IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 July 2022, accepted 1 August 2022, date of publication 5 August 2022, date of current version 11 August 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3196933*

RESEARCH ARTICLE

# Toward the InterPlanetary Health Layer for the Internet of Medical Things With Distributed Ledgers and Storages

**GIOELE BIGINI AND EMANUELE LATTANZI**
Department of Pure and Applied Sciences, University of Urbino, 61029 Urbino, Italy
Corresponding author: Gioele Bigini (g.bigini@campus.uniurb.it)

This research was funded by Regione Marche with Decreto del Dirigente della Posizione di Funzione (DDPF) n. 1189 and by Department of Pure and Applied Sciences, University of Urbino.

**ABSTRACT** With the dramatic increase of the Internet of Medical Things devices, self and remote health data monitoring is consistently receiving more attention. However, medical devices are usually challenging to deploy due to privacy regulations, and they generally leverage a centralized third party. Enabling data sharing would enhance new medical studies, formulate new treatments, and deliver new digital health technologies. Solving the issue will have a triple impact: we will handle sensitive information easily, contribute to international medical advancements, and enable personalized care. A possible solution is to exploit decentralization distributing privacy concerns directly to users. Solutions enabling this vision are closely linked to Distributed Ledger Technologies. Through its characteristics of immutability and transparency, this technology would allow privacy-compliant solutions in contexts where privacy is the first need. This paper envisions the InterPlanetary Health Layer and related real-world implementations in the Internet of Medical Things domain. The main idea of the proposed solution is to handle sensitive data by preserving privacy and guaranteeing data availability. Specifically, users can build their private network, collaboratively authorize operations among their data and manage their privacy conditions without relying on a third party. The results of several stress tests conducted on a real case study confirmed the feasibility of the proposed solution, which shows good scalability and a modest impact on the application performance measured during the decentralized data access.

**INDEX TERMS** Decentralized Internet of Medical Things, distributed ledger technology, InterPlanetary file system, InterPlanetary health layer.

## I. INTRODUCTION

The Internet of Medical Things (IoMT) is a subset of the Internet of Things (IoT) that involve medical IoT devices. The ability to deploy medical devices is mainly linked to the regulations in force of the specific deployment territory. The reason why policies are strict is usually related to the fact that medical devices can achieve remote monitoring and tracing of sensitive information. Enabling the data sharing in the IoMT will significantly impact people's lives since the wealth of valuable information could be used to increase their wellness, advance new medical techniques, and encourage new studies.

Traditional healthcare systems are mostly based on self-managed infrastructures. The trust of centralized entities represents the usual pursued solution for health data management, but centralized solutions are closed to external stakeholders and lack transparency. Moreover, the infrastructure needs to update over time to avoid malicious attacks and has limited capabilities in terms of data storage.

Collecting data using centralized entities impacts accessibility and availability since all the responsibility is charged to the provider that focuses the attention on being

The associate editor coordinating the review of this manuscript and approving it for publication was Barbara Guidi.

compliant with regulations. Generally, when different healthcare institutions need to collaborate, they agree on sharing and outsourcing health data to cloud computing services for medical studies, i.e. the ''registers of pathologies'' are shared tools to collect and analyze personal, clinical, and health data [1]. However, it does not help new data-sharing steps outside of specific agreements.

Because of the lack of decentralized approaches, the need for a shift to decentralized management in the healthcare sector seems desirable. Shifting to the decentralized paradigm should increase the efficacy of healthcare institutions and the wellness of the people. Distributed Ledger Technology (DLT) and Distributed File Storage (DFS) are promising technologies that could lead to decentralized healthcare. Some examples of DLTs derivatives are Bitcoin [2], and Ethereum [3], which are general-purpose implementations of a DLT: what is generally referred to as the blockchain. While examples of DFS are: the IPFS, a peer-to-peer (P2P) network to store large amounts of information, guaranteeing fault tolerance in a decentralized manner; the Solid platform, a project led by Sambra *et al.* at MIT that provides secure storage for the exchange of data on the new decentralized web [4]. We think combining these technologies could lead to secure and decentralized data layers with an international scope.

This work presents a new distributed data access layer, the InterPlanetary Health Layer (IPHL), which we believe could be useful to the healthcare industry to enable data sharing. The implementation of this layer is based on a DLT network built with the experimental IBM Hyperledger framework. The implementation considers the technological barriers of mobile devices present nowadays, namely computation and storage capabilities, and represents a starting point for our vision of secure decentralized data layers. The contributions of this work can be summarized as follow:

- we provide a new shared, agnostic, and permissioned decentralized data layer with enhanced data availability;
- we implement the proposed architecture on a real-world use case represented by a traditional IoMT application;
- we carefully characterize the architecture to assess its feasibility and performance;
- we provide for downloading the entire source code of the proposed architecture.

The rest of the paper is organized as follows: Section II describes the background, while Section III contains the related work. In Section IV we give our proposed solution by envisioning the IPHL first and later describing our implementation. In Section V, we describe the IoMT use case and in Section VI we discuss the experimental results. Finally, we draw conclusions and highlight the current limitations in Section VII.

## II. BACKGROUND

This section describes the concepts and technologies underlying the proposed software architecture.

### A. INTERNET OF MEDICAL THINGS

The Internet of Medical Things (IoMT) comprises medical devices and applications connected to individuals' health. A typical infrastructure includes several devices connected to centralized locations on the internet, enabling the user to collect medical data. Thanks to the intrinsic property of IoMT devices of being personal and interconnected, they can capture data about the individual and be used as a source of knowledge for the healthcare sector. Examples of IoMT-enabled scenarios include: remote patient monitoring of people with chronic or long-term conditions; tracking patient medication orders and patients' location; the storing of valuable information to professionals and caregivers [5], [6].

However, these devices represent a perfect target for malicious users. Consequently, most of the IoMT solutions available in the market make intensive use of anonymization, intending to remove personally identifiable information but impacting data quality. The latter could limit machine learning and artificial intelligence. So, enabling the data sharing in the IoMT could be essential for feeding the future artificial intelligence, i.e. the Artificial General Intelligence by Goertzel and Pennachin [7]. As underlined by Mesko *et al.*, the exploitation will allow the individuals' efficient healthcare while having more accurate diagnoses, monitoring, and better overall healthcare [8].

### B. DISTRIBUTED LEDGER TECHNOLOGY

DLTs provide an immutable ledger that guarantees tamperproof. The resistance to manipulation makes DLTs an up-and-coming technology for developing new types of applications where immutability and transparency represent a requirement.

Examples of these implementations can be found in general-purpose blockchains, i.e. Ethereum [3]. Other DLT implementations are domain-specific, such as Hyperledger Fabric [9], a distributed, enterprise-grade, proven, open ledger platform. It provides advanced privacy controls to share only specific data among network participants.

A concept generally related to DLT is the ability to support smart contracts. Smart contracts, the term initially coined by Szabo [10], are code deployed on a DLT that facilitates, verifies, or enforces the negotiation or execution of a contract. This code is executed deterministically by different participants in the DLT, who receive the same inputs and then perform a computation that leads to the same outputs while taking care of the integrity of the ledger. When a smart contract is deployed on the DLT, and the issuer is confident that the code embodies the intended and proper behaviour by reviewing the code, the transactions originated cut the middle-man.

In some specific DLTs as Hyperledger Fabric, in order to define a smart contract, writing the Chaincode is needed. The Chaincode is a program written in several common languages, such as Golang and Javascript, that implements

a prescribed interface generally referred to as business logic. It could be considered an envelope over smart contracts to avoid specific programming languages such as Solidity in Ethereum. This allows the definition of multiple smart contracts as its implementation is practically a container. Every chaincode comes with an endorsement policy that extends to each relevant smart contract linked to it that specifies which organization must sign a smart contract-generated transaction. In other words, it is a smart contract bound to an endorsement strategy.

### C. DISTRIBUTED FILE STORAGE

A DFS offers an alternative way to store files compared to traditional databases. It comprises a network of peer nodes with their storage, which follows the same protocol for content storing and retrieval, offering high data availability and resilience because of the data *replication*. Unlike a centralized server operated by a single entity, the storage consists of a peer-to-peer network of participants who hold a portion of all existing data, creating fault-tolerant file storage and sharing system network. A feature of DFS is that peer nodes do not have to trust each other to store and access data, which makes it similar to DLTs, but in contrast, there is no guarantee of immutability.

