ICT-56-2020 “Next Generation Internet of Things”

Grant Agreement number: 957218

IntellioT

Deliverable D3.8
Decentralized trust via secure interaction and contracts (final version)

Deliverable release date	31/03/2023
Authors	- AAU: Igor Donevski - AAU: Beatriz Soret - SANL: George Nikitakis, Antonios Paragioudakis - TSI: Charalampos Savvakos
Editor	Igor Donevski (AAU), Beatriz Soret (AAU)
Reviewer	Sumudu Samarakoon (UOULU), Konstantinos Fysarakis (SANL)
Approved by	PTC Members: (Vivek Kulkarni, Konstantinos Fysarakis, Sumudu Samarakoon, Beatriz Soret, Arne Bröring, Maren Lesche) PCC Members: (Vivek Kulkarni, Jérôme Härri, Beatriz Soret, Mehdi Bennis, Martijn Rooker, Sotiris Ioannidis, Anca Bucur, Georgios Spanoudakis, Simon Mayer, Filippo Leddi, Holger Burkhardt, Maren Lesche, Georgios Kochiadakis)
Status of the Document	Final
Version	1.0
Dissemination level	Public

Table of Contents

1 INTRODUCTION

1.1 Overview of IntellIoT framework

One of the pillars of the IntellIoT framework is trustworthiness (see the project's architecture overview in Figure 1, as defined in D2.3 ("High level architecture"). The majority of the relevant security and privacy enablers are addressed in WP4 ("Efficient, reliable and trustworthy computation & communication infrastructure"), specifically in Task 4.4 ("Trustworthy infrastructure by design"). Still, the third component of the trustworthiness, **Distributed Ledger Technologies (DLT) and smart contracts**, is specific of distributed systems and collaborative IoT, and therefore addressed in this Task 3.4 ("Decentralized trust via secure interaction & contracts"") belonging to WP3 ("Distributed, self-aware IoT applications & human-defined autonomy"). This deliverable D3.8 ("Decentralized trust via secure interaction and contracts") aims to describe the final version of the research and implementation tasks related to DLT and smart contracts.

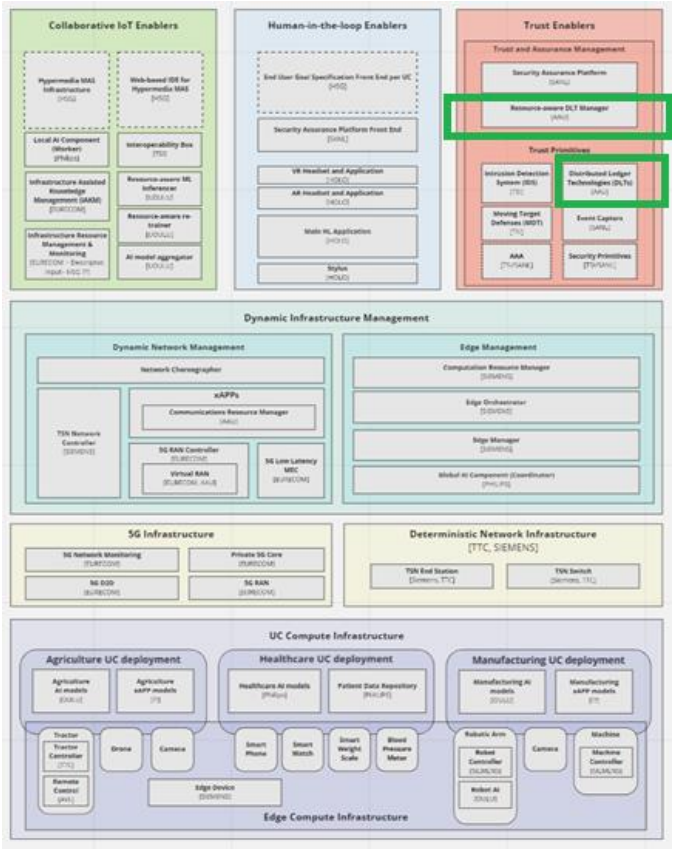


Figure 1. DLTs as a part of the core trust enablers (top right) of IntellIoT. The green boxes indicate the relevant components for this deliverable. The rest of components in the trustworthiness pillar are addressed in Task 4.4.

A DLT system offers a tamper-proof ledger that is distributed on a collection of distributed communicating nodes, all sharing the same initial block of information, the genesis block [1]. In order to publish data to the ledger, a node includes data formatted in transactions in a block with a pointer to its previous block, which relates a chain of blocks, the so called DLT. Starting with the adoption of Bitcoin in finance area, DLT has received a lot of attention in the realm of IoT, as the technology promises to help address some of the IoT security and scalability challenges [2].

For instance, in IoT deployments, the recorded data are either centralized or spread out across different heterogeneous parties. These data can be both public or private, which makes it difficult to validate their origin and

consistency. In addition, querying and performing operations on the data becomes a challenge due to the incompatibility between different application programming interfaces (APIs). Furthermore, Non-Governmental Organizations (NGOs), Public and Private sectors, and industrial companies may use different data types and databases, which leads to difficulties when sharing the data [3].

In the scope of IntellIoT project, the DLT provide a transparency, immutability, and distributed platform for distributed IoT application. DLTs solve the problem of trust between participant in a system via mathematical encryption and distributed consensus algorithms. Moreover, with the involvement of smart contracts which is considered as a key innovation of DLTs, this will bring an autonomous decision-making system with wide range of applications and areas.

1.2 Objectives and KPIs

Task 3.4, the outputs of which are documented herein, has a two-fold contribution to the IntellIoT framework:

- The **research activities** related to DLT and smart contracts for IoT environments. Specifically, the research questions that have been addressed are: (1) How to support DLT-based communication protocols in resource-constrained wireless IoT networks and what is the trade-off among trust, latency and energy?; (2) How to implement DLTs in distributed learning to generate a distributed trust sharing systems and considering resource-constrained IoT system?; (3) How to use DLT to do the valuation of data and incentivation for collaborative training and trading of data/models?.
- The **implementation** of the **DLT** software components for each of the UCs demos at the end of the project, with each UC having a different level of data privacy.

As with Task 4.4, these efforts contribute directly to Objective 4, which states the following:

Objective 4: Enable security, privacy and trust by design with continuous assurance monitoring, assessment and certification as an integral part of the system, providing trustworthy integration of third part IoT devices and services.

More specifically for Task 3.4, it contributes to the integration of third-party IoT devices and services through the integration in the architecture of the DLT paradigm. The achievement of this goal is related to the following requirement and KPI:

Requirement 1: Provide transparent and trusted distributed IoT application based on Distributed Ledger Technologies under the constraints of latency, bandwidth, computing capacities and integrate with third-party IoT services and devices.

[KPI-4.2] Delivery of at least 2 DLT implementations that can adjust the level of trust to capabilities of devices, integrate proxies, and conform to certain latency and reliability requirements, such that the level of decentralization of device participation is proportional to its computation-communication capabilities.

Task 3.4, being a key contributor of the trustworthiness pillar, contributes to the following impact KPI, too:

[KPI-i13.1] Delivery of more than three standalone and re-usable innovative security tools and technologies developed within IntellIoT, in a form ready-to-be-adopted in other domains (in addition to the domains covered by use cases).

Specifically, the IntellIoT DLT has been integrated in the security assurance platform (SAP) for recording of security and privacy-pertinent evidence from the SAP in the ledger, in a tamper-proof manner. See D4.8 for more details.

In terms of outputs, this task has provided 3 DLT implementations, one per UC, considering the communication and computation capabilities of the devices. The code of the client and the manager for each of the UCs is open and

available as described in Section 6. Besides, we have done research on DLT-based communication protocols for distributed IoT networks and the new challenges to the communication network: rather than the conventional uplink-intensive IoT traffic, we have to accommodate a more uplink-downlink balanced IoT traffic. Moreover, each IoT device becomes a DLT client with a digital identity, secured account to generate and exchange transactions in the network. The proposed DLT-based protocols can be deployed under resource-constrained IoT networks. We have also addressed the problem of scalability of centralized IoT networks by using off-chain DLT storage.

1.3 Outline and summary of modifications compared to previous deliverable (D3.4)

This deliverable has the following updates/modifications as compared to deliverable D3.4:

1. Updates in the integration with the UCs and the functional capabilities of the final products, towards the final (Cycle 2) integration, demonstration & validation of the framework.
2. The scientific advances have been expanded with regards to the previous version. For example, the novel and challenging aspect of data marketplaces is suggested and explained in the following sections.
3. The deliverable is organized in a way which accents the advancements and showcases the implementations for the integrated UCs, also emphasizing the completeness of the relevant KPI (i.e., KPI-4.2).

The rest of the deliverable is organized as follows: Section 2 explains the background of distributed ledger technologies and the process of selecting the best DLT for IntellIoT and remains unchanged with regards to the previous version of this deliverable. Section 3 has been reorganized to better show the premise, motivation, integration, and vision for the core Intelllot systems that stand to benefit from DLT. As there has been significant progress of the implementation of DLT & smart contracts for the demonstration of the three UCs, Section 4 has been upgraded to include the novel details. In addition, Section 4 expands on a couple of experimental research efforts, one of resource-constrained devices, and the second one of trustworthy and fair digital data marketplaces. Then, section 5 describes the two other research areas relevant for edge collaborative IoT: DLT implementation in distributed learning scenarios and exploitation of DLT for advanced contribution rewards systems. The link to the open source implementations is given in Section 6, and the deliverable is concluded in Sections 7.

2 BACKGROUND

2.1 Distributed ledger Technologies

The DLT is a peer-to-peer ledger that records transactions in a network [4]. The transactions represent transfer of data. The transactions are relayed on to the DLTs network, and the peers responsible for validating transactions (also called 'miners') group the transactions together into candidate blocks. The validating peer that successfully solves a complex mathematical puzzle involving the hash of the block, is selected to add its candidate block to the DLTs [5]. Each new added block contains the hash of the previous block. Linking the blocks this way makes the contents of the DLTs immutable, since any changes in a previous block are easily detectable. The remaining nodes verify that the new block is valid and update their copies of the DLTs. Every node carries the same copy of the DLTs, and 'trust-free' distributed networking is established [6].

A high-level overview of the process is shown in Figure 2. The DLT client start creating transactions which are signed with private keys and published to DLT miners. The transactions are here arranged in groups of blocks and broadcasted to all the miner in the DLT network. the initiator of a block or transaction first checks and verifies the difficulty of the block or transaction and then transmits the transaction and block through the gossip protocol. The transmission process requires the initiator to send an *inv* message to the receiver. The receiver uses the *inv* message to determine whether the block or transaction already exists locally. If the receiver does not store the block or transaction locally, the receiver will send *getdata* information to the initiator. Finally, the initiator sends the block or transaction to the receiver to complete the transmission of the block data. In the process of node interaction, the network latency mainly comes from the difficulty check of the sender and the hash verification of the block as well as the delay in the transmission of *inv* messages, *getdata* messages, and blocks or transactions between the sender and the receiver. In order to optimize the protocol of blocks and reduce the propagation delay, it was proposed in the literature [10–12] that single node optimization and pipelining of the propagation to reduce the propagation delay of DLT networks, respectively.

Among them, single-node optimization stipulates that the sender performs difficulty checking on the block, and the receiver performs hash verification on the block. The streamlining of the propagation process is to pass the block difficulty check and block hash verification to the receiver. Although these optimization schemes reduce the propagation delay to a certain extent, they also bring about other security problems and also affect the performance of the DLT network, and the effect of reducing the total network propagation delay is not obvious. Each miner has its verifier and block to ensure that the real data and the contributions of devices are updated. Each block contains a head and body parts. The blockhead contains a pointer to the next block, and the body part contains a set of validated transaction information. The local models are formed in transaction format and in order to make the solution scalable, the local models are recorded in IFPS storage, such that just a hash version of the data is recorded in the distributed ledger as shown in Figure 1.

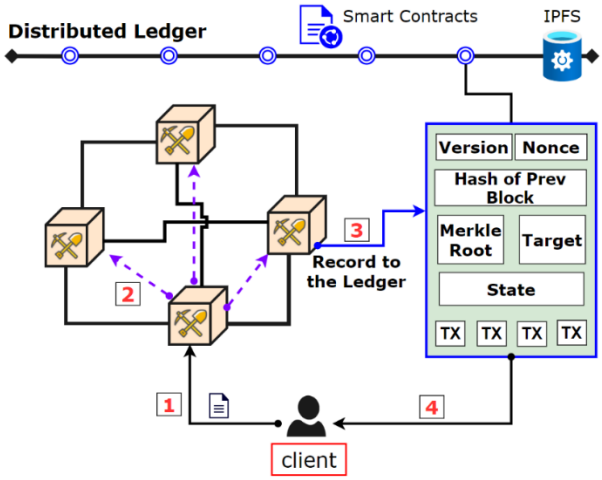


Figure 2. Distributed Ledger Technology process overview

2.2 Transaction Validation

'Proof-of-Work' is a way of authenticating new transactions into blocks by solving computational puzzles (PoW). The node that adds a new block to the DLTs in PoW is chosen based on a race to discover a nonce that, when hashed with the remainder of the block, generates a result with a specific number of leading zeroes. As a result, PoW-based consensus is computationally demanding and causes delay, both of which are undesirable on the Internet of Things.

'Proof-of-Stake' (PoS) [7] is an alternative consensus mechanism that picks the validating node for adding their candidate block in a pseudo-random manner. The likelihood of a node being picked is proportional to their percentage of the network's stake. PoS has the benefit of not requiring a lot of processing capacity to validate new transactions and add them to the DLTs. PoS, on the other hand, renders the network vulnerable to the 'nothing-at-investment' attack [8, in which nodes may simply establish forks in the chain without having any stake in it or investing a lot of processing resources. Monax and Ethereum's Casper [9], two DLT development platforms that adopt PoS, cope with nothing at stake attacks by reducing the stake of any nodes that produce bogus blocks. Because minimal processing needs are important for resource management, PoS allows us to further investigate the potential of DLTs in IoT.

Further to the above, Smart Contracts bring programmability contracts to the DLTs, in the sense that Smart Contracts execute defined rules autonomously. Smart contracts are deployed in the DLTs with specific addresses, so in order to invoke a function written in a smart contract, transactions are signed off by nodes and addressed to the smart contracts themselves. While smart contracts enforce terms and conditions for transactions in financial applications, they can enforce access control policies in the modular consortium architecture [10].

