Concurrency Conflicts Resolution for IoT Using Blockchain Technology

Amr Morgan†, Ashraf Tammam††, and Abdel-Moneim Wahdan†††

† Faculty of Engineering, Military Technical Collage, Cairo, Egypt
†† Faculty of Engineering and Technology, Arab Academy for Science, Technology and Maritime Transport (AASTMT), Cairo, Egypt
††† Computer and Systems Engineering Faculty of Engineering, Ain Shams University, Cairo, Egypt

Summary
The Internet of Things (IoT) is a rapidly growing physical network that depends on objects, vehicles, sensors, and smart devices. IoT has recently become an important research topic as it autonomously acquires, integrates, communicates, and shares data directly across each other. The centralized architecture of IoT makes it complex to concurrently access control them and presents a new set of technological limitations when trying to manage them globally. This paper proposes a new decentralized access control architecture to manage IoT devices using blockchain, that proposes a solution to concurrency management problems and enhances resource locking to reduce the transaction conflict and avoids deadlock problems. In addition, the proposed algorithm improves performance using a fully distributed access control system for IoT based on blockchain technology. Finally, a performance comparison is provided between the proposed solution and the existing access management solutions in IoT. Deadlock detection is evaluated with the latency of requesting in order to examine various configurations of our solution for increasing scalability. The main goal of the proposed solution is concurrency problem avoidance in decentralized access control management for IoT devices.

Key words:
Blockchain, distributed systems, decentralized access management, internet of things, concurrency management problem

1. Introduction

New aspects of information technology (e.g., sensors, smartphones, smart electronic devices, and wireless appliances) vastly grown in recent years. These new information forms introduce many security issues, privacy, and access management problems. The Internet of Things (IoT) has evolved as a combination of technologies derived from Wireless Sensor Networks (WSN) and smart electronic gadgets. It allows smart homes and organizations to perceive, react, and communicate through the online platform. It also serves as a point of access with critical infrastructure networks. At this time there are many attack vectors appeared and spread for these devices therefore, protection has become necessary and important. It is difficult to protect and secure IoT devices because of main vulnerabilities such as the insecure web interface, network services, and the cloud interface, lack of transport encryption, privacy concerns, insecure software/firmware, insufficient security features, and ineffective authentication/authorization. [1]

These features need the development of security solutions for IoT devices. [2]

This security design is complex and more complicated when considering the characteristics and restrictions of these devices, which include:

i) Resource constraints: Several IoT devices have limited resources, such as storage capability, memory, and computational power. Encryption-based security techniques aren't suitable also to be matched for these limited devices since they can't handle complicated encryption and decryption quickly enough to safely transfer data in real-time.
ii) Heterogeneous devices and communications: IoT systems often adopt different devices with different specifications and characteristics from hardware and software and also use various communication channels on a wide range of operating systems, which makes it impossible to apply the same or traditional security solutions to IoT systems.
iii) Privacy: many IoT systems need to use their information and collected data to realize their purposes and functions, this information needs to be protected to an acceptable degree due to its privacy.
iv) The large scale: The ever-growing scale makes the complexities of developing security solutions for IoT systems difficult.
v) Trust management: Trust computing is a key component of the design of security. Trust protection remains a major challenge in IoT with a large part of the IoT systems structured as peer-to-peer or ad-hoc networks.

Manuscript received July 5, 2021
Manuscript revised July 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.7.38

vi) Web, mobile, and cloud applications: these applications and services used for IoT devices data management, access, and processing should also be secure as part of multi-layered IoT security

vii) Manage device updates: there is a range of difficulties in applying changes to firmware or applications running on IoT devices and gateways, including security fixes. Many types of IoT devices cannot support over-the-air (OTA) updates, or updates without interruption, so devices might need to be physically accessed or temporarily pulled from production to apply updates. In this paper, we are trying to overcome some of the above challenging security issues focusing on the issue of managing access control keeping good privacy scenarios.

The paper is structured as follows. Section 2 examines what a blockchain is, and how a blockchain network operates ended with a taxonomy for blockchain and investigates the security challenges in IoT. Section 3 states the issue of decentralizing access management in IoT and concurrency problems. Section 4 describes the proposed architecture and management algorithm for managing the concurrency problem of decentralized access control in IoT devices and implementation. Section 5 describes the setup used to evaluate the performance of the proposed system. Finally, Section 6 concludes this paper.

2. IoT-Blockchain Challenges and Solutions

2.1 Security Challenges in IoT

Security is a requirement for IoT systems to protect sensitive data and critical physical infrastructures. The new features and special characteristics of IoT systems enforce to solve many security challenges due to its applications which are established for the analysis and collect critical information and data.

(Kewei Sha et al) Target to analyze security challenges resulting from the IoT systems' special characteristics and the IoT application's new features. [3]

Furthermore, open issues are also identified for each layer in IoT architecture. There are many challenges that appeared in IoT devices, one of the most important challenges is End-to-End security, decentralize access, which is a big problem.

Although resource constraints at the things layer are a limit in choices of available security techniques, there exist necessities of deploying these issues. One solution for supporting these challenges in IoT systems is to enhance and increase the available resources such as memory and computing the power capacity to IoT devices so that can utilize available security solutions. [4]

Another solution is to add an extra security-related layer (Edge Layer) that contains hardware and devices to trust with the things layer and use the edge layer as the security agent to manage IoT security needs (deploy decentralized access control at the edge layer) Fig. 1.

