On the Decentralization of Blockchain-enabled Asynchronous Federated Learning

Francesc Wilhelmi  
Nokia Bell Labs  
Stuttgart, Germany  
francisco.wilhelmi@nokia.com

Elia Guerra  
CTTC/CERCA  
Barcelona, Spain  
eguerra@cttc.es

Paolo Dini  
CTTC/CERCA  
Barcelona, Spain  
paolo.dini@cttc.es

Abstract—Federated learning (FL), thanks in part to the emergence of the edge computing paradigm, is expected to enable collaborative learning-based applications. However, its original dependence on a central server for orchestration raises several concerns in terms of security, privacy, and scalability. To solve some of these worries, blockchain technology is expected to bring decentralization, robustness, and enhanced trust to FL. The empowerment of FL through blockchain (widely known as FLchain) however, has some implications in terms of ledger inconsistencies that lead to forks and staleness, which are naturally inherited from the blockchain’s fully decentralized operation. Such issues stem from the fact that, given the temporary ledger versions in the blockchain, FL devices may use different models for training, and that, given the asynchronicity of the FL operation, stale local updates (computed using outdated models) may be generated. In this paper, we shed light on the implications of the FLchain setting and study how decentralization in blockchain affects the age of information (AoI) and FL accuracy. To that end, we provide a faithful simulation tool that allows capturing the decentralized and asynchronous nature of the FLchain operation.

Index Terms—Age of information, blockchain, federated learning, performance evaluation

I. INTRODUCTION

Edge intelligence, by means of edge computing, is expected to bring artificial intelligence (AI) applications and optimizations closer to users by providing low-latency responses both in training and inference phases [1]. Moreover, the edge computing paradigm is expected to provide more efficient usage of resources for handling massive amounts of user-generated data with respect to traditional centralized learning approaches, typically hosted at large data centers.

One paradigm increasing in popularity to enable edge intelligence is federated learning (FL), whereby a set of participants exchange model parameters to build a collaborative model [2]. Following this approach, an FL algorithm can potentially reduce the overheads and enhance the privacy of its centralized counterpart. FL has been typically realized through the orchestration of a central server, which is responsible for gathering local updates computed by FL clients, aggregating the information to generate a new version of the global model, and distributing new models to clients. Nevertheless, alternative decentralized architectures [3] aim to address the weaknesses of the centralized FL setting [4], where the central orchestrator may entail a bottleneck and a single point of failure, thus potentially compromising performance and security.

By providing secure, immutable, and trustworthy decentralized storage, blockchain technology [5] has emerged as an appealing solution to address several issues in FL. The serverless realization of FL through blockchain, namely FLchain [6], provides trust via cryptographic proof to federated ecosystems where multiple (often unreliable) parties cooperate to train a shared model. The FLchain solution does not only address centralization issues (e.g., single point of failure) but also provides complementary mechanisms that may boost FL settings:

- In a blockchain, the control is handed over to a decentralized network, thus improving resilience and enhancing the robustness of FL applications. Furthermore, thanks to the decentralization provided by FLchain, clients can read (get the latest global model) and write (submit a local update) to the blockchain asynchronously [7], which is useful to minimize the impact of stragglers [8].
- Blockchain provides tools for enhancing trust in FL, which are essential in collaborative environments (e.g., hospitals sharing insights on their data [9]). Blockchain concatenates blocks of information through advanced cryptography techniques (e.g., hashing techniques and digital signatures), which allows for providing immutability and, thus, security. Moreover, as the distributed ledger is replicated over the entire blockchain network, high transparency and democratization are achieved.
- Blockchain can encourage participation through economic incentives enabled by smart contracts [10], thereby potentially addressing the accuracy issues created by devices refusing to engage in the training procedure (e.g., to save computational resources) [11].

Different from typical centralized and synchronous FL settings, in FLchain, each client maintains its own FL timeline according to the set of blocks it uses for keeping track of global
model updates (see Fig. 1). This is due to the decentralized operation of blockchains, whereby blockchain nodes (miners) may use inconsistent versions of the ledger. The fact is that any participating miner can generate blocks, hence different miners may come across a valid solution for appending a block with the same depth simultaneously, which would lead to temporal forks, i.e., divergent (inconsistent) versions of the ledger.

