DECENTRALIZED ARCHITECTURES IN IOT DATA SHARING USING BLOCKCHAIN TECHNOLOGY

Cristinel **COSTEA**
Technical University of Cluj-Napoca, North University Centre of Baia Mare
cristinel.costea@ieec.utcluj.ro

Keywords: Content addressable storage, Blockchains, Distributed systems

Abstract*: The development of IoT applications generally relies on centralized architectures, whether they use solutions based on cloud or on premises environments, usually operated by a single entity. However, recent technologies and the increasing number of interconnected intelligent devices are driving new approaches, the essential requirements being scalability and fault-tolerance. This work studies an alternative concept based on a decentralised architecture useful in several scenarios and focuses on available technologies for potential practical implementations.*

1. INTRODUCTION

If IoT devices record data locally for later access by other processes, scalability and access time problems arise due to the volume and rate of data generation. A particular solution is provided by time-series databases, but cloud storage is the most common storage for IoT data [1]. This offers the advantage of easy access to data from anywhere and anytime, as well as the possibility of mass storage. By example, Hadoop is an open-source framework for storing and processing data in large sets, distributed on several servers. It can manage large volumes of structured and unstructured data, being a suitable solution for data generated by IoT equipment.

For latency-sensitive and real-time applications, new paradigms have been developed. In Edge computing the data is stored and processed at the "edge" of the network, close to where it is generated. This reduces latency and allows for a faster response to data. Fog computing storage is a mix between edge computing and cloud storage. In a fog computing

model, data is primarily processed at the "edge" of the network but can also be sent to the cloud for storage and further processing [2]. Storage on gateways: gateways are devices that connect individual IoT devices to the Internet. They can store data from multiple devices, allowing pre-processing and data management. Decentralized storage: approach where data generated by IoT devices is stored in a distributed network of nodes, rather than concentrated in one centralized location. This model provides scalability, resilience and fault tolerance, and with the help of advanced blockchain technology, users can improve data privacy and maintain sovereignty over their data.

2. BACKGROUND

In modern distributed systems the concepts of consistency, availability and partition tolerance play a crucial role in the design and implementation of scalable and fault-tolerant systems. These concepts are essential to understanding and managing decentralized storage, where data is distributed over a global network of nodes.

Consistency is ensured by implementing protocols and algorithms that coordinate and synchronize actions between network nodes. Among the most important consensus protocols studied are Paxos, Raft, Byzantine Fault Tolerance (BFT), Practical Byzantine Fault Tolerance (PBFT), Zab [3]. With the development of blockchain systems, each system perfected a specific protocol, the most famous being Proof of Work (PoW) for Bitcoin or Proof of Stake (PoS) for Ethereum (and not only).

But each distributed system can provide just two of three fundamental properties, not all three at the same time. Specifically, the CAP theorem [4], formulated by Eric Brewer in 2000, states that in a distributed system, consistency (C), availability (A), and partition tolerance (P) cannot be achieved simultaneously. Another theorem developed by Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson in 1985(FLP theorem) [5], states that in an asynchronous distributed system, where nodes can fail by stopping processing, there is no a deterministic distributed algorithm to make a safe decision when a decision node may fail.

2.1. Distributed ledgers

Distributed Ledgers are a form of data recording that does not require a central authority but uses a distributed network of nodes to validate and store information. There are various types of distributed ledgers, but most approaches refer to blockchains. Two other important approaches are DAG (Directed Acyclic Graph) and Hashgraph [6], each with distinct features and applications. Blockchain is basically a kind of linked list where transaction information is stored in sequential blocks. In blockchain, data is immutable,

meaning that once a transaction is recorded in a block and added to the blockchain, it cannot be changed or deleted.

Some outstanding features of these technologies relate to digital signatures, distributed consensus and smart contracts. Distributed ledgers are implemented on the Internet, so participants are assumed not to trust each other by default, and this is where hashing technologies and public key cryptography come into play.

2.2 Smart contracts

A smart contract is a piece of code that runs on the blockchain and is recorded in a block of the chain. The largest number of smart contracts are registered in the Ethereum network (in 2022 there were about 44 million[1], although only 15 million were original and functional); many other networks have developed similar technologies in various programming languages such as Solana, Cardano, Polkadot, Algorand, Hyperledger, Stellar, Arbitrum, Optimism, Cosmos.