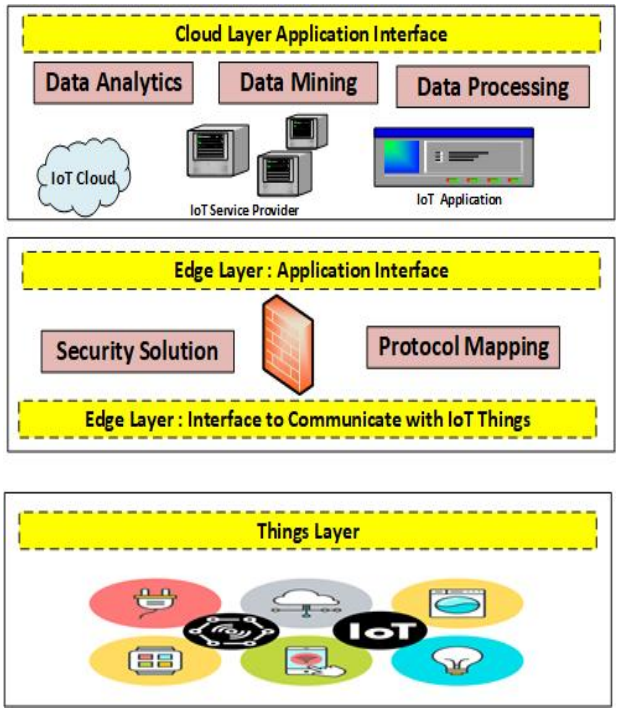

Fig. 1 Deploy Security Service at The Edge Layer.

There are several advantages to deploying security at the edge layer such as:

i) More resources are available at this layer.

ii) Edge layer devices are physically close to ending devices. This not only reduces the communication cost significantly but also improves the real-time performance of IoT applications.

iii) The edge layer also has more information than end devices about the whole system, thus it is possible to deploy more optimized security management at the edge layer.

2.2 Blockchain

The technology of blockchain was first introduced in 2009 with the creation of the first block of the chain by Satoshi Nakamoto. [5]

Blockchain is the mechanism that allows transactions to be authenticated and trusted without the need for a central authority. It provides a decentralized, distributed, shared, and immutable database ledger that stores a registry of assets and transactions across the network. The Blockchain may be consulted in an open and comprehensive manner, enabling

access to all transactions that have occurred since the system's inception. Transaction in the system and can be verified and classified by any entity at any time.[6] Blockchain also makes it is possible to develop new systems with more participatory decision-making and decentralized (autonomous) organizations, which can operate over a computer network without human intervention. The technology of blockchain has many properties such the following:

i) Decentralized, distributed, time-stamp records, shared, and immutable database ledger that stores registry of assets and transactions, also it does not need a central authority to dictate rules, and all the nodes of a network validate the transactions instead of a central entity.
ii) Anonymous transactions (very high privacy).
iii) Authentication, Authorization, and Privacy: blockchain smart contracts can provide an IoT device with decentralized authentication rules and logic to provide single-party and multi-party authentication. Smart contracts can also provide more efficient rules for authorizing access to connected IoT.
iv) Secure Communications: the blockchain completely eliminates the problems of key management and distribution, as each IoT device would have its own unique GUID, and asymmetric key pair installed and connected to the blockchain network.

2.3 Blockchain Solutions, and Open Challenges

This section describes how blockchain can be a key technology to deliver variable security solutions to IoT security challenges today.

(Minhaj Ahmad Khan et al) And (Ashwin Karale et al) present and survey major security issues for IoT. Also, discuss how blockchain, technology for bitcoin can be an important enabler to solve many IoT security problems in an efficient, secure, and trustworthy manner. One of the main challenges facing IoT devices is the decentralized access management for IoT. Blockchain has the ability to solve these challenges easily, securely, and efficiently, which is used widely for providing trustworthy, authorized identity registration, decentralize access management, ownership tracking, and monitoring, also achieving data authentication and Integrity in data transmitted by IoT. [4] [7]

3. Access Control Management in IoT and Concurrency Problems

We propose a decentralized access control architecture and management algorithm for managing distributed IoT networks devices, and managing the concurrency to avoid deadlock occurred in IoT resources.

3.1 Related Work

There is a lot of research available in literature integrating human access management and blockchain, but there is a lack of solutions offering IoT devices access management with blockchain coining IoT, decentralize access management. [8]

The structural design proposed in this paper presents a modified architecture for concurrent IoT access management using the proposed concurrency algorithm and private blockchain technology.

(Oscar Novo 2018 2019) Proposed an architecture for arbitrating roles and permissions in IoT which is a fully distributed access control system for IoT based on blockchain technology. [9][16]

This architecture faces 2 main issues summarized as follows:

i) Multi-Management concurrency problem: in this architecture, every IoT device has to belong to multiple numbers of managers which controlling the same device. There are many ways to move management authority from one manager to another or to add or remove several system managers. IoT devices may have several managers at same time, so if there are two or more managers access the same IoT device at the same time there is a concurrency problem will occur.
ii) Malicious Management hubs: IoT devices can connect to the closest management hub, the device first needs to discover the hub's IP address. There can be several mechanisms for discovering the closest management hub node but in this implementation, they assumed the IoT devices are connected to the default management hub without any type of authentication.

