On the decentralization of Mobile Crowdsensing in Distributed Ledgers: an architectural vision

Lorenzo Gigli*, Federico Montori*, Mirko Zichichi§, Luca Bedogni†, Stefano Ferretti‡*, Marco Di Felice*
* *Department of Computer Science and Engineering, University of Bologna, Italy*
† *Faculty of Maths and Physics, University of Modena and Reggio Emilia, Italy*
‡ *Department of Pure and Applied Sciences, University of Urbino, Italy*
§ *IOTA Foundation, Berlin, Germany*
Correspondent author's Email: lorenzo.gigli@unibo.it

***Abstract*—Mobile Crowdsensing (MCS) is a paradigm where a crowdsourcer recruits a set of workers through a campaign to collect data using sensors in their mobile device. This process greatly reduces the costs of data collection processes; however, most of the historically proposed systems are centralized. Since this makes the MCS platform a single point of failure, there is an increasing interest in decentralized blockchain-based solutions; regardless, most of the current proposals have a vertical focus and do not account for the heterogeneity of MCS. We propose a decentralized high-level architecture for MCS, based on Distributed Ledger Technology (DLT), that is adaptable to most MCS deployments. We then implement our architecture using the IOTA protocols and evaluate its performance over a real deployment in terms of scalability, showing its advantages over classic blockchains for MCS data.**

***Index Terms*—Mobile Crowdsensing, DLT, blockchain, IoT**

I. Introduction

Mobile Crowdsensing (MCS) [1] is a data collection paradigm that leverages the power of the crowd to assess and describe phenomena of common interest in the context of the Internet of Things (IoT). In MCS, a Crowdsourcer, an entity interested in certain data, publishes a data collection campaign where end users, called Workers, can participate by collecting data through their mobile devices. In this scenario, Workers are rewarded by the Crowdsourcer via monetary compensation or through digital assets, e.g. in the form of a game. Regardless of how it is practised, MCS has revolutionized data collection processes, as there is no need to either deploy field sensors or hire a dedicated group of experts to conduct such sensing tasks. This greatly reduces the costs of data collection campaigns, especially when occasional.

Research on MCS has been significantly active over recent years, as such a paradigm brings in several challenges that need to be solved, such as privacy, data quality, coverage, and incentivization [2]. Moreover, traditional MCS deployments rely on a centralized architecture: data is collected onto a central MCS platform, where most of the computation occurs. While efficient in data processing, this centralization implies significant risks. Workers must trust a single entity, creating a single point of failure that, if compromised, can lead to system-wide vulnerabilities, data integrity issues, and potential misuse of sensitive information [3]. Additionally, centralized systems can face scalability challenges, as increasing data volumes and user participation may lead to processing bottlenecks and increased operational costs.

In recent years, Distributed Ledger Technology (DLT) have arisen as a potential solution for decentralized and trustless architectures. DLT use a Peer-to-Peer (P2P) architecture, in which each of the participating nodes retains a copy of all the transactions that occurred in the system. A transaction denotes a change in the status quo and is validated across all peers through a consensus system. A *blockchain* is an example of a DLT, the first of its kind introduced with Bitcoin, where transactions are stored in a chain of blocks. With the advent of Ethereum, a blockchain may be capable of running *smart contracts*, which can be seen as programs that are deployed on-chain and can be invoked by peers, generally for a fee, through the Ethereum Virtual Machine (EVM). The interaction with smart contracts is handled through decentralized applications (Dapps), the main enabler for decentralized architectures.

Adopting a DLT for an architecture historically considered centralized does not come for free. It brings about a number of challenges, as well as a dramatic change in the architectural setup. There have been proposals in literature for applying existing DLT in data collection scenarios, most of them using ancillary distributed services to store big data. However, very few of them are focused on MCS, and the vast majority only target a vertical aspect of a particular MCS deployment. They do not consider the inherent heterogeneity of MCS ecosystems, making such solutions hardly applicable to a wide range of use cases.

In this paper, we propose a high-level decentralized architecture for MCS that uses DLT. Our proposed architecture is generic enough to accommodate different MCS paradigms and does not lean toward any specific technology, letting the developers choose the right tools. At the same time, we define the architecture components by providing examples of real-world technologies to show that they can be implemented using existing tools. We also implement the architecture on a real ledger using IOTA, a DLT whose design is especially suitable for IoT scenarios. In particular, we make the following three contributions:

- We define the architectural components of a fully decentralized MCS system through DLT by outlining the essen-

tial operations that take place throughout a data collection campaign, abstracting from vertical deployments.
- We implement our proposed architecture by using IOTA and describe how the components of our architecture are to be placed in a system that runs over the IOTA Tangle.
- We evaluate IOTA's performance and scalability on a large-scale deployment by using a real IOTA node and some MCS devices that upload data concurrently.

