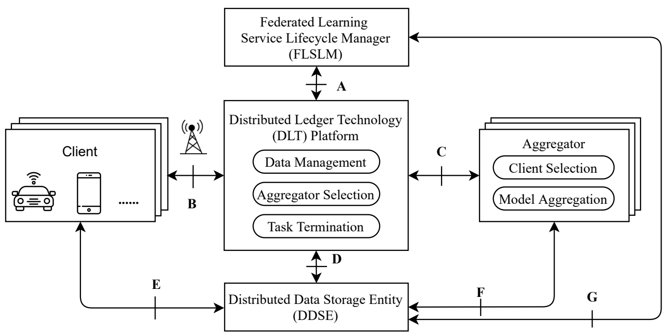

Federated Learning Service Lifecycle Manager (FLSLM)
A
Distributed Ledger Technology (DLT) Platform
Data Management
Aggregator Selection
Task Termination
Client
......
B
C
Aggregator
Client Selection
Model Aggregation
D
E
Distributed Data Storage Entity (DDSE)
F
G

Fig. 1: DLT-based FL architecture, where FLSLM undertakes FL service lifecycle management, the DLT platform assumes transparent and traceable data management, and DDSE is adopted for distributed and off-chain data storage. The smart contract selects an aggregator from a set of aggregator candidates to carry out client selection and model aggregation.

I. Introduction

With the development of 5G technologies and massive deployment of the corresponding infrastructure, communication networks are shifting from human-centric connectivity based on enhanced Mobile BroadBand (eMBB) to Internet of things based on Ultra-Reliable Low-Latency Communication (URLLC) and massive Machine Type Communication (mMTC) [1]. While 5G opens the door to the Internet of everything, 6G is expected to evolve into a platform for intelligent connectivity of everything.

One of the fundamental goals driving network architecture innovation is native AI support, where the ideal support for AI is taken into account in the design of the communication system [2]. Currently, there is still some distance from this vision, in that the central cloud serves as the primary functional center for data processing and analysis, with the communication architectures only being the channel for delivering data. However, computing in vertical industries is migrating to the edge of mobile communication systems because of the constraints of central cloud-based computing models in terms of privacy protection, latency requirements and scalability [3]. The mobile communication system, as an infrastructure with numerous terminal connections, will be a potential solution if it could enable the native support for AI.

Federated Learning (FL) is an emerging machine learning technique in which participating devices, known as *clients*, are coordinated by a *central server* to perform learning tasks, where only local model updates are shared without uploading the raw data to the server [4]. Clients are responsible for training the model with local data after downloading the global model, and then uploading the local model to the central server. The central server has three main functions. The first is to select clients to participate in the training for each global epoch, i.e., *client selection*; the second is to send configuration information of tasks to the selected clients, i.e., *training initialization*; and the final is to aggregate the local models uploaded by the clients to obtain the global model, i.e., *model aggregation* [5]. Due to the privacy-preserving nature, FL has great potential in mobile communication systems, especially in industry verticals, such as Vehicle to Everything (V2X) and Industrial Internet of Things (IIoT) [6].

Despite the great benefits mentioned, there are some issues in FL systems. It mainly adopts the central server-based network design, which may suffer from Single Point of Failure (SPOF) or Denial of Service (DoS) attacks. Besides, there is no native trustworthy management of data, including local and global models. Without proper management, there is a risk of exposure of sensitive or confidential data, resulting in personal privacy breach or financial loss [7]. Also, low-quality models can be uploaded to a central server by unreliable clients, leading to degradation or even a collapse of training. Meanwhile, data management by a single entity inherently imposes limitations on clients to participate only in tasks associated with that entity.

Trusted data management is critical to the robustness of the FL, as FL involves a large amount of intermediate data exchange between clients and aggregators during training iterations. One possible solution is to leverage Distributed Ledger Technology (DLT). The unique features of DLT, such as immutability, traceability, transparency and decentralization, make it ideal for supporting a decentralized data management system. DLT uses distributed and shared ledgers for recording transactions that are packaged in blocks with unalterable cryptographic signatures, then these blocks are linked in chronological order [8]. The data storage on immutable ledgers is referred as on-chain, and the opposite case is off-chain. To carry out trusted transactions and agreements without the need for a third-party authority, DLT utilizes smart contracts, which are self-executing computer programs or transaction protocols. Previous work, such as [9]–[12], explore trustworthiness of data management into FL systems by incorporating DLT, however, they failed to sufficiently consider privacy issues or resource efficiency.