An example of DFS is the InterPlanetary File System (IPFS) proposed by Benet [11]: a protocol that builds a distributed file system over a P2P network. The IPFS network stores and shares files and directories as content Identifiers objects (CID). The CID acts as a universal identifier linked to the resource stored in the network. This means that different nodes can retrieve the files and even share the same identifier in a decentralized manner.

### D. DECENTRALIZED ACCESS CONTROL MECHANISMS

The Access Control Systems (ACS) is a mechanism used to express conditions that determine whether or not specific computer system resources can be accessed. They usually rely on a central entity, carrying all the risk of a single point of failure. We could extend the potential of a traditional ACS by employing a distributed solution such as a DLT.

The literature proposes different approaches to decentralized ACS that extensively use smart contracts. The Mandatory ACS employ a solution similar to the one proposed by Zichichi *et al.* and consists of the ability of a subject to access data through smart contracts [12]. The Discretionary ACS by Zyskind *et al.* considers the data management off-chain using the DLT as a store for the access control policy [13]. Another possible solution comes by employing Role-based ACS by Cruz *et al.*, which achieves authentication based on user roles in the system [14].

## III. RELATED WORK

In the IoMT, many works focus on user privacy and user-centricity. Researchers attempt to envision a system where we could effectively store and own our data. Literature offers many insights into the properties needed to do so with all converging on several topics such as: decentralization of the system; authentication of data; handling scalability of data.

The first attempts in the field were mostly related to the healthcare institutional domain. Jiang *et al.* [35] pointed out how the direction of health information exchange should integrate blockchain-based systems. Along with them, several other researchers made the same assumption as Srivastava *et al.* [36], Seliem and Elgazzar [37], Uddin *et al.* [38], and Marangappanavar and Kiran [39]. As already said, the proposed visions were more ''institutions-oriented'' and so focused on the possibility of connecting healthcare institutions to nimbly share EHR data, in a few cases including even medical mobile devices. Other visions focused on emergency management Tantidham and Aung [40], telemedicine in the case of Kordestani *et al.* [41], and then other works as from Nascimiento *et al.* [42] propose the tracking of data flows that may occur in the network.

All these attempts have in common the usage of DFS, Smart Contracts, and DLT. The primary technology of choice for the DFS is generally IPFS or, in some cases, cloud storage, while the DLT could vary by the requirements required by researchers, from public blockchains to permissioned DLTs or even Directed Acyclic Graphs (DAG). Saweros and Song [15], Donawa *et al.* [16], Lucking *et al.* [17], and Cisneros *et al.* [18] give more relevance to scalability issues inherited by blockchains solutions and so, privileging the usage of DAGs through ledgers as IOTA. Other researchers such as Adlam and Haskins [19] and Fernandes *et al.* [20] gave more importance to the user's identification in the system and to the ability to set permissions on the ledger by using permissioned DLTs such as Hyperledger.

Over time, researchers seem to converge on decoupling data from DLTs to avoid sensitive data being immutably stored on the ledger and focus on this feature. They even came out with a concrete framework or solution embedding the IoMT scenario in a few cases. Among the earliest works, Dwivedi *et al.* focused on a solution to allow users to control the data but rely on centralized entities to receive an identity and register in the system [21]. Similar works followed by Arul *et al.*, Garg *et al.*, Stamatellis *et al.* and Mani et al proposing solutions for EHR data management [22]–[25]. Other researchers focused on the possibility of linking institutions employing DLT and investigating the scalability of these systems and the possibility of tracking data for provenance, i.e. Madine *et al.* [26], Nguyen *et al.* [27], and Xu *et al.* [28]. Interesting about the latter is the introduction of nodes interacting with external stakeholders. Moreover, other works in the field do not necessarily focus on long-lasting health data management but sometimes on short-lasting health data, like healthcare passports or clinical studies as in the work of Omar *et al.* [29].

So, the first works in the area of IoMT slowly came out, but still, there are not many concrete implementations available. They primarily provide guidelines with attempts to use the implementation investigating the data-sharing

**TABLE 1.** List of considered IoMT related works compared with the current proposal.

| Work | DLT | IoMT Oriented | Real-world experiments | User Participation |
|---|---|---|---|---|
| Saweros et al. [15] | DAG | ✓ | | |
| Donawa et al. [16] | DAG | | | |
| Lucking et al. [17] | DAG | | | |
| Cisneros et al. [18] | DAG | ✓ | | |
| Adlam et al. [19] | Permissioned | ✓ | | |
| Fernandes et al. [20] | Blockchain | ✓ | ✓ | |
| Dwivedi et al. [21] | Blockchain | ✓ | | |
| Arul et al. [22] | Blockchain | ✓ | | |
| Garg et al. [23] | Blockchain | ✓ | | |
| Stamatellis et al. [24] | Permissioned | ✓ | | |
| Mani et al [25] | Permissioned | | | |
| Madine et al. [26] | Blockchain | | | |
| Nguyen et al. [27] | Blockchain | ✓ | ✓ | |
| Xu et al. [28] | Permissioned | ✓ | | |
| Omar et al. [29] | Blockchain | | | |
| Kumar et al. [30] [31] [32] | Permissioned | ✓ | | |
| Egala et al. [33] | Permissioned | ✓ | | |
| Abdellafi et al. [34] | Permissioned | | | |
| Garg et al. [23] | Permissioned | ✓ | | |
| **This Work** | Permissioned | ✓ | ✓ | ✓ |

issue. Kumar and Chand, Kumar *et al.*, and Kumar and Tripathi [30]–[32] move toward including data from the IoMT, trying to provide a solution for medical device storage and authentication that can prevent security and privacy obstacles in the IoMT. The solution, called MedHypChain, is built on Hyperledger Fabric and focuses on authenticating and authorizing patient data while protecting the dissemination of medical device information in the blockchain network. Medical devices in the IoMT generate data transmitted to the blockchain network. The IPFS cluster is responsible for facilitating patient synchronization and providing secure information storage, while smart contracts are used to achieve consensus. Egala *et al.* [33] proposed an architecture based on IoMT and encryption methods that intend to provide privacy, security, traceability, low latency, low storage cost and data availability. It comprises three layers: IoT sensors, a combination of the edge and cloud paradigms, and the so-called ''data consumer layer,'' i.e., devices that leverage data from sensors. To achieve decentralized EHRs, they maintain a public ledger for medical records and critical events to provide traceability. Smart contracts are used to help medical professionals perform event-based automation tasks without human interference. Abdellafi *et al.* [34] propose the MEdge-Chain, which includes several e-health entities whose role is to monitor, promote and maintain people's health. The blockchain architecture is a consortium-based multichannel architecture that enables secure access, processing and sharing of medical data among different electronic health entities. Garg *et al.* [23] propose a real-time data solution with different use cases of implanted medical devices (IMDs), namely neurostimulators, cochlear implants, cardiac pacemakers and gastric stimulators. The system leverages an individual server node near the patient, which collects data from the IMDs. The server then transmits the collected data to the cloud server, which receives and stores it and then uses it for further processing. When remote users are interested in obtaining stored data, they must obtain access from a trusted authority.

Ultimately, our solution aims to distance itself from previous ones by giving a concrete implementation, supported by its source code, ensuring data availability and involving users in the data sharing process, leveraging the DLT and Smart Contracts to manage the information stored in the DFS efficiently. Data are never exchanged with remote users without consent and are subject to verifiable transactions and access mechanisms coded in smart contracts. The user can create a private network (Halo Network) to improve data availability and sharing. The insertion of a social component should make the architecture resilient to failures. Users can create networks of trust through which they share their choices about the data with their locations, and all are governed transparently. Along with the architecture, we also present an application of the IoMT called Balance that interacts with the network by demonstrating the real-world usage in a decentralized domain. We hope that through our work, we can provide the basis for creating an international secure data layer of sensitive information that all stakeholders can tap into.