2.3 DLTs Comparison

Although a large number of DLTs are available, the most prominent platforms include Bitcoin, Ethereum, IOTA, and Hyperledger Fabric. In the following, we compare these DLTs in five different aspects: scalability, latency, throughput, security, and the level of smart contract functionalities. Scalability, latency, and throughput are deeply related and of vital importance for IoT applications [11]. For instance, the large number of sensors in smart cities may generate millions of transactions per day. This requires high efficiency of the consensus mechanism, including the way in which transactions are processed by the peers, known as endorsing peers in Hyperledger Fabric and full nodes (peers) in Bitcoin and Ethereum. Regarding latency, the transaction confirmation time must be sufficiently short to avoid queueing in the DLT and to ensure consistency in the ledgers. Bitcoin and Ethereum confirmation times per transaction are around 10 minutes and 25 seconds, respectively. These latencies might not be suitable for real-time IoT monitoring, while the confirmation time of Fabric and IOTA is much lower [12]. Note that the transaction

confirmation *time is only part of end-to-end latency, as it does not account for the communication latency at the radio access network.*

The charge of fees to process the transactions, commonly known as gas is yet another factor to consider selecting the appropriate DLT. These may greatly increase the operational costs of the network, which negatively impacts the throughput of the DLT. On the one hand, transaction fees pose a problem in massive IoT scenarios if the generation of a large number of transactions is essential. On the other hand, these fees may contribute to minimize the number of redundant transactions generated by the sensors, which in turn offloads the DLT. Among the considered DLTs, Ethereum requires fee and gas for each transaction whereas Hyperledger Fabric and IOTA provide free solutions to exchange transactions [13]. It is clear that IoT applications will involve many stakeholders with different roles, functionalities, and information with access rules, identities and security factors.

An important factor to provide security is the support for permissioned and permissionless (i.e., hybrid) solutions to validate participating nodes. Both Ethereum and Hyperledger Fabric support public and private solutions, while Bitcoin and IOTA only provide public ones. Although IoT networks, such as smart cities, may have a large number of stakeholders willing to contribute to the security of a permissionless DLT network, permissioned networks could also be beneficial. For example, in smart homes where the homeowner wants to validate the transactions via home miners or validators [14]. Regarding security, public networks may be more secure than private ones if they are able to provide transparency and distributed storage. For instance, in a permissionless DLT, the data is encrypted and stored in all the devices, which makes it definitely transparent. Besides, the more users a permissionless DLT has, the more secure it is. However, permissionless DLTs are not ideal for enterprise use, where companies deal with highly sensitive data and cannot allow anyone join their network. A permissioned DLT can be altered by its owners, making it more vulnerable to hacking [15]. In addition, permissioned DLTs provide very low or no fee for validation and a faster consensus process.

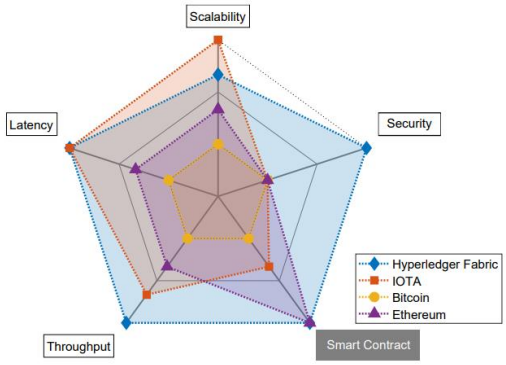


Figure 3. Comparison of different DLTs

Finally, smart contracts act as autonomous entities on the ledger that deterministically execute logic expressed as functions of the data that are stored on the ledger. Therefore, smart contracts can be established to have automatic reactions from the DLT network to specific events. For example, in case of carbon emissions, smart contracts can be used for real-time policy enforcement upon changes in the emission patterns. The smart contract feature currently is supported by Ethereum and Hyperledger Fabric (Chaincode) [16]. An IOTA smart contract type is still in progress [17]. Besides, only Hyperledger Fabric supports data confidentially via in-band encryption and guarantees the privacy of data by creating private channels. Hyperledger Fabric provides a solution with various features such as identity management, transaction integrity, and authorization with a trusted CA. These features are vital in a trusted IoT system.

The comparison of the DLTs mentioned above in these areas is illustrated in Figure 3, where each aspect has been given an abstract score based on the previous discussion. Note that the smart contract aspect is functionality rather than a strict performance indicator and can only be scored qualitatively. This makes it different to the rest of the aspects reflected in the figure, hence, it is shown in a grey background. In this IntellIoT project, because of requirements for implementation in cross-domain application, we have chosen the **Ethereum platform** to implement the system.

3 DLT-BASED COMMUNICATION PROTOCOLS AND STORAGE

In this section, we introduce and describe the DLT-based communication in resource-constrained IoT network and distributed storage solution for scalable networks. This section is updated with regards to the previous version to contain the working updates of the

3.1 DLT-based communication protocols in resource-constrained IoT

We assume a scenario where two IoT devices are connected to the peer-to-peer DLT network. The nodes of the network store the smart contracts, maintain the DLT, and perform updates when new information blocks are introduced. A nominal use of the DLT network is when a device can request to update the smart contract state, e.g., to attribute some credit to another device, as a form of documenting a transaction. Other devices, participating in the DLT network are informed of this update when they receive a new block containing the novel state. Finally, the recipient device can verify the new state and release a service to the sender.

The above mechanism is native to DLT systems and enables the exchange of credit in absence of mutual trust. This holds true since the economic transaction has been certified by a third party, i.e., the DLT network, and hardly reversible [18]. The objective is to assure that each device locally observes the same state of the smart contract, such that the key element in the system design is how each device stays synchronized with the most recent version of the DLT.

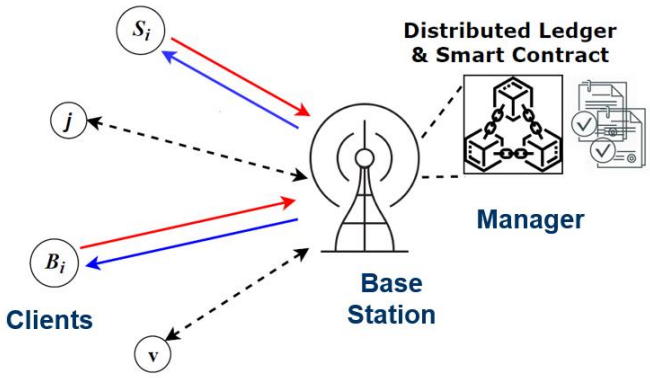


Figure 4. General architecture of DLT-based IoT networks

As anticipated by the previous version of the deliverable two main types of components are foreseen in this scheme as shown in Figure 4, where Si, Bi are two types of DLT clients that communicate with the same DLT manager via a wireless base station. In specific:

- **DLT lightweight client**: A light client or light node is a piece of software that connects to full nodes to interact with the DLT. Unlike their full node counterparts, light nodes do not need to run 24/7 or read and write a lot of information on the DLT. In fact, light clients do not interact directly with the DLT; they instead use full nodes as intermediaries. As such light clients participate in the DLT network by proxy. Thus, they rely on full nodes for many operations, from requesting the latest headers to asking for the balance of an account.
- **DLT manager**: DLT managers are nodes which have powerful capabilities to do mining process for verifying and have a copy version of entire distributed ledger. When a new set of transactions, packaged in a block, then is broadcast to the entire network by one of the nodes, every other node must verify its validity. DLT consensus is achieved when that block is added to all copies of the DLT. Note that, the DLT lightweight clients typically only download just enough DLT data to process and verify new transactions, and so their computational workload is minimal. Also note that full DLT generally represent large quantities of data.

The communication between DLT clients and DLT manager can be categorized into three protocols:

- **Protocol 1**: The DLT clients, such as Si and Bi in Figure 4, are assumed to have enough capacity to run a full copy of distributed ledger and can accomplish the synchronization process with full on-chain information. However, in realistic, the DLT client is implemented in resource-constraint devices and not capable for running a full node.
- **Protocol 2**: The DLT client Si and Bi are resource constrained devices which are not capable to run a DLT full ledger, so it just can synchronize a part of data from the distributed ledger, for example, just synchronize the header of blocks instead of synchronizing all data. Via headers of blocks, the client devices can still query the data from the ledger and guarantee about the transparent and availability of the data.
- **Protocol 3**: In this protocol, the IoT devices will not hold any fragment of data from the ledger, it only uses the open provided interface and send the raw data to the ledger.

As previously noted, the implementation for the IntellIoT use cases warrants the use of the **Ethereum** platform. And to be abler to provide device participation (in the DLT network) as proportional to the devices' computation-communication capabilities, device integrations are retained only in the form of **Protocol 3**.

The communication workflow between DLT clients and DLT manager is described in the Figure 5, to discuss the protocol overhead imposed by DLTs. A general message exchange in lightweight protocols is depicted in Fig. 5 below shown for the implementation that will encompass the needs of IntellIoT.

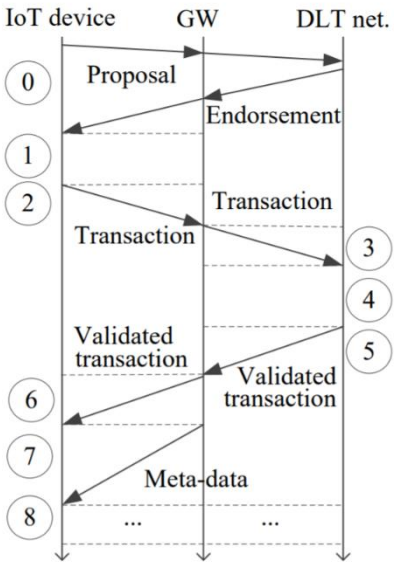


Figure 5. DLT-based IoT communication protocol

We note that some specific protocols implementations, e.g. Bitcoin, Ethereum, Fabric, and IOTA, skip some of the phases. Such implementations would incorporate the following changes in the communications protocol:

- For instance, phase 0 exists only in Fabric, serving to prepare the transaction by collecting the "endorsement", i.e., authentication, from nodes of the DLT network via the GWs.
- In phase 1, the IoT device invests energy to solve the PoW (this is only encountered in IOTA) and digitally signs the transaction (this happens in all the protocols)

- Then, in phase 2, it sends the transaction to the GW. The transaction is forwarded to the DLT network in phase 3, and included in the ledger at the end of phase 4 (this delay exists in blockchains but not in IOTA, where the PoW was done already in phase 1).
- In phase 5, the GW is informed of the validation of the transaction, subsequently informing the IoT device in phases 6 and 7. For instance, in Ethereum-like blockchains, the message in phase 6 is a receipt and the one in phase 7 contains block headers and the proof of inclusion [19]. In incentive-based protocols, step 6 may not be present, and the message in phase 7 may just contain an acknowledgment from a trusted GW. Observe that with incentive-based protocols, the amount of DLT metadata sent to the device is minimal, because it is filtered by the GWs.

3.2 Distributed Data Storage Implementation and Integration

In the IntellIoT project, the DLT platform is necessary to establish trustworthy data storage implementation that in order to also be transparent to all parties participating in the IoT network. Since this called for distributed data storage implementation such as the DLT network, the added trust and security comes at a cost in other areas. In addition to the processing and communication resource challenges, there are also data storage related challenges that must be addressed in the context of DLTs for IoT applications. More details are provided below.

Scalability Problem: Large files cannot be efficiently stored on DLTs. On one hand side, the DLT becomes bloated with data that has to be propagated within the DLT network. On the other hand, since the DLT is replicated on many nodes, a lot of storage space is required without serving an immediate purpose, especially if the node operator does not need to view every file that is stored on the DLT. It furthermore leads to an increase in the price of operating DLT nodes because more data needs to be processed, transferred, and stored.

IPFS (InterPlanetary File System) is a distributed file system that connects all computing devices to the same file system via peer-to-peer connections. Because the maximum amount of data that can be stored in a DLT block is around 1 MB, storing a big amount of data is problematic. In this project, we integrate IPFS with the whole system as a file system that permits just the hash value returned by IPFS to be kept in a smart contract, allowing a portion of the vehicle data to be uploaded.

Implementation: To develop a Distributed Application (DApp) [30] which are applications that run on a distributed P2P network rather than a single entity in a local environment more easily, a DLT network was built using Geth [22], an Ethereum RPC client that can be installed and used locally, and Truffle [20] is used as a framework for compiling and distributing smart contracts. The DApp was developed based on a web browser and a web server was built with Node.js. The DApp accesses the smart contract by communicating with the Ethereum network via JSON RPC through Web3.js [23], an Ethereum JavaScript API. In addition, the web server communicates with the IPFS network to upload data, and to access IPFS, go-ipfs [21], which can run the IPFS network in the local environment, was used. We used the free open source [21] that deals with the web browser-based DApp development tutorial using IPFS for the layout and basic implementation of a web browser-based DApp, and we implemented the system's four recording procedures to finish the system. For example, in Figure 6, we introduce an example of using IPFS for data storage in scale network.

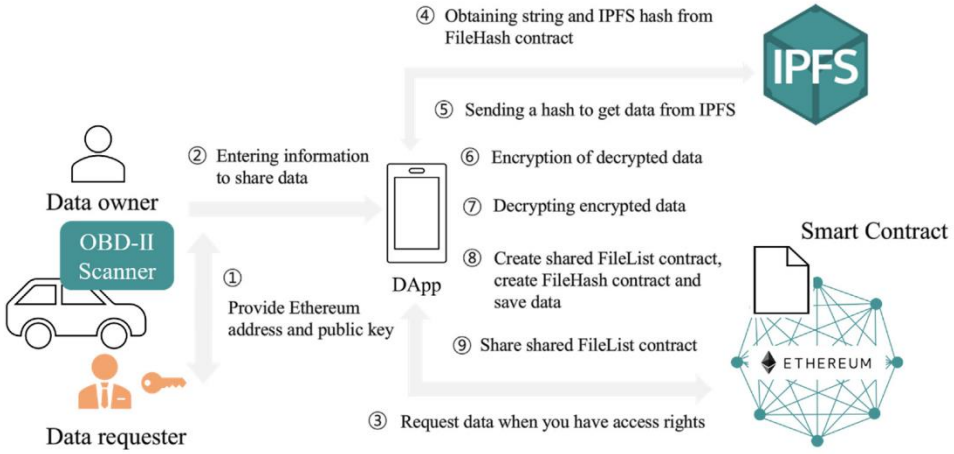


Figure 6. Example of Off-chain storage solution for DLT-based IoT networks [28]

Integration for diverse IoT platforms: As established the use of a properly managed DLT network establishes full trust and transparency of all records that were facilitated by it. As such the trust by design requirement of the IntellIoT objective has been satisfied by design through the use of DLT. However, the DLT design only guarantees trustworthiness once the client successfully submits the data to the DLT manager, that then informs the network of the updated block in the blockchain network. To satisfy trust when performing continuous assurance monitoring, assessment and certification the integration with third party IoT devices needs to facilitate trustworthy processes and methods between IoT device and DLT client.

Note that successful integration with IoT devices needs to not only play into the benefits and drawbacks of DLT usage but adapt to the devices' constraints itself. In other words, different devices have various constraints of latency, bandwidth, and computational capabilities. Moreover, depending on the device integration, implemented at device, the trustworthiness of the DLT to device connection can trade-off between implementational complexity. As such, we define three ways of integration with third party devices proposed for DLT clients:

[1] **Fully Integrated:** In the integrated DLT clients, the client code is directly injected into the device's codebase, in such a way that the entire device is a DLT client itself. In other works, the IoT device is natively made to support DLT functionalities, and the device maintainers are the ones responsible for the proper implementation of the client. Such an approach incorporates the maximum amount of trust, provides the lowest amount of latency. However, it is very inflexible and requires that the client runs at all times when the device is running. The worst drawback to this setup is however, that the client is very hard to synchronize with updates for networks with diverse IoT networks and requires device specific updates for such a client.