The remainder of the paper is organized as follows: Section II outlines recent works, Section III details our architectural proposal, Section IV shows our implementation using IOTA, Section V discusses the evaluation of our proposal, and, finally, Section VI concludes the paper.

II. Related Works

In this section, we review the state of the art for the different areas pertaining to this work, specifically MCS in Section II-A and blockchain in Section II-B.

A. Mobile Crowdsensing

MCS has been actively studied in recent years due to its ability to provide sensory data without needing specific infrastructure. There have been many different architectures to realize such a vision, most of which employ a centralized approach [1].

MCS present several challenges, which we can summarize as *(i) Privacy*, as personal information of workers sharing their data must be protected; *(ii) Reward*, to provide adequate compensation for the workers' services; *(iii) Data quality*, since data obtained through MCS is usually noisy.

Regarding privacy, the main issue to consider is protecting users' traces and routines, as subsequent geolocalized measurements can easily uncover potential users' habits and points of interest [4]. To this end, there have been several proposals in literature to protect the workers' anonymity [5] [6]. More recently, proposals have also focused on the privacy preservation of users by considering novel protocols through which workers can communicate without disclosing too much information [7] by employing correlation metrics between the workers and the central server.

Considering the reward, again, the challenge may have multiple different solutions and possibilities. It is also important to note that the reward is tied to privacy: to reward a user, the platform must know who provided data so that it is possible to later exchange the user's work for some form of compensation [1]. Determining the right amount of reward to provide to a user depends on several factors that are outside the scope of this work, and we refer to the work presented in [8]. Closer to this work is instead the possibility to provide a reward to the users while protecting their personal information, which is the scope of [9], in which the authors propose a novel feedback based protocol that allows the users to select the desired degree of information disclosed.

B. IoT and MCS with blockchain

In recent years, many attempts have been made to bind the concepts of MCS and distributed ledgers [3]. Early works such as [10] introduce the usage of a blockchain in MCS scenarios; however, the blockchain itself is used as a black box for data exchange for non-repudiation and non-tampering, without full architectural integration. These papers focus mostly on deep mathematical problems not very much influenced by centralized or decentralized architectures. More recent works propose their own DLT to adapt to the MCS paradigm. For instance, the work in [11] proposes BlockSense, a blockchain for MCS that is built on top of a novel consensus algorithm called Proof-of-Data (PoD), which allows miners to check for data quality instead of performing useless puzzles. To achieve this, authors use zk-SNARKS and Homomorphic Encryption on top of each data point. Another work [12] proposes a similar model based on Delegated Proof of Reputation (DPoR) as the consensus mechanism and multiple contracts to constrain the behaviour of the actors. The aspect of fair rewarding is considered in ABCrowd [13], where authors bring the well-known reverse auction paradigm into a decentralized ecosystem as an incentive. They also design an MCS system to counter misbehaving workers over Ethereum [14] by letting actors interact with each other through smart contracts. This solution is, however, hardly sustainable nowadays because of the high transaction costs of Ethereum.

Most of the cited works are focused vertically on a single MCS aspect, and they often do not consider existing DLT. Rather, they assume a generic blockchain without a deep architectural discussion or an implementation that justifies its scalability in the real world. Pioneering work for IoT has been proposed in [15], where a global IoT market is envisioned, and owners of IoT devices are recruited through oracles on top of their reputation to provide sensory data. A similar concept can be applied to MCS; however, MCS differs from pure IoT in many aspects.

Within a broader scope, works exist centred on leveraging sensors sensed by sensors and stored in decentralized systems, addressing overarching framework issues to establish data marketplaces [16]. These endeavours emphasize the creation of data marketplaces without delving into specific crowdsensing aspects [17].

III. System Architecture

This paper aims to present an architectural solution for deploying MCS systems in a fully decentralized scenario powered by distributed ledgers. In this section, we present a high-level view of such an architecture, outlining the main actors as well as the necessary components (see Figure 1). The goal here is to first abstract from the technologies and focus on how the different parties are supposed to interact.