In this paper we modified this architecture to overcome the multi-management problem and concurrency.

3.2 Concurrency Issues in Database Environment:

The following are the most common concurrency issues in database environment: [10]

3.2.1 Dirty Reads

A transaction reads data that has been written by a recently uncommitted transaction Fig. 2. For example, power grid manages power consumption for a set of factories:

- Factory F1 issues power ON transaction (TF1), but not executed yet (no committed).
- Power grid G1 reads the status of the factory (TG1) and found its ON.
- factory F1 decided to rollback and stop power ON procedures
- Here power grids have a dirty read that the factory ON while it's not.

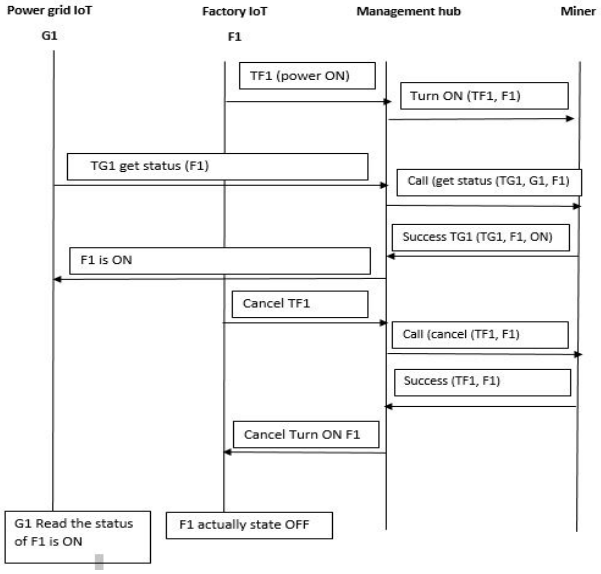

Fig. 2 Dirty Read Diagram.

3.2.2 Non-Repeatable (Fuzzy) Reads

When a transaction rereads data it has already read, it discovers that the data has been updated or removed by another committed transaction Fig. 3. For example:

- The Power grid (G1) read the power consumption of a factory (F1) to adopt power providing for that factory by increasing or decreasing power generation to meet its demands.
- The Power grid reads (G1) power consumption of a factory (F1) and found the reading is value X, while it preparing to adopt power providing to X, and before it's become ready, its return to check the reading again and then found the reading is changed to value Y.

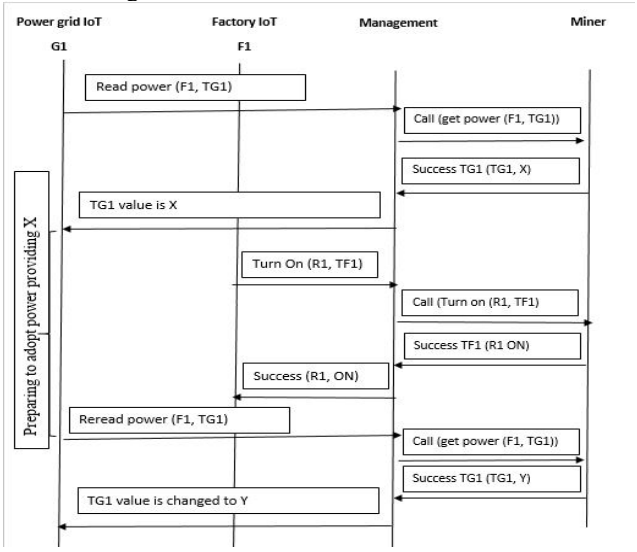

Fig. 3 Non-Repeatable (Fuzzy) Reads Diagram.

3.2.3 Phantom Reads

When a transaction is a set of results returned by a query that fulfils search criteria and then it is discovered that another committed transaction has added additional rows that meet the criterion Fig. 4. For example:

- The power grid checks the factories that working to take action for generating power providing for these factories.
- When the power grid checks the status of factories it found that factories (F1, F2, F4) are ON.
- While the power grid starting to prepare its self for generating the power, factory (F3) issues power ON transaction.
- When the power grid preparation is ready and before executes, power generation returns to check the reading again, then it found the reading was (F1, F2, F3, F4) is ON.
- More data satisfies the query criteria than before, but it doesn't like the case in a fuzzy read (the previously read data is unchanged).