Finally, in Table 1 we summarize the important details related to the previously mentioned works. Without considering the desirability of providing open source code, almost all works provide possible IoMT solutions without concern for how they might involve the same user in the decentralization of the system and data availability, assuming that this is likely to be held by those who want to create the network, i.e. central institutions. In addition, few works provide real-world applications of such works, with tangible experiments and combinations of DLTs and IoMT use cases. Moreover, to the best of our knowledge, no work seems to speculate on how the user constitutes an active operator in the network, that

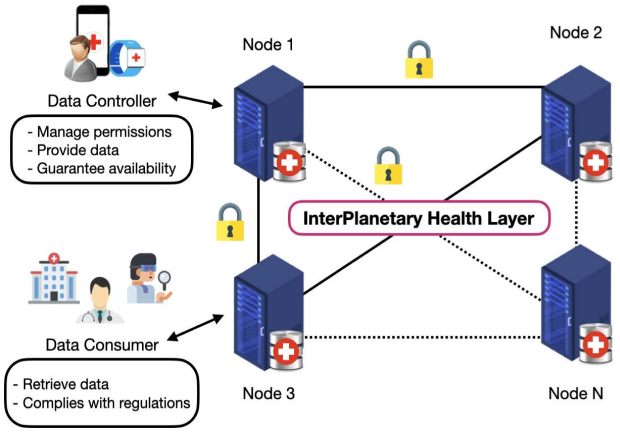



**FIGURE 1.** Conceptualizing the InterPlanetary Health Layer.

is, making the user responsible for the network rather than relying exclusively on known entities for decentralization.

## IV. PROPOSED SOLUTION

The literature does not explicitly mention the need for a shared, agnostic, and international layer that aims to collect sensitive data of individuals. We could look at the layer as a subset of the web of data where information is kept private in a decentralized manner. Below, we describe the big picture of it, and then we go through our implementation.

### A. THE InterPlanetary HEALTH LAYER

We propose the InterPlanetary Health Layer (IPHL) as an international, shared, agnostic, and permissioned decentralized data lake containing the IoMT data. The IPHL should enable individuals to act as data managers of their data. We describe two main roles: the data controllers (or subjects), which are the users in control over their data; the data consumers, which are the users interested in retrieving the data.

The big picture of the system is described in Figure 1:

- **The IPHL**: it is a network where individuals can exchange information without relying on a central authority for storing information. The network allows establishing private and secure communications channels for data sharing and enables individuals to manage permissions on data. Moreover, participants in the network should act as operators, which means they maintain it in case of failure. The participants in the network could be everyone interested in sharing IoMT data, and so the medical device owners as well as researchers, doctors, and healthcare professionals in general.
- **The Data Controller**: users IoMT devices, such as smartphones or smartwatches acting as medical devices, interact with the IPHL providing data and managing permissions. They are the Data Controllers in the network, being able to interact with nodes (possibly their nodes) and manage the data made available by their applications. They could be citizens, ordinary individuals.
- **The Data Consumer**: Remote users are the stakeholders. They could be researchers, doctors, healthcare research institutions, universities, and healthcare professionals. They interface with the IPHL network and may be interested in collecting information from the IoMT devices. They interact with the nodes triggering the data access mechanisms and initiating the process of sharing.

### B. IMPLEMENTATION

We called the proposed IPHL implementation Halo Network, as described in Figure 2.

The Halo Network is based on two layers: a permissioned Hyperledger Fabric (HLF) network that support smart contracts and the InterPlanetary FileSystem (IPFS). Hyperledger Fabric is an experimental DLT from IBM and part of the Linux Foundation. The reason for considering a combination of the technologies is that DLTs expose immutable information, which is never advisable for personal data, even in the case of permissioned ledgers. So, storing data off-chain is the available alternative.

In the following, we devise the Halo Node implementation in more detail and later focus the attention on the Chaincode. The network's source code is open source and can be found by following this link: *https://github.com/BigG-DSC/fabric-network*.

#### 1) THE HALO NODE

The Halo Node acts as an access point to users, allowing them to participate in the Halo Network and manage the personal data submitted by the IoMT devices. The source code can be found by following this link: *https://github.com/BigG-DSC/halo-node*. The node can be divided into four main parts as described in 2:

- **Graphic User Interface**: the visualization panel consists of a web interface designed as a single web page application. It is accessible on the installed machine and was designed to allow the easy management of the node from the user. It is built with Jinja on top of the Flask framework, serviced by a Web Server Gateway Interface. It lets the user establish the connection and easily retrieve updates from the node. The event notification service is triggered every time the user accesses the visualization interface and tries to establish the connection with the node. Among the enabled functions, it allows transferring data to the node, accepting requests, and participating in the voting session.
- **Core**: it contains the node's logic, and it deals with the GUI and both the DFS and the Peer. Whenever interactions in the GUI occur, the Flask server dialogues with the corresponding underlying DLT and DFS applying the logic required. The module was specifically thought for managing current implementations and enabling future works acting as a middleware for the different underlining technologies. This ensures that the

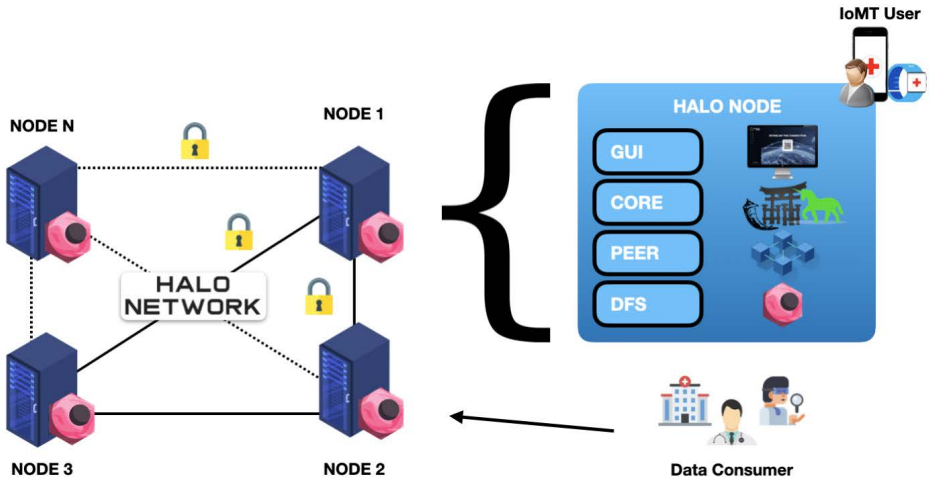



**FIGURE 2.** Halo Network Architecture.

new layers can be easily added and that all operations pass through it.

- **Peer**: it allows interaction with the Halo Network. It is a Hyperledger Fabric peer communicating with the rest of the network performing invocations to Chaincodes and a NodeJS application to allow easy interaction with the peer. The Peer consists of holding the shared ledger, ensuring immutability and transparency, and making it possible to store the history of transactions in the network. It is built by implementing the peer provided by the HLF and an application written in NodeJS employing the Fabric SDK. The code helps to easily interact with the DLT when requests come from the Core. As soon as a request is received, the Core can dialogue with the peer responsible for the readings and writings on the ledger.
- **DFS Module**: IPFS is a distributed storage that does not organize data for querying. To solve the problem, we use OrbitDB: a decentralized database built on top of the IPFS, independent and secure, that allows data to be stored in a distributed manner with many replicas. As discussed by Shapiro *et al.*, the replicas are constantly synchronized among all available peers and rewritten according to Conflict-free Replicated Data Types [43]. Thanks to OrbitDB, IPFS-based applications have a register to consult to handle the IPFS as a database but distributed. In addition, it allows participants to set read and write permissions reflected in the ability not to disclose data stored. It is built by implementing the OrbitDB library with an application written in NodeJS.