A. Actors

First, it is necessary to specify the actors involved in MCS scenarios. Specifically, MCS systems are powered by an exchange of data and money between two parties: the

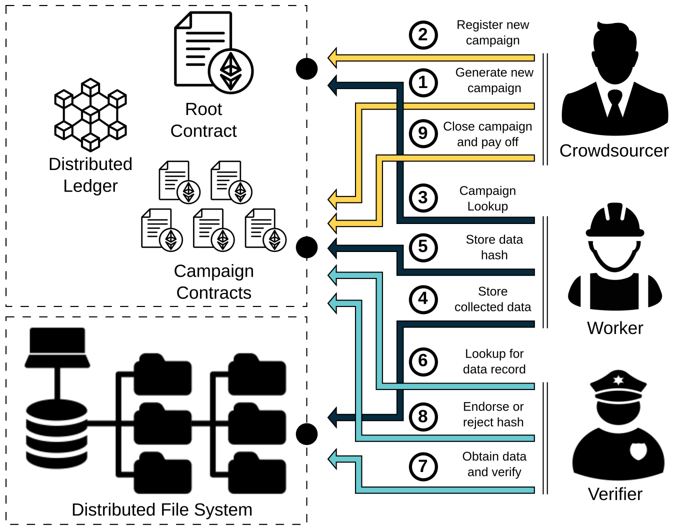

Fig. 1. Our proposed decentralized MCS architecture.

Crowdsourcers, who need data, and a crowd of **Workers**, who possess the capabilities to collect such data. In MCS, a Crowdsourcer publishes a data collection *campaign*, which is advertised through an MCS platform so that any Worker can retrieve information on how to participate. A campaign generally specifies a handful of metadata, such as what kind of data is needed (e.g. accelerometer, gyroscope, pictures, etc.), an area within which the data is actually of interest, a validity over time, and a reward for a set of data points. By publishing the campaign, the Crowdsourcer also stakes a certain amount of money – the campaign budget – to be paid to Workers. Workers then may sign up for the campaign and collect sensor data through their devices (a smartphone, a wearable, or any other portable sensor). The Worker then upload data to the MCS platform in return for a reward.

In many cases, Workers are considered untrustworthy: they may upload fake data to obtain more money or simply be ingenuous users who may make mistakes in the data collection process. For this reason, many MCS scenarios adopt data quality countermeasures [18]. In our case, we rely on a third type of actor: the **Verifier**. Verifiers are other MCS users committed to checking data uploads for their integrity and validity. How this process is carried out is out of the scope of the paper, as it may change depending on the nature of the campaign (a decentralized solution for this process is briefly explained in section III-D). In this paper, we assume they have a way to endorse or reject a data point, so the rewarding mechanism occurs only after the data has been endorsed a certain number of times.

B. Components

As specified earlier, this section aims to outline a high-level and technology-agnostic architecture. That being said, the feasibility of such a system is unavoidably tied to a certain number of components that are necessary for its deployment. First of all, the main ingredient for a fully decentralized system is a **Distributed Ledger Technology (DLT)**, which is natively tamper-proof and assures non-repudiation. We intentionally do not constrain this definition to a blockchain, as, despite being the most common example of DLT, it is not the only one (see the IOTA Tangle, Section IV-A). **Smart contracts** are a necessary component of our architecture; thus, we assume that the chosen DLT is also enabled for running smart contracts. Finally, our architecture makes use of a **Distributed File System** (DFS) for storing MCS data. In fact, in most cases, MCS data may be too large or simply too expensive for being stored directly on a smart contract. A typical approach in literature consists of making use of a DFS, such as InterPlanetary File System (IPFS) [19] or Swarm [20], for the actual storage and then uploading on the smart contract only the reference/hash of the data [21]. This is much more sustainable as, in a DFS, not all peer nodes need to keep the full database in memory as in DLT, only a part of it.

C. Interactions

Once specified the actors that take part in the system and the necessary components for its deployment, we outline the macro actions that each actor must carry out within a full data collection process. With respect to the steps numbered in Figure 1, we describe them below in detail.

1) Generate new campaign: The Crowdsourcer must create a new smart contract uniquely representative of a new campaign. In doing so, the smart contract needs a definite set of functions that must adhere to some kind of standards so that Workers can interact with different campaigns using the same schema. Much like standards established for fungible and non-fungible tokens (ERC-20 and ERC-721), we could imagine a similar standard for MCS campaigns that allow Workers to upload data and obtain the metadata of the campaign and allows the Crowdsourcer to stake a certain amount of tokens which will be used as a reward once the data is verified. This operation may cause the Crowdsourcer to additionally spend gas fees.

2) Register new campaign: When the campaign is deployed, it needs to be reachable through a “campaign hub”, which is defined as a *root contract*, i.e., a smart contract that works as an indexer and maintains a list of active campaigns. In a global IoT market, we could envision multiple root contracts to be deployed on-chain, each of them requiring a fee for indexing a campaign.

3) Campaign lookup: Workers can query a root contract that they know to look up the address of active campaigns. Once the address is known, they can also query each of the campaign contracts they are interested in for their metadata and discover if they are eligible to participate (e.g. if they possess the right sensors, if they are in the zone of interest, etc.).