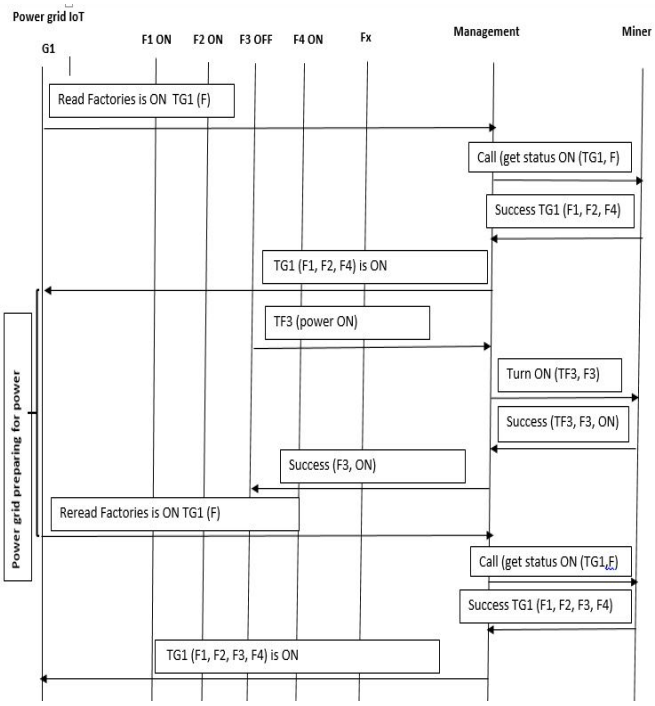

Fig. 4 Phantom Reads Diagram

4. Implementation

4.1 Proposed Concept

The concept assumptions to our proposed algorithm need to define the following terms in order to avoid concurrency problems with distributed databases in IoT:

i) Resources: In the registration phase, Each IoT device should define or broadcast its resources with allowed type of locking mechanisms and the release time for this resources. The lock must not interfere with device critical resources such (Alarm generator – Power ON or OFF in critical situation – Important recourse that cannot be locked).

ii) Transaction: All IoT operations should be atomic (all transactions must be executed as a single block or canceled as a single block), atomic operations are organized into a set of transactions carried out as a unit and finally either committed or rolled back, any transaction must define its isolation level .

iii) Lock Type: All IoT devices should define the allowed locking type to its resources. ANSI locking mechanism to define isolation level Table 1:
   a) Read uncommitted.
   b) Read committed.
   c) Repeatable read.
   d) Serializable.

Table 1: Shows the Concurrency Side Effects Allowed by the Different Isolation Levels.

Isolation Level	***Dirty Read***	***Non-Repeatable Read***	***Phantom Read***
Read uncommitted	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible
Repeatable read	Not possible	Not possible	Possible
Serializable	Not possible	Not possible	Not possible

iv) Release timeout: Is the time between granting lock and release in case locker entity does not release it, IoT devices should define lock timeout to prevent infinite wait loop condition in case of disconnecting locking entity while holding a lock or deadlock**.**

v) Modifying the smart contract: Blockchain should modify the smart contract to include a locking table to be published with each blockchain nodes. Miner should obtain both contract and locking table while mining chain.

4.2 Proposed architecture:

This section explains the differed interactions between the different components of our architecture Fig. 5. It presents an overview of a modified decentralized access control management system architecture for arbitrating roles and permission as follows: [11]

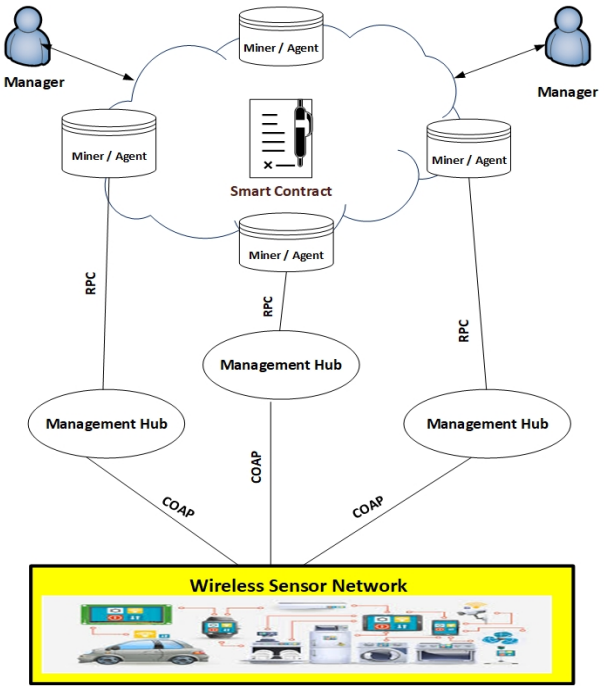

Fig. 5 Proposed Decentralized access control system architecture

i) Wireless sensor networks
Which is a communication network that contains IoT devices with constrained resources.

ii) Management hubs
It is an interface that translates the IoT devices' information encoded in CoAP messages into JSONRPC messages that the blockchain nodes understand. The management hub is directly related to a blockchain node.
A copy of the blockchain must be present on each node in a blockchain network so the blockchain may grow to be rather large. Most IoT devices will be unable to contain blockchain data owing to their limitations. As a result, all entities in our architecture will use blockchain technology except IoT devices and management hub nodes. [12]

iii) Managers
A manager is an entity responsible for maintaining a collection of IoT devices' access control permissions .It does not store blockchain data or validate the transaction of the blockchain. In addition, all registered IoT devices in the system have to belong to at least one registered manager and it can also belong to multiple managers. [13]

iv) Agent node

A particular blockchain node responsible for deploying the single smart contract, and it's the owner of this smart contract during its lifetime.

v) Smart contract

The system of access management is controlled by the operations specified in a single smart contract which cannot be removed from the system. Therefore, in the smart contract, all the operations permitted in the access system are specified and are triggered by blockchain transactions. Once an operation is triggered by a transaction, the miners will keep the transaction details available globally. [14]

4.3 System Interactions

This section explains the sequence of the operation in our proposed system. As shown in Fig. 6

The interactions between system components can be divided into three distinct stages.

i) Transaction initiation: during this phase, the transaction will be created when the device (D1) is trying to access the resource (R2) with locking type (Lx), then it needs to search for this resource in the locking table and check if required resource and lock type is allowed.