To achieve the goal of supporting FL as a native service within mobile communication system, the main contributions of this paper are summarized as follows:

1) A native trustworthy data management architecture is proposed, where the DLT platform assumes transparent and traceable data management, Distributed Data Storage Entity (DDSE) is adopted to realize distributed and off-chain data storage.
2) Decentralization of the central server is achieved by smart contracts selecting an aggregator from multiple aggregator candidates, where the selected aggregator is responsible for implementing client selection and model aggregation.
3) In addition, this work also presents FL service lifecycle management, and gives a concrete implementation of task execution and data upload and download procedures, which are used to realize the exchanges of model parameters between the selected clients and aggregators.
4) To investigate the impact of introducing this data management mechanism on the time spent in the FL system, we set up a simulation system to simulate task execution, with Hyperledger Fabric as the DLT platform, Kademlia-based Distributed Hash Table (DHT) as DDSE and PyTorch-based FL framework.

II. PROPOSED ARCHITECTURE

A. System architecture

The novel DLT-based FL system architecture is composed of five functional entities and the corresponding seven interaction interfaces for data communication (as shown in Fig. 1): 1) Federated Learning Service Lifecycle Manager (FLSLM): responsible for lifecycle management of FL services, consisting of three phases, *service initialization*, *service Operation and Maintenance (O&M)* and *service termination*, in charge of *task preparation*, *task execution* and *task termination* respectively. Here, *task* refers to learning services involving multiple clients, and is identified by task ID, for the system may execute multiple tasks simultaneously. FL service lifecycle management is described in detail in Section. III-A. 2) Client: responsible for downloading global model parameters, training the local model with local datasets and then uploading the trained local model parameters. 3) Aggregator: responsible for model aggregation and client selection. Model aggregation refers to the aggregation of local models into a global model according to the aggregation algorithm, and client selection refers to the selection of a specified number of clients to join the model training according to the client selection strategy. 4) DDSE: responsible for raw data storage, in this novel design for storing model parameters, as well as task configuration information and client registration information introduced later on. 5) DLT Platform: responsible for three functions, namely data management, aggregator selection and task termination. Data management refers to the management of data upload and download access permission and data activity record. Aggregator selection means the selection of aggregators according to the aggregator selection strategy. Task termination means collection of all the records on the DLT platform generated by consuming the FL service. It occurs when a task termination request from FLSLM is received or the task termination criteria is satisfied, e.g., the maximum global epoch is reached.

FLSLM, DDSE and the DLT platform are considered to be native network entities in the mobile communication system. A client can be any connected terminal or device capable of data collection and model training computations, such as mobile phones, cars, robots, etc. The aggregator could be within mobile communication systems, such as in the form of network entity with capabilities for model aggregation. Alternatively, it could belong to a third-party organization, in which proper registration and authentication procedure are required in order to certify such entity to be utilized within a mobile communication system.

Distinguished from the traditional FL architecture, the principal features include: First, the introduction of the DLT platform and DDSE to perform on-chain hash storage and off-chain raw data storage, which replaces the direct communication mode between client and aggregator for model parameters exchange; Second, decentralized the central server into a number of aggregators; Last, the implementation of aggregator selection strategy and task termination criteria in the form of smart contract in the DLT platform.

B. Transaction definition

To enable reliable data management, data operations are stored as transactions in the unalterable ledger. Thus, in

TABLE I: Transaction type definition

Transaction	Symbol	Definition
Data upload	T_{up}	Upload requester requests permission to upload data.
Data download	T_{down}	Download requester requests permission to download data.
Transaction confirmation	T_{con}	DDSE sends transaction confirmation to the DLT platform after accomplishing the required operations.
Aggregator selection	T_{sel}	A smart contract deployed on the DLT platform is invoked to perform aggregator selection.
Task termination	T_{term}	A smart contract is invoked to perform task termination.

addition to the system architecture and functional entities described above, a transaction set is defined in Table. I.

Download requester refers to the entity that needs to download data, including FLSLM, clients, aggregators, network functions of the service provider network or a third-party application provider. Upload requester is for entity with needs to upload data, including FLSLM, clients or aggregators.