[2] **Containerized:** This is a flexible yet powerful type of integrating DLT clients. To elaborate, a containerized DLT client implementation is a developer managed on-device implementation of an app (such as the client) that operates internally on the same machine but operates in a virtual-like environment that is segmented from the operational system. Such an implementation requires interfacing between the client and the process generating and demanding the data from the device. This type of integration offers very high trustworthiness to the final integration process, without sacrificing much on the latency and computational capability.

[3] Remote: The most flexible type of DLT client integration that acts as a standalone process from the devices' ones. As such, this kind of integration can be applied both on-device in favour of trustworthiness, or remote from the IoT device in favour of easing the computational resources of the IoT device. This implementation is excellent for devices with low trust requirements (such as maintaining a simple logbook). And it is recommended for systems with very low computational or power resources.

In general, the fully integrated and containerized platforms operate in a poll-basis operation, where they query the device for updates and push such updates to the DLT network. The remote DLT client is more applicable for push-based interaction with the DLT network. This means that a device that sporadically pushes updates or request data from the network is recommended a remote DLT client setup. This would also contribute to the reduced on-device computation for the IoT device.

3.3 DLTs as a Key Enabler within IntellIoT's Trustworthiness Pillar

While DLT networks keep the stored data trustworthy, the DLT implementation does not guarantee the trustworthiness of data produced by the IoT device. For this, the IntellIoT project defines trust enablers associated to the DLT module. These enablers are developed within Task 3.4 and there is also a strong link and associated work being carried out in the context of Task 4.4 ("Trustworthy infrastructure by design"), as DLTs are a core part of the trust-by-design approach within the IntellIoT framework.

More specifically, these joint efforts pertain to the integration of the DLT enablers with the rest of the Trust Enablers of IntellIoT which are being developed in Task 4.4, including the Security Assurance Platform (SAP), the Trust Intrusion Detection System (Trust IDS), the Moving Target Defences (MTDs), and the multi-layer monitoring through Event Captors and the EVEREST Monitoring Engine of SAP. For a full list of IntellIoT's Trust Enablers, their specification, and their interplay, we refer the reader to deliverable D2.3 – "High level architecture (first version)", where all of these have been extensively documented.

From the above Trust Enablers, there is a direct integration between the SAP and the DLT Manager. This is done to facilitate the trust-by-design approach of IntellIoT, as it allows the recording of security and privacy -pertinent evidence from said Trust Enablers, via SAP, in the Ledger, in a tamper-proof manner. This provides an additional layer of trust for said evidence aggregated at the SAP, which may include:

- Evidence generated internally at the SAP (e.g., vulnerability or dynamic testing assessment results).
- Evidence the SAP collects from monitoring and interacting with other Trust Enablers (e.g., Trust IDS alerts, or triggered MTD strategies)
- Monitoring Evidence from monitoring the Event Captors deployed across the various layers of IntellIoT and the protected deployment.

Thus, through interaction between the SAP and the DLT Manager, the above types of Evidence are recorded in the Ledger in an automated manner, and the entries (more specifically, transaction and block IDs) are returned to the SAP, to be provided to the SAP operator for verification (e.g., in the case of any audit). As such, this SAP-DLT integration offers the highest form of physical system verification in combination with stored data verification. Therefore, IoT devices that have simultaneous direct integration with both Trust Enabler systems receive the highest form of trust and security. Additional details on this integration, and how each of the above three cases are relayed from the SAP to the DLT Manager, were provided in deliverable D4.4 – "Trust mechanisms (first version)" of Task 4.4. Said deliverable also included details on the design and development of the rest of the Trust Enablers highlighted above.

For Cycle 2, further updates took place in the integration between SAP & the DLT Manager, to support the final (refined) Use Case scenarios, as detailed in deliverable D2.4 – "Use Case specification & Open Call definition (final version)". These updates include richer (REST-based) interfacing capabilities, as well as further work on the common deployment of key DLT components through the same process (i.e., alongside) the rest of the trust enablers.

4 SEAMLESS DLT INTEGRATION WITH DIVERSE IOT NETWORKS

Below we provide details on the application of DLTs in the context of the project's different use cases and the framework itself. As part of the project there are four DLT implementations depending on the IoT device's requirements with regards to level of trust and the capabilities of said devices. Moreover, the implementations were created flexible with regards to the latency and reliability requirements of the device. This was done with the goal to adjust each device's participation to the DLT network to reflect and adjust to its computation-communication capabilities. With this, the each DLT implementation implies various levels of decentralization (the less devices it has the more centralized it is. This is a consequence of the dynamic size of the DLT network that changes with device participation (and device participation can be dynamic.). Besides the description of each implementation within the scope of IntellIoT, we also provide for each of them a vision and future lines of extension after the project finishes.

4.1 DLT for Agriculture (Use Case 1)

4.1.1 DLT AS A RELIABLE TRACTOR TRACKER

The premise: Agricultural information sharing refers to farm-related activities and administrative data across value chains with the goal of getting the correct data and information to the right person or devices at the right time. Because of the benefits, trusted data sharing in agriculture is essential for the full realization of electronic records. It has the great potential to improve food security while reducing post-harvest losses, facilitating trusted communication and data sharing among critical farm ecosystem participants to enable farm-level decision making, supply and demand visibility, and efficient, trusted, data-driven agriculture.

We carried out an in-depth investigation of the challenges of farming (e.g., tractor management) by conducting design thinking sessions with stakeholders in the ecosystem. Fragmented processes conveyed high transaction cost, complex, and wireless network communication among the participants of the small-scale farming ecosystem makes it very difficult to undertake agribusiness transactions efficiently and transparently across the value chain.

In the standard system, there are some drawbacks. First, the lack of information makes it difficult to analyze farming's development and productivity. This would especially be important in developing countries, where food security is closely linked to small-scale farming's performance. In this area, there are three options for modernization and digitalization that can considerably improve the quicker movement of commodities, information, and money [10].

Second, traceability is currently quite restricted in agricultural ecosystems, and introducing end-to-end visibility across all parts of the agriculture ecosystems can be crucial in assuring food safety and provenance. Third, monetary transactions such as trade, credit, and insurance may considerably benefit from access to data from the previous two, making these vital business operations more automated, data-driven, and paperless.

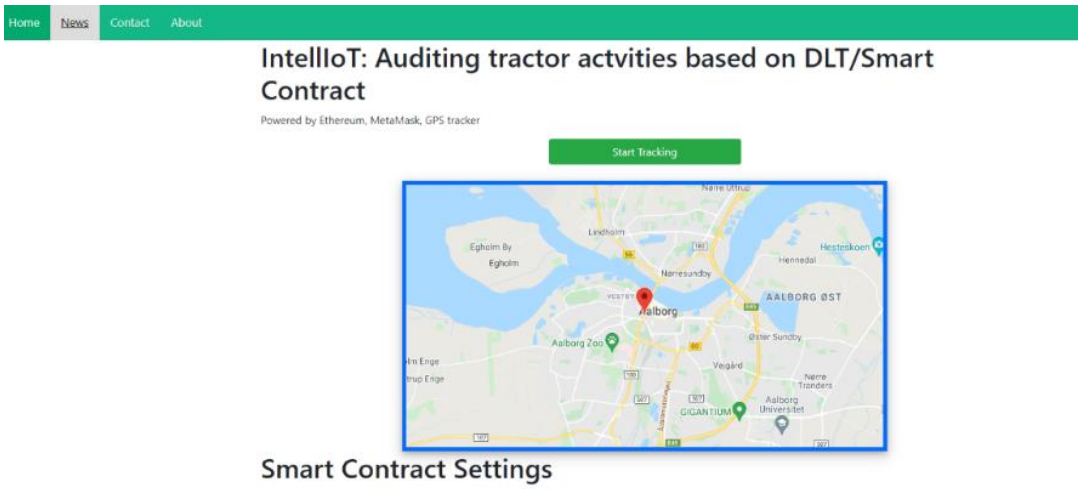


Figure 7. DLT-enabled Trust location tracking

The role of DLT: In the scope of IntellIoT, initial DLT implementations and smart contracts for recording the location GPS trajectory of the tractor will be implemented in the Tractor Controller. The GPS sensor of the tractor will record the location data via satellites and then format the raw GPS data into DLT transactions.

A lightweight version of a DLT client is setup in the tractor and communicating via secured link with the DLT manager, which is installed in the IoT infrastructure. The DLT client then signs the transaction which includes the GPS data and uploads to the distributed ledger via smart contract function to record the data. The data then will be verified and synchronized among the nodes and server in the DLT network. The GPS data will be transparent and immutable for the administrators as well as customers who rent the tractor. Each tractor owner and customer will be provided a DLT digital identity for the management and control of the tractor.

The smart contracts provide an autonomous strategy for accounting and operating the network. In the agriculture use case, a Smart Contract can manage the tractors activities, location and other statistic metrics. An example of a code snippet for a smart contract to register a tractor with the DLT system is shown below:

```
pragma solidity ^0.5.0;
import "assets.sol";

contract TractorTracker {
    string id;

    function setId(string serial) public {
        id = serial;
    }

    function getId() public constant returns (string) {
        return id;
    }
}
```

Besides, we provide a DLT-based management system for the tractor, where users can manage the tractor via GUI, which can interact with DLT network. Figure 7 shows a proof of concept for the trusted DLT-based GPS tracking.

Integration with DLT: Having defined the premise and the role of DLT in the tractor setting, the next step is to cover the method of integration with the tractor client. The DLT and tractor client operate in a hybrid setting that can be incorporated in either of the three abovementioned integrations. In detail, it is a non-containerized python client that can be directly integrated into the tractor one, containerized, or simply used as a remote client. In the current state,

the integration was done in a remote setting, where the tractor client is on a different device, accessed by HTTP, and the URL of the tractor client is called as:

```
parser = argparse.ArgumentParser(description="Poll coordinates from AVL")
parser.add_argument("--avl_url",  type=str, help="Assign the URL of the AVL platform", 
```

Once, the client is run, it probes the DLT manager with a test transaction. It establishes connection to the HTTP server of the tractor to pull the coordinates.

```
        def send_tx():
                try:
                        response = requests.request("GET", url, data=payload, verify=False)
                except RuntimeError as error:
                        print("Failed to get coordinates. Please try again...")
                        print(error)

                        return
```

Afterwards, it encodes the data and sends it to the DLT manager, after which it receives a confirmation of transaction and prints the following:

```
==============================================================================
{"seq":10431304,"latitude":48.097100550335135,"longitude":14.350143418151582,"heading":0.0}
Transaction:
{'value': 0, 'gas': 22820, 'gasPrice': 1, 'chainId': 1337, 'nonce': 24, 'to': '0xDca4261fB51136E65700Ef6
3fb7f18943A658094', 'data': '0x0b06b91f0000000000000000000000000000000000000000000000000000000000000200
000000000000000000000000000000000000000000000000000000005b7b22736571223a31303433313330342c226c61746
974756465223a34382e30393731303035353033333353133352c226c6f6e676974756465223a31342e3335303134333431383131353
13538322c2268656164696e67223a302e307d0000000000'}
------------------------------------------------------------------------------
Successfully sent to Ethereum
gas used: 22820
contract address: 0xDca4261fB51136E65700Ef63fb7f18943A658094
tx: 0xc931369944c651267e5e6020532132575dcd9c1b5f4329b81bcdc1575a8627bb
block: 0x39a9ffe6567cc706bbd70c164abb4b4caf14d54d0604d665cf8c9a87723e036b
==============================================================================
```

On the DLT manager, we can visually verify the functioning integration as:

```
ganache          | eth_chainId
ganache          | eth_estimateGas
ganache          | eth_getBlockByNumber
ganache          | eth_getBlockByNumber
ganache          | eth_getBlockByNumber
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_getBlockByHash
ganache          | eth_chainId
ganache          | eth_sendRawTransaction
ganache          |
ganache          |   Transaction: 0xc931369944c651267e5e6020532132575dcd9c1b5f4329b81bcdc1575a8627bb
ganache          |   Gas usage: 22820
ganache          |   Block Number: 25
ganache          |   Block Time: Wed Feb 08 2023 13:45:54 GMT+0000 (Coordinated Universal Time)
ganache          |
ganache          | eth_getTransactionReceipt
```

Vision and Future: Once the block has been submitted to the DLT network, the coordinates are stored across all participating DLT devices. In such a case, a voluntary auditor can take participation in the DLT network and inspect the entire history of usage of the tractor. As such the verification process not only been made more secure and tamperproof, but the inspection process is easier to facilitate. As such the vision for this particular use-case has been

fulfilled. This, however, does inspire a future outlook onto including more data, or the passing of more compressed version of the data in order to save DLT resources.

4.1.2 DLT AS A VERSITILE AND TRANSPARENT FARMER'S LOGBOOK

The premise: Oftentimes, agricultural equipment is smaller and has lesser computational capabilities opposed to the previous example from the tractor. Such an example is the autonomous self-navigating all-terrain ground vehicle as the Greenbot vehicle from iKnowHow partners in IntellIoT. In more detail, the robot is fitted with a robotic arm that is used for targeted 3D spot spraying on tomato plants affected by Tuta Absoluta (a pest).

It is noteworthy that this agricultural implementation has high consequences when failing to eradicate the said pest, or when it unnecessarily over-sprays the plant and adds to the pollution and potentially become harmful to human health. Due to the implication of both errors of the operation of the robot, there is a need for a trust facilitator for during the main course of operation.

In a standard design of such automated operations, the robot would keep only a local copy of the history of its operation. In a case of audit, the robot would be taken aside from operation, inspected, and then returned to operation if found to be not detective. Additionally, having the data stored as such, bigger malfunctions that harm the electronic equipment can also damage the history registry of the robot and render the audit process useless. It is also worth to mention that in the case of voluntary tampering with the device (i.e., stimulate over-spraying so that less crops are affected by the pest at the future harm of humans), the responsible actors can purposefully damage the records to hide their tracks. To avoid the short comings of the above, the Greenbot has been envisioned to operate in collaboration with a Farmer's logbook that records most of the actions from the robot directly in a trustworthy manner.

The role of DLT: In accord with this deliverable, the trust facilitator of the logbook is the DLT client. A smart contract would be deployed by the DLT manager that allows bidirectional flow of information to and from a lightweight client. The sensors located on robot, contains information regarding the correct operation of the robotic arm, and/or the position of the entire vehicle-robot will be passed down to the logbook in a sporadic manner once relevant events have been noticed. The

In such a manner a DLT client is required that operates through a secure link with the IoT device on the robot to communicate the data. The client's role is to be on standby for any novel information from the robot. Moreover, the client's role is expanded to act as an intermediary for the storing and withdrawing past information from the DLT network. Note that once information has been written down in a block of the DLT network, it is therefore immutable and transparent to all nodes of the DLT network. As such all DLT clients that employ the logbook client can share and request data from other robots that participate in similar tasks and extract important information such as predict larger infestations and/or anticipate seasonal variations due to the behavior of the bugs. Since the trust between operators of the logbook client is established, information provided by participatory devices is invaluable to establish important markers of the aforementioned phenomena. The system is envisioned to operate such as shown in Figure 8.