#### 2) THE CHAINCODE

A Chaincode is created above each node. A Chaincode is an implementation containing several smart contracts available on each peer. The stored data structure is a list of lists that represent the ACS constituted by the Pools and the list of votes from participants. Its functionality is described below:

- **CreatePoll**: this operation is dedicated to inserting a new vote when it is requested by an external entity, such as a physician. Following such a request, the receiving node notifies, through the DLT, that a new request has been received and creates an entry in the ledger so that all nodes can cast a vote.
- **Approve/Decline**: the operation allows voting on the ballot. The operation allows the ledger to be updated with the vote of the considered user. Each user has an identity within the ledger and can write its vote within the smart contract. Based on the policy chosen for vote validation, the node interested in closing the vote will wait for all votes to be received and only then can it do so.
- **ClosePoll**: the closing operation and an update performed by a participant at the time he or she has the opportunity to do so. It consists of confirming the outcome of the vote and then granting or not granting permission for access to the data by an external user.
- **GetPoll/GetAllPolls**: allows the retrieval of information saved on the ledger. This can be done by requesting a specific voting id or by recalling all votes.

The source code of the Chaincode can be found at this location *https://github.com/BigG-DSC/fabric-contracts*.

## V. THE USE CASE

In this work, as a use case, we consider a platform called Balance, collecting human postural stability data from signals gathered by the Inertial Measurement Unit (IMU) provided by a common smartphone.

### A. BALANCE SMARTPHONE APPLICATION AND THE CURRENT IoMT SCENARIO

Balance consists of a mobile application and a backend that resides in a centralized location, as shown in Figure 3. The data collected are based on the Romberg test, which is a standard test performed by professionals to assess postural

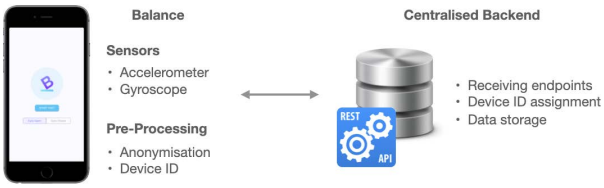



**FIGURE 3.** Balance Centralised Architecture.

stability. They ask the patient to keep their eyes open or closed to understand the influence of the visual system on posture. The application allows for repeatable tests performed through the smartphone in a few seconds, and the appropriate mode (eyes closed or open) can be chosen on the home screen. Specifically, the analysis generates the sway-path (SWP), which represents the projection of the displacement of the Center of Gravity (COG) over the floor. Normally it is also represented by its components in the anteroposterior (AP) and mediolateral (ML) directions w.r.t. the body position. Starting from the SWP the Balance application automatically extracts global and structural parameters: the former belongs to methods whose aim is to estimate the overall size and features of the sway patterns, while the latter is based on the decomposition of sway trajectories into sub-units that can potentially be related by their role w.r.t. the underlying motor control processes [44]. For the complete list of the numerical features collected by Balance refer to Table 2 while the complete source code of the application can be found at: *https://github.com/ComputerScienceUniUrb/balance-mobile*.

The smartphone application performs all the pre-processing onboard to comply with privacy regulations. For this reason, the user receives a unique token ID not associated with him or his device to refer to his data correctly. Moreover, as part of the required data, the user needs to provide some personal information such as height, age, gender, and some anamnesis data such as any postural, hearing, and vision problems, and some information about personal habits such as alcohol and medicine assumption or sports activities.

### B. DECENTRALISING BALANCE WITH THE HALO NETWORK

We modified the source code of Balance to integrate it with the Halo Node. Interactions with the network are based on the possibility that third parties can communicate with the peer target. Once a request is received, the peer collects it and notifies the mobile device in its first interaction.

The mobile device interacts with its node, can send its data, and is updated on external requests to vote and receive information about network participants. Figure 4 shows the GUI that interacts with the Halo Node and, consequently, the Halo Network:

- The Halo Network: the function allows the user to see the users added to their network and who participate in maintaining the data by contributing to its availability.

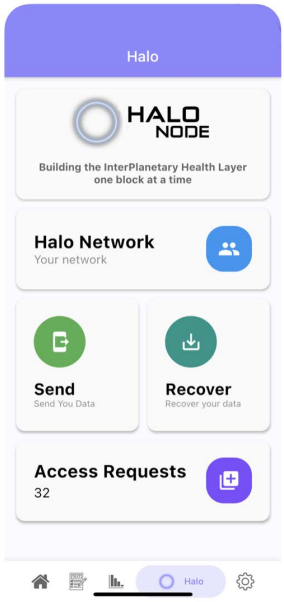



**FIGURE 4.** Halo Node GUI Integration on Balance.

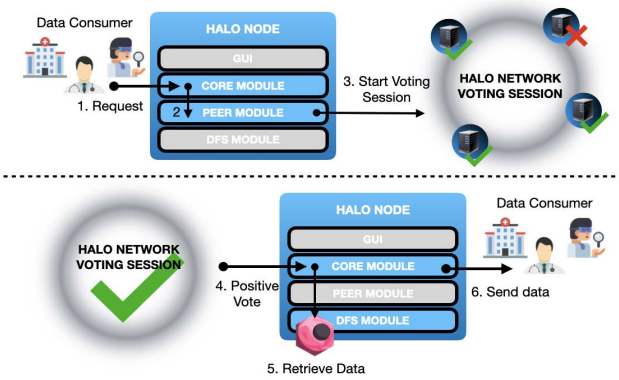



**FIGURE 5.** The Sharing Phase.

- Send: the function allows the user to transfer their data to their repository on IPFS through OrbitDB.
- Recover: the function allows the user to recover his data in the case he needs to, i.e. if the device is re-initialized.
- Access Requests: the function allows the user to keep track of all requests made by the network and vote on them while maintaining control over his data.

Finally, we can distinguish two primary workflows: the Sharing Phase and the Storing/Retrieving Phase are described below. The source code of the smartphone application can be found at the following location: *https://github.com/BigG-DSC/balance-decentralized*.

#### 1) THE SHARING PHASE

The Sharing Phase involves the interaction with the DLT and retrieving data from the distributed storage. It could be costly, including encryption techniques, and implies increasing response times. The workflow is described in Figure 5,

**TABLE 2.** List of parameters computed by Balance application (extracted from the work presented by Lattanzi *et al.* in 2020 [44]).

| List of features | | | |
|---|---|---|---|
| **Category** | **Symbol** | **Dimension** | **Description** |
| **Time domain** | $SWP$ | $mm/s$ | Sway-path: the total length of the COG trajectory. |
| | $SWA$ | $mm^2/s$ | Sway-area: the area spanned from the COG trajectory. |
| | $DIST$ | $mm/s$ | Mean distance from the COG center. |
| | $STD_{AP,ML}$ | $mm/s$ | Standard deviation of the COG displacement on AP and ML axes. |
| | $R$ | $mm/s$ | Range of the COG: the maximum distance between two points of the trajectory |
| | $AR$ | $adimens.$ | Axes rate: the rate between the axes of the ellipse containing 90% of the data points |
| **Freq. domain** | $FP_{AP,ML}$ | $Hz$ | Frequency peak of the spectrum for AP and ML axes. |
| | $FM_{AP,ML}$ | $Hz$ | Mean frequency of the spectrum for AP and ML axes. |
| | $F80_{AP,ML}$ | $Hz$ | The frequency band containing 80% of the power of the spectrum calculated for AP and ML axes. |
| **Structural** | $NP$ | $unit$ | Mean number of SDC peaks per seconds. |
| | $MT$ | $s$ | The average time distance between two consecutive peaks of the SDC. |
| | $ST$ | $s$ | Standard deviation of MT. |
| | $MD$ | $mm$ | The average spatial distance between two consecutive peaks of the SDC. |
| | $SD$ | $mm$ | Standard deviation of MD. |
| | $MP$ | $s$ | The average duration of the SDC peaks. |
| | $SP$ | $s$ | Standard deviation of MP. |
| **Gyroscopic** | $GR_{x,y,z}$ | $degrees/s$ | Range of the gyroscopic signal for x,y, and z axes. |
| | $GM_{x,y,z}$ | $degrees/s$ | Max value of the gyroscopic signal for x,y, and z axes. |
| | $GV_{x,y,z}$ | $degrees/s$ | Variance of the gyroscopic signal for x,y, and z axes. |
| | $GK_{x,y,z}$ | $adimens.$ | Kurtosis index of the gyroscopic signal for x,y, and z axes. |
| | $GS_{x,y,z}$ | $adimens.$ | Skewness index of the gyroscopic signal for x,y, and z axes. |

and it involves the following steps: (i) the remote user makes the request; (ii) the request is registered by the peer who received it on the DLT; (iii) voting begins on the authorization of the request; (iv) when voting is closed, the remote user receives the requested information (if authorized), otherwise is declined.