![Fig. 1. Overview of the FLchain operation.](image)

In the literature (see, e.g., [7], [12]), a widely adopted assumption is that all the FL clients in FLchain can perfectly access the blockchain’s main chain simultaneously, thus disregarding the effect of temporal ledger inconsistencies. Nevertheless, blockchains solve inconsistencies as miners progressively switch to the most adopted chain. The consensus procedure requires time (e.g., network propagation delays), so FL devices may use inconsistent models for generating local model updates.

In this paper, we focus on the ledger inconsistencies resulting from the blockchain operation, whose impact remains unclear. Such inconsistencies lead to using stale and disparate model updates during collaborative FL training. To delve into these aspects, we feature an age of information (AoI)-based metric [13], introduced as the age of block (AoB), to evaluate the freshness of model updates in FLchain and analyze its relationship with the global algorithm performance (i.e., convergence time and accuracy). The analysis is driven by the provided integration of the BlockSim simulator [14] (which characterizes any type of blockchain system) with FL through Pytorch [15]. We name this integration BlockFLSim [16], which, to the best of our knowledge, is the first of its kind. Our implementation is evaluated for a federated text recognition application, using the MNIST dataset [17].

The rest of the paper is structured as follows: Section II overviews the literature on server-less FL. Section III presents the details of the blockchain-enabled FL implementation, which deals with asynchronous model updates. Section IV provides simulation results to showcase the performance of asynchronous FL, as well as the set of implications that stem from the blockchain operation. Section V concludes the paper.

### II. RELATED WORK

The trend of distributing and decentralizing machine learning (ML) models has been embraced as an appealing solution for addressing many issues of centralization (connectivity, privacy, and security). One appealing distributed learning solution is FL [2], whereby different devices collaborate to train a model by exchanging model parameters that are obtained from local (and unshared) data. The FL framework has received a lot of attention in recent years, and its implementation in real-world applications is extensive [18]. Nevertheless, traditional FL settings still require the figure of a central server, responsible for client orchestration and model aggregation.

A prominent solution to fully decentralize FL is FLchain [6], [19], [20], where a blockchain system allows FL clients to submit and retrieve model updates without the need for a central server. Blockchain technology, beyond enabling decentralization, provides important enablers for trust, including security, privacy, and traceability, thus outperforming other existing decentralized FL solutions like the ones in [21] (based on gossip learning) or [22] (based on BitTorrent).

FLchain, thanks to the abovementioned blockchain properties, opens the door to an unprecedented way of building powerful collaborative ML models through the participation of multiple untrusted parties, and without the need for any regulating third party. In this regard, we find FLchain realizations for novel use cases like autonomous vehicles [23] or fog computing [24]. For further details on the integration of FL and blockchain, we refer the interested reader to the surveys in [25], [26].

While promising decentralization and trust, FLchain entails a set of limitations that, to date, have been barely studied. A major difference stemming from the architectural shift to decentralization is related to how model updates are contributed by FL clients and aggregated. Unlike centralized settings where a server decides the set of participating users in each FL round, in blockchain, FL clients submit model updates asynchronously [12], [27], which are added to the blockchain by the miners. Moreover, blocks may have different lengths, which depend on the blockchain configuration and its status [28], thus leading to irregular model updates in terms of the number of included local updates.

An important implication of the asynchronous learning setting is that FL devices may contribute to global training with outdated models (i.e., models that have been updated using past gradient information, or models that have not been updated at all). This poses a set of interesting challenges and trade-offs for incorporating data from slow devices. The model staleness property of asynchronous FLchain has been considered in a limited (and recent) number of contributions [7], [12], [27], [29] and it is closely related to concept drift [30], also related to data inconsistency. The concept drift has been studied for the FL setting in [31], where both temporal and spatial drifts were shown to severely affect the validity of global models. Different from the concept drift problem, where the distribution of the training data is non-stationary, in this paper, we assume static and well-balanced datasets, so that...


we can focus on the staleness generated by the blockchain operation itself.