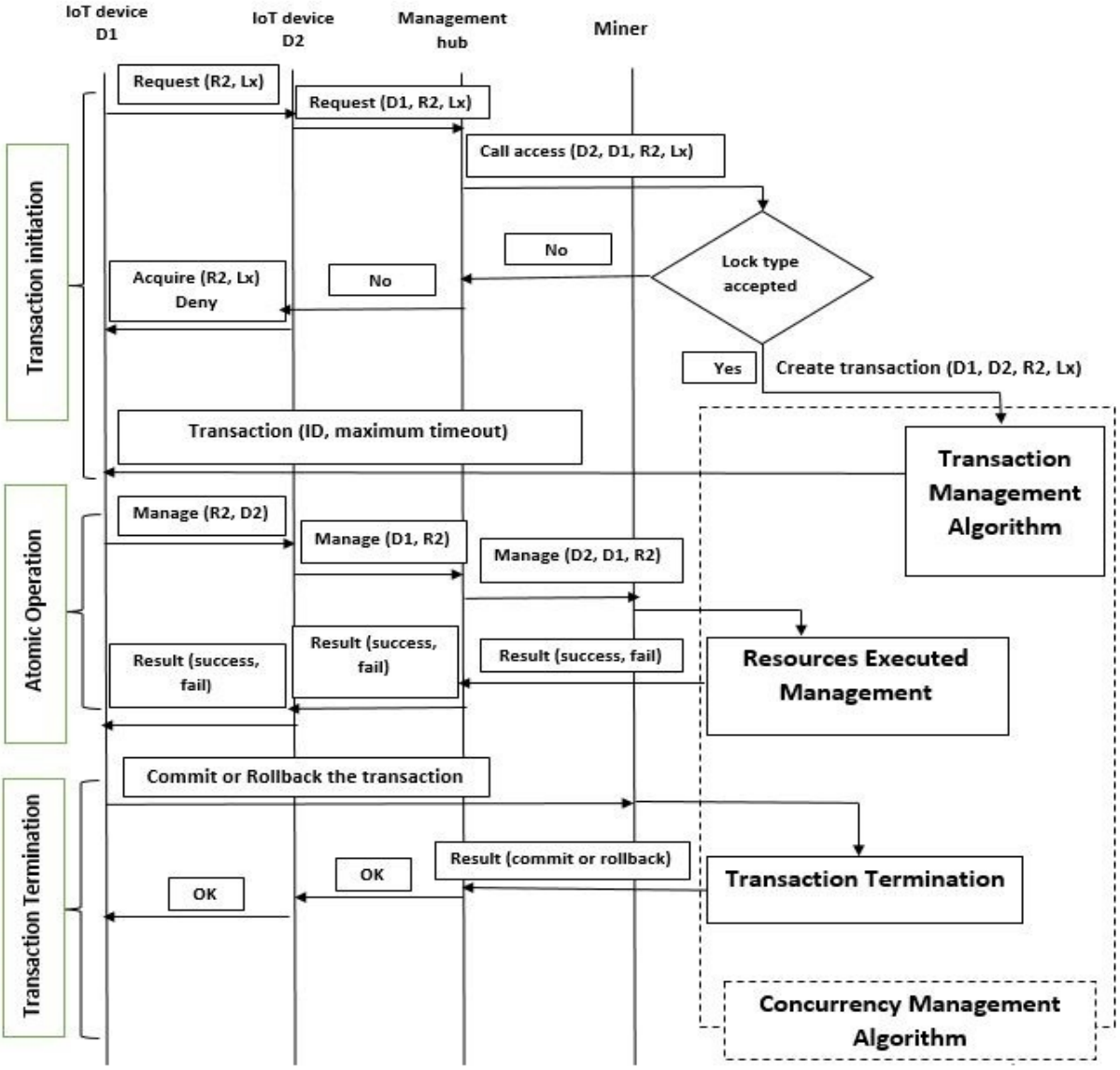

Fig. 6 IoT Network set-up , Registration, and Discovery of the Policy

If it is allowed the transaction will start and then it adds the resource and related information (Timeout, lock type, locking device) into the lock table and replies to the requested device with transaction ID and a maximum management time lock (maximum timeout). If it is not allowed this transaction will be rejected and terminated.

ii) Atomic Operation: during this phase, the whole transaction will be executed and continue to the next state for committing and terminating. In the case of any failure or time-out operations, the algorithm will continue to the next state for termination and rollback.

iii) Transaction Termination: It depends on the previous state. For a successful transaction, an ACK will be sent to the device (D1) to release the resources. In case of failed transactions, trying again till reaching the time-out value. Otherwise, the whole transaction will be terminated and send NACK to release the requested resources.