The full sequence involves the following ten steps reported in the diagram shown in Figure 6:

1) A data consumer, i.e. a doctor, requests access to the receiving peer's data.
2) The node creates the poll by interacting with the DLT.
3) The DLT replies with and acknowledges.
4) The receiving node updates the mobile device regarding the request.
5) The receiving user initiates the vote, sending its own to the node.
6) The node redirects the request to the DLT, recording the vote on the smart contract.
7) The DLT replies with and acknowledges.
8) After voting, the user keeps waiting for the votes of the remaining participants.
9) The DLT replies with and acknowledges.
10) When all the votes have been received, the poll is closed.
11) A positive outcome triggers the retrieval of the data, granting permissions to the remote user along with requested data.
12) A negative outcome results in an access denied.

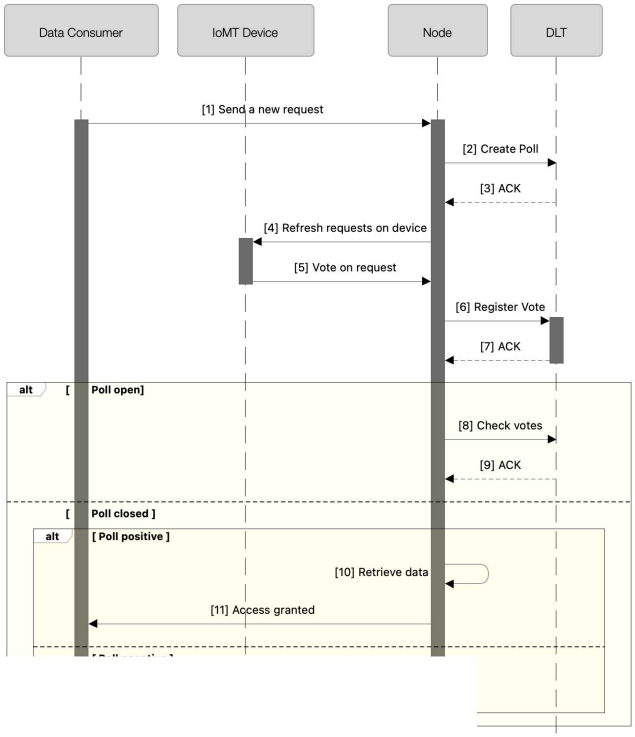



**FIGURE 6.** Sharing Phase sequence diagram.

### 2) THE STORING/RETRIEVING PHASE

The Storing/Retrieving Phase, described in Figure 7, can both involve the interaction through the mobile medical device or a data consumer. Specifically, it is triggered when the IoMT application establishes the connection with the Halo Node to transfer newly collected data or a data consumer asks for data. The sequence diagram describing the Retrieving Phase is highlighted in Figure 8. The Storing Phase is very similar and obtained by substituting the data consumer with the producer, sending the data to OrbitDB, storing it in the IPFS and receiving the final acknowledgement as the last step 6. The Retrieving Phase considers the following six steps:

1) A request for storing data or retrieving is received, i.e. following a remote request or by the user himself attempting to store or retrieve data.

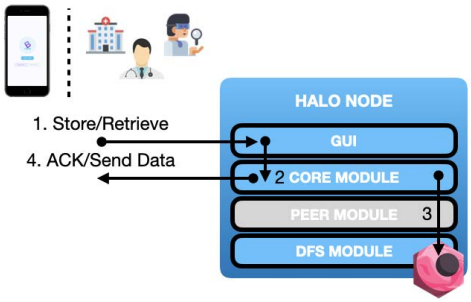



**FIGURE 7.** The Storing/Retrieving Phase.

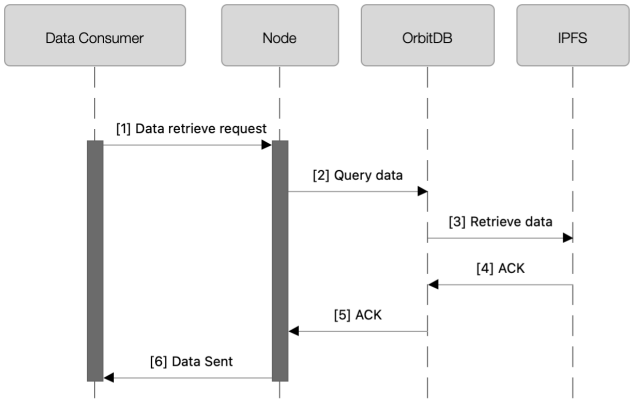



**FIGURE 8.** The Retrieve Phase sequence diagram.

2) The Core sends the query to the decentralized database OrbitDB.
3) The data is retrieved from IPFS with the help of OrbitDB.
4) The IPFS acknowledges the operation to OrbitDB
5) The OrbitDB acknowledges the operation to the Core
6) The data is finally sent back to the requestor

## VI. EXPERIMENTAL RESULTS

This section describes the evaluation of the implemented architecture and the tests performed. We focused on testing the interaction with DLT and DFS. As mentioned in Section IV, the DFS is responsible for storing data, while the DLT is responsible for data management.

### A. EXPERIMENTAL SETUP

For testing the DLT, we started a Hyperledger Fabric network of four nodes in a real-use case scenario in several positions around the globe: one in Europe, another one in the United States, and the last in China. The nodes have the following specifications: two cores, 4 GB RAM, 50 GB storage, and run Ubuntu 18.04 LTS. Since Fabric needs that each participant maintains its infrastructure, it is possible to facilitate the deployment through an overlay network with Docker Swarm. The fourth node is then responsible for synchronizing the ledger since Fabric consensus algorithm is deterministic.

For what concerns the DFS, we deployed the experiment on two VMs with the same specification as the previous one used for Fabric. The *read* and *write* tests have been carried on one machine only, that we call producer, while for testing the *replication* capabilities, we added the second one called a consumer. Both the producer and the consumer have the ability to interface with IPFS and use OrbitDB, which builds a decentralized database on top of it. Moreover, the producer is the creator of the the OrbitDB database, that means the consumer relies on it thanks to its replica. So, once it has established its own IPFS node, it does not bother to create a decentralized database but instead connects to the one created by the producer, sharing an actual database of its own.

The experiments were performed by interacting with the Halo Node through a remote client developed in NodeJS. Scripts written in Python were used for getting the host machine's performance (such as CPU load and network traffic).

### B. TESTING THE DISTRIBUTED LEDGER SYSTEM

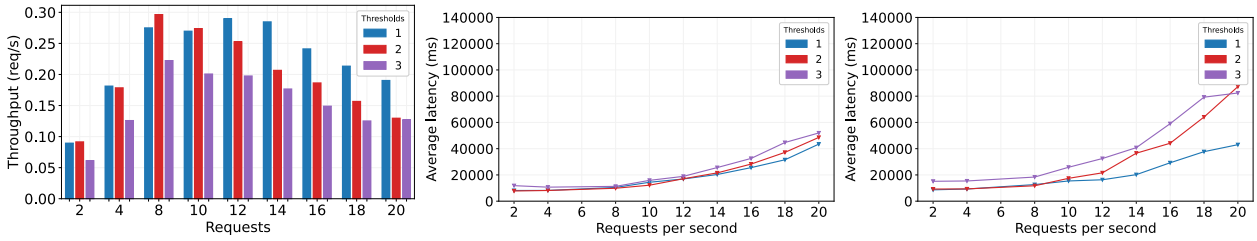
The workflow is composed of three primary operations accessible through the smart contract: (i) CreatePoll; (ii) Approve/Decline; (iii) ClosePoll. During the test, we simulate the delay of the real user in reacting to a new request to vote with a parameter randomly given by a Poisson Process with a mean $\lambda = 1000$ms. We collected information about the following parameters and metrics:

- Fixed parameters: the number of the maximum DLT nodes $n$ was set to 3. For each test, the same requests were repeated five times. This means that we average the timings of the same tests.
- Independent parameters: the threshold $t$ of the $(t, n)$-threshold scheme varies in the tests from 1 to 3, representing an increasing load on the DLT. A second parameter is the *number of requests per second* generated by the remote users, which varies from 2 to 20.
- Dependent metrics: the *request latency* which is the time between the submission of the request and its actual completion. Notice that this time is built on the first 7 contributions of the steps described in section V-B1.