4) Store collected data: Once a Worker decides to participate in a campaign, it starts collecting data on their personal device. When a certain number of data points is collected – this is commonly a parameter of the campaign – the Worker uploads the data on a DFS and gets back its hash and index as proof of the upload.

5) Store data hash: The Worker finally uploads the hash of its collected data onto the campaign contract. In the case of EVM-like blockchains, this is not a view/pure transaction; thus, it may cause the worker to invest a small gas fee to ensure the transaction. This depends on the DLT technology; however, this investment may discourage Workers from uploading fake/inaccurate data. It is also worth mentioning that because the hashing function is known, a Worker may even calculate the hash locally and upload it before the actual data upload is finished on the DFS. This is because uploading data on DFS is generally time-consuming (cfr. Section V), while it may also be convenient for a Worker to secure a reward beforehand.

6) Lookup for data record: Here is where the Verifiers come into play. A Verifier opportunistically discovers ongoing campaigns, much like Workers do in step 3 (omitted here for Verfiers for conciseness) and, subsequently, for unverified hashes. A Verifier must be eligible to verify the data of such a campaign in order to proceed. Verifiers eligibility can be assessed in various ways, such as expertise or location. This aspect is out of the scope of the paper and largely discussed in literature [17], [18].

7) Obtain data and verify: Once the Verifier obtains the hash to verify, it queries the DFS for the actual piece of data.

8) Endorse or reject hash: The Verifier then checks the uploaded data and posts a verdict (endorse or reject) on the campaign contract. This may also involve a fee to be paid in EVM-like blockchains, which makes sense, considering that the Verifier will eventually be paid for this operation. Putting some rewards at stake should encourage the Verifier to perform this task diligently.

9) Close the campaign and pay off: When the data collection is complete, the Crowdsourcer closes the campaign. This may occur because the campaign has expired or because the number of data points collected has reached a threshold. Regardless, once this condition is met, the Crowdsourcer must pay off all the participants using the tokens staked at the beginning of the process. A simple yet effective way to do it would be to pay a fixed amount to Workers for each endorsed hash and a second fixed amount to Verifiers for each vote confirmed by the community. For instance, if a Verifier rejected a hash that was instead endorsed by the majority of other Verifiers, then such Verifier will not be rewarded and will lose the money put at stake, if any. Several other reward systems in the literature may differently reward Workers with budget constraints, depending on how valuable is indeed the collected data. Nonetheless, this is out of the scope of the paper, and we redirect the interested reader to [22].

D. Extensions

The architecture shown is intentionally generic to encompass the majority of decentralized MCS deployments. This admittedly leaves apart some important aspects that may be felt as limitations of the proposed architecture; however, they can all be addressed through well-known solutions.

First, the architecture, as is, suits a *pull-based* MCS setup, where Workers can freely join any campaign and contribute as much and as often as they want. Indeed, many MCS deployments in literature imply a *push-based* paradigm, where the MCS platform actively recruits Workers by deciding who is committed to performing which task, based on costs, capabilities, context, etc. This requires our architecture to add a recruitment phase by keeping track of all enrolled Workers and implementing a selection function within the contract. The same recruitment scheme may be used to assign Verifiers to hashes to validate because letting them choose what to validate may easily lead to collusion attacks. These operations can be decentralized using a Decentralized Oracle solution. For instance, the Chainlink protocol [23] enables the creation of a decentralized oracle network where each node, i.e., a single Validator and/or Worker, can perform the data uploads or the validation task if they are appointed to. If the majority of the nodes in the Oracle network vote in favor, the data can be considered valid. The interactions with the contracts, however, remain almost the same. This aspect may be traced back to other architectures, such as our recent proposal [15].

Another issue is that once data is on a DFS, and the hash of it is public, every peer can read such data. Additionally, ensuring data ownership to a buyer may seem challenging once the transaction is performed. However, several works deal with data encryption on DFSs as well as using NFTs to grant data ownership [24]. One last aspect to take into account is privacy, which is a crucial matter in MCS, as data points are almost always geolocalized. Actually, some of the proposed privacy schemes in literature do not necessarily imply a trusted party, and they can be almost equally applied to our decentralized architecture [7].

IV. System Implementation

In this section, we concretely envision how our proposed decentralized MCS architecture would be implemented using one of the latest technologies introduced for decentralized IoT environments: IOTA.