Another relevant implication of FLchain is related to the temporal inconsistencies of the ledger across blockchain nodes, resulting from the blockchain’s partitioning due to forks. Since blockchain nodes provide global models to FL devices, forking events may lead FL nodes to train on different models. This aspect has not been examined in the literature and, typically, blockchain systems have been assumed to act as a central server with a perfect synchronization among blockchain nodes (see, e.g., [3]). In this paper, we bridge this gap in the literature and study the problem of model staleness against accuracy by introducing an AoI metric of the FL models generated at each device and distributed throughout the blockchain.

III. DECENTRALIZED ASYNCHRONOUS FLCHAIN

We adopt a Proof-of-Work (PoW)-based blockchain system, where transactions are organized in blocks (with size \( S^B \)), which are generated (mined) at regular intervals \( BI \) (e.g., every 15 seconds) [32]. In FLchain, the transactions are the local updates submitted by FL devices, which are gathered and spread throughout the blockchain’s peer-to-peer (P2P) network by a set of full blockchain nodes \( \mathcal{M} \) (with \( M = |\mathcal{M}| \)), acting as miners, as well. At this point, it is important to acknowledge that the blockchain mining procedure and the FL operation are executed in parallel, mostly independently one from the other.

When it comes to the FL operation (illustrated in Fig. 1), a dataset \( \mathcal{D} \) distributed across a set of clients \( K \) (with \( K = |K| \)) is used to train a global model (represented by the model weights \( w \)) collaboratively. Under the FLchain setting, each FL client \( k \in K \) with computational power \( \xi(k) \) and local dataset \( \mathcal{D}(k) \) of length \( |\mathcal{D}(k)| \) generates local model updates \( w(k) \) by following the next asynchronous steps (and described also in Algorithm 1):

1) Initialization with global model \( w_0 \).
2) Compute a local model update \( w(k) \), using the latest received global model. To do so, the client runs \( E \) epochs of SGD based on the target local loss function \( l(k) \) and the batch size \( B \) applied to \( |\mathcal{D}(k)| \) local data points, so \( w(k) \leftarrow w(k) - \eta \nabla l(k)(w, \mathcal{D}(k)) \), where \( \eta \) is the learning rate.
3) Submit the local update \( w(k) \) to the blockchain, to be included in a future block by a given miner.
4) Receive a new block and extract \( U \) updates for global model aggregation. Aggregation is performed following FedAvg, so that \( w \leftarrow \sum_{u=1}^{U} \alpha(u) w(u) \), where \( \alpha(u) \) is a scalar indicating the importance of model \( w(u) \).
5) Repeat steps (2)-(4) until convergence.

Regarding the freshness of global model updates in FLchain, we define the AoB as the mean peak AoI [13] across all the transactions in a block. The AoB can be computed as the mean difference between the time clients generate local updates and the time it takes to include them in a block. Formally, the AoB of block \( i \) (\( \Delta_i \)) is given by:

\[
\Delta_i = \frac{1}{U} \sum_{j=1}^{U} \delta_{i,j} = \frac{1}{U} \sum_{j=1}^{U} t_{i,j}^{(2)} - t_{i,j}^{(1)},
\]

where \( U \) is the number of client updates included in block \( i \), \( t_{i,j}^{(1)} \) is the time at which client \( j \) generates a local model update, and \( t_{i,j}^{(2)} \) is the time at which node \( j \)’s local update is included into block \( i \). It is worth noting that the AoB depends on FL clients’ communication capabilities and blockchain parameters (e.g., block size, block interval, and P2P links capacity).

IV. PERFORMANCE EVALUATION

We study the impact of the introduced AoB metric on the learning performance under different types of blockchain networks (e.g., Ethereum) and blockchain parameters (e.g., the maximum block size). The targeted FL application is based on hand-written digit recognition. For that, we define a four-layered feed-forward neural network (FNN) model (with 784, 200, 200, and 10 neurons in each layer), which is trained individually by FL clients using stochastic gradient descent (SGD) on the well-known MNIST dataset [17], whose weights are aggregated using FedAvg [33]. The MNIST dataset consists of 60,000 and 10,000 data samples for training and testing, respectively. For evaluation purposes, we have further split the test dataset into test (30%) and validation (70%).