In Ethereum a smart contract has its own persistent data storage that can be modified by specific transactions (calling methods from the smart contract). Anyone with an Ethereum account can initiate a transaction that calls a method on the contract and can change its state. In fact, when a user changes the value of a (state) variable, that transaction returns a new state of the contract, which will be recorded with the transaction that called the contract.

Information about contract states is stored in a trie data structure: Merkle Patricia Tree (MPT) [7]. Like in a radix tree in MPT the key is split into sequential characters; in the lookup procedure each character of the encoded key is used to match the encoded path. Updating a value only involves recalculating the affected paths in the tree, not the entire tree. A new transaction that will modify the same variable will not require a node to go through the entire transaction history in the blockchain.

2.3 Decentralized storage

In the decentralized storage method, the data is distributed over several nodes, often in a peer-to-peer (P2P) network, avoiding concentration in a single location (single point of failure) and avoiding the control of a single centralized entity.

The outstanding example is the Inter-Planetary File System (IPFS) which defines both a protocol and a P2P file system to store and share content on the Internet. IPFS use content addressing: each piece of content is addressed by a unique cryptographic hash named Content Identifiers (CIDs) and generated based on the content.

A distinctive attribute of P2P (Peer to Peer) systems, which include both blockchain and IPFS, is the use of the Distributed Hash Table (DHT) [8] protocol. It offers a scalable way of organizing without the need for a centralized server. The nodes are arranged in a ring

configuration, they can enter and exit the system without significantly disrupting the operation.

When a node joins the IPFS network, it generates a pair of public and private keys, and the public key is used to identify the node in the network. The node must connect to another node in the DHT, usually a bootstrap node that provides information about other nodes in the network. It is an entry point to the network for new nodes or those that want to reconnect. The address of a bootstrap node is often recorded in the source code because these nodes are constantly available and accessible to ensure network stability. Following the connection request the new node will be able to build a routing table and which nodes it should connect to directly. Based on a finger table neighboring nodes will update a replica of records to ensure the appropriate level of redundancy.

2.4 Decentralized applications

A web application can interact directly with Ethereum via a local Ethereum node (with JSON-RPC) or via a Web3 service provider (such as Infura, Alchemy, Chainstack, QuickNode or BlockDaemon). The most popular JavaScript library used in these operations is Web3.js. The web application creates a Web3.js instance and connects to Metamask to communicate with the Ethereum network. Metamask is a browser extension that allows users to interact with decentralized applications (DApps), being a digital wallet required to access the Ethereum network. Under these conditions, a web application can read and write data to Ethereum smart contracts, send transactions and receive events from the blockchain.

3. EXPERIMENT AND RESULTS

The system architecture considered for the simulation includes the following possible components: IoT smart sensors provide messages in JSON format to devices with raw data pre-processing capability, for example gateway devices. They were simulated through a web application that logs the data to a decentralized storage system based on the IPFS protocol. A unique hash is generated for each file stored. This obtained CID is sent to a smart contract to be registered in the Ethereum blockchain. Later, for testing the data retrieval mechanism, a similar contract is used to obtain the CID based on data retrieved from IPFS.

To implement the contract, the Solidity language was used (the latest version being 0.8.24) with online IDE Remix. Each Ethereum user has a wallet that is identified by a specific cryptographically generated address. In this case Metamask (Chrome plug-in) was used to interact with the Web3 application. For the Ethereum Sepolia testnet network there are several online services (faucets) that offer free fractions of ETH coins for the purpose of testing applications. The user can send and receive cryptocurrencies (ETH or other tokens) and

interact with smart contracts to register, update or run.

A smart contract is uploaded to the Ethereum network through a special transaction - contract creation transaction. Such a transaction will include the bytecode of the contract, the address of the sender - the user sending the transaction to the network. The recipient of the transaction is in this case the address 0x0, and the specified value is usually zero. The nonce is a number that increments with each sender transaction and is used to prevent duplicate transactions or to determine the order of transactions. Like any other transaction, it will contain the gas, which is the units of calculation needed to process the transaction, and the gas price, which is the rate paid for each unit of *gas*.

After the contract becomes active, users can interact with it by sending transactions to the smart contract address and calling the functions and methods defined in the contract. The contract registration can be verified (like any other transaction) with Etherscan Explorer. It gives users access to detailed information about blocks, transactions, wallet addresses, smart contracts and more, which are stored on the Ethereum network.

