

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

FlowChain: The Playground for Federated Learning in Industrial Internet of Things Environments

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:

Bellavista, P., Foschini, L., Montanari, R., Romandini, N. (2022). FlowChain: The Playground for Federated Learning in Industrial Internet of Things Environments. IEEE INTERNET OF THINGS MAGAZINE, 5(2), 78-83 [10.1109/IOTM.001.2100188].

Availability:

This version is available at: <https://hdl.handle.net/11585/899277> since: 2025-03-19

Published:

DOI: <http://doi.org/10.1109/IOTM.001.2100188>

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (<https://cris.unibo.it/>).
When citing, please refer to the published version.

(Article begins on next page)

FlowChain: the playground for Federated Learning in Industrial Internet of Things Environments

Paolo Bellavista, Luca Foschini, Rebecca Montanari, and Nicolò Romandini

Department of Computer Science and Engineering, University of Bologna, 40100 Bologna, Italy
{paolo.bellavista, luca.foschini, rebecca.montanari, nicolo.romandini}@unibo.it

ABSTRACT

Industrial Internet of Things (IIoT) lays the foundations for a new industrial revolution, the so-called Industry 4.0, in which every element, from machines to processes, is interconnected and fully automated, producing a huge amount of valuable data, crucial for making industrial processes more efficient and profitable. For this reason, it is common to run Machine Learning algorithms in order to extract useful information from these data. At the same time, due to bandwidth and privacy issues, it is often unfeasible to transfer all these large-scale data to a centralized location, where these algorithms can be performed. To address these scenarios, Federated Learning (FL) has been gaining ground in recent years. FL leaves the training data on the devices where they are produced and builds a global model by aggregating locally computed models, preserving the user privacy and avoiding overwhelming the network with unnecessary raw data. However, there are still important challenges and limitations to the application of FL in Industry 4.0, mainly due to security issues and the fact that many solutions still suffer from single-points-of-failure and bottlenecks. In this paper we present FlowChain, a framework that integrates FL with blockchain technology and Decentralized Identifiers (DIDs) to create an infrastructure that offers the possibility to easily exploit FL in Industry 4.0 scenarios. By using Smart Contract technology to automate the aggregation of partial models, it is possible to make FL fully decentralized. In addition, the fact that the blockchain is immutable and every transaction is verified and traceable makes FL secure. Furthermore, we exploit DIDs to be able to uniquely identify each element participating in industrial processes. This allows a finer-grained control of authorizations, so that only IIoT devices with a known and authorized identity can actually participate in the FL training process. Finally, we present some preliminary results that demonstrate the feasibility of the proposed approach.

Introduction

In recent decades, a fourth industrial revolution, known as Industry 4.0, has emerged. It leverages digital technology developed in recent years, such as the Internet of Things (IoT) and cyber-physical systems, to make industrial processes more efficient and secure, offering a more comprehensive, interconnected, and holistic approach to manufacturing. One of the main technological evolutions that have enabled the development of Industry 4.0 is undoubtedly the Internet of IoT. Applied to the industrial world, it is known as Industrial IoT, an evolution of the concept of the Internet of Things that closely concerns industrial processes and aims to make them more efficient and secure. The basic idea is to make the elements of industry, from production machines to energy systems, to large infrastructures, smarter by installing sensors that can collect all kinds of data, such as operating status or electricity consumption. These data are then used by to extract important and useful information about the functioning of devices and infrastructures, in order to make them more efficient, safe, and productive. To do this, the traditional approach is based on collecting produced data in a centralized location where a wide range of Machine Learning (ML) algorithms can be run to produce models, which represent what has been learned from the data. However, collecting such data for analysis into centralized storage facilities is costly and time-consuming. Another critical concern is privacy and user confidentiality as training data might contain sensitive information, and

it is a risk to transfer these data outside the devices where they are collected. Furthermore, in recent years, legal restrictions, such as the General Data Protection Regulation (GDPR), are emerging that make data aggregation practices less feasible [1]. In this scenario, FL, which distributes model training to the devices from which data originate, emerged as a promising alternative ML technique. FL can cooperatively complete ML tasks without directly sacrificing the data privacy: a ML model is trained across multiple distributed clients with local datasets and aggregated on a centralized server. However, FL still relies on sending model data to a central server to compute the global model (*Centralized Federated Learning*) and this leads to infrastructure and security issues. The central server is both a single-point-of-failure and a bottleneck in the architecture; it is also an element of fragility regarding security, as it can be compromised, resulting in a potential tampering of published models. Another important aspect is related to the trust participants place in this central entity, as it may not be independent but conditioned by individual predominant participants. Participants have to trust that the central entity is doing its job correctly. Moreover, above all, the central entity must trust the models that the participants send to it. In fact, it is possible that malicious participants try to disturb the FL process by sending partial models that have been appropriately forged. The purpose of this attack could be either to prevent the global model from converging, or to create a global model with ad-hoc vulnerabilities.