All the simulations are done in BlockFLsim [16], which allows studying in detail the real phenomena in FLchain. Table I collects all the simulation parameters. During the simulations, we keep track of (i) the training accuracy of the main chain (i.e., for the FL devices including transactions in each block), (ii) the training loss of every FL device when

\[
\text{Algorithm 1 Dec. Asynchronous FLchain (a-FLchain)}
\]

1. Initialize: \( \eta, \alpha, B, T^k, \xi(k) \)
2. procedure MODEL_TRAINING():
3. while training do
4. Retrieve block \( b \) (with \( U \) updates) from miner \( m \)
5. Model aggregation: \( w(k) \leftarrow \sum_{u=1}^{U} \alpha(u) w(u) \)
6. for \( e = 1, \ldots, E \) do
7. \( w_{c}^{(k)} = w_{c}^{(k-1)} - \eta \nabla l_{c}^{(k)}(w, \mathcal{D}(k), B, \xi(k)) \)
8. end for
9. Send model update \( w_{E}^{(k)} \) to miner \( m \)
10. end while
11. end procedure

---

1In the literature, forks have been considered to affect the transaction confirmation time only (see, e.g., [7]), so their implications on the FL training procedure remain unclear.

2Notice that blocks are mined sequentially one after another, regardless of the incoming FL model updates.

3As a result of the IIDness properties of the considered MNIST dataset, we have opted for a simple hold-out cross-validation method. Nevertheless, other cross-validation approaches (e.g., k-Fold cross-validation) are envisioned for more complex datasets and distributions.

4For the sake of openness and reproducibility, all the source code used in this paper is open and can be accessed at https://gitlab.cttc.es/supercom/blockFLsim/-/tree/BlockFLsim (commit: 8fa5c48e).
computing local model updates, and (iii) the test accuracy of the resulting global model when the FL procedure stops.

### TABLE I
**MODEL/SIMULATION PARAMETERS**

| Parameter   | Description                  | Value                           |
|-------------|------------------------------|---------------------------------|
| $S^B$       | Max. block size              | {5, 10, 20} trans.             |
| $T_2$       | Transaction length           | 796.84 Kbits                   |
| $T_h$       | Blockheader length           | 20 Kbits                       |
| $M$         | Number of miners             | 10                              |
| $C_{p2p}$   | P2P links' capacity          | {10, ∞} Mbps                   |
| $C_r$       | FL devices links' capacity   | 1 Mbps                          |
| $BI$        | Block interval               | {5, 15, 60} s                  |
| $NB$        | Number of simulated blocks   | 50                              |
| $N$         | Number of FL devices         | {10, 50, 100}                  |
| $ξ$         | Devices comp. power          | {4.744, 83,000} MIPS           |
| $E$         | Number of local epochs       | 3                               |
| $B$         | Batch size                   | 20                              |
| $q_{fin}$   | FNN's neurons per layer      | [784, 200, 200, 10]            |
| $a$         | Activation functions         | ReLU / Softmax                  |
| $o$         | Optimizer                    | SGD                             |
| $η$         | Learning rate                | 0.01                            |
| $t_{fl}$    | Time to compute local updates| Exp($ξ$)                        |

A. **Performance in the main chain**

We evaluate the performance of FLchain in the main chain, which would match clients’ performance if the blockchain’s information could be simultaneously accessed by all the FL nodes across the network (i.e., without experiencing ledger inconsistencies). This assumption has been widely adopted in the literature but, as shown in the next subsections, does not hold in reality due to the network heterogeneity and temporal forks in the blockchain. Nevertheless, the main chain’s accuracy is a solid indicator of the overall performance.

Fig. 2 shows the main chain’s test and training accuracy for various scenarios. While Fig. 2(a) focuses on the impact of the block interval $BI$ on the learning accuracy, Fig. 2(b) illustrates the effect of the FL devices’ computational capabilities. In particular, we consider FL scenarios of $N = \{10, 50, 100\}$ clients with two different types of computational power $ξ = \{4.744, 83,000\}$ MIPS (matching Raspberry Pi 2 and Intel Core i5-2500K computational capabilities). Furthermore, we depict different types of blockchain networks with $BI = \{5, 15, 60\}$ seconds and $S^B = \{5, 10, 20\}$ transactions. Notice that the impact of $S^B$ is captured in each box of the plot. The propagation delays in the blockchain are neglected in this part of the analysis (i.e., $C_{p2p} ≈ ∞$).