III. SYSTEM DESIGN

Based on the system architecture in Section. II, this section describes FL service lifecycle management and task execution. Besides, it presents a concrete implementation of data upload and download procedures, which are used to realize the exchanges of model parameters between the selected clients and aggregators.

The whole work is based on the following assumptions. First, entities of the core network of the mobile communication system or deployed by third parties are honest but curious and that clients may be malicious. Second, FLSLM has completed the initial interaction with the DLT platform, and each has obtained the communication address of the other. Third, aggregators are within the scope of mobile communication systems, or that belonging to a third party have followed a proper registration and authentication procedure, and the aggregators' registration information have been stored on the DLT platform. Similarly, clients have been authenticated by and attached to the mobile network, during which the hash and raw data of *client registration information* are stored in the DLT platform and DDSE, respectively. Given that client registration information may contain private data, it is not suitable to be stored directly in the DLT platform [13]. Last, all the procedures are described without the exceptional case, e.g., invalid permission. When the DLT platform or DDSE encounters abnormal access, it will directly return a message indicating that the access is denied.

A. Federated learning service lifecycle management

Federated learning service lifecycle management consists of three phases. Fig. 2 illustrates these phases and the corresponding task operations. The three phases are:

1) Service initialization: During this phase, FLSLM performs *task definition*, i.e., it defines the task ID and task configuration information, which includes but is not limited to model architecture, hyperparameters, initial model parameters, loss function, aggregation algorithm, global epoch, local epoch, hash algorithm and client selection strategy. Next, FLSLM deploys aggregator if needed. Then it deploys aggregator selection strategy and task termination criteria, both in the form of smart contract, on the DLT platform, which occurs in the operation called *smart contract deployment*. Finally, FLSLM uploads the task configuration information, which implements in the operation named *task configuration*.

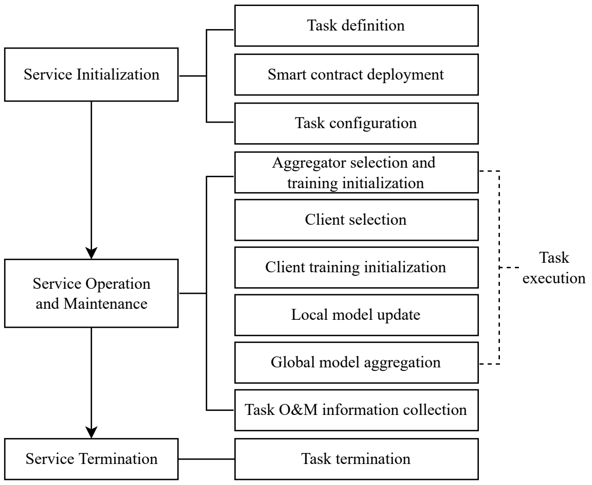

Service Initialization
Task definition
Smart contract deployment
Task configuration
Service Operation and Maintenance
Aggregator selection and training initialization
Client selection
Client training initialization
Local model update
Global model aggregation
Task O&M information collection
Task execution
Service Termination
Task termination

Fig. 2: The three phases of FL service lifecycle management and the corresponding task operations, where *task execution* takes place in service operation and maintenance.

2) Service operation and maintenance: During this phase, the *task execution*, which will be described later specifically in Section. III-B, takes place. Besides, collects information, such as participating clients and aggregators IDs, from the DLT platform and DDSE as part of the *task O&M information collection* on the basis of which further processing (e.g., data monitoring or data analysis), can be performed.

3) Service termination: During this phase, FLSLM terminates services when the pre-defined conditions are met. Then, a smart contract is invoked to perform task termination, and FLSLM collects the result.As the last step, FLSLM removes the deployed aggregator if necessary.

B. Task execution

Task execution is achieved through the operations of four entities (i.e., client, aggregator, DLT platform, and DDSE) and the interaction between them (see Fig. 3). It can be divided into several global epochs, and each global epoch consists of the following five tasks:

1) *Aggregator selection and training initialization*: In each global epoch, the DLT platform invokes the smart contract to select one aggregator. Then each aggregator will contact the DLT platform periodically to check whether it was selected or not. Afterwards, the selected aggregator checks whether it has task configuration information. If not, it needs to perform training initialization, i.e.,

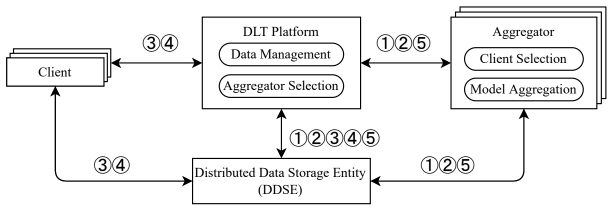


Fig. 3: Task execution in one global epoch, which is composed of five steps, namely ① Aggregator selection and training initialization, ② Client selection, ③ Client training initialization, ④ Local model update and ⑤ Global model aggregation.