To address all above issues, this work presents FlowChain, an architecture based on blockchain and DIDs [2], able to secure and decentralize FL processes. Blockchain ensures transparency, fairness, and impartiality to help FL build trust. By design it also introduces fault tolerance by the nature of peer-to-peer (P2P) network that is more resilient in computation and improves the integrity of the entire system. The distributed nature of FlowChain grants that there is no need for any trusted central entity; trust is only placed in the cryptography at the core of the blockchain. In particular, by exploiting the Smart Contracts technology, it is possible to use the Blockchain not only as a distributed repository for partial models, but also to perform their aggregation in a fully automated way. Moreover, DIDs are a new type of globally unique identifiers that do not require a centralized registration authority because control of the identifier can be proved using cryptography. They are considered the cornerstone of the Self-Sovereign Identity approach [3]. By using this type of identifiers, FlowChain allows to identify every single IIoT device in order to create an infrastructure where only known identities can take part in the FL process. By doing so, FlowChain enables an a-priori secure environment, where each operation is tracked and linked to a precise identity, by implementing a distributed, decentralized, and secure framework to perform FL tasks, thanks to the integration of blockchain and DIDs. Finally, thanks to the integration with the Flower FL framework on the client side, FlowChain is a ready-to-deploy solution in many different real-world scenarios, including those with a large number of heterogeneous clients participating in the FL process.

Background

This section presents the technological background related to blockchain, FL and DID.

Blockchain

Blockchain is a distributed ledger composed of a chain of interconnected blocks containing tamper-proof information. The blockchain ensures the integrity of the stored blocks using intensively cryptographic techniques, such as hash and digital signature making it secure and non-counterfeit. The blocks are linked together to form a chain by referencing the hash value of the previous block and any change of the data within the blocks results in a completely different hash value, destroying the preceding chain structure. Therefore, it is impossible to tamper the data that are already stored in the Blockchain. Another essential building block of the blockchain is the consensus mechanism, aimed to validate data within the blocks and to ensure that all the nodes participating in the

blockchain store the same data. The most prevalent consensus mechanism is Proof of Work (PoW) in which nodes race to solve a computationally difficult problem to win the ability to produce the next block in a blockchain. PoW can defend against attacks as long as the total computational resources of the malicious devices are less than the sum of the honest devices. The blockchain networks can be divided into two different access models: permissionless and permissioned. In the first model, participation is public and open access: anybody can participate in the network and in the consensus process. In the second model, participation is permissioned: participants have either restriction on writing (validation), or on both reading (access) and writing. Some blockchains also support Smart Contracts, which are applications or programs that run on a blockchain. Typically, they work as a digital agreement that is enforced by a specific set of rules. These rules are predefined by computer code, which is replicated and executed by all network nodes.

Federated Learning

Federated Learning can be expressed as the decentralization of the Machine Learning. Opposed to centralized ML, in a FL scenario, the training data remain at the premises of their owners instead of transmitting them to a central location. The ML model is primarily distributed among the data holders, who train it using their private data, and then sent back to the ML model owner. Partial models are thus aggregated and then a single consolidated and improved global model is sent back to the devices. In other words, FL allows for ML algorithms to gain experience from a broad range of data sets located in different sites, by enabling multiple organizations to collaborate on the development of models, without needing to directly share secure data with each other. The authors in [4] propose a baseline algorithm, namely FederatedSGD (FedSGD), where stochastic gradient descent (SGD) is naively applied to the federated optimization problem. On each round, each client locally takes one step of gradient descent on the current model using its local data, and the server then takes a weighted average of the resulting models. The authors also propose a slightly varied version of the baseline algorithm, namely FederatedAveraging (FedAVG), that adds more computation to each client by iterating the local update multiple times before the averaging step. Most existing algorithms assume a synchronous aggregation of partial models. To be more precise, this means that in order to generate the global model it is necessary to wait for the partial models of all clients; however, recently, asynchronous approaches are also emerging. For an extensive survey on FL, interested readers may refer to [5]. Let us conclude this brief overview noting that although FL is designed to increase the privacy of sensitive data, its distributed nature inevitably increases attack surfaces. Indeed, there are many elements that can be subject to violations, from the individual training participants to the centralized entity performing the aggregation.

Decentralized Identifiers

DIDs [2] are a new type of identifier that enables verifiable, decentralized digital identity, promoted by the W3 Consortium. A DID refers to any subject (e.g., a person, organization, thing, etc.) as determined by the controller of the DID. In contrast to federated identifiers, DIDs have been designed to be decoupled from central authorities, such as identity providers and certificate authorities. Consequently, DIDs are stored in distributed ledgers such as blockchains, which are not managed by a single entity. Technically speaking, DIDs are Uniform Resource Identifiers (URIs) that associate a DID subject with a DID document allowing trustable interactions associated with that subject. The URI is a simple text string consisting of three parts: the did URI scheme identifier, the identifier for the DID method, and the DID method-specific identifier. The URI resolves to a specific DID document. The DID document can express cryptographic material, verification methods, or

services, which provide a set of mechanisms enabling a DID controller to prove control of the DID. Figure 1 shows an overview of the relationships between the various components of the DID.

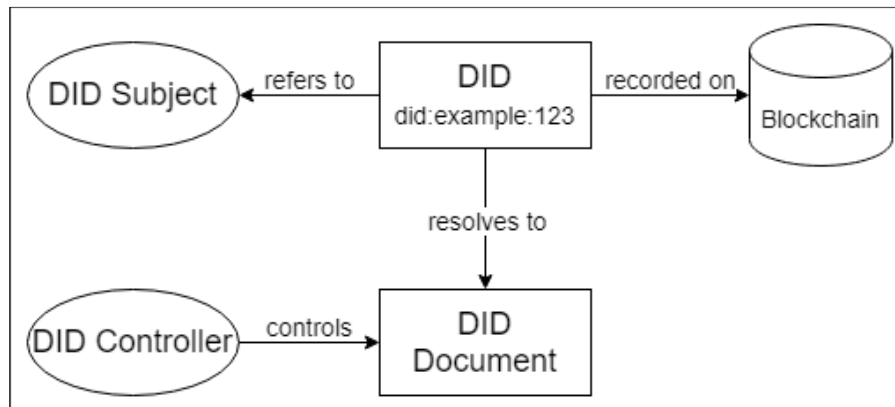


Figure 1 Overview of DID architecture and the relationship of the basic components.