As shown in Fig. 2, the test and training accuracy decreases as the number of clients increases. This is because the whole dataset is split into the considered number of devices, so FL local updates become more meaningful as $N$ decreases. Regarding the effect of $BI$ on the accuracy (see Fig. 2(a)), it has a different effect depending on the selected number of FL devices. For small $N$ values, the block interval $BI$ has a moderate effect on the test and training accuracy. However, for $N = 100$, we observe a decrease in the training accuracy as $BI$ increases, which is due to the extra delay required to incorporate all the clients’ local updates in the global model.

When it comes to the implications of using devices with different computational capabilities (see Fig. 2(b)), we observe that the best results are achieved for $ξ = 83,000$ MIPS.

**B. Age of Blocks**

We now focus on the impact of the AoB (presented in Section III) on the FL performance. Fig. 3 illustrates, for each number of FL devices $N = \{10, 50, 100\}$, both the AoB and training accuracy achieved as the main chain increases. For the sake of illustration, we show the simulations for $BI = 5s$ and $BI = 60s$, since they represent extreme cases with low and high AoB.

As it can be observed in Fig. 3, both the training and the validation accuracy increase with the main chain’s block number in all the cases. As previously observed, a low $N$ grants higher accuracy values. As for the AoB, an important conclusion is that, even if reaching very high values (e.g., as for $BI = 60s$), it has a low impact on the training accuracy. This indicates that, for the targeted FL application, untimely local updates can contribute to improving the performance of a collaboratively trained model. Another observation is that a higher AoB may be reached for high computational capabilities ($ξ$) and high block intervals ($BI$) because the clients are processing faster than the miners and the blockchain system is not able to incorporate all the local updates into blocks on time.

Next, Fig. 4 serves to further analyze the sensitivity of the AoB to blockchain parameters. Each subplot represents a different number of FL clients, whereas solid and dashed
bars refer to the two types of FL devices in terms of computational capabilities ($\xi = 4.744$ MIPS and $\xi = 83.000$ MIPS, respectively). This time, we plot the mean AoB ($\overline{\Delta}$) obtained throughout the entire simulation.

As previously illustrated in Fig. 3, Fig. 4 shows that the mean AoB increases for larger numbers of FL devices ($N$). This effect is further accentuated for $\xi = 83.000$. An important conclusion is that, due to the non-linearity shown by the AoB across different scenarios, blockchain parameters (e.g., block size, block interval) must be carefully selected to optimize the blockchain delay, which would allow minimizing the AoB. The optimization of blockchain parameters has been previously targeted in [28].

C. Blockchain inconsistencies

An important consideration of FLchain lies in the models used by individual FL devices for training. These models are retrieved from the closest miner’s latest block, which, due to the blockchain’s decentralized operation, may not be the same for all the FL clients. One remarkable reason for such type of inconsistencies lies in forks, which occur when two or more miners generate a valid block simultaneously. The acceptance of inconsistent blocks leads to forks, which create different temporary versions of the ledger. These types of forks are eventually solved thanks to the consensus protocol that states that the longest chain (the chain with the highest invested computational power) is the valid one. Nevertheless, during the FL operation, different FL devices can potentially work with different models in case forks occur.

To showcase the effect that ledger inconsistencies in the form of forks have on the FL operation, we focus on the P2P links’ capacity. In the previous subsections, we depicted the ideal case whereby blockchain delays were disregarded so that all miners were virtually synchronized. This approach has been widely adopted in the existing literature but differs from reality. To illustrate the effect of the capacity of the FL operation, Fig. 5 compares the FL training accuracy achieved during the simulation in two cases: i) the blockchain communication delays are disregarded (i.e., for $C_{p2p} = \infty$), and ii) forks can occur as a result of using a limited capacity in P2P links (i.e., for $C_{p2p} = 10$ Mbps). The block interval is fixed to $BI = 5$s (the worst case with respect to forks), while $S_B = \{5, 20\}$ transactions are considered.