4.4 Logical State Machine Diagram and Pseudo Code for Proposed Concurrency Management Algorithm

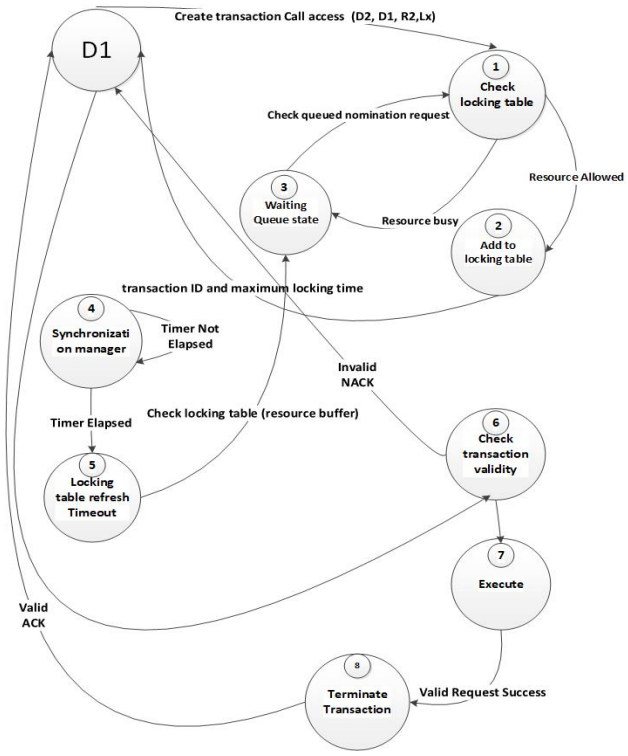

Fig. 7 State Machine Diagram for proposed Management Algorithm

In this section we explain the sequence of the operation in our proposed algorithm as shown in fig. 7:

i) In first state: when a transaction is created and requests a resource, it starts to check to lock table statues for:
   a) Search for the resource in the locking table.
   b) Check if the acquired lock is allowed.

   If the resource is available and the device allowed to use it, then it moves to the second stage, else if the resource is busy it moves to the third stage.

ii) In this state, the resource is added to the locking table with its related information (Timeout, lock type, locking device), also creates a transaction to send to the acquiring device with a transaction ID and a maximum management time lock**.**

iii) In the waiting queue state, the created transaction is added to the waiting for (FIFO) queue if the resource is busy and waits for check lock table signal from synchronization manager module to use provided resource buffer to select nominated queues and nominated requests. Each resource is assigned to a distinct queue and requests to that resource are added to that resource queue and become nominated when it becomes the first request in the queue, then it waits for the locking table signal to dequeue the nominated requests to be executed.

iv) Synchronization manager state: basically it's a timer to check and validate the locking table and keep it synchronized, at each clock timeout the locking table state is visited to manage nominated requests.

v) In the locking table refresh timeout state, the locking table is scan for timed out and released resources then:
   a) Cache timed out and related resources in a resource buffer.
   b) Clear the locking table from timed out and released resources.
   c) Send resource buffer through check lock table signal to the waiting queue.
   d) Waiting queue uses resource buffer to determine nominated queues to release them for execution.

vi) Check transaction validity state:
   a) Check if the transaction ID exists in the locking table.
   b) Check if the transaction not timed out.

vii) Execution state: this state approves the execution of the operation against the resource**.**

viii) Transaction termination state:
   a) Commit or roll back any pending transaction.
   b) Mark transaction and associated resource as released in locking table, which enables synchronization manager to clear.

The Pseudo Code for Proposed Concurrency Management Algorithm shown in fig. 8

```
Call Begin_S0_Tranaction ();
        If   New Transaction call access (D2, D1, R2, Lx)
        Then Call S1 (D2, D1, R2, Lx);
        If   Manage Transaction (D2, D1, R2)
        Then Call S6 (D2, D1, R2);
End S0;
Call Begin_S1_Tranaction (D2, D1, R2, Lx);
        If   R2 is locked by another request  //Resource busy
        Then Call S3 (D2, D1, R2, Lx);
        Else
        Call S2 (D2, D1, R2, Lx);
End S1;
        Call Begin_S2 (D2, D1, R2, Lx);
    Create transaction Tid; //create transaction id
    Add_to_lock_table (Tid, D2, D1, R2, Lx, timeout);
    //timeout is the maximum time allowed to lock the resource
    Return to caller D1 (Tid, max_timeout);
End S2;
Call Begin_S3 (D2, D1, R2, Lx);
    Add_to_wait_queue (D2, D1, R2, Lx);
    If check_locktable_signal then
    Dedueue nominated request list;
    For each request Rq in nominated list Rq (D2, D1, R2, Lx);
    Call S1 (D2, D1, R2, Lx);
    End for;
End S3;
Call Begin_S4;
    Loop1;
        Sleep (refresh timeout);

        Call S5;
    Goto loop1;
End S4;
Call Begin_S5;
    Create empty nominated buffer
    For each Rq (D2, D1, R2, Lx) in lock table;
        If Rq is timeout and its resource cleared
            Then add to nominated buffer (Rq);
        End if;
    End for;
If   notempty (nominated_buffer)
    Then send signal to S3;
End S5;
```

```
Call Begin_S6 (D2, D1, R2, Lx);
    If valid _transaction (D2, D1, R2, Lx)
        Then Call S7;
    Else return to caller D1 (D1, Tid, max_timeout, fail_nack);
End if;
End S6;
Call Begin_S7 (D2, D1, R2, Lx);
    Let exec result = execute transaction (D2, D1, R2, Lx);
        Call S8 (D2, D1, R2, Lx, exec_result);
End S7;
Call Begin_S8 (D2, D1, R2, Lx, exec_result);
    exec_result = success;
    Then commit (D2, D1, R2, Lx);
    Return to caller D1 (D2, D1, R2, Lx, success transaction ack);
End S8;
```