FlowChain: Architecture and Implementation

FlowChain is a distributed, decentralized, and secure framework for the execution of Federated Learning tasks. These features are achieved through the exploitation of blockchain technology and Decentralized Identifiers.

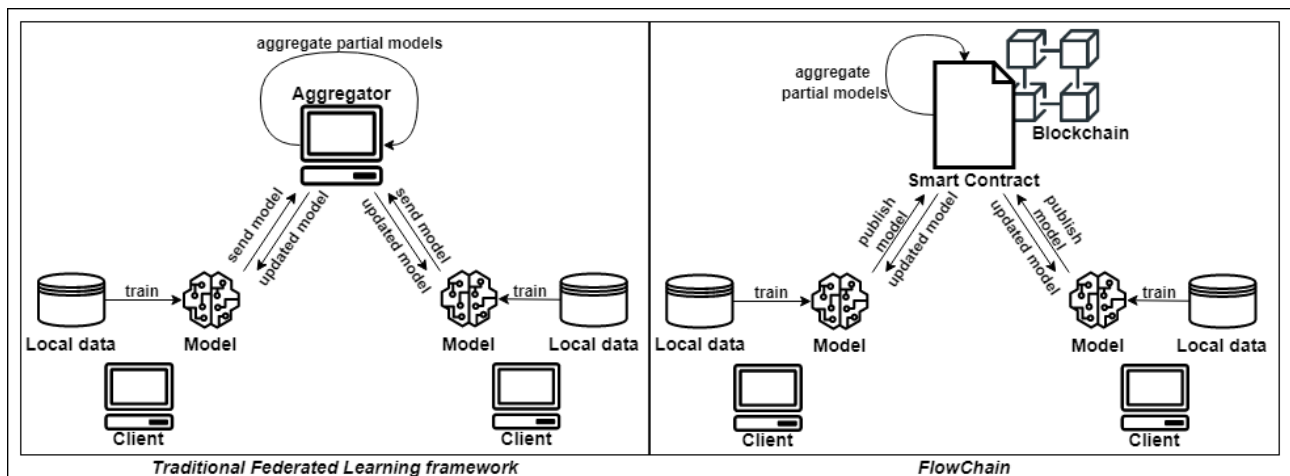


Figure 2 Comparison between the architecture of the traditional FL framework and FlowChain.

As Figure 2 shows there are significant differences between a traditional FL framework and FlowChain. The first difference is the exploitation of the blockchain technology as core building block to provide decentralization. FlowChain exploits blockchain technologies to support not only partial models' storage, but also to their aggregation in a fully automated way, by relying on Smart Contract technology. This makes the framework fully decentralized and distributed. Furthermore, as key feature, each FlowChain Client has a DID that certifies its identity. This allows us to ascertain who sent the specific partial model and to check accordingly whether or not it is entitled to participate in the FL process, thus achieving a priori security.

Architecture and Workflow

The FlowChain architecture consists of two main interacting elements: the blockchain at the core of the framework and the participants in the FL process.

FlowChain clients are the elements in charge of performing the logic of the training process. More in details, clients use their own local data to perform training of a partial model. The modalities of this training are based both on local choices, such as the algorithm for calculating gradients and optimization, and on architectural choices, such as the model aggregation algorithm. The latter, in fact, discriminates which data should be published on the blockchain, gradients or simple weights, and the frequency of publication. As it stems from Figure 2, Clients in the traditional framework and Clients in FlowChain are similar. The main difference is obviously how they interact with the entity in charge of calculating the global model, which in FlowChain is the Smart Contract component, through the execution of appropriate transactions towards the blockchain.

At the core of the architecture the blockchain replaces all the functionalities of the central entity present in traditional Federated Learning schemes, namely store all partial models and calculate the global one. In particular, a Smart Contract is deployed on the blockchain, which exposes the methods that can be invoked to execute the FL process, namely the method to publish partial models and a method to read the global model. As primary role, the FlowChain Smart Contract validates the partial models it receives, by checking that the signature on the transaction comes from a known and enabled identity. It then keeps track of these models in its internal state for their subsequent aggregation and for the generation of the global model. It is worth noticing that, given the immutability property of the blockchain, partial models can be retrieved not only from the Smart Contract state but also from the blockchain, by analyzing the sequence of all executed transactions. Another distinctive functionality of the Smart Contract is to automatically aggregate partial models to generate the global model. Toward this goal the FlowChain Smart Contract integrates several algorithms, such as FedSGD or FedAVG. To activate partial model aggregation, it is necessary to perform an explicit transaction submitted to the Smart Contract. This depends on the fact that the blockchain does not allow self-execution of Smart Contracts. In FlowChain different aggregation models can be envisioned. On the one hand, in a synchronous aggregation context, where all participants are expected to submit their models, one client could exploit the publication of its last model to perform the aggregation. On the other hand, in an asynchronous context a specific third entity can be delegated to submit the transaction enabling the Smart Contract to perform the aggregation.