As shown in Fig. 5, the ideal case ($C_{p2p} = \infty$) provides better training accuracy than the realistic one ($C_{p2p} = 10$ Mbps). Such a difference is more noticeable for $S_B = 20$ transactions (see Fig. 5(b)), which, as shown later in this section, leads to a higher probability of experiencing forks.

Finally, we focus on the impact that forks have on the test accuracy, which is obtained by evaluating the global model from the last simulated block on the test dataset. Fig. 6 shows the difference in the test accuracy for the two capacities considered above ($C_{p2p} = \{10$ Mbps, $\infty\}$). The results are shown for $BI = \{5, 15, 60\}$ s and $S_B = \{5, 10, 20\}$ transactions, and each boxplot includes the results for all the considered $\xi$ and $N$ values.

As shown in Fig. 6, the forks occurrence increases with the maximum block size ($S_B$) and for shorter $BI$ values, which allows reducing the block propagation delay directly (the most stable configuration is obtained for $BI = 60$ s and $S_B = 5$). As for the difference in the test accuracy, it increases with the fork probability, where more significant ledger inconsistencies are expected. Still, the difference is very low, which suggests a very interesting result: even in the presence of inconsistencies, FL devices can still learn collaboratively.

V. Conclusions

FLchain emerges as an appealing solution for a secure
and robust decentralized learning framework. However, the implications of blockchain decentralization on FL optimization have not been studied closely yet. In this paper, we have studied the implications of running an FL application in a blockchain, including the temporal inconsistencies and the age of information that are implicit in such a technology. To do so, we have proposed a simulation tool (BlockFLSim, an extension of BlockSim) for realistic FL chain operation. Our results show that both the age of information and model inconsistencies (forks) have a low impact on the accuracy of an FL application (trained with the MNIST dataset) running over a blockchain. This result suggests that FL is a robust framework that works effectively even if clients use outdated information or different models for collaborative training. Future work includes studying more complex datasets with unbalanced settings and non-IID data, as well as datasets varying over time. Further improvements in the definition of AoI metrics are also expected. In particular, the current AoB metric fails to capture data freshness, which is critical for the concept drift problem.