Fig. 8 Pseudo Code for Proposed Concurrency Management Algorithm

5. Evaluation

We established a proof-of-concept (PoC) implementation of our decentralized access control system model by using and adapting the blockchain into a decentralized access control manager, we establish an initial implementation to test and evaluate our architecture. [15]

This section is divided into two parts. The setup and instruments for evaluating management systems are explained in the first section. The second part of the section aims at evaluating the performance and comparing our proof-of-concept implementation with existing IoT standard-based access management systems.

5.1 Experiment Setup

The experiments were done on an Ubuntu-20.04.2 desktop with Intel Core i7-9700T@4.3 GHz. We used Docker version 19.03.8 and an image called vertigo/ethereum12, which is derived from implementation image client-go of the Ethereum protocol ethereum/client-go. Vertigo has been slightly modified to make it simpler to run a private Ethereum network. To dimension our experiments, we use a benchmark tool called CoAPBench, which uses Californium14 as the CoAP implementation baseline. CoAPBench is a tool that resembles ApacheBench and uses virtual clients to meet the defined concurrency factor. [16]

Available: https://github.com/mcollina/node-coap
Available: https://github.com/obgm/libcoap/releases/tag/v4.3.0
Available: https://projects.eclipse.org/projects/iot.tinydtls
Available: https://github.com/ethereum/tests
Available: https://github.com/vertigobr/ethereum
Available: https://blog.ethcore.io/performance-analysis/
Available: https://www.docker.com

5.2 Evaluation and Performance

The objective is to study the scalability of the existing access management method's performance with the performance of the proposed system. Traditional access management solutions rely on a single server for scalability but in our proof-of-concept, we use decentralized access management implementation to handle multiple access with the management hubs at once, also deal with concurrency issues to avoid deadlock, and handle an unlimited number of IoT resources in the blockchain network. [9] [16]

For this experiment, the first scenario evaluates the deadlock detection. In this scenario we assume a set of virtual IoT clients from the CoAP Bench tool requests to access the resources concurrently and calculates the average number of deadlock events that will occur. Fig. 9

For simplicity, we assume ideal cases for both architectures.

For deadlock detection, when the number of clients is increasing, the probability of deadlock events occurring is increasing gradually. While in our proposed architecture the probability of deadlock occurrence is avoided by employs a certain time (timeout) for using the resource.

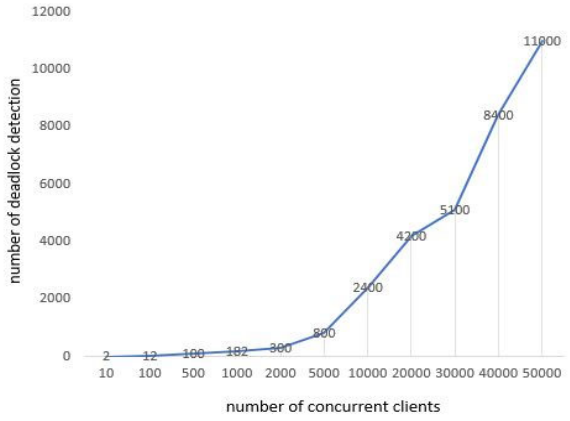

Fig. 9 Compare between deadlock detection in the system with our algorithm and the system with standard algorithm.

The second scenario evaluates the performance and latency of resource access control operations, as one IoT device demands resource information from another IoT device and waiting until receiving this data. [17]

This scenario evaluates the latency using a fixed number of resources. In order to better understand the distribution of latencies, all the processes are analyzed and the outcomes are plotted into cumulative distribution functions. CDFs. Fig. 10 The median latency graph shows clearly that the latency is increasing in the proposed solution more than in the standard one. The latency is relatively small for a standard solution until reaching a saturation point, where the number of requested access exceeds the number of available resources. At this point, the latency drastically increases for the standard model while the proposed model is keeping the increase in latency almost linearly, which is the most significant point.

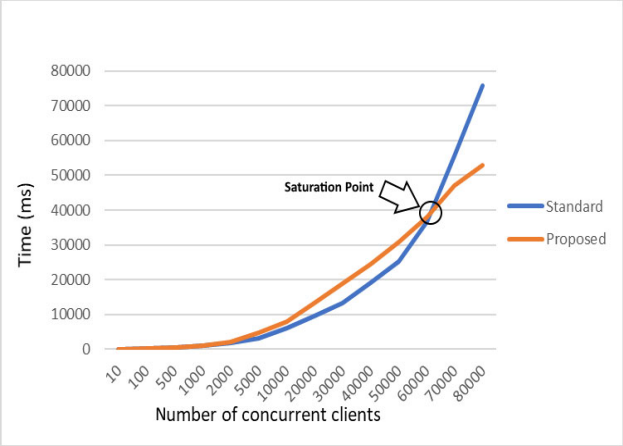

Fig. 10 Compare Latency between standard system and the proposed system with enhancement algorithm.