The other important feature that FlowChain exhibits is the support of identity management via DID standard. Each device participating in the FL process must have an associated DID published on the blockchain so that an a priori secure execution environment can be created. In fact, by certifying the identities of the devices, it is possible both to control who can and cannot participate in the training process and to track who has published a particular partial model. Apart from access control, DIDs can also be used to keep track of information about devices involved in industrial processes, such as the status of the device, its location, and other data that may be useful. In addition, it is also possible to assign a DID to the personnel working on these devices, in order to record any interaction with them, such as installation or maintenance operations.

To be clearer and more precise, the main workflow among the FlowChain architecture components is summarized below. To simplify explanation, FedAVG is used as the aggregation algorithm, and the synchronous aggregation context is considered, which means we wait for all models from the participants before aggregating them. Furthermore, all participants possess certified identities and are therefore authorized to perform operations on the blockchain, or at least to invoke the methods of the Smart Contract.

1. The first step is for all clients to agree on the architecture of the neural model to be used. In other words, each participant must have the same starting configuration (weights and biases) with regard to the model on which to perform the training process. This can be achieved either by configuring the model statically, namely setting up all Clients with the same hard-coded model, or dynamically, where one Client publishes the initial configuration of its model, and all the others adopt it.
2. Once all participants have the model, they can start the training phase using their own local data. The training of a neural network is based on repeating the update of the network weights several times, according to the gradient of the error function. In the traditional case, training ends when a particular model accuracy threshold is reached. In our case, the number of updates depends on configuration parameters of the entire system, namely how often we want to send the updated partial model. Once the partial training of the model is complete, a transaction is performed on the blockchain, using the proper method of the deployed Smart Contract, to publish the weights of the model. Figure 3 summarizes this step.

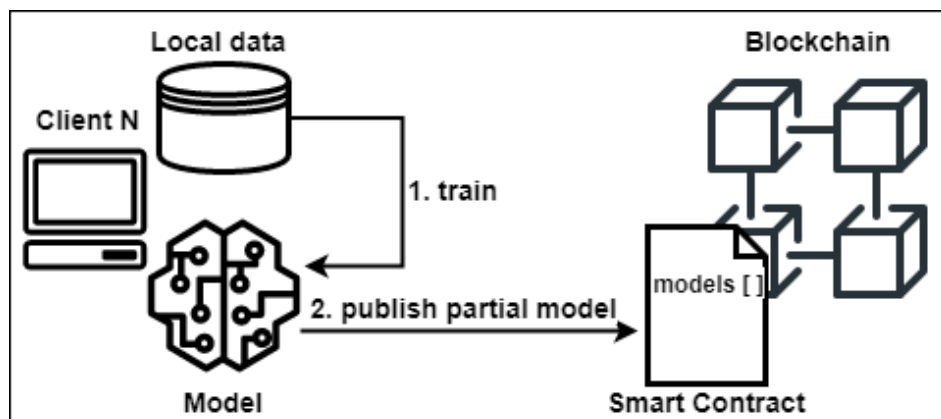


Figure 3 The Client first trains the model with its own local data (1), then publishes it on the Blockchain (2).

3. Once all participants have published their partial models, namely the weights of the networks, the Smart Contract can proceed to perform partial model aggregation (see Fig. 4). In this case, it is simply a matter of averaging all the weights. Once this is done, the Smart Contract updates the state of the blockchain, publishing the global model. This is done by saving the newly generated global model in its internal state. Different global model notification schemas can be realized on FlowChain, following a push or a pull approach. In a

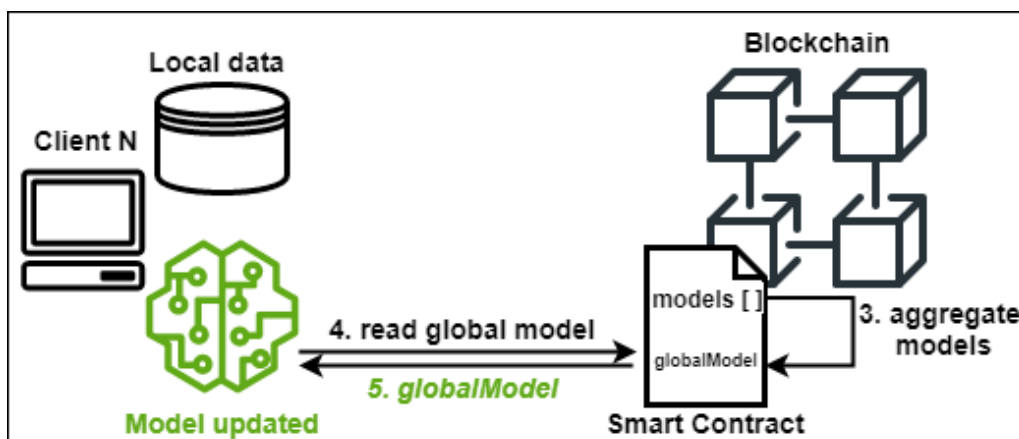


Figure 4 After receiving all partial models, Smart Contract generates a new global model by aggregating them (3). Clients can now receive the new global model (4) (5).

push approach it is possible to notify the participants of the aggregation, possibly even sending them the updated global model at the same time. In a push approach, participants are responsible for retrieving the new updated global model through the specific Smart Contract method.

4. Once participants receive the updated global template, the training process can continue. In short, the previous points in the list are repeated. Depending on the needs, the process may or may not end. For example, it may stop when the accuracy of the global model is satisfactory. Conversely, if, for example, the training data change over time, it is necessary to continue updating the global model to keep track of these changes.