A. IOTA: Background

IOTA [25] is a DLT that runs on the Tangle, a data structure replicated across a network of nodes. It forms a directed acyclic graph of blocks (a block-DAG), where each newer block is attached to multiple older ones. In IOTA's Tangle, every participant who wants to issue a block containing a transaction is required to validate two previous blocks. This validation process replaces the concept of transaction fees, as each participant contributes to the security and functionality of the network by validating other transactions. When a user wants to issue a new block, it must perform a small Proof-of-Work (PoW) computation. The PoW is a cryptographic puzzle that requires a certain amount of computational effort to solve. However, the difficulty level is intentionally kept relatively low compared to traditional blockchain networks like

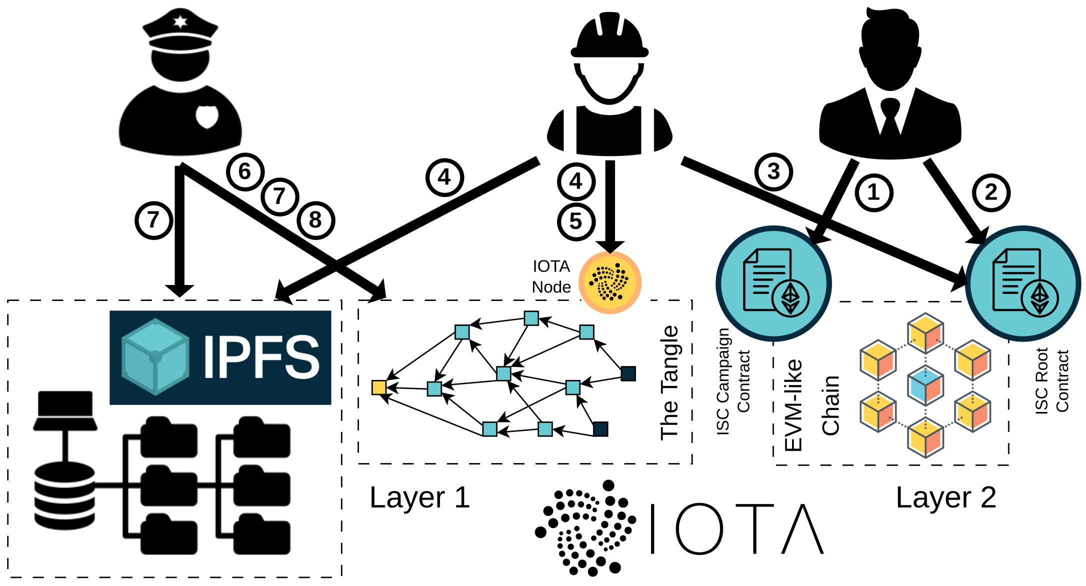

Fig. 2. Our proposed architecture implemented using IOTA and IPFS.

Bitcoin or Ethereum. This design choice makes it feasible for a wide range of devices, including IoT devices with limited computational power, to participate in the network and issue transactions.

Blocks usually contain core payloads that are processed by all nodes. The payload concept generally offers a flexible way to combine and encapsulate information in a block on the IOTA Tangle. The most flexible way to extend an existing object is by adding arbitrary data. The Tagged Data Payload provides a way to do just that. This type of payload can be used by automated devices or users to share messages in the Tangle in a feeless way.

Additionally, the IOTA environment provides the capability to execute smart contracts thanks to the IOTA Smart Contracts (ISC) protocol. Considering the IOTA Tangle as a Layer 1 (L1) network, the ISC protocol builds on a Layer 2 (L2) network of nodes executing an EVM-compatible blockchain. The ISC protocol is an L2 framework that brings quasi-Turing complete smart contracts to the IOTA technology stack. ISC is a versatile, multi-chain environment that can run numerous parallel L2 Blockchains on top of the L1 IOTA ledger. Each chain possesses its own independent ledger state, using an account-based model anchored to a specific IOTA unspent transaction Output (UTXO) ledger account on L1. Every chain can host multiple smart contracts that are fully composable via synchronous calls within the chain. At the same time, cross-chain transactions are enabled through an anchoring mechanism on L1, promoting asynchronous composability. This design lets smart contracts interact trustlessly across different IOTA Smart Contract chains.