6. Conclusion

This paper provided a proof-of-concept architecture that uses the suggested concurrency algorithm and private blockchain technology. It implemented a modified decentralized model for IoT devices access control, where the credentials and permissions to access various IoT resources are recorded globally on the blockchain.

Furthermore, we evaluated the new model for latency and deadlock times using CoAP Bench and vertigo/ethereum12 based blockchain network. Using of the timeout to release this resource avoids the probability of resource deadlock, although for latency the proposed model is applying more latency until reaching the saturation point where the standard model latency significantly increases while the new model is keeping the increase in latency almost linearly.

In summary, continuing in this research area, we can have a full framework for decentralized distributed access control mechanisms that using the blockchain technology for IoT which proved a successful model in cryptocurrencies.

References

[1] M. O. I. A. T. H. F. A. Fadele Ayotunde Alabaa, "Internet of Things security: A survey," *Journal of Network and Computer Applications,* vol. 88, no. June 2017, pp. 10-28, June 2017.

[2] A. K. ·. S. Kaur, "Internet of Things (IoT), Applications and Challenges: A Comprehensive Review," vol. 114, no. September 2020, p. 1687–1762, September 2020.

[3] W. W. a. T. A. Y. a. Z. W. b. W. S. c. Kewei Sha a, "On security challenges and open issues in Internet of Things," *Future Generation Computer Systems,* vol. 83, no. June 2018, pp. 326-337, June 2018.

[4] AshwinKarale, "The Challenges of IoT addressing Security, Ethics, Privacy and Laws," *Internet of Things,* no. June 2021, p. 100420, June 2021.

[5] B. O. Daniel Minoli, "Blockchain mechanisms for IoT security," *Internet of Things,* Vols. 1-2, no. September 2018, pp. 1-13, September 2018.

[6] A. S. I. G. V. B. Md.Ashraf Uddin, "A Survey on the Adoption of Blockchain in IoT: Challenges and Solutions," *Blockchain: Research and Applications,* no. February 2021, p. 100006, February 2021.

[7] K. S. b. Minhaj Ahmad Khan a, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems,* vol. 82, no. May 2018, pp. 395-411, May 2018.

[8] N. K. Y. Kenta Yokogi, "Access Control Model for IoT Environment," *IEEE 42nd Annual Computer Software and Applications Conference,* no. July 2018, pp. 616-621, July 2018.

[9] O. Novo, "Scalable Access Management in IoT Using Blockchain: A Performance Evaluation," *IEEE Internet of Things Journal,* vol. 6, no. June 2019, pp. 4694 - 4701, June 2019.

[10] M. I. H. W. M. I. Mohammad Goudarzi, "An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments," *IEEE Transactions on Mobile Computing,* no. January 2020, January 2020.

[11] S. S. K. b. R. J. c. P. G. d. Ali Dorri a, "LSB: A Lightweight Scalable Blockchain for IoT security and anonymity," *Journal of Parallel and Distributed Computing,* vol. 134, no. December 2019, pp. 180-197, December 2019.

[12] R. D. S. C. A. W. L. SHUANG SUN, "Blockchain-Based IoT Access Control System:," *IEEE Access,* vol. 9, no. February 2021, pp. 36868 - 36878, February 2021.

[13] J. C. C. L. 2. K. F. 3. A. H. L. SHENG DING 1, "A Novel Attribute-Based Access Control," *IEEE Access,* vol. 7, no. March 2019, pp. 38431 - 38441, 2019.

[14] K. C. a. M. Devetsikiotis, "Blockchains and Smart Contracts for the Internet of Things," *IEEE Access,* vol. 4, no. May 2016, pp. 2292 - 2303, May 2016.

[15] C. M. J. C. E. S. M. D. Ana Reyna *, "On blockchain and its integration with IoT. Challenges and opportunities," *Future Generation Computer Systems,* vol. 88, no. November 2018, pp. 173-190, November 2018.

[16] O. Novo, "Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT," *IEEE Internet of Things Journal,* Vols. 5, no. 2, no. April 2018, pp. 1184 - 1195, April 2018.

[17] K. G. B. c. N. K. a. Sotirios Brotsis a, "On the suitability of blockchain platforms for IoT applications: Architectures, security, privacy, and performance," *Computer Networks,* vol. 191, no. May 2021, p. 108005, May 2021.

Amr Morgan He received his B.Sc. in Electrical Engineering – Computer Engineering from Military Technical College (MTC), Cairo, Egypt in 2006.

Ashraf Tammam. He is an Assistant Professor of Computer Engineering at Faculty of Engineering and Technology, Arab Academy for Science, Technology and Maritime Transport (AASTMT), Cairo, Egypt. He received his B.Sc. in Electrical Engineering – Computer Engineering from Military Technical College (MTC), Cairo, Egypt in 1994. He received his M.Sc. and PhD in Electrical Engineering - Computer and Systems Engineering from Faulty of Engineering, Ain Shams University, Cairo, Egypt in 2004 and 2011 respectively. He was the Chairman of Information and Decision Support Center (IDSC), Egyptian Cabinet in 2014. His research interest includes Computer and Network Security, IoT, and GIS.

.

Abdel-Moneim Wahdan He is Professor of Computer and Systems Engineering Faculty of Engineering, Ain Shams University, he has supervised many master's and doctoral thesis