Implementation¹

To implement the FlowChain blockchain component, we opted for Hyperledger Fabric, a framework that is part of the large family of Hyperledger projects supported by the Linux Foundation. It is a decentralized operating system for permissioned blockchains that can execute distributed applications (Dapps) written in general-purpose programming languages. The Fabric's modular architecture and versatile design fit into a variety of industrial distributed ledger use cases. Fabric-based blockchain applications are enterprise-grade and offer a high level of security, scalability, confidentiality, and performance. More specifically, recent work [6] has shown that in an industrial environment Fabric can support about 200 transactions per second, with an average latency of about 0.16 seconds, and up to 100,000 participants. These values are a good compromise between the benefits that the blockchain introduces, and the possible overheads involved in its use.

As first step in building our architecture we deployed a Fabric network. A Fabric network consists of nodes that can be clients that propose transactions to execute and broadcast them for ordering, peers that maintain the ledger and the state of the latter, ordering service nodes that establish the order of all the transactions. Moreover, each node must have an identity given by a membership service provider. Indeed, Hyperledger Fabric uses a public key infrastructure (PKI) to verify the actions of all network participants. Every node, network administrator, and user submitting transactions needs to have a public certificate and private key to verify their identity. These identities need to have a valid root of trust, for example a Certificate Authority (CA), establishing that the certificates were issued by an organization that is a member of the network. In our implementation we used the test network that the Fabric developers made available. This is a network that includes two peer nodes, two CAs (one per peer) and an ordering node.

After configuring and launching the Fabric network, we built and deployed the Smart Contract to run the FL process. Hyperledger Fabric allows the use of various programming languages; we adopted the Node.js SDK to implement the Smart Contract and the methods for publishing a partial model and for reading the global model. We implemented FedAVG as the adopted aggregation algorithm. This implies that the partial models are simple weights and biases of the neural networks. Furthermore, we have implemented synchronous aggregation, allowing us to automatically generate the global model following the publication of the last partial model. Hyperledger Fabric also enables to implement a push notification approach through event management. The generation of a new global model causes an event to be emitted to the registered Clients. This event also contains the new model configuration. So, having N participants, for each training cycle exactly N transactions are executed on the blockchain, one for each partial model. The N -th also includes

¹ The code for this first implementation is available at <https://github.com/nickromandini/FlowChain>

the generation of the new global model. As stated in [6], Fabric can handle a very large N without performance or scalability issues.

As mentioned above, Hyperledger Fabric is a permissioned blockchain that manages identities via PKI. Unfortunately, to date, it does not support the DID standard. However, this is not a problem since, being permissioned, anyone performing a transaction must have a known identity. In this way, the control of who can execute Smart Contract methods can be performed directly via the verification of the signature on the transaction. If using Hyperledger Fabric, the use of DIDs seem useless. However, DIDs can also be used to keep track of all devices in the environment, together with relevant information about them. For this reason, we decided to implement support for DIDs in Hyperledger Fabric anyway, pointing out, however, that this technology is not used for access control in the specific case of the Hyperledger Fabric-based implementation. To achieve this support, we relied on a project, called TrustID [7], incubated in Hyperledger Labs, the development space for new Hyperledger projects. By using a Smart Contract, TrustID implements the management of the entire lifecycle of a DID, from its creation to its update and revocation. Moreover, it allows to assign a DID also to Smart Contracts already deployed on the blockchain, so that they could be seamlessly discovered and accessed. By doing so, it is possible to use the TrustID Smart Contract to execute other Smart Contracts by exploiting the support of DIDs as identity checks.

For the devices participating in the training process, we decided to use a new framework for FL called Flower [8]. Flower provides FL infrastructure to ensure low engineering effort which enables the programmer to concentrate on his own ML use case. It is compatible with a wide range of existing and future ML frameworks, for example Tensorflow and Pytorch. Moreover, the use of Flower allows FlowChain to be ready-to-deploy in any real-world context, since Flower has been built to meet various real-world configurations with a large number of clients and to be interoperable with different operating systems and hardware platforms to work well in heterogeneous environments. In addition, Flower includes means to simulate real-world system conditions such as limited computational resources that are common for typical FL workloads. However, the integration between Flower and Hyperledger Fabric was not straightforward. In fact, Flower is implemented, as is the norm in the ML field, using Python and, unfortunately, Hyperledger Fabric does not yet have an official SDK for Python. In addition, the SDK for interacting with the TrustID Smart Contract is implemented in Node.js. For these reasons, we decided to implement the Clients as consisting of two communicating parts: one part in Python to perform the ML model manipulation procedures, and one part in Node.js for communication with the blockchain. The integration between these two parts is done via gRPC, a modern open-source high performance Remote Procedure Call (RPC) framework. By doing this, we remain consistent with the Flower framework, the TrustID project, and the official Hyperledger Fabric SDKs. Figure 5 shows a high-level