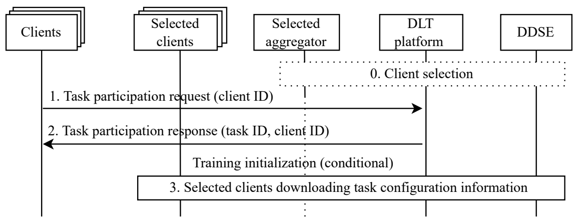
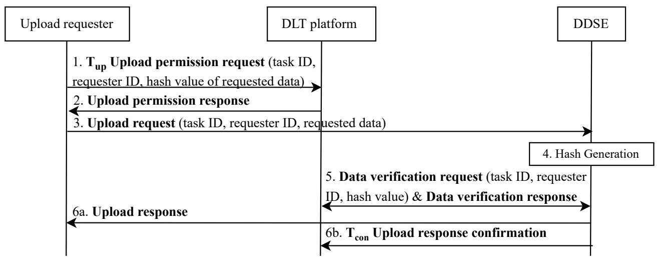
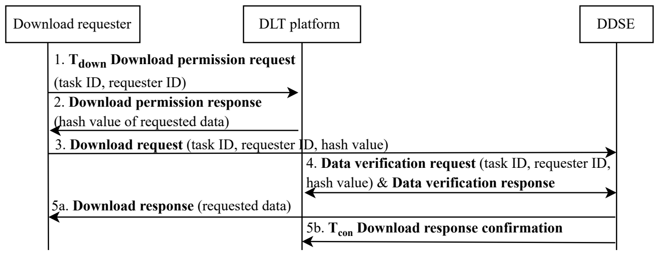

Clients
Selected clients
Selected aggregator
DLT platform
DDSE
0. Client selection
1. Task participation request (client ID)
2. Task participation response (task ID, client ID)
Training initialization (conditional)
3. Selected clients downloading task configuration information

Fig. 4: Client training initialization. After the selected aggregator performs client selection, clients periodically check in, and subsequently selected clients undergo optional training initialization, which needs to be executed only once per client for a task.


Upload requester
DLT platform
DDSE
1. T_{up} Upload permission request (task ID, requester ID, hash value of requested data)
2. Upload permission response
3. Upload request (task ID, requester ID, requested data)
4. Hash Generation
5. Data verification request (task ID, requester ID, hash value) & Data verification response
6a. Upload response
6b. T_{con} Upload response confirmation

(a) Data upload procedure


Download requester
DLT platform
DDSE
1. T_{down} Download permission request (task ID, requester ID)
2. Download permission response (hash value of requested data)
3. Download request (task ID, requester ID, hash value)
4. Data verification request (task ID, requester ID, hash value) & Data verification response
5a. Download response (requested data)
5b. T_{con} Download response confirmation

(b) Data download procedure

Fig. 5: According to the proposed trusted mechanism, data interactions need to be permitted by the DLT platform, and then data upload and download are implemented within the DDSE.

downloading task configuration information from the DLT platform and DDSE; If so, the procedure goes to the next step.

2) *Client selection*: The selected aggregator downloads client registration information, then it performs client selection according to the client selection strategy. Finally, the aggregator returns the IDs of selected clients to the DLT platform.
3) *Client training initialization*: Clients periodically communicate with the DLT platform to check the selection results. Selected clients need to check whether they have task configuration information. If not, they need to perform training initialization; If so, they directly perform the corresponding operations. The concrete procedure is illustrated in Fig. 4.
4) *Local model update*: Selected clients download the latest global model parameters, train local models with the local dataset, and then upload the local model parameters.
5) *Global model aggregation*: The selected aggregator downloads the relevant local model parameters, aggregates them in keeping with the aggregation algorithm to get the global model, then uploads the model parameters.