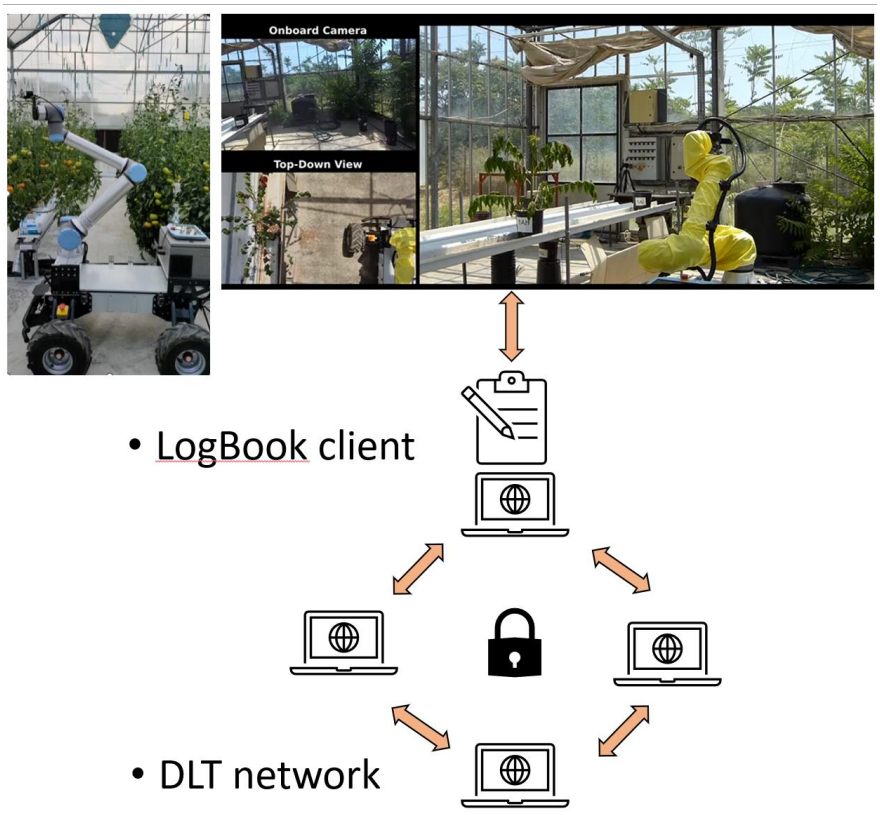


Figure 8 The role of the LogBook client in the overall use case

Integration with DLT: As the Greenbot operates on batteries and its computational capabilities are not as powerful, the DLT integration follows the remote implementation. In detail, the logbook client operates outside of the framework of the robot, or it can be also run on-device, but interface through the device's internal network. To facilitate the integration as such, the DLT acts as an HTTP server itself with two procedures. This type of integration also allows for the logbook to be useful for more than one robot, as it can be accessed on the LAN network, and can operate at all times to also be allowed to monitor devices outside its local network.

Running the client first initializes with a check to see if it is still connected to the DLT manager, and hence the DLT network. Following this the client opens port 9090 on the pre-set IP address (set to the loopback by default). After this the DLT client is open for block submission requests or block retrieval. I.e., to send the first log into the logbook the IoT device needs to invoke the POST function and submit the encoded data to the DLT client. The DLT client receives the data and displays the below:

```
================================================================================
Setting gas price strategy. Please wait...
Estimated gas price 1E-9 gwei
================================================================================
tx: 0xcc9864c2c09c08c2c8c9736981d4f4f3f9d97dc60a3cd1453f8ad74c8ac5c26a
block: 0x729aa2c15a86cc628b3cc20172c307a03f76845457bdd7d37e34061075aa5d5c
================================================================================
127.0.0.1 - - [08/Feb/2023 16:03:17] "POST / HTTP/1.1" 200 -
```

At the IoT device's side, the HTTP POST returns the block number in which the information was written down in the blockchain. Any other user that participates in the DLT network can use the GET method to later retrieve the information from that specific block. On the DLT manager's side this appears as:

```
ganache          |   Transaction: 0x305ae48a9488d69e86d6bc33470546c3e8d274d3cb77697d27a17cd4865800bf
ganache          |   Gas usage: 21616
ganache          |   Block Number: 206
ganache          |   Block Time: Wed Feb 08 2023 15:18:41 GMT+0000 (Coordinated Universal Time)
ganache          |
ganache          | eth_getTransactionReceipt
ganache          | eth_getTransactionByBlockNumberAndIndex
ganache          | eth_getTransactionByBlockNumberAndIndex
ganache          | eth_getTransactionByBlockNumberAndIndex
ganache          | eth_getTransactionByBlockNumberAndIndex
ganache          | eth_getTransactionByBlockNumberAndIndex
```

Vision and Future: The logbook application goes in accord of using the DLT network as a data storage. In this case, the bidirectional communication with the DLT network allows for more efficient distribution of the logged data. Thus, the vision of having an efficient data distribution structure that facilitates trust, is complete. Future implementations might look at scale and latency implications of using the DLT network for trustworthy communication between various agricultural devices.

4.2 DLT for Healthcare (Use Case 2)

The premise: Medical data comprises large files such as medical photographs of patients and films of diagnostic procedures, as well as smaller data items such as text and digital files. As data can only be added to a Blockchain and not removed from it, it results in a DLT ledger's constant extension, which leads to the chain's storage pressure being very high [25]. Furthermore, the DLT system necessitates the local storage of a full ledger by each node. Hence, data redundancy is introduced into the system as a result of recurrent data storage, and increased storage space requirements are imposed for newly added nodes.

The role of DLT: In the IntellIoT project, we are addressing two problems: i) security and privacy of data storage and ii) the scalability of data storage in a large network. First, in order to address the security and privacy of data in a healthcare scenario, we integrate the DLT system with the Security Assurance Platform and MTD to detect the malicious sources and record the filtered logs to the DLTs to make sure the data is recorded precisely and correctly. An example of recording medical data using a smart contract is described in the snippet below:

```
pragma solidity ^0.5.5;
/* ABIEncoderV2 is needed to return an array from a function. */
pragma experimental ABIEncoderV2;

contract HeathCareContract {

  /* Every patient or user has a list of medical records. */
  mapping (address => string[]) public medicalRecords;

  function addDocument(string memory documentHash) public returns (uint) {
    address from = msg.sender;
    // push returns the array length
    return medicalRecords[from].push(documentHash) - 1;
  }

  /**
   * Get all medical records of a user.
   */
  function getmedicalRecords(address user) public view returns (string[] memory) {
    return medicalRecords[user];
  }
  mapping (address => address[]) public doctorsPermissions;

  /**
   * Allow a doctor to view all your medicalRecords.
   */
  function giveAccessToDoctor(address doctor) public {
    doctorsPermissions[doctor].push(msg.sender);
  }

  /**
   * Revoke a doctors access to your medicalRecords.
   */
  function revokeAccessFromDoctor(address doctor, uint index) public {
    require(doctorsPermissions[doctor][index] == msg.sender, 'revoke the medical Records of patient.');
    delete doctorsPermissions[doctor][index];
  }
```

Second, in order to address the data storage problem, we deploy an off-chain storage solution. The IPFS system will automatically slice the data and store it in IPFS nodes around the world. Finally, after the hash value of each slice is spliced together, the unique hash address corresponding to the file is calculated. At the same time, IPFS uses content-based addressing instead of traditional address-based addressing. In other words, if the contents of the transmitted file are the same, the corresponding IPFS hash value will not change due to different local storage locations, thus solving the problem of data redundancy in the network caused by repeated data storage. IPFS is therefore an ideal platform for storing bulk medical data.

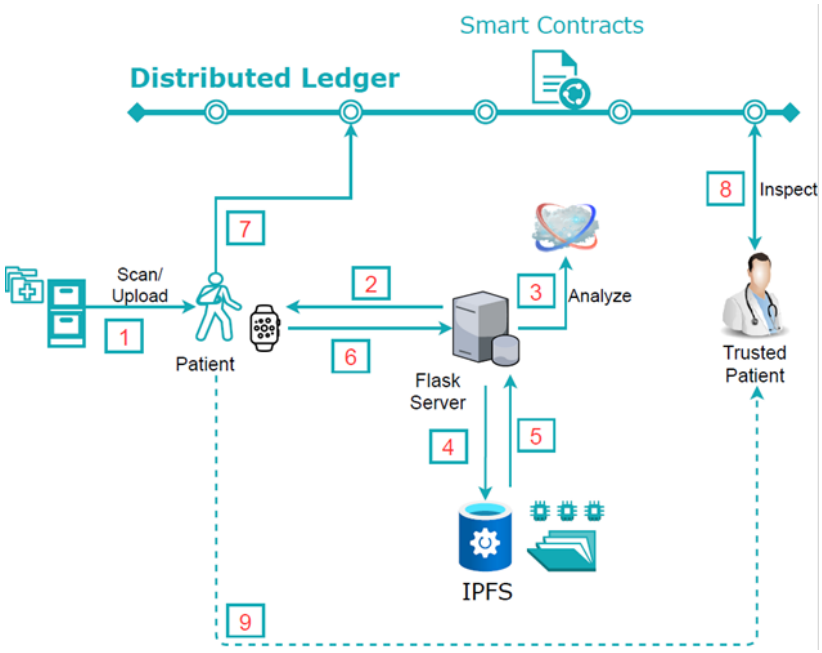


Figure 9. The application of DLT in trusted medical record

Figure 9 shows an example of how to integrate DLT in healthcare system. The urge for innovation is insatiable when it comes to electronic medical records and data. Hospitals continue to employ age-old data management systems for patient data, and the way patient health information are maintained and safeguarded today does not reflect our technical advances in this field over the last decade. This is partially due to rigorous privacy and security restrictions governing medical data, which have hampered the adoption of cutting-edge technology to make medical data management more visible and beneficial for both patients and clinicians. The application depicts the platform via the eyes of four different actors:

1. The administrator is in charge of a group of hospitals and has the highest degree of access in the hierarchy. On their dashboard, they may add a new organization (hospital) to the conglomerate and assign/de-assign hospital administrators.
2. The organization (hospital) administrator is in charge of a specific hospital within the conglomerate/solution. They can add new users with the role of patient or doctor, as well as remove existing members.
3. The doctor is an organization user with the proper position who has the ability to upload docs for their patients as well as download/view papers for whom they have been authorized access.
4. The patient is an organization user with the proper role who may upload documents, examine them, check the document access records, and manage access to their documents on their own. The medical records of patients can be stored in the distributed ledger and available for permissioned actors, such as doctors or hospital. The demo application developed in this context is shown in Figure 10.

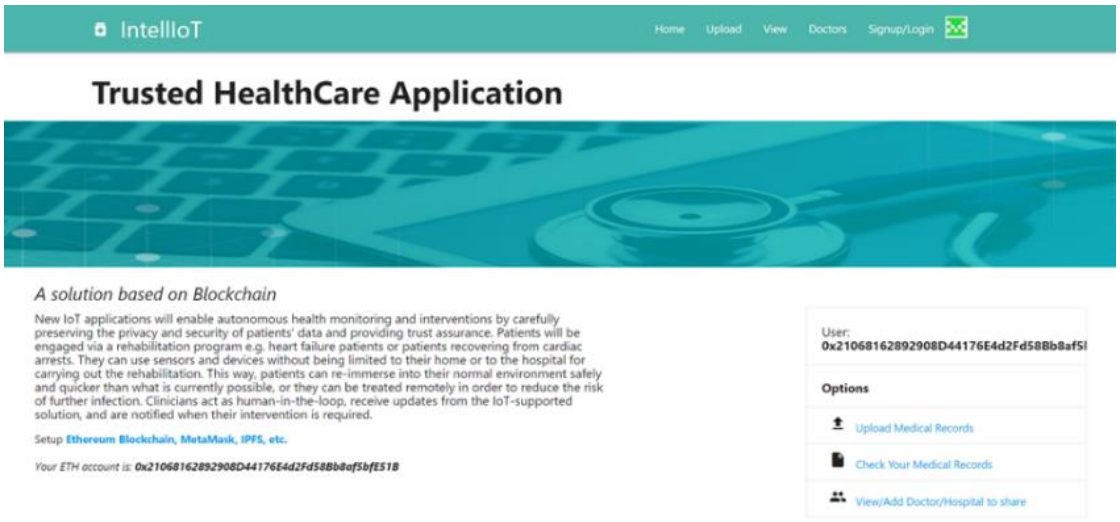


Figure 10. GUI of the developed application.

Integration with DLT: DLT technology presents numerous opportunities for health care; however, it is not fully mature today nor a panacea that can be immediately applied. Several technical, organizational, and behavioral economics challenges must be addressed before a health care DLT can be adopted by organizations nationwide.

Vision and Future: Despite the current immaturity of the DLT system and the hindrances of the organizational healthcare system a model system can be used to test several approaches for the healthcare usage. As mentioned, healthcare institutes most important resource to more advanced diagnosis/treatments is the use of data gathered from clients. Unfortunately data protection harms any effort of objectively evaluating the data before gathering it in a single database (gathered from many healthcare institutions) with the purpose of developing healthcare solutions. Accordingly, privacy preserving federated learning implementations in conjunction with DLT implementations are researched to provide a decentralized reward framework for the advancement of common ML solutions.

The integration efforts are still in the early phase of research and mainly concern with creating an accurate reward system that create a share based system. In detail, the share system is aimed towards creating a DLT managed financial reward system that is superficial whilst in the development phase but has solid value once a marketable solution has been provided by the FL contributors. Such a system suffers from several issues which were explained in detail and provisionally addressed in research efforts that are described in the research section of this report (Section 5.)

4.3 DLT for Manufacturing (Use Case 3)

The premise: In the manufacturing industry DLT can enable a completely new manufacturing business model by increasing visibility across all elements of the process, from suppliers, strategic sourcing, procurement, and supplier quality through shop floor operations, which include machine-level monitoring and servicing. The grounds for such claims are founded in the concept of Manufacturing as a digital service marketplace. This concept is not novel by itself; however, marketplace adoption is always slow due to the trust factor. In short, service providers and customers would need to trust the marketplace itself to rely on its usage for the needs. To counter this issue DLT can be a facilitator for the trust process of such a manufacturing marketplace. In fact, all industrial organizations that rely on supply chains, and most of them can benefit from DLT's. Manufacturers will be able to better meet delivery dates, improve product quality, and ultimately sell more by expanding supplier order accuracy, product quality, and track-and-traceability. And finally, the DLT implementation can offer an easy audit process to find supply chain faults and poor invoicing.