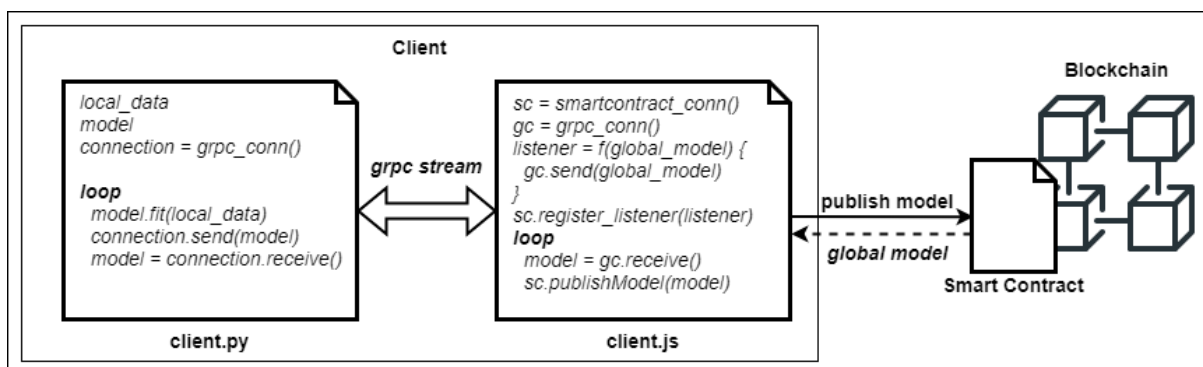


Figure 5 High-level overview of Client implementation.

view of the structure of the Clients. Compared to a traditional FL configuration, the Client-related part remains essentially very similar. For this reason, leaving aside the overhead due to the communication with the blockchain, we do not expect a difference in resource consumption between our solution and the traditional one.

Related Work

This section presents some related work related to the use of FL, starting with those related to the integration between FL and blockchain and then those related to the integration between FL and DIDs. To the best of our knowledge, there is no project that integrates FL with blockchain and DIDs.

The combination of FL and blockchain technology is being used in a variety of fields, including Internet of things [9, 10] and healthcare [11, 12]. Researchers in [13] propose a system architecture consisting of two main components: a permissioned blockchain and Federated Learning component. The blockchain is used to store transactions performed by IoT devices and peer-to-peer data sharing, which eases the auditing of data flows. The authors in [10] propose BlockFL, a blockchain-enhanced Federated Learning architecture that allows participating members to exchange updates of local device models and provides rewards for participation. BlockFL improves Federated Learning by introducing decentralization and providing a monetary incentive for blockchain network members.

Regarding the integration between FL and DIDs, two main projects have been found. The first one [14] proposes DID-eFed, a FLaaS (Federated Learning as a Service) system that encompasses the construction of a Federated Learning System and the provisioning and access management of services, leveraging the DID standard. The second research project [15] presents a privacy-preserving decentralized workflow that facilitates trusted federated learning among participants, defining a trust framework instantiated using decentralized identity technologies.

In this context, FlowChain is unique among the presented works, as it integrates both blockchain and DID technology. Moreover, as shown, the blockchain is not only a tool for storing models and information, as in the related work presented. In fact, it is also an active element in the architecture, having the task of aggregating partial models in a fully automated way.

Conclusion and Future Work

One of the cornerstones of the new Industry 4.0 paradigm is the use of IIoT. Thanks to the deployment of sensors on industrial machines and infrastructure, it is possible to obtain a large amount of data, which is crucial to extract useful information to make industrial processes safer and more efficient. Although traditional FL frameworks seem to be a valid solution for the execution of ML tasks, their application in such a distributed and uncontrolled context involves a lot of issues. To solve them, in this work we presented FlowChain, a distributed, decentralized, and secure framework for the execution of FL tasks in Industry 4.0. By using the blockchain, we are to overcome the problems associated with the centralized entity in traditional frameworks, which can be both a single point of failure and a bottleneck. Furthermore, FlowChain not only uses the blockchain for storage, as in other related shown solutions, but also for the automated aggregation of partial models using Smart Contract technology. Moreover, using DIDs, we can create a secure and trusted environment. We have also tested the validity and feasibility of our framework by means of an initial implementation, which, as shown, proved to be fully achievable and functioning. This implementation, thanks to the use of the Flower framework, is ready to be deployed in a real Industry 4.0 scenario.

As future work, we aim to study in more detail how architecture can be enriched. Indeed, FL is not only a very recent technique, but is also constantly expanding. Therefore, it will be necessary to keep up with technological advances in this field and to integrate new solutions into our architecture, such as new algorithms for aggregating partial models. Finally, an important part will be related to the fine-tuning of various parameters of the framework, such as the frequency of publication of partial models, to make it more efficient. A possible approach could be based on leveraging Reinforcement Learning (RL) algorithms, in order to make the framework also able to adapt to possible changes in the environment.

References

- [1] European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC," Official Journal, pp. 1-88, 2016.
- [2] World Wide Web Consortium (W3C), "Decentralized Identifiers (DIDs) v1.0," [Online]. Available: <https://www.w3.org/TR/did-core/>.
- [3] A. Preukschat and D. Reed, *Self-Sovereign Identity: Decentralized digital identity and verifiable credentials*, 2021.
- [4] H. B. McMahan et al., "Communication-Efficient Learning of Deep Networks from Decentralized Data," AISTATS, 2017.
- [5] P. Bellavista, L. Foschini and A. Mora, "Decentralised Learning in Federated Deployment Environments: A System-Level Survey," ACM Computing Surveys, vol. 54, no. 1, pp. 1-38, 2021.
- [6] M. Kuzlu et al., "Performance Analysis of a Hyperledger Fabric Blockchain Framework: Throughput, Latency and Scalability," 2019 IEEE International Conference on Blockchain (Blockchain), 2019.
- [7] A. de la Rocha and M. T. Nieto, "TrustID: A New Approach to Fabric User Identity Management," 2020. [Online]. Available: <https://www.hyperledger.org/blog/2020/04/21/trustid-a-new-approach-to-fabric-user-identity-management>.
- [8] D. J. Beutel et al., "Flower: A Friendly Federated Learning Research Framework," arXiv preprint arXiv:2007.14390, 2020.
- [9] D. Preuveneers et al., "Chained anomaly detection models for federated learning: An intrusion detection case study," Applied Sciences, 2018.
- [10] H. Kim et al., "Blockchained on-device federated learning," IEEE Communications Letters, 2020.
- [11] N. K. Bore et al., "Promoting Distributed Trust in Machine Learning and Computational Simulation via a Blockchain Network," IEEE International Conference on Blockchain and Cryptocurrency, 2019.
- [12] T.-T. Kuo and L. Ohno-Machado, "ModelChain: Decentralized Privacy-Preserving Healthcare Predictive Modeling Framework on Private Blockchain Networks," 2018.
- [13] Y. Lu et al., "Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT," IEEE Transactions on Industrial Informatics, 2019.
- [14] J. Geng et al., "DID-eFed: Facilitating Federated Learning as a Service with Decentralized Identities," EASE 2021: Evaluation and Assessment in Software Engineering, 2021.