C. Data upload and download procedure

With the introduction of the DLT platform, data access no longer implies direct interactions between clients and the central server. To enable trusted data management, the upload requester is required to obtain authorization from the DLT platform by sending an upload permission request including hash values of the uploaded data. Similarly, the download requester needs to request for download permissions from the DLT platform. Afterwards, DDSE must implement the data validation before returning the requested data or writing down the uploaded data.

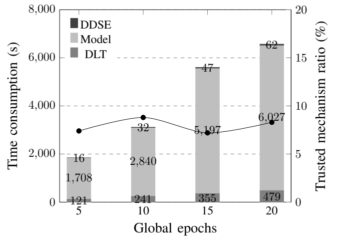
Fig. 5a describes the upload procedure, which includes the following steps:

1) The upload requester sends an upload permission request, including task ID, requester ID and hash value of uploaded data, by initiating Transaction T_{up} to the DLT platform.
2) Based on the permission of the upload requester, the DLT platform returns an upload permission response.
3) The upload requester initiates an upload request to the DDSE including task ID, requester ID and uploaded data.
4) After receiving the data to be uploaded by the requester, DDSE generates a hash value based on the hash algorithm.
5) DDSE initiates a data verification request to the DLT platform and receives the corresponding response.
6) DDSE writes the requested data down, and sends an upload response confirmation as Transaction T_{con} to the DLT platform.

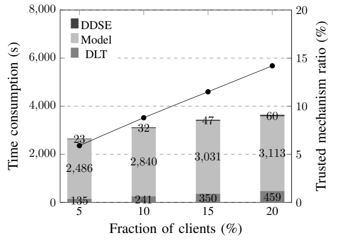
The data download procedure, which is shown in Fig. 5b, is similar to the data upload procedure. Therefore, we will describe it here.

IV. PERFORMANCE ANALYSIS

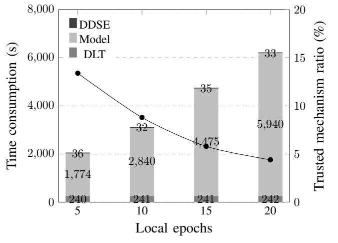
To investigate the impact of introducing a trustworthy mechanism on the overall system, a simulation system is set up to simulate the task execution introduced in Section. III-B



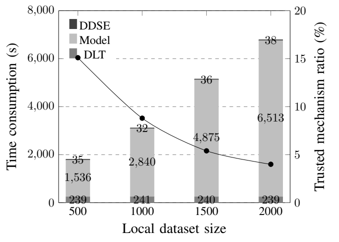
(a) Global epoch vs Time



(b) Fraction of clients vs Time



(c) Local epochs vs Time



(d) Local dataset size vs Time

Fig. 6: The consumed time in terms of (a) Global epoch, (b) Fraction of clients, (c) Local epoch and (d) Local dataset size. The impact of introducing trusted mechanisms on the overall system time spent is dependent on the parameters of FL.

where the data upload and download procedure are shown in Section. III-C.

The simulation is performed in a $64\,\mathrm{bit}$ Ubuntu 20.04 on Oracle VirtualBox Manager with 4 processors and $12\,\mathrm{GB}$ RAM, which is launched on the host machine with 1 Intel CPU ($1.08\,\mathrm{GHz}$ and 8 cores).

FL model training framework is built with the PyTorch by referring to [4]. Since the whole system focuses on data management and does not involve model training itself, it has no impact on training accuracy. Considering the need for multiple carriers to collaborate in an immutable ledger and for all participants to be authenticated, a consortium blockchain is more appropriate than a public blockchain, which is completely open to peers with no identity verification, and a private blockchain, which is under the control of a single entity. Hyperledger Fabric, as a distributed ledger application supporting consortium blockchain, is adopted as the DLT platform because of its great performance in terms of scalability, performance and trust through the execute-order-validate architecture [14]. DDSE is implemented as a DHT based on the Kademlia protocol for its remarkable search efficiency [15].

The FL goal is to predict the digit based on the learning task with the MNIST dataset, which consists of 60000 small images of handwritten single digits between 0 and 9. The training model is a Convolutional Neural Network (CNN), including convolutional layers and linear layers, with ReLU as activation function, SGD as optimizer, cross-entropy loss as loss function, and a total of parameters of 21840. We set the batch-size to 8, the learning rate to 0.01, and the total number of clients to 100 by default.