Essentially, once a transaction interacting with a smart contract is confirmed and included in a block, the contract state is updated to reflect the changes made by the transaction. Thus, when a user accesses a variable or calls a read function, the value is returned from the contract state, which is updated and maintained by each node in the network.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract SaveMetadataString {
    mapping(address => string) public savedData;

    event IoTDataSaved(address indexed user, string newMetadataString);

    function saveMetadata(string memory _string) public {
        savedData[msg.sender] = _string;
        emit IoTDataSaved(msg.sender, _string);
    }

    function getData() public view returns (string memory) {
        return savedData[msg.sender];
    }
}
```

Fig. 1. Smart contract fragment in Solidity

A fragment of the smart contract is shown in *fig.1* to allow a brief discussion on the working mechanism. This simple example smart contract written in Solidity allows users to save a string associated with their Ethereum address and retrieve it later.

The *mapping* statement associates an Ethereum address (in this case of the user accessing the contract) with a string. The public variable will be accessible directly from

outside the contract. The *IoTDataSaved event* is emitted whenever a user saves a string, thus allowing applications to react to on-chain events. *function getData* returns the string associated with the address of the user who called the function. The function is declared view, which means it will not change the state of the contract and only returns a value.

In this example, there is no restriction limiting access to the contract's *saveMetadata* function. Thus, any user who owns an Ethereum address and has access to the Ethereum network can send a transaction to the smart contract address and record a string using the *saveMetadata* function. Each user who submits a string registration transaction will have their string saved within the contract associated with their Ethereum address. This means that each user can register and retrieve their own string using their Ethereum address. Multiple strings can be stored for each address: for example, a mapping of mappings, where the outer key is the user's address, and the inner key can be a unique identifier associated with each string.

Data immutability on the blockchain does not mean that data cannot be updated, but rather that changes are made by adding new values or updating the state of the contract, and existing data remains intact and available for consultation transparently. There are three types of variables in the Solidity coding language: local variables, global variables and state variables (*World State*). Global variables hold transaction information and blockchain properties like sender or origin, but the state variables are permanently stored in the chain (*smart contract storage*).

Table 1. Time and cost evaluation

1	Time to write in IPFS (JSON 592 bytes)	1570 - 1748 msec
2	Time to read from IPFS	695 - 1020 msec
3	Cost of smart contract deployment (gas) Sepolia Ethereum Testnet	0.00108179 eth 0.00398563 eth
4	Time to write in blockchain	5234 - 9263 msec
5	Cost to write CID (IPFS hash) in blockchain	0.00023343 eth 0.00025173 eth
6	Gas price	2.507532742 Gwei 2.704124527 Gwei
7	Time to read (46b) from blockchain	191 - 197 msec

Updating a variable in a smart contract in Ethereum involves using a transaction recorded on the chain that will update the state of the contract. Thus, reading a variable does not require traversing the entire transaction history.

Table 1 show the times for write and read and the gas used in this experiment; there are one billion gwei (Giga wei) to one ether and one quintillion wei (18 zeros) to one ether.

Ethereum solves the problem of updating state variables through the concept of "world state". At a given moment, world state represents the current state of the smart contract. World

state is a mapping between the Ethereum addresses of accounts and the states of these accounts. When a smart contract is called and changes a state variable, the blockchain history is not changed, but the world state is updated to reflect the new values of the state variables. To implement these functionalities in an Ethereum node there is a database (usually LevelDB or RocksDB) that stores the world state. Regarding the large number of smart contracts (over 45 million), Ethereum manages this by using *gas*. Each instruction executed in Ethereum requires a certain amount of gas, which is paid by the one who initiated the contract call.

4. RELATED WORK

In recent years, more studies are oriented on introducing blockchain technology into the IoT applications – smart grids, smart cities, healthcare [9], supply chain and logistics, autonomous vehicles [10], decentralized identity [11].