#### 1) RESULTS

Figure 9 shows the latency measured for the voting process, writing into DLT, and updating into DLT with an increasing request per second and different threshold values. An increased threshold means the nodes in the network concurrently access the ledger. In general, the results show a clear dependence on the number of requests per second and on the value of $t$.

Plot (a) shows the system's throughput when increasing the requests per second. The chart shows a peak performance increase when 8 requests are sent to the node. Before that threshold, the system suffers under usage, and after that threshold, the overall performance degrades. The chart provides a measure of scalability in the sense that as the number of requests per second increases, the system is less efficient.




(a) System Throughput during Voting Phase (b) Write on DLT (c) Update on DLT

**FIGURE 9.** DLT throughput and requests latencies.

The results obtained give an indication of the system as a whole, considering the user's interaction with the voting system. As is easy to imagine, having the system to wait for more votes, it is no longer able to respond efficiently as the number of voters increases. Nevertheless, it remains usable and scalable despite the increase in waiting time.

For what concerns the inner operations Write and Update, it is interesting to highlight how they end up being different. The rightmost chart related to the Update operation in Figure 6 is the one in which the difference in the spread of the thresholds curves is most evident. Indeed, it seems that simultaneous interaction of multiple nodes (higher value of $t$) would cause longer wait times on the ledger, probably due to the access conflicts management. The increase of concurrent nodes updating the same data on the ledger confirms that the Fabric DLT uses lock-free optimistic concurrency, with rollback in case of dirty reads/writes.

Unlike the latter, the effect of an increasing number of nodes does not seem to affect the Write operation (Figure 9. (b)), probably because they only insert new entries in the ledger which do not generate conflicts.

In the best-case scenario, the DLT should establish about 8 concurrent network connections as we could suppose that performances will degrade over this number, being unable to guarantee low latencies on average. In the worst case, the average latency could almost double when the configuration is set to 20 data consumers.

### C. TESTING THE DISTRIBUTED FILE SYSTEM

The phases considered consist of the following operations:

- **Read**: this is the operation through which we stress the architecture by retrieving data on IPFS. At each step, an $N$ number of records is requested by a remote user in order to verify the response times of OrbitDB (linked to IPFS).
- **Write**: this is the operation through which we stress the architecture by injecting data on IPFS. At each step, an $N$ number of records is inserted by a remote and local user in order to check the response time of OrbitDB and IPFS.
- **Replication**: IPFS and OrbitDB work in tandem as information storage and organizer, this means that when a user has his own IPFS node, he only has to connect to a second OrbitDB node to know all about his data. OrbitDB can set permissions, which means that knowledge of information on IPFS can be locked and selective. By establishing the connection with the other node, the data is replicated. During the replication, we test the latencies between the producer (writing node) and the consumer (the reading or receiver node) as the number of records increases.

We observed the following parameters and metrics during the tests:

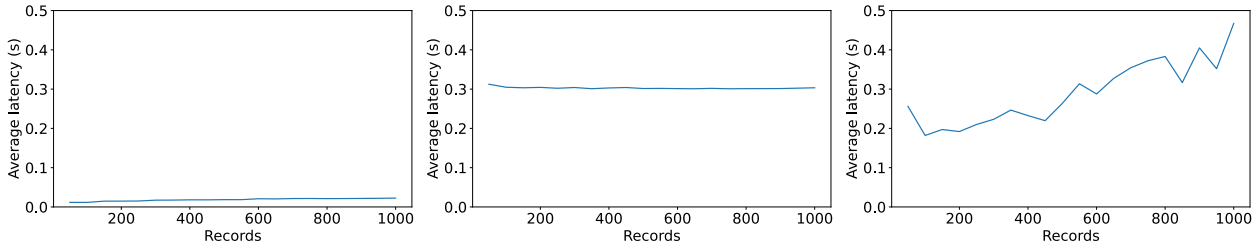
- Controlled parameters: the *number of records* requested/inserted/replicated. In this experiment, the number of independent tests at each step i, with a step increment size of 50 records, was 10. This means that each experiment performed at step i is repeated 10 times and then averaged. The active nodes are always two: Producer and Consumer.
- Dependent metrics: we measure: the *latency* to accomplish a request, the *CPU load*, and the *network traffic* of the machine running the Halo Node. Notice that the latency values also include the contribution due to the network transmission.

#### 1) RESULTS

Reading and writing were measured on the producer, while replication was evaluated on the consumer. The charts were produced with the following conditions:

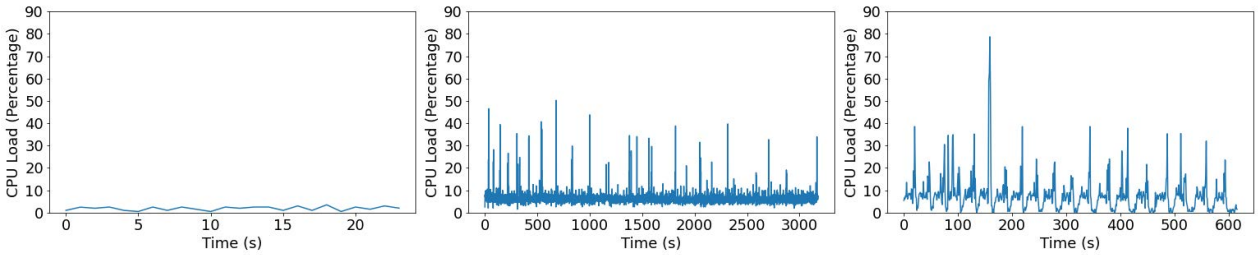
- Figure 10: latencies were evaluated through a client that makes the request and waits for responses. The requests are issued in parallel. The *read* and *relication* waits for data while the *write* waits for ACKs.
- Figure 11 and Figure 12: CPU and network activity values were collected directly on the machine involved. The producer for *read* and *write* while in the case of *replication*, the evaluation is done on the consumer.

Figure 10 shows the measured latency when varying the number of the records during read, write, and replicate operations. The results highlight that the three operations on OrbitDB are quite efficient, with the *replication* being more burdensome than the others. Also, in the case of the *replication*, the latency never reaches 500 ms, which is, however an acceptable value. This is an important detail because the *replication* operation tells us how the size of the database to be replicated contributes to a deterioration of the general performance. Although the values are normalized, as the



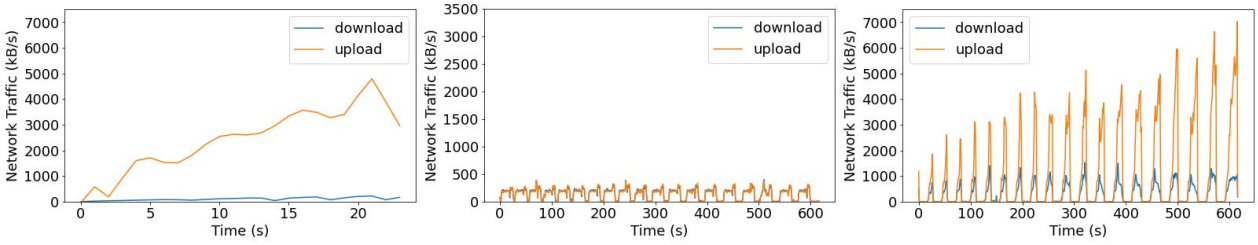

(a) Read (b) Write (c) Replicate

**FIGURE 10.** DFS requests latencies.




(a) Read (b) Write (c) Replicate

**FIGURE 11.** CPU load.




(a) Read (b) Write (c) Replicate

**FIGURE 12.** Network traffic.

information increases, it is presumably more expensive to keep the decentralized database integrity. For what concerns the read and write operations, we measure negligible latency values not exceeding about 300ms, and that shows no dependence on the number of records.