The role of DLT: First, the physical machines and robots act as DLT clients reporting the operation data in form of DLT transactions via edge devices. The DLT managers are set up in powerful edge nodes. The data then is published to the distributed ledger via smart contracts. Smart contracts are responsible for accepting the published data and

autonomously executing the agreements between actors, for example, the machine rental agreements between plant operators and customers. The operation logs of machines are recorded transparently in the distributed ledger, so that it is available for all the actors, and it make convenience for accounting and auditing process. In the scope of this research, we introduce a new integration framework between Ethereum Blockchain and the Universal Robot 5 (UR5) in Fig. 11. The Ethereum DLT with its smart contracts interacts with UR5 via interfaces for accounting the activities and agreements between actors.

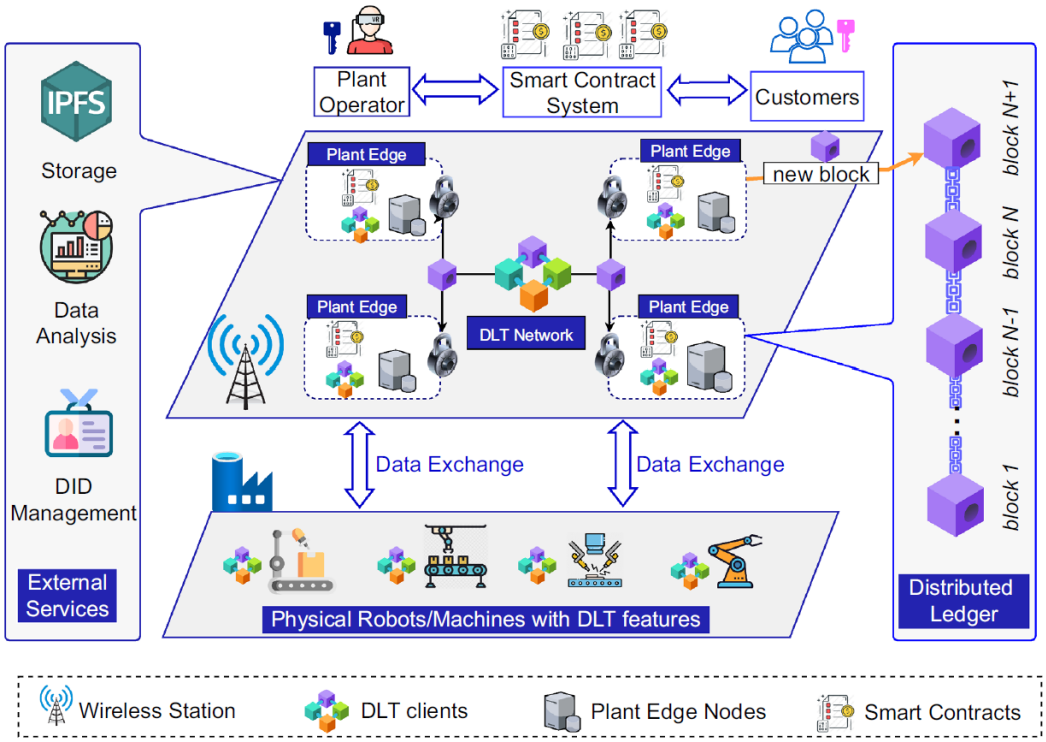


Figure 11. General Architecture of DLTs in Manufacturing UC

Integration with DLT (as a manufacturing gateway): To interact with the UR5 robot, we implemented the UR-RTDE [library, which allows to query and command the robot via an open interface. Through RTDE, the user can control and manage the robot arm. The smart contract is designed to communicate with the UR5 robot through RTDE, and account the robot activities, for example, working time and movement positions. In a real scenario, it opens the opportunity of pay-per-use schemes in a manufacturing plant between multiple participating organizations. As shown in the snippet below, the RTDE is designed based on Python.

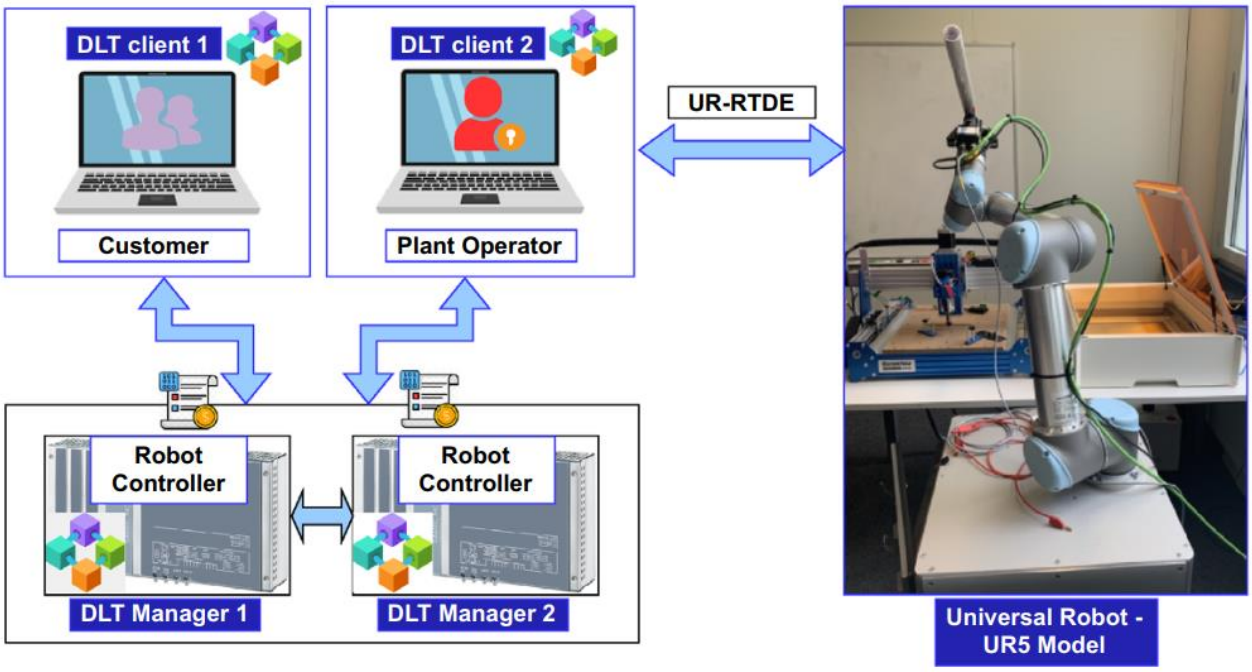


Figure 12. Implementational diagram of the use of the DLT for manufacturing applied to the UR5.

```
#parameters
parser = argparse.ArgumentParser()
parser.add_argument('--host', default='10.0.2.4', help='name of host to connect to (robot controller)')
parser.add_argument('--port', type=int, default=30004, help='port number (30004)')
parser.add_argument('--samples', type=int, default=0, help='number of samples to record')
parser.add_argument('--frequency', type=int, default=125, help='the sampling frequency in Herz')
parser.add_argument('--config', default='record_configuration.xml', help='data configuration file to use (record_configuration.xml)')
parser.add_argument('--output', default='robot_data.csv', help='data output file to write to (robot_data.csv)')
parser.add_argument("--verbose", help="increase output verbosity", action="store_true")
parser.add_argument("--buffered", help="Use buffered receive which doesn't skip data", action="store_true")
parser.add_argument("--binary", help="save the data in binary format", action="store_true")
args = parser.parse_args()

logging.basicConfig(level=logging.INFO)

if args.verbose:
    logging.basicConfig(level=logging.INFO)

conf = rtde_config.ConfigFile(args.config)
output_names, output_types = conf.get_recipe('out')

con = rtde.RTDE(args.host, args.port)
con.connect()

# get controller version
con.get_controller_version()

# setup recipes
if not con.send_output_setup(output_names, output_types, frequency = args.frequency):
    logging.error('Unable to configure output')
    sys.exit()

#start data synchronization
if not con.send_start():
    logging.error('Unable to start synchronization')
    sys.exit()
```

Integration with DLT (as a manufacturing supervisor): The pay-per-use scheme does not need to be enforced by a DLT. In example the query process can be facilitated by an individual party that receives and pushes tasks based on each manufacturing use-case individually. Thus, the job initiation and command of a robot can be designed without the DLT as a manufacturing gateway, but on its own proprietary implementation. To facilitate trust without the gateway operations, the DLT can be implemented as manufacturing supervisor that probes the IoT device for the operational status.

In this type of integration of manufacturing and DLT, the DLT client is more lightweight and versatile as it can easily be adapted for different manufacturing applications m-APPs than the gateway one. As the IntellIoT project has a plenty of different m-APP systems that interact for UC3, this type of client is more applicable to the entire UC3. The

The integration setup can accept three different arguments. Since the manufacturing IoT devices are powered by the RTDE API accessed by HTTP, the first argument and the URL of the tractor client is called as:

```
253 parser = argparse.ArgumentParser(description="Poll app states for FLT")
254 parser.add_argument("--mode",  type=str, help="Selects the operation mode: ied, test, remote", default="ied")
255 parser.add_argument("--edp",  type=str, help="Selects the edge device to pool", default="all")
256 parser.add_argument("--dlt_mgr_ip",  type=str, help="Assigns the IP of the DLT manager", default="0.0.0.0")
```

Once, the client is run, it probes the DLT manager with a test transaction. For successful connection with DLT, the appropriate IP of the DLT manager needs to be passed. After a test transaction, the client establishes connection to the HTTP server of the manufacturing device. The client implementation was done with flexibility in mind in a way that can probe a specific m-APP regardless of the client's deployment. As such the client can operate as on-device containerized implementation by setting the `--mode ied`, or if the client is on a different device, `--mode remote`. Since the DLT client can probe any of the participating m-APPs, the client needs to be specified the edge device to probe for information by setting `--edp laser/robot/HIL` for any of the specific devices or set `-edp all` to probe all devices. Such an implementation is shown in Table 1 with console output from the implementation.

```
2023-02-09 13:39:31,046 Mode is ied
2023-02-09 13:39:31,046 Device is all
2023-02-09 13:39:31,046 Manager IP is edge.fritz.box
2023-02-09 13:39:31,046 starting DLT
2023-02-09 13:39:32,258 Using IED at 172.17.4.1
2023-02-09 13:39:32,258 Update interval is 21600s
2023-02-09 13:39:32,258 Start automatic token renewal scheduler
2023-02-09 13:39:32,284 Success!!! Received token=9664c64d-0119-4f00-b5bc-622833a34cd3 at 2023-02-09 13:39:32 expiring
at 2023-02-10 01:39:32
2023-02-09 13:39:32,284 Next renewal will be executed at 2023-02-09 19:39:32
2023-02-09 13:39:32,285 Start app poll scheduler
2023-02-09 13:39:42,297 Get app list
2023-02-09 13:39:42,398 Get app stats
2023-02-09 13:39:42,565 Fetching state from Engraver Laser
2023-02-09 13:39:42,593 {"state":"safetylock"}
2023-02-09 13:39:43,834 Fetching state from HIL Service
2023-02-09              13:39:43,862              [{"creationTime":"2022-12-14T15:06:30.938Z","willExpireAt":"2022-12-
14T16:06:30.938Z","takenOverAt":"2022-12-14T15:06:47.707Z","finishedAt":"2022-12-
14T15:07:30.118Z","status":"finished","workResult":"completed","hilAppIpAddress":"192.168.99.99","numberReassignments
":0,"description":{"aiSessionId":"46840","robotId":"UR5","cameraId":"robot-
camera","sessionType":"wood_piece_picking"}},{"creationTime":"2023-01-30T12:17:40.964Z","willExpireAt":"2023-01-
30T13:17:40.964Z","takenOverAt":"2023-01-30T12:17:46.827Z","finishedAt":"2023-01-
30T12:18:11.131Z","status":"finished","workResult":"completed","hilAppIpAddress":"192.168.99.99","numberReassignments
":0,"description":{"aiSessionId":"HilSession_1","robotId":"UR5","cameraId":"robot-
camera","sessionType":"wood_piece_picking"}}]
2023-02-09 13:39:44,869 Fetching state from Robot Controller
2023-02-09   13:39:44,892   {"inMovement":false,"tcp_link_to_UR5":"common.status.disconnected","status":"waiting   for
commands","actual_named_position":"home","ai_allowed_range":{"x":{"min":0.08,"max":1.05},"y":{"min":0.365,"max":0.5},
"alpha":{"min":-20,"max":25}},"positions":{"home":{"x":0.07,"y":0.745,"z":0.328,"alpha":0,"beta":0,"gamma":-
3.14},"home_joint":{"j1":1.42,"j2":-1.44,"j3":-1.94,"j4":3.4,"j5":4.5,"j6":4.63},"engraver_load":{"x":0.561,"y":-
0.413,"z":0.21,"alpha":0,"beta":0,"gamma":1.58},"engraver_pre":{"x":0.35,"y":-
0.413,"z":0.21,"alpha":0,"beta":0,"gamma":1.58},"milling_machine_pick":{"x":0.472,"y":0.47,"z":0.153,"alpha":0,"beta"
:0,"gamma":1.57},"milling_machine_place":{"x":0.472,"y":0.47,"z":0.153,"alpha":0,"beta":0,"gamma":1.57},"milling_mach
ine_pre_1":{"x":0.38,"y":0.47,"z":0.153,"alpha":0,"beta":0,"gamma":1.57},"milling_machine_pre_2":{"x":0.5,"y":0.47,"z
":0.25,"alpha":0,"beta":0,"gamma":1.57},"machine_waypoint":{"x":0.41,"y":0.1,"z":0.35,"alpha":0,"beta":0,"gamma":1.58
},"storage_waypoint":{"x":-0.57,"y":0.1,"z":0.36,"alpha":0,"beta":0,"gamma":-
1.57}},"joints_near_to_home_range":{"j_1":{"min":1,"max":2},"j_2":{"min":-2,"max":-1},"j_3":{"min":-1.65,"max":-
```

1}},"actual_joint_positions_in_rad":{"j_1":2.856009006500244,"j_2":-1.918042008076803,"j_3":-1.4403656164752405,"j_4":-2.8714969793902796,"j_5":-0.2849038282977503,"j_6":4.669086933135986}}
2023-02-09 13:39:56,170 Get app list
2023-02-09 13:39:56,300 Get app stats
2023-02-09 13:39:56,450 Fetching state from Engraver Laser
2023-02-09 13:39:56,473 {"state":"safetylock"}
2023-02-09 13:39:57,820 Fetching state from HIL Service
2023-02-09 13:39:57,851 [{"creationTime":"2022-12-14T15:06:30.938Z","willExpireAt":"2022-12-14T16:06:30.938Z","takenOverAt":"2022-12-14T15:06:47.707Z","finishedAt":"2022-12-14T15:07:30.118Z","status":"finished","workResult":"completed","hilAppIpAddress":"192.168.99.99","numberReassignments":0,"description":{"aiSessionId":"46840","robotId":"UR5","cameraId":"robot-camera","sessionType":"wood_piece_picking"}},{"creationTime":"2023-01-30T12:17:40.964Z","willExpireAt":"2023-01-30T13:17:40.964Z","takenOverAt":"2023-01-30T12:17:46.827Z","finishedAt":"2023-01-30T12:18:11.131Z","status":"finished","workResult":"completed","hilAppIpAddress":"192.168.99.99","numberReassignments":0,"description":{"aiSessionId":"HilSession_1","robotId":"UR5","cameraId":"robot-camera","sessionType":"wood_piece_picking"}}]
==========
Successfully connected to Testnet
Attempting to send transaction from 0x2E6d29B17a526334843141F8BEf0f0424760199e to 0x2E6d29B17a526334843141F8BEf0f0424760199e
Test transaction to Ethereum successful
Setting gas price strategy. Please wait...
Estimated gas price 1E-9 gwei
==========
Transaction:
{'value': 0, 'gas': 21736, 'chainId': 1337, 'nonce': 49, 'gasPrice': 1, 'to': '0xDca4261fB51136E65700Ef63fb7f18943A658094', 'data': '0x0b06b91f0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000000167b227374617465223a227361666574796c6f636b227d0000000000000000000'}