- [15] P. Papadopoulos et al., "Privacy and Trust Redefined in Federated Machine Learning," Machine Learning and Knowledge Extraction, 2021.

Biographies

Paolo Bellavista (SM'06) received MSc and PhD degrees in computer science engineering from the University of Bologna, Italy, where he is now a full professor of distributed and mobile systems. His research activities span from pervasive wireless computing to location/context-aware services, from edge cloud computing to middleware for Industry 4.0 applications. He is currently the scientific coordinator of a large H2020 big data innovation action called IoTwins about distributed digital twins for the manufacturing industry. He serves on the Editorial Boards of IEEE Communications Surveys and Tutorials, ACM Computing Surveys, IEEE T. on Network and Service Management, Elsevier Pervasive Mobile Computing, and Elsevier J. on Network and Computing Applications, among the others.

Luca Foschini (SM'19) graduated from the University of Bologna, Italy, where he received a Ph.D. degree in computer science engineering in 2007. He is now an associate professor of computer engineering at the University of Bologna. His interests span from integrated management of distributed systems and services to wireless pervasive computing and scalable context data distribution infrastructures and context-aware services. Currently, he is working on mobile crowdsensing and crowdsourcing and management of Cloud systems for Smart City environments.

Rebecca Montanari graduated from the University of Bologna, where she received a Ph.D. degree in computer science engineering in 2001. She is now an associate professor of computer engineering at the University of Bologna. Her research primarily focuses on semantic-based middleware supports for service provisioning, context-aware services, security solutions for pervasive environments, policy-based service management, and adaptive and scalable middleware solutions for system and service management.

Nicolò Romandini graduated from the University of Bologna, Italy, where he received MSc degree in computer science engineering. He is currently a Ph.D. student at the Department of Computer Science and Engineering at the University of Bologna. His research focuses mainly on blockchain, cybersecurity and machine learning, and how to integrate them into IoT domains.

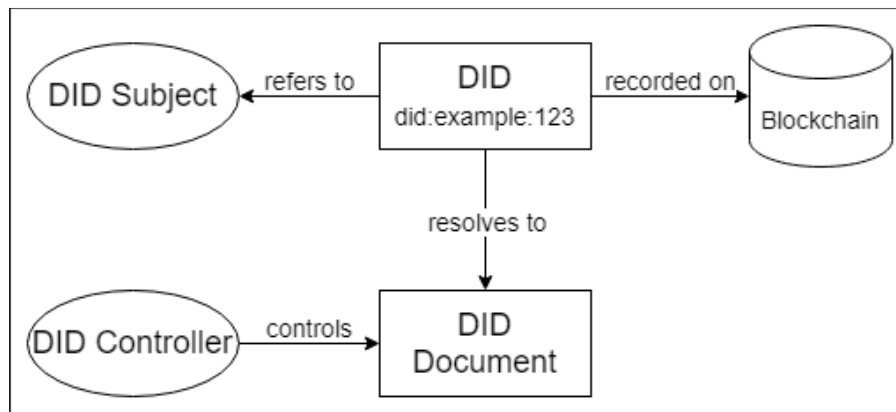


Figure 1 Overview of DID architecture and the relationship of the basic components.

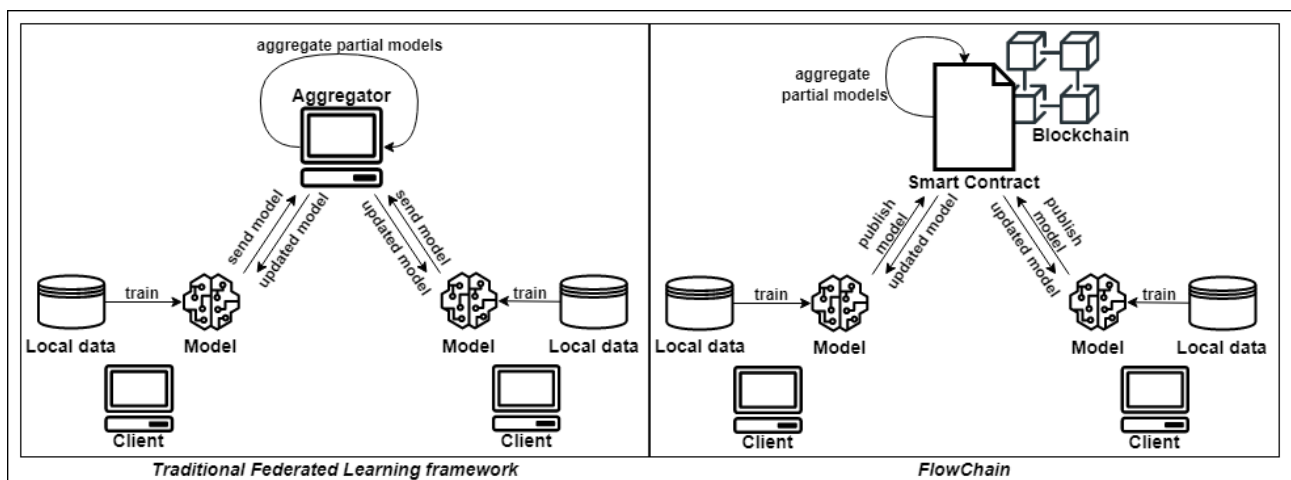


Figure 2 Comparison between the architecture of the traditional FL framework and FlowChain.