REFERENCES

[1] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, “Edge intelligence: Paving the last mile of artificial intelligence with edge computing,” Proceedings of the IEEE, vol. 107, no. 8, pp. 1738–1762, 2019.
[2] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, “Federated learning: Strategies for improving communication efficiency,” arXiv preprint arXiv:1610.05492, 2016.
[3] S. R. Pokhrel and J. Choi, “A decentralized federated learning approach for connected autonomous vehicles,” in 2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW). IEEE, 2020, pp. 1–6.
[4] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” IEEE Signal Processing Magazine, vol. 37, no. 3, pp. 50–60, 2020.
[5] I. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” Decentralized Business Review, p. 21260, 2008.
[6] H. Kim, J. Park, M. Bennis, and S.-L. Kim, “Blockchained on-device federated learning,” IEEE Communications Letters, vol. 24, no. 6, pp. 1279–1283, 2019.
[7] L. Feng, Y. Zhao, S. Guo, X. Qiu, W. Li, and P. Yu, “Baffi: A blockchain-based asynchronous federated learning framework,” IEEE Transactions on Computers, vol. 71, no. 05, pp. 1092–1103, 2022.
[8] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated learning: Challenges, methods, and future directions,” IEEE Signal Processing Magazine, vol. 37, no. 3, pp. 50–60, 2020.
[9] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, “A survey on federated learning systems: vision, hype and reality for data privacy and protection,” IEEE Transactions on Knowledge and Data Engineering, 2021.
[10] N. B. Somy, K. Kannan, V. Arya, S. Hans, A. Singh, P. Lohia, and S. Mehta, “Ownership preserving ai market places using blockchain," in 2019 IEEE International Conference on Blockchain (Blockchain). IEEE, 2019, pp. 156–165.
[11] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, “A learning-based incentive mechanism for federated learning,” IEEE Internet of Things Journal, vol. 7, no. 7, pp. 6360–6368, 2020.
[12] Y. Liu, Y. Qu, C. Xu, Z. Hao, and B. Gu, “Blockchain-enabled asynchronous federated learning in edge computing,” Sensors, vol. 21, no. 10, p. 33335, 2021.
[13] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, “Age of information: An introduction and survey,” IEEE Journal on Selected Areas in Communications, vol. 39, no. 5, pp. 1183–1210, 2021.
[14] M. Alharby and A. Van Moorsel, “Blocksim: a simulation framework for blockchain systems,” ACM SIGMETRICS Performance Evaluation Review, vol. 46, no. 3, pp. 135–138, 2019.
[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” in Advances in Neural Information Processing Systems 32. Curran Associates, Inc., 2019, pp. 8024–8035.
[16] F. Wilhelmi, E. Guerra, and P. Dini, “BlockFLsim: Blockchain Federated Learning Simulator,” 2022. [Online]. Available: https://gitlab.cttc.es/supercom/blockFLsim/-/tree/BlockFLSim
[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.
[18] M. Aledhari, R. Razrak, R. M. Parizi, and F. Saeed, “Federated learning: A survey on enabling technologies, protocols, and applications,” IEEE Access, vol. 8, pp. 140 699–140 725, 2020.
[19] U. Majeed and C. S. Hong, “Fichain: Federated learning via nec-enabled blockchain network,” in 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 2019, pp. 1–4.
[20] Y. Bao, C. Su, Y. Xiong, W. Huang, and Y. Hu, “Fichain: A blockchain for auditable federated learning with trust and incentive,” in 2019 5th International Conference on Big Data Computing and Communications (BIGCOM). IEEE, 2019, pp. 151–159.
[21] C. Hu, J. Jiang, and Z. Wang, “Decentralized federated learning: A segmented gossip approach,” arXiv preprint arXiv:1908.07782, 2019.
[22] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, “Braitorrent: A peer-to-peer environment for decentralized federated learning,” arXiv preprint arXiv:1905.06731, 2019.
[23] S. R. Pokhrel and J. Choi, “Federated learning with blockchain for autonomous vehicles: Analysis and design challenges,” IEEE Transactions on Communications, vol. 68, no. 8, pp. 4734–4746, 2020.
[24] Y. Qu, L. Guo, T. H. Luan, Y. Xiang, S. Yu, B. Li, and G. Zheng, “Decentralized privacy using blockchain-enabled federated learning in fog computing,” IEEE Internet of Things Journal, vol. 7, no. 6, pp. 5171–5183, 2020.
[25] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, “Federated learning meets blockchain in edge computing: Opportunities and challenges,” IEEE Internet of Things Journal, 2021.
[26] D. Hou, J. Zhang, K. L. Man, J. Ma, and Z. Peng, “A systematic literature review of blockchain-based federated learning: Architectures, applications and issues,” in 2021 2nd Information Communication Technologies Conference (ICTC). IEEE, 2021, pp. 302–307.
[27] R. Wang and W.-T. Tsai, “Asynchronous federated learning system based on permissionless blockchain,” Sensors, vol. 22, no. 4, p. 1672, 2022.
[28] F. Wilhelmi, S. Barrachina-Muñoz, and P. Dini, “End-to-end latency analysis and optimal block size of proof-of-work blockchain applications,” IEEE Communications Letters, pp. 1–1, 2022.
[29] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, “Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles,” IEEE Transactions on Vehicular Technology, vol. 69, no. 4, pp. 4298–4311, 2020.
[30] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” ACM computing surveys (CSUR), vol. 46, no. 4, pp. 1–37, 2014.
[31] E. Jothimurugesan, K. Hsieh, J. Wang, G. Joshi, and P. B. Gibbons, “Federated learning under distributed concept drift,” in International Conference on Artificial Intelligence and Statistics. PMLR, 2023, pp. 5834–5853.
[32] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf, and S. Capkun, “On the security and performance of proof of work blockchains,” in Proceedings of the 2016 ACM SIGSAC conference on computer and communications security. 2016, pp. 3–16.
[33] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Artificial Intelligence and statistics. PMLR, 2017, pp. 1273–1282.