B. Our proposal with IOTA

In Figure 2, we show the schematic deployment of our architectural proposal in an IOTA environment. The basic components are the IOTA Tangle, which works as an L1 ledger, and an L2 ISC EVM-like blockchain, in which our two smart contracts – the campaign contract and the root contract – are deployed. The DFS used here is IPFS, one of the most well-known DFS. IPFS uses a content-addressing manner to identify pieces of data, which means that the hash of the data points collected by a Worker is the only information needed to retrieve such data points. The Crowdsourcer replicates exactly steps 1 and 2 of Section III-C, only interacting with the L2 chain. The Croudsourcer would only stake the tokens that serve as payments for Workers and Verifiers and, if present, a small fee for registering the campaign in the root contract. Step 3, taken by the Worker, is straightforward, while a separate discussion must be made for steps 4 and 5. Differently from a full EVM-compatible deployment, here, the IOTA Tangle serves as the storage where the data is collected. Now, depending on the nature of the data for the campaign, we can take two separate paths. If the data is relatively lightweight (e.g. a JSON object containing sensor readings and a handful of metadata), then we can directly encapsulate such data within a single IOTA block and save it on the L1 ledger. If, instead, the data is larger than 32KB (multimedia, pictures, videos, etc.), then we adopt the classic procedure of saving it onto IPFS and writing only its hash on the IOTA block, encapsulated in a JSON object. In both cases, data integrity is maintained by the IPFS URI or by the block ID (that corresponds to the hash digest of the block payload, i.e., the data or the IPFS URI). In any case, the Worker here submits to the campaign contract on L2 the block ID that points to the block in the L1 ledger. The Verifier then takes the same steps as in Section III-C, checking first the block on L1 and, if needed, the data on IPFS. The block ID on the campaign contract can be marked as endorsed or rejected. After a number of votes, the approved blocks are presented to the Crowdsourcer, who is then committed to closing the campaign and releasing the tokens. The Crowdsourcer can then use the data within (or pointed by) the blocks in L1.

V. Evaluation

In this section, our primary objective was to assess the viability of our proposed architecture, leveraging IOTA and, in some instances, IPFS for data storage. We aimed to determine whether the performance metrics met the demands typical of a crowdsensing environment. The results presented will shed light on the practicality of our approach and its potential fit in real-world scenarios.

A. *Experimental Setup*

This section provides a detailed overview of the infrastructure and procedures we adopted to assess our proposed architecture's performance and scalability.

For the experiments, we set up an IOTA Hornet full node v2.0.0-rc.6[1] configured with 10 workers and deployed on a cloud server. This server was equipped with an Intel Xeon Processor E7330 @ 2.40GHz, with 64GB RAM and an SSD. This configuration ensured the node was adequately robust to manage the incoming data and execute the necessary PoW to insert new transactions on the IOTA Tangle. We developed a multi-threaded application to simulate the behavior of multiple devices transmitting crowdsensing data. This application was executed on a machine powered by an Intel Xeon Gold 6238R Processor @ 2.20 GHz, with 256GB RAM and an SSD. Given that this machine has 56 logical CPUs, it was suitable for simulating up to 50 devices concurrently, offering a realistic representation of a dense MCS environment.

Our experimental methodology was designed to evaluate two primary strategies for data storage. The first involved directly storing the entire payload on IOTA. In our case, we simulated a campaign where workers were asked to measure the temperature and humidity of the environment and upload a JSON object with such information. The second strategy involved storing the payload on IPFS and then recording only the CID on IOTA. In our case, we simulated a campaign where the workers were asked to take a picture. Each of these strategies was further differentiated based on the location of the execution of the PoW: either locally by the device or remotely by the IOTA node. The experiments were designed to introduce the workload incrementally. We began with a single device dispatching a new measurement every 10 seconds. Gradually, we scaled the setup, introducing 5 devices, then 10, followed by 25, and finally reaching our peak load with 50 devices. To ensure the reliability of our findings, we repeated each phase five times and then averaged the results.

In these experiments, we primarily focused on the time metric, measuring the duration from the initiation of the data send request to its successful storage on the system. In the following section, we will present the results derived from these experiments, providing insights into the performance of our proposed architecture.

[1]https://github.com/iotaledger/hornet

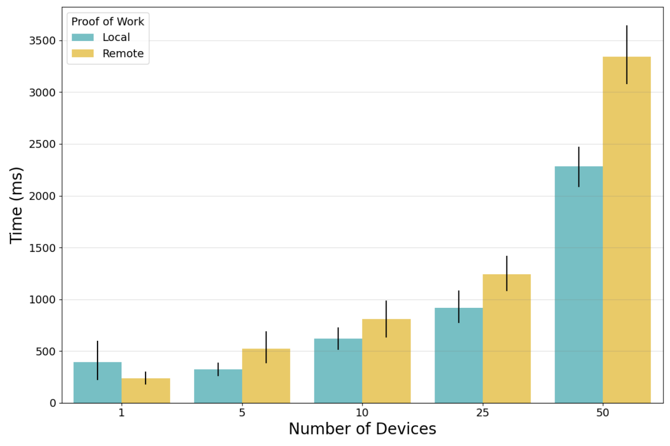

Fig. 3. Time analysis for data transmission to IOTA with local vs. remote PoW by device count.