Successfully sent to Ethereum
gas used: 21736
contract address: 0xDca4261fB51136E65700Ef63fb7f18943A658094
tx: 0xdd8b82c856a275c6e4410c500291611430a2b14efd4dff9fb24d0cce91b837fa
block: 0xfdb72898e683eb457f30e405c43232a11b054b5de1a928e0d5f6f84739b7e053
==========
Transaction:
{'value': 0, 'gas': 33892, 'chainId': 1337, 'nonce': 50, 'gasPrice': 1, 'to': '0xDca4261fB51136E65700Ef63fb7f18943A658094', 'data': '0x0b06b91f0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000030a5b7b2263726561746964f6e54696d65223a22323032322d31322d31345431353a30363a33302e3933385a222c2277696c6c4578706972654174223a22323032322d31322d31345431363a30363a33302e3933385a222c2274616b656e4f7665724174223a22323032322d31322d31345431353a30363a34372e3730375a222c2266696e6973686564174223a22323032322d31322d31345431353a30373a33302e3131385a222c2273746174757223a2266696e6973686564222c22776f726b526573756c74223a22636f6d706c65746564222c2268696c41707049704164647265737373223a223139322e3136382e39392e3939222c226e756d626572526561737369676e6d656e7473223a302c226465736372697074696f6e223a7b2261695365737373696f6e4964223a2248696c536573696f6e5f31222c22726f626f744964223a22555235222c2263616d6572614964223a22726f626f742d63616d657261222c2273657373696f6e54797065223a22776f6f645f70696563655f7069636b696e67227d7d2c7b2263726561746964f6e54696d65223a223230323332d30312d33305431323a31373a34302e3936345a222c2277696c6c4578706972654174223a22323032332d30312d33305431323a31373a34362e3832375a222c2266696e6973686564174223a22323032332d30312d33305431323a31383a31312e3133315a222c2273746174757223a2266696e6973686564222c22776f726b526573756c74223a22636f6d706c65746564222c2268696c41707049704164647265737373223a223139322e3136382e39392e3939222c226e756d626572526561737369676e6d656e7473223a302c226465736372697074696f6e223a7b2261695365737373696f6e4964223a2248696c536573696f6e5f31222c22726f626f744964223a22555235222c2263616d6572614964223a22726f626f742d63616d657261222c2273657373696f6e54797065223a22776f6f645f70696563655f7069636b696e67227d7d5d0000000000000000000000000000000000000000000000000000'}

Successfully sent to Ethereum
gas used: 33892
contract address: 0xDca4261fB51136E65700Ef63fb7f18943A658094
tx: 0x070158d113df7a3c65454b117f02d1ad249700df3c6b92f2690576aec5d77952
block: 0xbbd3ddb0d4c3446d652ea9537e3e1f46e081691d300100e3719010a0469a2958
==========
Transaction:
{'value': 0, 'gas': 43072, 'chainId': 1337, 'nonce': 51, 'gasPrice': 1, 'to': '0xDca4261fB51136E65700Ef63fb7f18943A658094', 'data': '0x0b06b91f0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000005477b22696e4d6f76656d656e74223a66616c73652c227463705f6c696e6b5f746f5f555235223a22636f6d6d6f6e2e7374617475732e646973636f6e6e6563746564222c22737461747573223a2277616974696e6720666f7220636f6d6d616e6473222c22616374756c5f6e616d65645f706f736974696f6e223a22686f6d65222c2261695f616c6c6f7765645f72616e6765223a7b2278223a7b226d696e223a302e3082c226d6178223a312e30357d2c2279223a7b226d696e223a302e3336352c226d6178223a302e357d2c22616c706861223a7b226d696e223a2d32302c226d6178223a32357d7d2c22706f736974696f6e73223a7b22686f6d65223a7b2278223a302e30372c2279223a302e3734352c227a223a302e3332382c22616c706861223a302c2262657461223a302c2267616d6d61223a2d332e31347d2c22686f6d655f6a6f696e74223a7b226a31223a312e34322c226a32223a2d312e34342c226a33223a2d312e39342c226a34223a332e342c226a35223a342e352c226a36223a342e36337d2c22656e6772617665725f6c6f6164223a7b2278223a302e3536312c2279223a2d302e3431332c227a223a302e32312c22616c706861223a302c2262657461223a302c2267616d6d61223a312e35387d2c22656e6772617665725f707265223a7b2278223a302e33352c2279223a2d302e3431332c227a223a302e32312c22616c706861223a302c2262657461223a302c2267616d6d61223a312e35387d2c226d696c6c696e675f6d616368696e655f7069636b223a7b2278223a302e3437322c2279223a302e34372c227a223a302e3135332c22616c706861223a302c2262657461223a302c2267616d6d61223a312e35377d2c226d696c6c696e675f6d616368696e655f706c616365223a7b2278223a302e3437322c2279223a302e34372c227a223a302e3135332c22616c706861223a302c2262657461223a302c2267616d6d61223a312e35377d2c226d696c6c696e675f6d616368696e655f7072655f31223

```
a7b2278223a302e33382c2279223a302e34372c227a223a302e3135332c22616c706861223a302c2262657461223a302c2267616d6d61223a312e
35377d2c226d696c6c696e675f6d616368696e655f7072655f32223a7b2278223a302e352c2279223a302e34372c227a223a302e32352c22616c7
06861223a302c2262657461223a302c2267616d6d61223a312e35377d2c226d616368696e655f776179706f696e74223a7b2278223a302e34312c
2279223a302e312c227a223a302e33352c22616c706861223a302c2262657461223a302c2267616d6d61223a312e35387d2c2273746f726167655
f776179706f696e74223a7b2278223a2d302e35372c2279223a302e312c227a223a302e33362c22616c706861223a302c2262657461223a302c22
67616d6d61223a2d312e35377d7d2c226a6f696e74735f6e6561725f746f5f686f6d655f72616e6765223a7b226a5f31223a7b226d696e223a312
c226d6178223a327d2c226a5f32223a7b226d696e223a2d322c226d6178223a2d317d2c226a5f33223a7b226d696e223a2d312e36352c226d6178
223a2d317d7d2c2261637475616c5f6a6f696e745f706f736974696f6e735f696e5f726164223a7b226a5f31223a322e383536303039303036353
0303234342c226a5f32223a2d312e393138303432303038303736383033 32c226a5f33223a2d312e343430333635363136343735323430352c226a
5f34223a2d322e383731343936393739333930323739362c226a5f35223a2d302e323834393033383238323937373530332c226a5f36223a342e3
6363930383639333331333539383 67d7d0000000000000000000000000000000000000000000000000000000000000000'}
--------------------------------------------------------------------------------
Successfully sent to Ethereum
gas used: 43072
contract address: 0xDca4261fB51136E65700Ef63fb7f18943A658094
tx: 0x71f9c5f60e66d2562aa0481e51b448a7075affadb4b82b0d2665da34dd566190
block: 0x0b26e0e3ee1625f95208ed3f4f9c26b0ce4b0d58961862d9f08ae250b52c33aa
================================================================================
Transaction:
{'value':     0,     'gas':     21736,     'chainId':     1337,     'nonce':     52,     'gasPrice':     1,     'to':
'0xDca4261fB51136E65700Ef63fb7f18943A658094',                                                                    'data':
'0x0b06b91f0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000167b227374617465223a227361666574796c6f636b227d0000000000000000000000'}
--------------------------------------------------------------------------------
```

Table 1. Test example in UC3

The integrated operation of m-APPs differs from the application. As it can be noticed the laser transmits the least amount of information as it is mostly concerned with time of operation. While the robot that has potential security and hazard in harming nearby humans needs to log the entirety of its movement data for audit-level troubleshooting. This makes the pay-per-use scheme more precise and allows for clear and definite control and invoicing of each m-APP performance.

Vision and Future: The manufacturing framework has an unlimited number of options expanding into future uses. The blockchain can be adapted to allow for more efficient, payment, review and refund processes following a human led audits. Moreover, in cases of detailed data submissions, such as in the case of the robot, a more resource-aware blockchain solution can be integrated in the system. This is in alignment with the IPFS system in UC2 for verification of information using hashes.

A possible extension after IntellIoT is to create a DLT implementation that has a billing client. The vision is that the billing client is a customer facing application with streamlined user experience UX. The goal of such a UX optimized application is to bring and accentuate the transparency of the DLT system proposed by IntellIoT baseline demo. In example, the client can display the progress of the m-APP job, with the DLT provided details of, the time of the update, the anticipated DLT induced delay. The billing client also envisions an easy display of all past jobs, percent of using m-APP, total owed. And finally, the most important feature of the billing client is invoicing and auditing. Instead of ordering the money flow through a separate APP, the DLT Ethereum implementation can bill the accurate amount to the final user that can be verifiable through the DLT as a trust facilitator.

5 DLT-ASSISTED DISTRIBUTED LEARNING

In this section, we introduce a general framework that integrates the DLTs with Federated Learning to improve the security and privacy of Federated Learning in IoT environments. This extra feature has application in the use cases addressed by IntellIoT, e.g., for the protection of the very sensitive data exchanged in healthcare. In addition, we study the value of data generated from IoT devices which has been typically neglected in the literature.

5.1 Overview

Federated Learning (FL) is a machine learning scheme where multiple entities (clients) collaborate via a central server in training a model without sharing their available raw data [27]. Instead, clients perform computations on their data and transfer obtained local learning parameters updates to the server for the aggregation process. The aggregated model, i.e., the global model, is broadcast back to the clients for the next round of local computations resulting in the local learning parameters. The interaction between the server and clients to solve the learning problem continues until an acceptable level of model accuracy is achieved [5]. The overall process is shown in Figure 13.

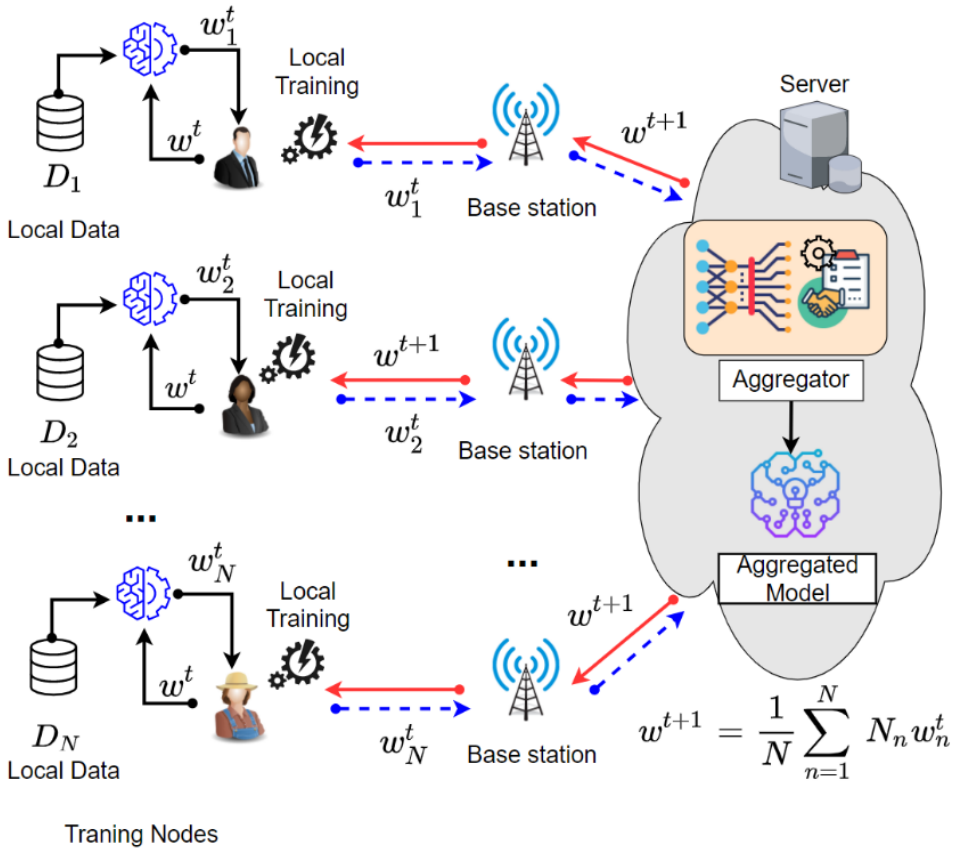


Figure 13. Typical Federated Learning scheme

In this manner, FL offers (i) privacy-preserving benefits in the model training approach by not requiring clients to share their local data to the server, and consequently, (ii) lower communication overhead by offering a distributed model training paradigm and exchange of model parameters only. Therefore, FL can serve as an enabling technology for ML model training at edge networks.

However, today's systems, for example, a common hospital IT environment, are often centralized and lack data integrity, trust, security, and transparency. Traditional data markets are often implemented as a centralized service platform that collects data from data owners and sells raw or processed data to the consumers. This approach leaves

the platform as a single point of security vulnerability for the data market, and corruption on the platform servers may lead to severe security issues including leakage of private data, faulty computation results, and manipulation of data price.