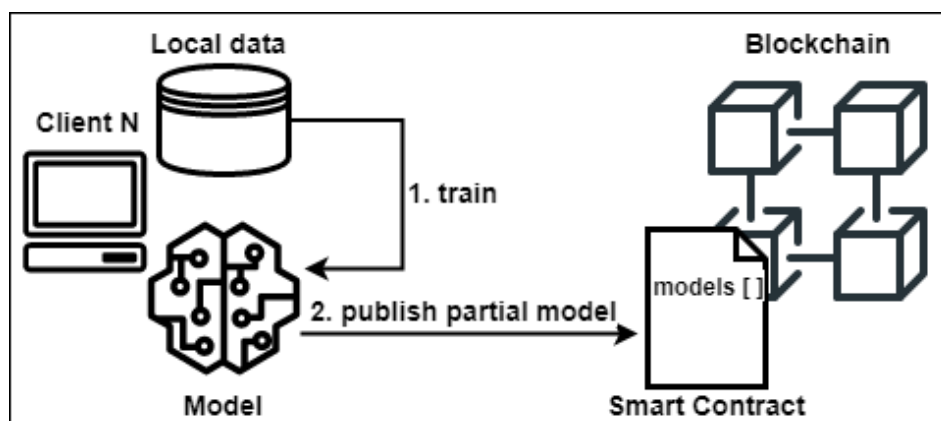


Figure 3 The Client first trains the model with its own local data (1), then publishes it on the Blockchain (2).

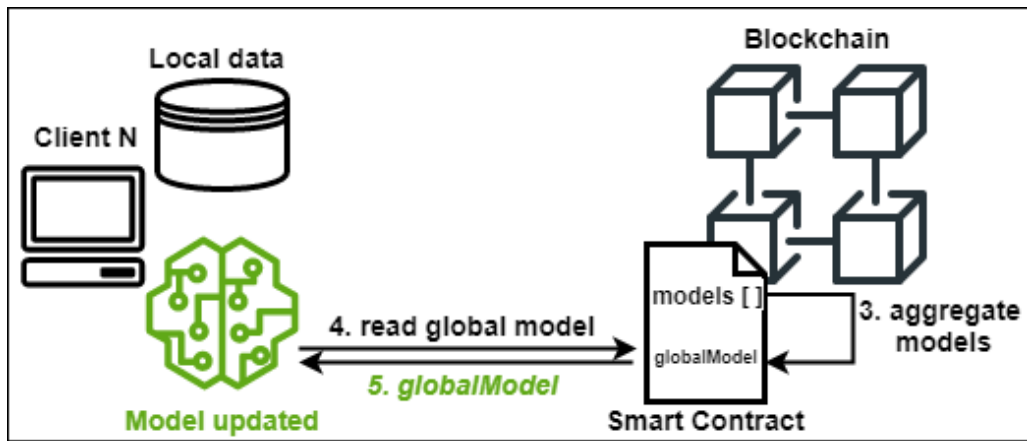


Figure 4 After receiving all partial models, Smart Contract generates a new global model by aggregating them (3). Clients can now receive the new global model (4) (5).

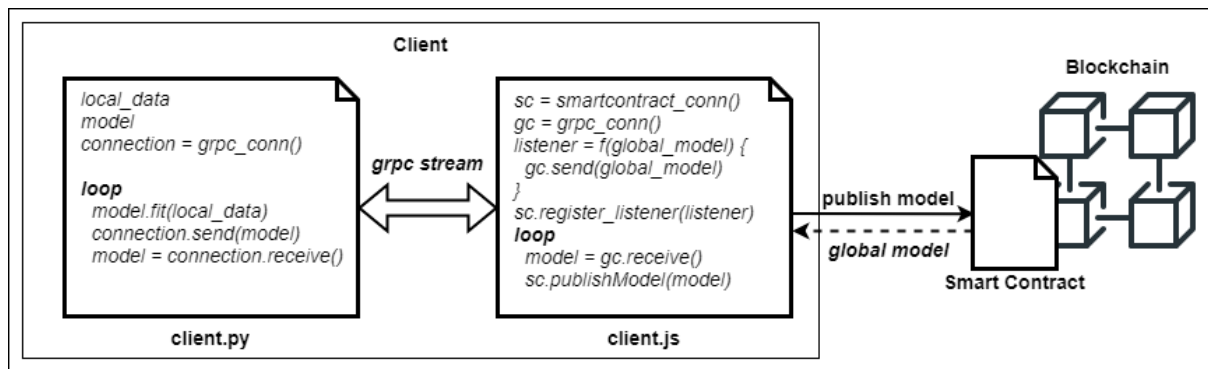


Figure 5 High-level overview of Client implementation.