In power systems blockchain can be used to build decentralized and secure P2P energy trading platform. An example of existing commercial implementations is PowerLedger. Relevant apps in smart grids are also secure metering and grid monitoring. The paper [12] presented two methods for voltage monitoring and storage. The authors considered that the centralised method is more vulnerable to attacks but the second decentralised solution using Ethereum IPFS and Python enhances security and relatively slow.

As decentralized storage solutions expand and improve, the prices of these services also become more affordable than cloud storage [13]. One of the most well-known storage providers using IPFS is Filecoin which operates a blockchain based on the FIL coin to provide incentives to participants. Thus, this mechanism does not require trust between the contractual parties.

Although the automatic monetization of IoT [14] data appears to be an attractive aspect for data owners, there are still serious regulations issues related to the recognition of cryptocurrencies. However, one promising aspect could be the introduction of Central Bank Digital Currencies (CBDC) [15].

5. CONCLUSIONS

In this paper we have reviewed some aspects related to decentralised data storage and the applications of these techniques for IoT systems. Most solutions use blockchain for access control and data recording in IPFS. We carried out a simple experiment by implementing a web solution with current technologies to analyze the flow of this technique. The level of complexity is higher than in the case of classical solutions, and the access time can be

restrictive for some applications. At the same time, offers benefits: transparency, enhanced data security, autonomy or monetization potential of IoT data.

REFERENCES

[1] T. Samizadeh Nikoui, A. Masoud Rahmani, and A. Balador, H. H. Seyyed Javadi, *Internet of Things architecture challenges: A systematic review*, International Journal of Communication Systems, 2020, 34.

[2] M. Laroui, B. Nour, H. Moungla, M. Ali Cherif, H. Afifi, et al. *Edge and fog computing for IoT: a survey on current research activities & future directions*. Computer Communications, 2021, 180, pp.210-231. 10.1016/j.comcom.2021.09.003.

[3] L. Tseng, *Recent Results on Fault-Tolerant Consensus in Message-Passing Networks*,

[4] Brewer, E. (2000). *Towards robust distributed systems* (PODC keynote). Proceedings of the Nineteenth Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Portland, Oregon, USA, pp. 1.

[5] Fischer, M. J., Lynch, N. A., & Paterson, M. S. (1985). *Impossibility of distributed consensus with one faulty processor*. Journal of the ACM (JACM), 32(2), pp. 374-382.

[6] K. Dinesh Kumar, N. Duraimutharasan, HJ. Shanthi, G. Vennila, B. Prabu Shankar, P. Senthil, *Comparative Analysis of Transaction Speed and Throughput in Blockchain and Hashgraph: A Performance Study for Distributed Ledger Technologies*, Journal of Machine and Computing 3(4), 2023.

[7] K. Jezek, *Ethereum Data Structures*, University of Sydney, [available online] https://arxiv.org/pdf/2108.05513.pdf , 2021.

[8] B. Confais, B. Parrein, J. Lacan, F. Marques. *Characterization of the IPFS Public Network from DHT Request*s. Transactions on Large-Scale Data and Knowledge-Centered Systems, 2023, Lecture Notes in Computer Science, 14280, pp.87-108. 10.1007/978-3-662-68100-8_4.

[9] S. Alam, S. Bhatia, M. Shuaib, MM. Khubrani, F. Alfayez, AA. Malibari, S. Ahmad, *An Overview of Blockchain and IoT Integration for Secure and Reliable Health Records Monitoring*, Sustainability. 2023; 15(7):5660. https://doi.org/10.3390/su15075660

[10] S. Mathur, A. Kalla, G. Gür, M. Kumar Bohra, M. Liyanage, *A Survey on Role of Blockchain for IoT: Applications and Technical Aspects*, Computer Networks, Vol. 227, 2023.

[11] C.D. Nassar Kyriakidou1, A.M. Papathanasiou, G.C. Polyzos, *Decentralized Identity With Applications to Security and Privacy for the Internet of Things*, Computer Networks and Communications, 2023, vol.1(2), 244-271. DOI 10.37256/cnc.1220233048.

[12] S. Mishra, S. Kumar, *Smart Voltage Monitoring: Centralised and Blockchain-based Decentralised Approach*, 2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), BALI, Indonesia, 2021, pp. 49-55.

[13] RW6, M. Irfan Khalid, I. Ehsan, A. Khallel Al-Ani, J. Iqbal, S. Hussain, S. Sajid Ullah, Nayab, *A Comprehensive Survey on Blockchain-Based Decentralized Storage Networks*, IEEE Access, 11:10995-11015, 2023.

[14] RW4, M. Abbasi, J. Prieto, A. Shahraki, J.M. Corchado, *Industrial data monetization: A*

blockchain-based industrial IoT data trading system, Internet of Things, vol.24, 2023.

[15] RW5, N. Pocher; M. Zichichi, *Towards CBDC-based Machine-to-Machine Payments in Consumer IoT*, Conference Proceedings: The 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22), Brno, 2022.