Global epochs, fraction of clients participating in each global epoch, local epochs and local dataset size are used as independent variables of this study, and time used for interacting with DHT, for interacting with DLT, and for model training locally by clients as the dependent variables. The initial settings are global epoch of 10, local epoch of 10, fraction of clients of 10, and local dataset size of 1000.

As shown in Fig. 6, each subgraph is simulated for the studied variables while ensuring that the other variables remain constant. The primary axis shows the time used by the three components and the secondary axis shows the percentage of time used by the trusted mechanism as a percentage of the system, i.e., the proportion of time spent on DLT and DDSE interactions to the total time spent. In Fig. 6a, time on DLT, model training and DDSE are linearly related to the number of global epochs because task execution is in global epochs. In each global epoch, clients and aggregators need to communicate with DLT and DDSE for a certain amount of times for model downloading and uploading, so the increase in global epochs leads to rise in time on DLT and DHT, similar to changes to model training. As a result, the proportion of time spent on trustworthy mechanisms is almost the same. Fig. 6b shows that time spent on DLT and DDSE interactions varies with the number of clients selected for each global epoch, given that a specified number of clients need to interact with DLT and DDSE. While for model training, on the other hand, time used does not vary significantly because multiple clients train locally in parallel, leading to the increase of percentage of trustworthy mechanism. Then in Fig. 6c, the number of local epochs is only concerned with the local model training, thus there is almost no influence on time spent on DDSE and DLT, and the percentage keeps decreasing, which is similar to the result of local epoch variation in Fig. 6d.

In summary, the feasibility of the proposed architecture is verified by simulations. The results show that the impact of the introduction of trusted mechanisms on the overall system time spent is related to the FL settings, with the overhead being 15% in the worst case.

V. Related Work

There are a number of papers exploring auditability and accountability on data management in FL systems by incorporating DLT.

Majeed et al. in [9] proposed a blockchain-based network architecture with the concept of channel introduced by Hyperledger Fabric, where each FL task has a dedicated channel for information logging. Clients read the global model directly from the blockchain network, train models based on local dataset, and then upload local model updates to the blockchain, after which the blockchain platform computes the global model. They gave an initial idea of how to apply DLT in data management of FL, however, model aggregation via the DLT platform implies that all local model parameters of one global epoch are stored in a single transaction, which is a big burden in terms of transaction size and storage.

Another blockchained FL architecture was presented in [10], in which each client trains the local model and updates it to the associated peer, who then generates the block and appends it to a blockchain, eventually each client downloads the block and aggregates it to get the global model. This approach actually asks each participating client to download a local model block and then to act as an aggregator, resulting in each client having access to all local models and increasing the difficulty of privacy protection.

The architecture named FL-Block introduced in [11] leverages DHT as a data storage entity, in which key-value pairs are stored into different nodes on the table based on the hash value of keys. The pointer of the global updates is stored on the blockchain and DHT is employed to store raw data off-chain, after which edge servers download the blocks and act as aggregators, then the global model is distributed to clients. Instead of storing data directly on the blockchain platform, which is inefficient for the blockchain storage and may leak private information hidden in data due to transparency, DHT-based off-chain storage is adopted in this work to ensure data unalterable. However, all the edge servers involved have access to the block containing local model update, which also poses the risk of privacy leakage, and duplicate model aggregation computation is a waste of resources.

Moreover, a solution is designed in [12] to execute FL processes in the form of smart contracts, including clients getting the latest global models and uploading local models, and aggregation of the local models, in which way trust and security are improved. This work proposes a new approach to automate FL with smart contracts, while various privacy issues or data management problems mentioned above have not been properly addressed.

Notwithstanding pioneers accomplishing reliable operational records in DLT-based data management, there are some pitfalls to the aggregation approach, since computation within smart contracts puts great pressure on the blockchain storage, and computation by clients increases the difficulty of privacy protection. It is also worth noting that client selection is a significant step in FL, and there is some research on this subject [16]. However, the current literature lacks specificity on how to implement FL, including client selection, with the proposed architecture from a system design perspective. Therefore, despite the aforementioned attempts, a concrete and practical architecture for decentralized and trustworthy FL is so far lacking in research domains.