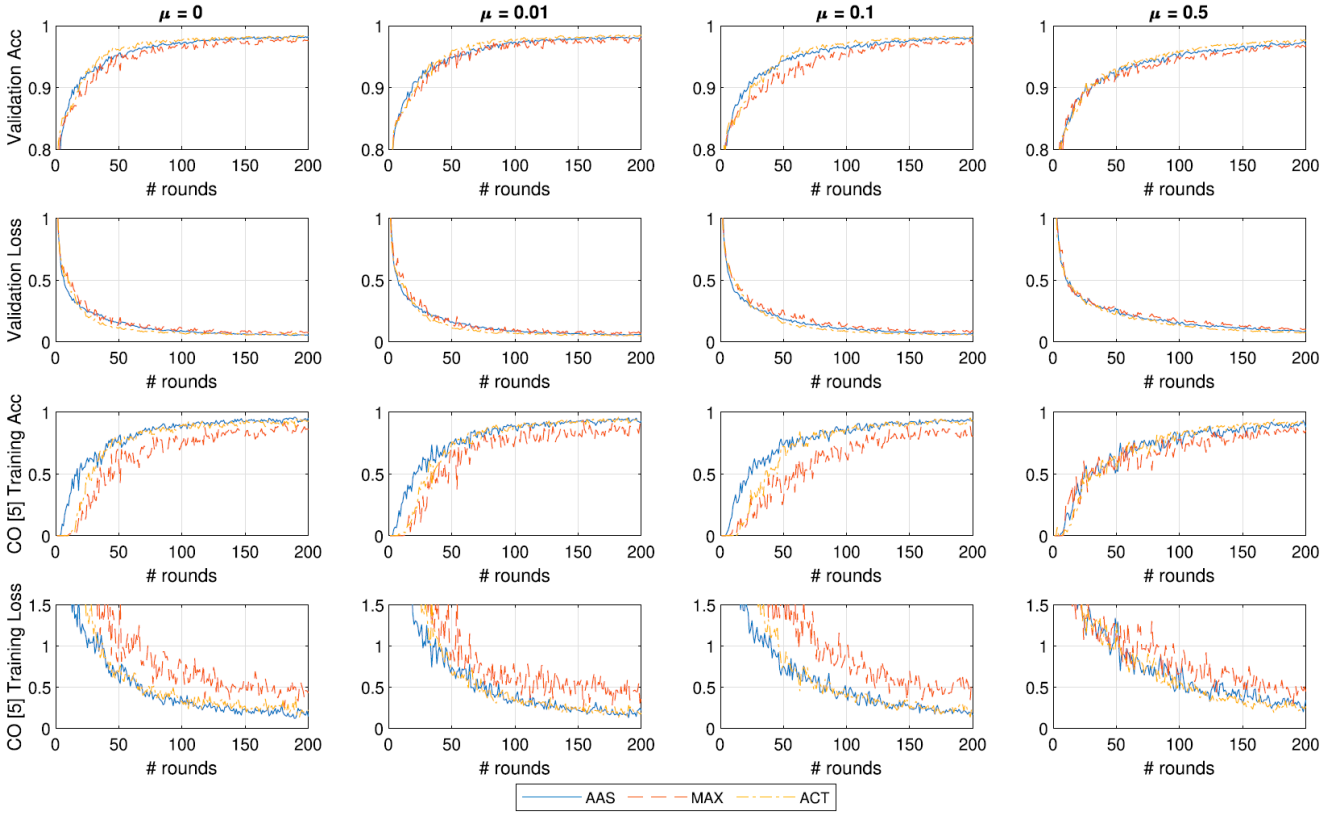


Figure 14 Federated learning in the presence of critical objects (CO) for different FL implementations. [32]

In addition, the data can be both public or private, which makes it difficult to validate their origin and consistency. Besides, querying and performing operations on the data becomes a challenge due to the incompatibility between different application programming interfaces (APIs) among organizations that may use different data types and databases, which leads to difficulties when sharing the data. On the other hand, traditional exchange systems (e.g., Paypal or Ebay) are vulnerable to a single point of failure, the lack of trust, transparency, and incentive for data trading, which is preventing the availability of digital information from data providers to customers.

Despite the benefits of FL, there is a major trade-off to the benefit of added privacy. FL implementations have a more difficult time learning when compared to a classical centralized ML implementation. This condition can be aggravated by unequal processing speeds at each device and non-even distribution of data for the so-called learner devices. In example, if FL is deployed with the purpose of detection a single critical object (CO) that does not appear that often, the training of the ML algorithm can be nearly impossible to converge. Therefore, the radio resources can be allocated in a way where learning equality is maintained despite computational differences through minimizing the average anchored staleness (AAS), or even better an estimation on the learner's past contributions can be used to introduce a reactive (ACT) radio resource allocation to give the performances shown in Figure 14.

5.2 DLT-based infrastructure for Federated Learning

In DLT-based Federated Learning, each learning device trains its own model with its own local data set to guarantee privacy. Each device is connected virtually to a miner in the DLT, i.e., the connection between the miner and device is variable and we assumed that they are connected automatically. This strategy ensures constant communication

between the device and the DLT layer, even if the miner fails. The miner connected to the device acts as the leader of the device and is responsible for uploading and downloading data of the device's data. In each iteration, the device receives the latest global model stored in the DLT system and trains a new round of local models using local data. After completing local training, the device uploads the local model to the adjacent miner and starts a global aggregation process. The process allows all devices to download the latest information from adjacent miners to receive the devices' scores and global model updates. More details on this process and the device scores will be described later in this section. Finally, each device uploads its local model and scores and proceeds to a new iteration

The training process is repeated until the global model has achieved satisfactory accuracy and convergence. In the DLT layer, duplicated versions of distributed ledger are kept by all miners. The DLT miners are members of specific DLT networks and actively participate in the trading process. They validate transactions through their own distributed ledger and store the scores of the models and devices in FL. The miners can be personal computers, enterprise servers, or cloud-based nodes if they voluntarily download the mining software. Each miner has its verifier and block to ensure that the real models and the devices' scores are updated. Each block contains a head and body parts. The blockhead contains a pointer to the next block, and the body part contains a set of validated transaction information. The local models are formed in transaction format and to address scalability issues, the local models are recorded in IFPS storage, and just hash versions of models are recorded in the distributed ledger.

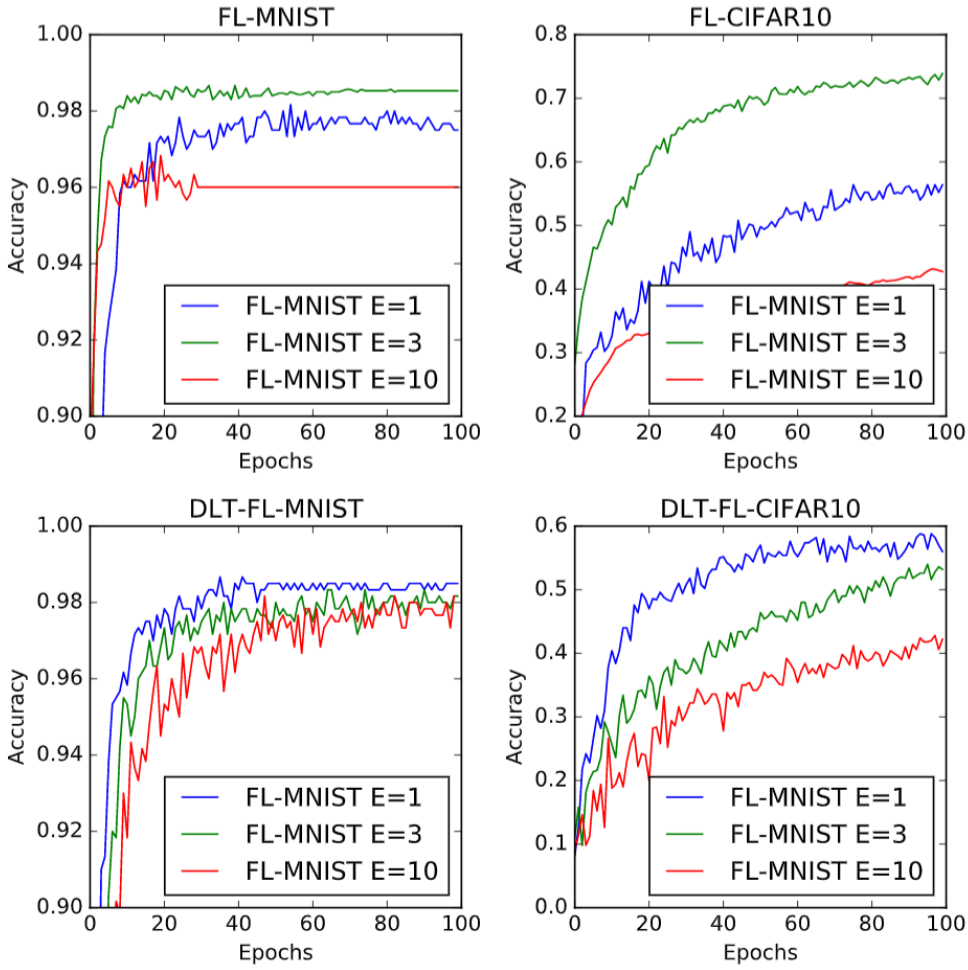


Figure 15. Performance comparison of standard FL and DLT-based FL

The basic comparison between standard FL and DLT-based FL is presented in Figure 15. The accuracy is similar in both standard FL and DLT-based FL, but the time required for convergence of DLT-based FL is higher than standard FL because of extra verification and consensus in the system

5.3 Freshness and valuation of data

In 2025, the volume of sensing data generated by personal IoT devices is expected to reach 79.4 ZB globally. Many attempts have been made to improve and adapt business workflows to exploit the availability of IoT data [2], for example, using IoT data for training machine learning models and sharing data are the most popular approaches. Various services for using and sharing of IoT data are emerging, connecting various devices and distributed IoT data sources, thereby facilitating data providers to exchange their data.

For example, in the medical surveillance, vital data gathered from the patients, e.g., heartbeat, x-ray that could be used for learning diagnostic support systems for detect diseases might be in possession of different hospitals, each of which have different data sources, for instance, from a specific geographical region with different demographics , the data needed to train good models often exists but is not easy to leverage as it is distributed and owned by multiple involved parties. Therefore, by combining the available data from various data silos, the hospitals could for their medical applications using their own data. As hospitals would benefit from better machine learning models obtained through data sharing there is an incentive for patients to give access to their personal data and a need for a distributed ML model marketplace.

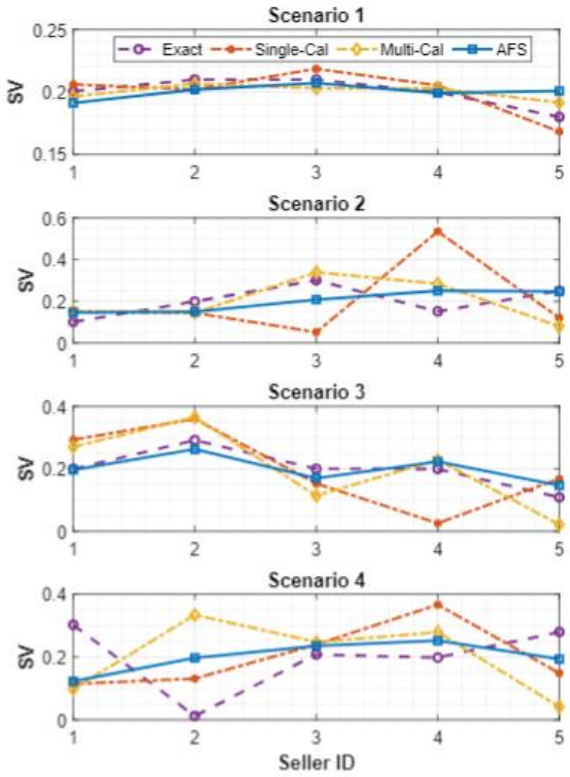


Figure 16: Valuation of data with different Algorithms

A conventional data sharing is often deployed as a centralized service platform that gathers and sells raw or processed data from data owners (e.g., the trained learning models) to the consumers [12] [13]. This leads to two important concerns. First, this strategy exposes the platform as a single point of security risk; the malfunctioning platform servers has serious security concerns including data leakage, inaccurate calculations results, and manipulation of data price. Second, collaboration for model training raises questions in terms of how to motivate participants to participate

in such a training endeavour. The incentive for each IoT client based on their contribution should be fair and transparent. These features are not present in a standard FL setup, as in many applications there is no clear and natural incentive mechanism for involved participants to provide quality information. This calls for a carefully designed mechanisms to reward parties economically and thus incentivize participation [14], [15]. For example, a fixed price per data point could motivate participants to collect massive amounts of low quality or fake data if there is no intermediary process to check quality of training data. Besides, another reason that may disincentives parties from sharing data could stem from privacy and integrity concerns regarding the use of participant's data once it is shared. For instance, the sellers can re-use the data which has already been sold.

With the aforementioned motivation, we propose a Blockchain-based model trading system which enables a secured and trusted marketplace to collaboratively train models as well as guarantees fair incentives for every participant and privacy of data. Based on the quality of the uploaded models, which is quantified by using a distributed Data Shapley Value (DSV), the participants can get the incentive based on the updated models, for example, as tokens or fiats. Note that based on our proposed system, the parties do not need to share their local data, but only provide customized models or query interface to the marketplace. Consequently, the proposed system allows multiple participants to jointly train the ML models on the marketplace based on their own training data. Buyers who need to train their ML model will pay to the market for the improvement of their model, and sellers who sell their contribution to train the ML models will get paid by the market via smart contracts. In Figure 16, we show the contribution of different involved participants calculated based on Shapley Value.

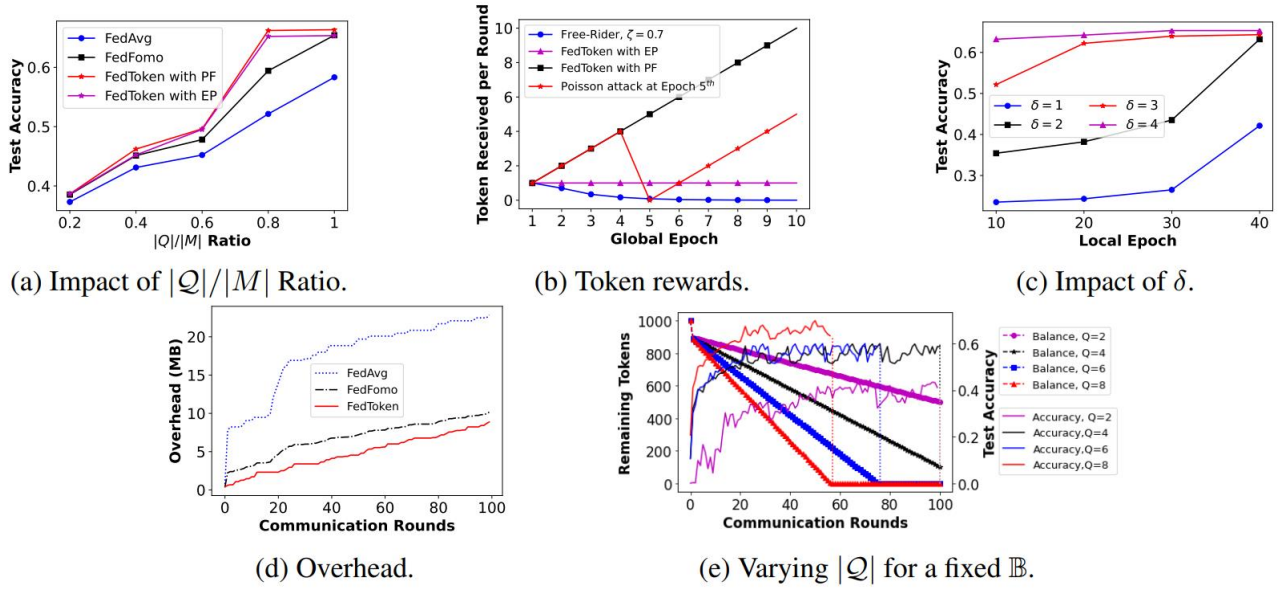
5.4 Incentivizing device contributions in Federated Learning with tokens

While there are many advantages of using data valuation to accelerate FL processes in distrustful machine learning networks, it is an issue that suffers from the difficulty of implementation. In such FL communities the learner (i.e., the medical centre of the second Intelllot use case) possesses valuable data, and is reluctant to share the data to improve the common FL model without any reward. Thus, a reward based learning performance per learner-device needs to be implemented to avoid purposeful data concealment (the learner restricts the usage of highly valuable data) or slacking (the learner restricts its computational resources). Thus, it is crucial to research the setting of an FL network with distrustful parties, and whereas in the real world data and processing capabilities are not equally distributed among learners.