The CPU load is reported in Figure 11. Notice that the *read*, *write*, and *replication* graphs report in the x-axis the value of the time taken to perform the complete tests whose latency measures are shown in the previous figure. Since the latencies of the three operations are appreciably different, the total duration of every single test varies considerably, resulting in different time scales. In general, what can be seen right away is how efficient is the read. In fact, it always remains quite low all the time despite the number of read records increases. On the other hand, a higher load is visible for *replication* and *writing* showing easily distinguishable peaks in the correspondence of the execution of the query operations. Moreover, the former always seems more CPU demanding than the latter. The impression is that it occurs in chunks, requiring a higher computational effort that does not result in *writing*. This behaviour is closely linked to the implementation of OrbitDB in the case of *writing* and *replication*.

The results for network utilization can be seen in Figure 12. They confirm what was suggested by previous charts, with *read* and *write* being very efficient and with the *replication* more wasteful. Also, in this case, during the execution of the *replication* tests, the upload and download peaks are easily distinguishable. The *read* graph shows that although upload increases (as more records are returned), this has no real impact on total latency that is significant. Note that upload activity only has significance in *read* and *replication* because they contact the client to provide the data back. As for download activity, it is very low in the case of *read*, and this is justified by the fact that the operations do not involve write operations. In the case of *replication*, on the other hand, it is much higher, indicating an increase in activity during *replication*. The same thing occurs during *writing* but

with significantly lower network activity (notice the scale is different to appreciate the behaviour in this case). This is justified by the fact that in contrast to *replication*, when *writing*, the data make use of the API as protocol (instead of OrbitDB replication protocol), probably leading to lower network activity values. The remaining upload activity is related to the IPFS, since once the node receives the data, it must forward it to IPFS.

## VII. CONCLUSION

Nowadays, the heterogeneous resources, the massive amount of data coming from mobile devices and the people's privacy needs are suggesting the need for new methods of storing data for effective sharing.

We have developed a solution based on a combination of DFS and DLT capable of ensuring communication, sharing, and participation. Combining the two allowed us to store and share data between trusted individuals without relying on a centralized entity. Substantial help in going forward with the implementation came from OrbitDB; creating a layer above IPFS allowed the usage of IPFS as a database, making meaningful and complex queries. Starting from these technologies, we envisioned the InterPlanetary Health Layer through which research centres and institutions could safely retrieve personal medical data. The proposed implementation called Halo Network has been extensively tested by connecting it with a modified IoMT application called Balance. The results obtained confirmed the feasibility of the proposed solution showing good scalability and a modest impact on the performance of the application. Moreover, the ability of users to create their network makes it possible to ensure the availability of data that otherwise, in a decentralized context, would remain doubtful at any instant. Such a network guarantees the stakeholder always gains access to user data and avoids a single point of failure.

Assuming that the field will progress, we hope that this work would incentivize new research works and implementations that allow the free flow of information and adequate tracking of information, also accessible by external authorities such as entities and institutes. Our vision is that these technologies, along with self-sovereign identities, will push further the development of increasingly secure user-centric applications. Moreover, the proposed InterPlanetary Health Layer could be used for beneficial purposes as medical advancements and as a data layer able to feed the future artificial intelligence that will need data to be used continuously.

### A. LIMITATIONS AND CHALLENGES

Mobile devices currently represent a technological barrier to decentralizing the IoMT. Computational power and energy consumption issues are the most significant issues. We proposed a solution that uncouples smartphones from distributed technology to overcome the barrier and deliver a tangible implementation. At the same time, the decentralized web will hopefully move toward integrating these technologies. The importance of this step relies on the fact that mobile devices are the most diffused and could offer even more availability of separate nodes. In the long run, we think these devices will almost certainly constitute the decentralized web of the future.

The DLT growth rate represents another issue. Maintaining the ledger of a DLT is costly over time, so it is expensive keeping its integrity. The storage increases over time, and more investigations on reducing the amount of storage should be carried out for these specific solutions based on mobile devices.

### B. FUTURE WORKS

It would be of considerable interest to modify the architecture to enable the usage of IoMT data with machine learning applications. Because these data are excluded from today's datasets and those shared by research centers are not very comprehensive, these layers would enable beneficial purposes, such as advances in the medical field and feeding the future artificial intelligence.

Along with these developments, introducing a Decentralized Identity (DID) could provide users with an unambiguous way to be identified in such a network and keep track of similar networks in other areas. We imagine that in the future, introducing a Self-Sovereign identity could provide a unique reference technology to access different data layers from each other in a decentralized manner.

## REFERENCES

[1] M. Trojano, R. Bergamaschi, M. P. Amato, G. Comi, A. Ghezzi, V. Lepore, M. G. Marrosu, P. Mosconi, F. Patti, M. Ponzio, P. Zaratin, and M. A. Battaglia, "The Italian multiple sclerosis register," *Neurolog. Sci.*, vol. 40, no. 1, pp. 155–165, Jan. 2019.

[2] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, vol. 21260, pp. 1–9, Oct. 2008.

[3] V. Buterin and V. Griffith, "Casper the friendly finality gadget," 2017, *arXiv:1710.09437*.

[4] A. V. Sambra, E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga, and T. Berners-Lee, "Solid: A platform for decentralized social applications based on linked data," MIT CSAIL & Qatar Comput. Res. Inst., Ar-Rayyan, Qatar, Tech. Rep. MIT-QCRI-2016, 2016.

[5] G. Bigini, V. Freschi, and E. Lattanzi, "A review on blockchain for the Internet of Medical Things: Definitions, challenges, applications, and vision," *Future Internet*, vol. 12, no. 12, p. 208, Nov. 2020.

[6] G. Bigini, V. Freschi, A. Bogliolo, and E. Lattanzi, "Decentralising the Internet of Medical Things with distributed ledger technologies and off-chain storages: A proof of concept," in *Proc. Int. Conf. Smart Objects Technol. Social Good*. Cham, Switzerland: Springer, 2021, pp. 80–90.

[7] B. Goertzel and C. Pennachin, *Artificial General Intelligence*, vol. 2. New York, NY, USA: Springer, 2007.

[8] B. Meskó, G. Hetényi, and Z. Győrffy, "Will artificial intelligence solve the human resource crisis in healthcare?" *BMC Health Services Res.*, vol. 18, no. 1, pp. 1–4, Dec. 2018.

[9] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, and D. Enyear, "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Apr. 2018, pp. 1–15.

[10] N. Szabo, "Formalizing and securing relationships on public networks," *1st Monday*, vol. 2, no. 9, Sep. 1997.

[11] J. Benet, "IPFS—Content addressed, versioned, P2P file system," 2014, *arXiv:1407.3561*.

[12] M. Zichichi, S. Ferretti, G. D'Angelo, and V. Rodriguez-Doncel, "Personal data access control through distributed authorization," in *Proc. IEEE 19th Int. Symp. Netw. Comput. Appl. (NCA)*, Nov. 2020, pp. 1–4.