VI. Conclusion and Outlook

Through the architecture proposed in this paper, data management is achieved in a trusted way as clients interact with the aggregators through the DLT platform, and the system stores hash values on-chain and raw data off-chain, realizing privacy protection without placing a huge burden on DLT storage. Besides, the central server is decentralized by selecting an aggregator among multiple aggregator candidates, which reduces a series of pitfalls brought by centralization. By introducing a DLT platform and a data storage entity, data management is decoupled from the aggregation approach, and the clients could participate in multiple tasks rather than the limited tasks related to the central server. It is likely to change the existing business model, where somehow client data is bound to the corresponding central server.

Based on this work, there are several interesting aspects that deserve further consideration. For example, in the aggregator selection, multiple aggregators could be selected for aggregation instead of just one to avoid excessive computational workload when a great number of local model parameters need to be combined. The specific architecture of aggregators can be implemented in the form of a hierarchy.

Acknowledgments

This work is supported by Munich Research Center, Huawei Technologies Duesseldorf GmbH and Technical University of Munich.

References

[1] ITU-R, “M.2083 IMT Vision – ‘Framework and overall objectives of the future development of IMT for 2020 and beyond’,”

[2] P. Z. Wen Tong, “6G: The Next Horizon: From Connected People and Things to Connected Intelligence,” in Cambridge University Press, 2021.

[3] Y. Mao, C. You, J. Zhang, K. Huang and K. B. Letaief, “A Survey on Mobile Edge Computing: The Communication Perspective,” in IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322-2358, Fourthquarter 2017, doi: 10.1109/COMST.2017.2745201.

[4] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, Blaise Agüera y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” in CoRR, vol. 1602.05629, 2016.

[5] K. A. Bonawitz, et al., “Towards federated learning at scale: System design,” in CoRR, vol. abs/1902.01046, 2019.

[6] Y. Liu, X. Yuan, Z. Xiong, J. Kang, X. Wang, and D. Niyato, “Federated learning for 6g communications: Challenges, methods, and future directions,” in China Communications, vol. 17, no. 9, pp. 105–118, 2020.

[7] Lian Jye Su, “Data Governance: Definitions, Challenges, And Universal Framework,” in ABi research, Feb. 2022, retrieved from https://go.abiresearch.com/lp-data-governance-definitions-challenges-and-framework?utm_source=media&utm_medium=email

[8] Nakamoto, Satoshi. (2009). “Bitcoin: A Peer-to-Peer Electronic Cash System,” in Cryptography Mailing list at https://metzdowd.com.

[9] U. Majeed and C. S. Hong, “FLchain: Federated Learning via MEC-enabled Blockchain Network,” in 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2019, pp. 1-4, doi: 10.23919/APNOMS.2019.8892848.

[10] H. Kim, J. Park, M. Bennis and S. -L. Kim, “Blockchained On-Device Federated Learning,” in IEEE Communications Letters, vol. 24, no. 6, pp. 1279-1283, June 2020, doi: 10.1109/LCOMM.2019.2921755.

[11] Y. Qu et al., “Decentralized Privacy Using Blockchain-Enabled Federated Learning in Fog Computing,” in IEEE Internet of Things Journal, vol. 7, no. 6, pp. 5171-5183, June 2020, doi: 10.1109/JIOT.2020.2977383.

[12] A. R. Short, H. C. Leligou and E. Theocharis, “Execution of a Federated Learning process within a smart contract,” 2021 IEEE International Conference on Consumer Electronics (ICCE), 2021, pp. 1-4, doi: 10.1109/ICCE50685.2021.9427734.

[13] X. Yan, X. An, W. Ye, M. Zhao and J. Wu, “A Blockchain-based Subscriber Data Management Scheme for 6G Mobile Communication System,” 2021 IEEE Globecom Workshops (GC Wkshps), 2021, pp. 1-6, doi: 10.1109/GCWkshps52748.2021.9682154.

[14] E. Androulaki, et al., “Hyperledger fabric: A distributed operating system for permissioned blockchains,” CoRR, vol. abs/1801.10228, 2018.

[15] P. Maymounkov and D. Mazieres, “Kademlia: A peer-to-peer information system based on the XOR metric”, In Proc. 1st Int. Workshop on Peer-to-Peer Systems (IPTPS), pp. 53-65, 2002-Mar

[16] Peter Kairouz, et al., “Advances and Open Problems in Federated Learning,” CoRR, vol. abs/1912.04977.