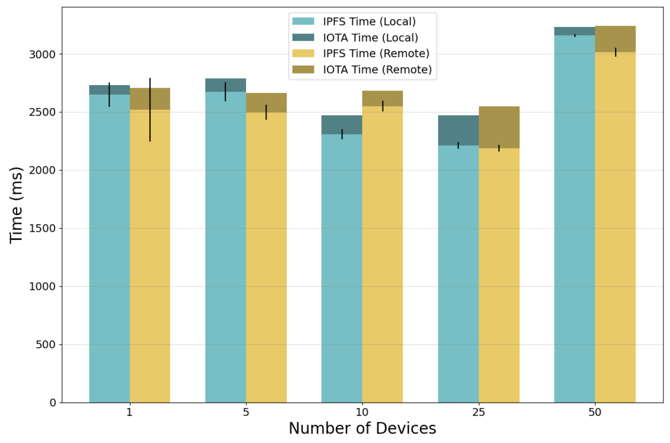

Fig. 4. Time analysis for payload storage on IPFS and CID transmission to IOTA with local vs. remote PoW by device count.

B. *Results*

Figure 3 shows the time performance of our system using the first strategy, where the full payload of the measurement performed by each Worker is stored on the Tangle. The bar chart shows two different cases: blue bars denote a PoW executed on the Worker's device, while yellow bars denote a PoW delegated to the IOTA full node. We can see how time increases as the number of blocks to be added to the Tangle increases as well. This operation is executed by the IOTA node almost sequentially – the node has been configured with 10 worker threads. We also note how performing local PoW saves time, as the PoW tasks are not centralized on the IOTA node, and we can assume that the time taken for the actual PoW is close to a constant. We also observe that the time taken for local PoW is greater than for remote PoW for only one device. This is also expected, as the computational power of one Worker is less than that of the IOTA node. However, the trend rapidly inverts as the number of devices increases, and the IOTA node needs to perform multiple tasks.

Figure 4 shows the plot when adopting the second strategy, which involves storing the full payload on IPFS and only the CID in IOTA. This plot reveals a very different trend, where the number of devices minimally influences the overall time taken. This is primarily because the upload on IPFS consumes a significant portion of the operation time. The difference between the time taken for calculating the PoW and the upload on IOTA versus the upload on IPFS is evident in the same figure. It is noticeable how the second is an order of magnitude greater than the first, even if it seems not affected by the number of devices, at least not at this scale.

For clarity in visualization, we plotted the confidence interval only for IPFS. The confidence interval for IOTA was negligible, and plotting both together resulted in a confusing representation on the chart. These experiments highlight the scalability of IOTA for our use case. For instance, even when 50 devices upload simultaneously using the same IOTA node, the process completes in less than 5 seconds. However, it's important to highlight that in real-world scenarios, the likelihood of 50 or more of them synchronously uploading data at the same time is low.

Moreover, these findings provide insights into sizing an IOTA node's deployment to handle specific loads. For instance, a crowdsourcer aiming to support its campaign can decide to deploy dedicated IOTA nodes, ensuring a controlled environment tailored to their specific needs. On the other hand, the crowdsourcer can also choose to leverage already established public nodes within the IOTA network. While this might introduce variability in response times and load, it can be a cost-effective strategy, reducing the need for initial infrastructure setup and maintenance.

VI. Conclusion

In this paper, we proposed an architectural solution for MCS in a decentralized fashion to overcome the problem of centralization. Our solution is based on DLT and DFS, and it is generic enough to encompass the vast majority of existing MCS deployments. We then implemented a system that follows the guidelines of our architecture using IOTA as a DLT and IPFS as a DFS in order to prove its superiority for use cases that are common in MCS. In the future, we aim to design a standard for smart contracts to comprehensively describe heterogeneous MCS campaigns.

References

[1] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities," *IEEE communications surveys & tutorials*, vol. 21, no. 3, pp. 2419–2465, 2019.

[2] F. Montori, P. P. Jayaraman, A. Yavari, A. Hassani, and D. Georgakopoulos, "The curse of sensing: Survey of techniques and challenges to cope with sparse and dense data in mobile crowd sensing for internet of things," *Pervasive and Mobile Computing*, vol. 49, pp. 111–125, 2018.

[3] Z. Chen, C. Fiandrino, and B. Kantarci, "On blockchain integration into mobile crowdsensing via smart embedded devices: A comprehensive survey," *Journal of Systems Architecture*, vol. 115, p. 102011, 2021.

[4] L. Bedogni and M. Levorato, "Rising user privacy against predictive context awareness through adversarial information injection," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2018.

[5] Y. Cheng, J. Ma, and Z. Liu, "A lightweight privacy-preserving participant selection scheme for mobile crowdsensing," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1509–1514, 2022.

[6] J. W. Kim, K. Edemacu, and B. Jang, "Privacy-preserving mechanisms for location privacy in mobile crowdsensing: A survey," *Journal of Network and Computer Applications*, vol. 200, p. 103315, 2022.