[13] G. Zyskind and O. Nathan, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Secur. Privacy Workshops*, May 2015, pp. 180–184.
[14] J. P. Cruz, Y. Kaji, and N. Yanai, "RBAC-SC: Role-based access control using smart contract," *IEEE Access*, vol. 6, pp. 12240–12251, 2018.
[15] E. Saweros and Y.-T. Song, "Connecting personal health records together with EHR using tangle," in *Proc. 20th IEEE/ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput. (SNPD)*, Jul. 2019, pp. 547–554.
[16] A. Donawa, I. Orukari, and C. E. Baker, "Scaling blockchains to support electronic health records for hospital systems," in *Proc. IEEE 10th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Oct. 2019, pp. 550–556.
[17] M. Lucking, R. Manke, M. Schinle, L. Kohout, S. Nickel, and W. Stork, "Decentralized patient-centric data management for sharing IoT data streams," in *Proc. Int. Conf. Omni-Layer Intell. Syst. (COINS)*, Aug. 2020, pp. 1–6.
[18] B. Cisneros, J. Ye, C. H. Park, and Y. Kim, "CoviReader: Using IOTA and QR code technology to control epidemic diseases across the US," in *Proc. IEEE 11th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2021, pp. 610–618.
[19] R. Adlam and B. Haskins, "A permissioned blockchain approach to the authorization process in electronic health records," in *Proc. Int. Multidisciplinary Inf. Technol. Eng. Conf. (IMITEC)*, Nov. 2019, pp. 1–8.
[20] A. Fernandes, V. Rocha, A. F. D. Conceicao, and F. Horita, "Scalable architecture for sharing EHR using the hyperledger blockchain," in *Proc. IEEE Int. Conf. Softw. Archit. Companion (ICSA-C)*, Mar. 2020, pp. 130–138.
[21] A. D. Dwivedi, G. Srivastava, S. Dhar, and R. Singh, "A decentralized privacy-preserving healthcare blockchain for IoT," *Sensors*, vol. 19, no. 2, p. 326, 2019.
[22] R. Arul, Y. D. Al-Otaibi, W. S. Alnumay, U. Tariq, U. Shoaib, and M. D. J. Piran, "Multi-modal secure healthcare data dissemination framework using blockchain in IoMT," *Pers. Ubiquitous Comput.*, vol. 2021, pp. 1–13, Feb. 2021.
[23] N. Garg, M. Wazid, A. K. Das, D. P. Singh, J. J. P. C. Rodrigues, and Y. Park, "BAKMP-IoMT: Design of blockchain enabled authenticated key management protocol for Internet of Medical Things deployment," *IEEE Access*, vol. 8, pp. 95956–95977, 2020.
[24] C. Stamatellis, P. Papadopoulos, N. Pitropakis, S. Katsikas, and W. J. Buchanan, "A privacy-preserving healthcare framework using hyperledger fabric," *Sensors*, vol. 20, no. 22, p. 6587, Nov. 2020.
[25] V. Mani, P. Manickam, Y. Alotaibi, S. Alghamdi, and O. I. Khalaf, "Hyperledger healthchain: Patient-centric IPFS-based storage of health records," *Electronics*, vol. 10, no. 23, p. 3003, Dec. 2021.
[26] M. M. Madine, A. A. Battah, I. Yaqoob, K. Salah, R. Jayaraman, Y. Al-Hammadi, S. Pesic, and S. Ellahham, "Blockchain for giving patients control over their medical records," *IEEE Access*, vol. 8, pp. 193102–193115, 2020.
[27] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain for secure EHRs sharing of mobile cloud based E-health systems," *IEEE Access*, vol. 7, pp. 66792–66806, 2019.
[28] J. Xu, K. Xue, S. Li, H. Tian, J. Hong, P. Hong, and N. Yu, "Healthchain: A blockchain-based privacy preserving scheme for large-scale health data," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8770–8781, Oct. 2019.
[29] I. A. Omar, R. Jayaraman, K. Salah, M. C. E. Simsekler, I. Yaqoob, and S. Ellahham, "Ensuring protocol compliance and data transparency in clinical trials using blockchain smart contracts," *BMC Med. Res. Methodol.*, vol. 20, no. 1, pp. 1–17, Dec. 2020.
[30] M. Kumar and S. Chand, "MedHypChain: A patient-centered interoperability hyperledger-based medical healthcare system: Regulation in COVID-19 pandemic," *J. Netw. Comput. Appl.*, vol. 179, Apr. 2021, Art. no. 102975.
[31] R. Kumar, N. Marchang, and R. Tripathi, "Distributed off-chain storage of patient diagnostic reports in healthcare system using IPFS and blockchain," in *Proc. Int. Conf. Commun. Syst. Netw. (COMSNETS)*, Jan. 2020, pp. 1–5.
[32] R. Kumar and R. Tripathi, "Towards design and implementation of security and privacy framework for Internet of Medical Things (IoMT) by leveraging blockchain and IPFS technology," *J. Supercomput.*, vol. 77, no. 8, pp. 7916–7955, 2021.
[33] B. S. Egala, A. K. Pradhan, V. Badarla, and S. P. Mohanty, "Fortified-chain: A blockchain-based framework for security and privacy-assured Internet of Medical Things with effective access control," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11717–11731, Jul. 2021.
[34] A. A. Abdellatif, L. Samara, A. Mohamed, A. Erbad, C. F. Chiasserini, M. Guizani, M. D. O'Connor, and J. Laughton, "MEdge-chain: Leveraging edge computing and blockchain for efficient medical data exchange," *IEEE Internet Things J.*, vol. 8, no. 21, pp. 15762–15775, Nov. 2021.
[35] S. Jiang, J. Cao, H. Wu, Y. Yang, M. Ma, and J. He, "BlocHIE: A blockchain-based platform for healthcare information exchange," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Jun. 2018, pp. 49–56.
[36] G. Srivastava, J. Crichigno, and S. Dhar, "A light and secure healthcare blockchain for IoT medical devices," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2019, pp. 1–5.
[37] M. Seliem and K. Elgazzar, "BIoMT: Blockchain for the Internet of Medical Things," in *Proc. IEEE Int. Black Sea Conf. Commun. Netw. (BlackSeaCom)*, Jun. 2019, pp. 1–4.
[38] M. Uddin, M. S. Memon, I. Memon, I. Ali, J. Memon, M. Abdelhaq, and R. Alsaqour, "Hyperledger fabric blockchain: Secure and efficient solution for electronic health records," *Comput., Mater. Continua*, vol. 68, no. 2, pp. 2377–2397, 2021.
[39] R. K. Marangappanavar and M. Kiran, "Inter-planetary file system enabled blockchain solution for securing healthcare records," in *Proc. 3rd ISEA Conf. Secur. Privacy (ISEA-ISAP)*, Feb. 2020, pp. 171–178.
[40] T. Tantidham and Y. N. Aung, "Emergency service for smart home system using Ethereum blockchain: System and architecture," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2019, pp. 888–893.
[41] H. Kordestani, K. Barkaoui, and W. Zahran, "HapiChain: A blockchain-based framework for patient-centric telemedicine," in *Proc. IEEE 8th Int. Conf. Serious Games Appl. Health (SeGAH)*, Aug. 2020, pp. 1–6.
[42] J. R. Nascimento, J. B. S. Nunes, E. L. Falcão, L. Sampaio, and A. Brito, "On the tracking of sensitive data and confidential executions," in *Proc. 14th ACM Int. Conf. Distrib. Event-based Syst.*, Jul. 2020, pp. 51–60.
[43] M. Shapiro, N. Preguiça, C. Baquero, and M. Zawirski, "Conflict-free replicated data types," in *Proc. Symp. Self-Stabilizing Syst.* Berlin, Germany: Springer, 2011, pp. 386–400.
[44] E. Lattanzi, V. Freschi, S. Delpriori, L. C. Klopfenstein, and A. Bogliolo, "Standing balance assessment by measurement of body center of gravity using smartphones," *IEEE Access*, vol. 8, pp. 96438–96448, 2020.

**GIOELE BIGINI** received the M.Sc. degree in computer science and engineering from the Politecnico di Milano, Milan, Italy, the M.Sc. degree in informatics engineering from the Universidad Politécnica de Madrid, Madrid, Spain, in 2019, and the Diploma degree in data science from the European Institute of Innovation and Technology (EIT Digital), Berlin, Germany, in 2019. He is currently pursuing the Ph.D. degree in research methods in science and technology with the University of Urbino. He is also part of the Alumni of the EIT Digital. In 2022, he was visiting the Ontology Engineering Group, Universidad Politécnica de Madrid, as a Visiting Researcher with Prof. V. Rodríguez Doncel. His research interests include big data, artificial intelligence, distributed ledger technologies, distributed systems, and the Internet of Things.

**EMANUELE LATTANZI** received the Laurea *(summa cum laude)* and Ph.D. degrees from the University of Urbino, Italy, in 2001 and 2005, respectively. Since 2001, he has been with the Information Science and Technology Institute, University of Urbino. In 2003, he was with the Department of Computer Science and Engineering, Pennsylvania State University, as a Visiting Scholar with Prof. V. Narayanan. From 2008 to 2020, he was an Assistant Professor of computer engineering at the Department of Pure and Applied Sciences (DiSPeA), University of Urbino, where he is currently an Associate Professor of computer engineering. His research interests include wireless sensor networks, wireless embedded systems, energy-aware routing algorithms, artificial intelligence, and distributed ledger technologies.