A research scenario is provisioned for the application in the second use-case of Intelllot due to the immense benefits of the cross-hospital collaborations on discovering difficult-to-find cures/diagnoses. Moreover, the FL infrastructure has the capability of protecting the hospital's clients privacy by avoiding to send data to third parties. Given a difference in scale between the hospital research institutes, creating a common solution is unfavourable to large institutes that painstakingly gather data from clients. Due to the more powerful computational platforms, and the larger data sets stored, large institutes have little incentive to collaborate to finding healthcare solutions as they have the higher likelihood of discovering those themselves. Due to this, the smaller institutes have much bigger incentives to join a collaboration that will lead to novel treatments. Obviously, when creating a common FL system between the healthcare parties, it is crucial to correctly reward the contributions. Unfortunately, when searching for a Machine Learning solutions to difficult problems such as complicated diagnosis or treatments finding one is not always guaranteed. Due to this, it makes sense to create an investment type system that progressively rewards the contributors with superficial tokens that may become valuable once the FL networks succeeds in finding a solution that is marketable. The tokens thus act as a form of shares of the FL solution, which once found, will give value to the to the tokens.

As previously mentioned, evaluating the data that goes into the FL contributions is not trivial. Thus, more technical research into proper incentive allocation for compensating the involved costs in the decentralized training of a Federated Learning (FL) model was researched. We initially engineered a key stimulus for clients' long-term participation in the FL network. The difficulty of this problems sources in three key information pieces may be missing from the framework: the data quality and properties for each device is missing, the value of each learning contribution, and a trustworthy facilitator of the monetary incentives. Addressing the previous can lead to an accelerated Web 3.0 adoption and addressing the machine learning objectives without violating the data privacy of each device. Thus, we

proposed a scheme named FedToken backed by blockchain technology that facilitates the trust process of fair allocation of tokens amongst the clients that corresponds to the data value and model training that each device has contributed to the FL implementation. Using the previously referred Shapley-base scheme, we aim to lower the communication rounds required by the FL network to converge to a usable solution of the ML model. Moreover, we aim to anchor ways to allocate affordable tokens under the constrained monetary budget.



(a) Impact of $|\mathcal{Q}|/|M|$ Ratio. (b) Token rewards. (c) Impact of δ.

(d) Overhead. (e) Varying $|\mathcal{Q}|$ for a fixed $\mathbb{B}$.

Figure 17 Performance Analysis of FedToken: a) impact of jQj value, b) token Rewards, c) impact of δ = f1; 2; 3; 4g on training performance, d) communication overhead (in MB), and e) varying jQj. 3.3 Tokenized Incentives for Data Contributions

The analysis of the proposed FedToken system is shown in Figure 17 and compared to the classic FedAvg and FedFomo (an implementation that utilizes the concept of opportunity cost between learners as a method for incentivation.) The tokenized incentives were consistently providing better results over the other two implementations with very quick allocations of tokens. The FedToken implementation also tested two methods of token allocation, a proportional fair (PF) allocation, where the tokens are distributed as per the proportion of contribution in improving the global model performance, versus the Equal Pay (EP), where all participating clients are allocated equal shares of available tokens. The PF allocation provided improvements based on the inequalities between learners in terms of data. Finally, the FedToken implementation provided much smaller overhead versus the benchmarks.

6 OPEN SOURCE

The implementation of the DLT for the three UCs is available in the following link:

https://gitlab.eurecom.fr/intelliot/dlt

The DLT manager

The manager is the same for all types of clients.

For testing it is the easiest to set the manager to the same machine where client integration is done. Use the following repository for the manager:

https://gitlab.eurecom.fr/intelliot-project/security/distributed-ledger-technology/-/tree/igor_master_branch/sim-dlt-net

Requirements:

- Docker
- Docker-compose
- NodeJs
- Python3

Use *docker-compose up –build* to deploy the manager.

The DLT clients (one for each UC)

Each application in each UC has a different client depending on the application use case. The code is available in the following folders:

DTL-for-Agriculture

DLT-for-Manufacturing

DLT-for-healthcare

7 CONCLUSION & FUTURE DIRECTIONS

In the previous deliverable (the first phase of the project - i.e., Cycle 1), we defined the DLT-based mechanisms for the three use cases healthcare, agriculture, and manufacturing. Having established that DLT provides the transparency and immutability platform for building a trusted system (with the deployment of smart contracts) we proceeded with the second phase (cycle 2) of the project where we aimed to successfully integrate with the different type of distributed applications for various domains. Having finished integration with the partner platforms, in this deliverable we showcase practical performance of experiments of DLTs in various application domains. This enabled us to also showcase, the valuation of IoT data in each use cases and the reliability of communication and computation overhead of actors in wireless IoT networks.

In conclusion, we documented the integration and application of Distributed Ledger Technologies (DLTs) to the three domains. All domains stand to benefit from the inherent characteristics of DLTs. Thus, DLT has been successfully exploited for accounting and auditing of recorded data, and thus nearly guarantee the trustworthiness of the system. To avoid more privacy and security concerns in the Intelllot framework, the DLTs system was integrated with HyperMAS and the SAP/MTD.

Furthermore, in this second version of the deliverable, an advancement in the research of data valuation and contribution incentivization has been showcased. Mainly, we have provided results that may bring the DLT implementation of fair healthcare systems to reality. The benefits of such systems are immeasurable as cross-collaboration of large and small healthcare institutions can be carefully balanced and adequately rewarded. Given the Shapley data valuation framework, the research advanced to also include the communication overhead of integrated systems that exploit DLT and federated learning. Finally, the FedToken implementation was proposed and tested, The results showed promising use of FedToken in unfair and distrustful FL networks by both lowering the communication overhead while improving the learning performance of the entire network.

The resulting designs and developments as part of IntellIoT have contributed to the following publications:

- Lam Duc Nguyen, Shashi Raj Pandey, Soret Beatriz, Arne Bröring, and Petar Popovski, "A Marketplace for Trading AI Models based on Blockchain and Incentives for IoT Data". IEEE Transaction (2021; *under review*)
- Nguyen, Lam Duc, Israel Leyva-Mayorga, Amari N. Lewis, and Petar Popovski. "Modeling and analysis of data trading on blockchain-based market in iot networks." *IEEE Internet of Things Journal* 8, no. 8 (2021): 6487-6497.
- Nguyen, Lam Duc, Amari N. Lewis, Israel Leyva-Mayorga, Amelia Regan, and Petar Popovski. "B-ETS: A Trusted Blockchain-based Emissions Trading System for Vehicle-to-Vehicle Networks." VEHITS 2021) - *Best Paper Award*.
- Soret, Beatriz, Lam D. Nguyen, Jan Seeger, Arne Bröring, Chaouki Ben Issaid, Sumudu Samarakoon, Anis El Gabli, Vivek Kulkarni, Mehdi Bennis, and Petar Popovski. "Learning, Computing, and Trustworthiness in Intelligent IoT Environments: Performance-Energy Tradeoffs." IEEE Transaction on Green Communication and Networking (2021; *under review*).
- Pandey, Shashi Raj, Lam D. Nguyen, and Petar Popovski. "A contribution-based device selection scheme in federated learning." IEEE Communications Letters 26.9 (2022): 2057-2061.
- Nguyen, Lam Duc, et al. "Analysis of distributed ledger technologies for industrial manufacturing." Scientific Reports 12.1 (2022): 18055.

While this is the last deliverable on the DLT implementation and integration with the partnered domains the work does not stop here. Within the scope of the integration with UCs, the initial testing and validation has commenced in all the three use cases, and it will be described in WP5 deliverables. In terms of research, we envisioned several promising directions after IntellIoT. Fully decentralized FL leads to distrustful and unfair networks as more practical matters are at issue. Specifically, the issue of added computation of data evaluation in terms of delay and energy needs to be considered. As the strongest flaw of the DLT system is its energy usage, an evolvement towards the proof of stake

system implemented by Ethereum needs to be considered when proposing a futureproof solution to the three domains of the Intelllot project.

REFERENCES

[1] Urquhart, Andrew. "The inefficiency of Bitcoin." Economics Letters 148 (2016): 80-82.
[2] L. D. Nguyen, I. Leyva-Mayorga, A. N. Lewis and P. Popovski, "Modeling and Analysis of Data Trading on DLT-Based Market in IoT Networks," in *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6487-6497, 15 April15, 2021, doi: 10.1109/JIOT.2021.3051923.
[3] Fernández-Caramés, Tiago M., and Paula Fraga-Lamas. "A Review on the Use of Blockchain for the Internet of Things." Ieee Access 6 (2018): 32979-33001.
[4] Huh, Seyoung, Sangrae Cho, and Soohyung Kim. "Managing IoT devices using blockchain platform." In 2017 19th international conference on advanced communication technology (ICACT), pp. 464-467. IEEE, 2017.
[5] Wang, Wenbo, Dinh Thai Hoang, Peizhao Hu, Zehui Xiong, Dusit Niyato, Ping Wang, Yonggang Wen, and Dong In Kim. "A survey on consensus mechanisms and mining strategy management in blockchain networks." IEEE Access 7 (2019): 22328-22370.
[6] Tschorsch, Florian, and Björn Scheuermann. "Bitcoin and beyond: A technical survey on decentralized digital currencies." IEEE Communications Surveys & Tutorials 18, no. 3 (2016): 2084-2123.
[7] Gaži, Peter, Aggelos Kiayias, and Dionysis Zindros. "Proof-of-stake sidechains." In 2019 IEEE Symposium on Security and Privacy (SP), pp. 139-156. IEEE, 2019.
[8] Gaži, Peter, Aggelos Kiayias, and Alexander Russell. "Stake-bleeding attacks on proof-of-stake blockchains." In 2018 Crypto Valley Conference on Blockchain Technology (CVCBT), pp. 85-92. IEEE, 2018.
[9] Buterin, Vitalik, Daniël Reijsbergen, Stefanos Leonardos, and Georgios Piliouras. "Incentives in Ethereum's hybrid Casper protocol." International Journal of Network Management 30, no. 5 (2020): e2098.
[10] Zhang, Yuanyu, Shoji Kasahara, Yulong Shen, Xiaohong Jiang, and Jianxiong Wan. "Smart contract-based access control for the internet of things." IEEE Internet of Things Journal 6, no. 2 (2018): 1594-1605.
[11] Putra, Dwiyan Rezkia, Bayu Anggorojati, and Ardhi Putra Pratama Hartono. "Blockchain and smart-contract for scalable access control in internet of things." In 2019 International Conference on ICT for Smart Society (ICISS), vol. 7, pp. 1-5. IEEE, 2019.
[12] Onik, Md Mehedi Hassan, and Mahdi H. Miraz. "Performance analytical comparison of blockchain-as-a-service (baas) platforms." In International Conference for Emerging Technologies in Computing, pp. 3-18. Springer, Cham, 2019.
[13] Pustišek, M. and Kos, A., 2018. Approaches to front-end IoT application development for the ethereum blockchain. Procedia Computer Science, 129, pp.410-419.
[14] Dorri, Ali, Salil S. Kanhere, Raja Jurdak, and Praveen Gauravaram. "Blockchain for IoT security and privacy: The case study of a smart home." In 2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops), pp. 618-623. IEEE, 2017.
[15] Hong, Sunghyuck. "Survey on analysis and countermeasure for hacking attacks to cryptocurrency exchange." Journal of the Korea Convergence Society 10, no. 10 (2019): 1-6.
[16] Androulaki, Elli, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." In Proceedings of the thirteenth EuroSys conference, pp. 1-15. 2018.
[17] Androulaki, Elli, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." In Proceedings of the thirteenth EuroSys conference, pp. 1-15. 2018.
[18] Androulaki, Elli, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." In Proceedings of the thirteenth EuroSys conference, pp. 1-15. 2018.

[19] Peng, Kun. "A general, flexible and efficient proof of inclusion and exclusion." In Cryptographers' Track at the RSA Conference, pp. 33-48. Springer, Berlin, Heidelberg, 2011.
[20] D. -L. Nguyen, I. Leyva-Mayorga and P. Popovski, "Witness-based Approach for Scaling Distributed Ledgers to Massive IoT Scenarios," *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, 2020, pp. 1-6, doi: 10.1109/WF-IoT48130.2020.9221269.
[21] L. D. Nguyen, A. E. Kalor, I. Leyva-Mayorga and P. Popovski, "Trusted Wireless Monitoring Based on Distributed Ledgers over NB-IoT Connectivity," in *IEEE Communications Magazine*, vol. 58, no. 6, pp. 77-83, June 2020, doi: 10.1109/MCOM.001.2000116.
[22] https://geth.ethereum.org/
[23] https://nodejs.org/en/
[24] Bermeo-Almeida, O., Cardenas-Rodriguez, M., Samaniego-Cobo, T., Ferruzola-Gómez, E., Cabezas-Cabezas, R. and Bazán-Vera, W., 2018, November. Blockchain in agriculture: A systematic literature review. In International Conference on Technologies and Innovation (pp. 44-56). Springer, Cham.
[25] Hölbl, M., Kompara, M., Kamišalić, A. and Nemec Zlatolas, L., 2018. A systematic review of the use of blockchain in healthcare. Symmetry, 10(10), p.470.
[26] Hölbl, M., Kompara, M., Kamišalić, A. and Nemec Zlatolas, L., 2018. A systematic review of the use of blockchain in healthcare. Symmetry, 10(10), p.470.
[27] Lu, Y., Huang, X., Dai, Y., Maharjan, S. and Zhang, Y., 2019. Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. IEEE Transactions on Industrial Informatics, 16(6), pp.4177-4186.
[28] Ye, H.; Park, S. Reliable Vehicle Data Storage Using Blockchain and IPFS. Electronics 2021, 10, 1130.
[29] Danzi, Pietro, Anders E. Kalor, Rene B. Sorensen, Alexander K. Hagelskjær, Lam D. Nguyen, Cedomir Stefanovic, and Petar Popovski. "Communication aspects of the integration of wireless iot devices with distributed ledger technology." IEEE Network 34, no. 1 (2020): 47-53.
[30] https://ethereum.org/en/dapps/
[31] https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/
[32] Donevski, Igor, Jimmy Jessen Nielsen, and Petar Popovski. "On addressing heterogeneity in federated learning for autonomous vehicles connected to a drone orchestrator." Frontiers in Communications and Networks 2 (2021): 709946.
[33] Pandey, Shashi Raj, Lam Nguyen, and Petar Popovski. "FedToken: Tokenized Incentives for Data Contribution in Federated Learning." Workshop on Federated Learning: Recent Advances and New Challenges (in Conjunction with NeurIPS 2022).
[34] Nguyen, Lam Duc, et al. "Analysis of distributed ledger technologies for industrial manufacturing." Scientific Reports 12.1 (2022): 18055.