[7] F. Montori and L. Bedogni, "Privacy preservation for spatio-temporal data in mobile crowdsensing scenarios," *Pervasive and Mobile Computing*, vol. 90, p. 101755, 2023.

[8] X. Xu, J. Cheng, J. Liu, Y. Yuan, H. Li, and V. S. Sheng, "A Survey of Blockchain-Based Crowd Sensing Incentive Mechanism," in *Advances in Artificial Intelligence and Security* (X. Sun, X. Zhang, Z. Xia, and E. Bertino, eds.), (Cham), pp. 245–259, Springer International Publishing, 2022.

[9] L. Bedogni and F. Montori, "Joint privacy and data quality aware reward in opportunistic Mobile Crowdsensing systems," *Journal of Network and Computer Applications*, p. 103634, 2023.

[10] S. Zou, J. Xi, H. Wang, and G. Xu, "Crowdblps: A blockchain-based location-privacy-preserving mobile crowdsensing system," *IEEE transactions on industrial informatics*, vol. 16, no. 6, pp. 4206–4218, 2019.

[11] J. Huang, L. Kong, L. Cheng, H.-N. Dai, M. Qiu, G. Chen, X. Liu, and G. Huang, "Blocksense: Towards trustworthy mobile crowdsensing via proof-of-data blockchain," *IEEE Transactions on Mobile Computing*, 2022.

[12] J. An, J. Cheng, X. Gui, W. Zhang, D. Liang, R. Gui, L. Jiang, and D. Liao, "A lightweight blockchain-based model for data quality assessment in crowdsensing," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 1, pp. 84–97, 2020.

[13] M. Kadadha, R. Mizouni, S. Singh, H. Otrok, and A. Ouali, "Abcrowd an auction mechanism on blockchain for spatial crowdsourcing," *IEEE Access*, vol. 8, pp. 12745–12757, 2020.

[14] M. Kadadha, H. Otrok, R. Mizouni, S. Singh, and A. Ouali, "Sensechain: A blockchain-based crowdsensing framework for multiple requesters and multiple workers," *Future Generation Computer Systems*, vol. 105, pp. 650–664, 2020.

[15] L. Gigli, I. Zyrianoff, F. Montori, C. Aguzzi, L. Roffia, and M. Di Felice, "A decentralized oracle architecture for a blockchain-based iot global market," *IEEE Communications Magazine*, vol. 61, no. 8, pp. 86–92, 2023.

[16] M. Zichichi, S. Ferretti, and G. D'Angelo, "A framework based on distributed ledger technologies for data management and services in intelligent transportation systems," *IEEE Access*, 2020.

[17] M. Bonini, M. Zichichi, G. D'Angelo, and S. Ferretti, "Proof of location through a blockchain agnostic smart contract language," in *Proc. of the 43rd IEEE International Conference on Distributed Computing Systems (ICDCS 2023)*, IEEE, July 2023.

[18] F. Restuccia, N. Ghosh, S. Bhattacharjee, S. K. Das, and T. Melodia, "Quality of information in mobile crowdsensing: Survey and research challenges," *ACM Transactions on Sensor Networks (TOSN)*, vol. 13, no. 4, pp. 1–43, 2017.

[19] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[20] S. team, "Swarm: storage and communication infrastructure for a self-sovereign digital society." https://www.ethswarm.org/swarm-whitepaper.pdf, 2021. Version 1.0 of June 13, 2021.

[21] H. Huang, J. Lin, B. Zheng, Z. Zheng, and J. Bian, "When blockchain meets distributed file systems: An overview, challenges, and open issues," *IEEE Access*, vol. 8, pp. 50574–50586, 2020.

[22] X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, K. Xing, and X. Mao, "Incentives for mobile crowd sensing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 54–67, 2015.

[23] L. Breidenbach, C. Cachin, B. Chan, A. Coventry, S. Ellis, A. Juels, F. Koushanfar, A. Miller, B. Magauran, D. Moroz, *et al.*, "Chainlink 2.0: Next steps in the evolution of decentralized oracle networks," *Chainlink Labs*, vol. 1, pp. 1–136, 2021.

[24] M. Di Francesco, L. Marchesi, and R. Porcu, "Kryptosafe: managing and trading data sets using blockchain and ipfs," in *2023 IEEE/ACM 6th International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, pp. 5–8, IEEE, 2023.

[25] S. Müller, A. Penzkofer, N. Polyanskii, J. Theis, W. Sanders, and H. Moog, "Tangle 2.0 leaderless nakamoto consensus on the heaviest dag," *IEEE Access*, vol. 10, pp. 105807–105842, 2022.