Blockchain Network Platform for IoT Data Integrity and Scalability

Chung Yup Kim[1,†], Bryan C.K. Ng[2], Jyoti Sahni[2], Normalia Samian[3,‡] and Winston K.G. Seah[2]
[1]Research and Business Foundation, Sungkyunkwan University, Seoul, KOREA
yup.kim@skku.edu
[2]Sch of Engineering & Computer Science, Victoria University of Wellington, Wellington, NEW ZEALAND
{bryan.ng, jyoti.sahni, winston.seah}@ecs.vuw.ac.nz
[3]Dept of Communication Technology & Networks, Universiti Putra Malaysia, UPM Serdang, Selangor, MALAYSIA
[‡]corresponding author, normalia@upm.edu.my

***Abstract*—Decentralised technology backed by blockchain has gained a huge popularity in recent years as it secures autonomous ecosystems without needing a central authority. The origin of the blockchain concept began in the financial domain using cryptocurrency, but over the last few years, blockchain has been applied to a variety of industries. In the era of Industry 4.0, most industries are leveraging automation by using Internet of Things (IoT). Despite numerous applications of blockchain in the industries, due to the significant latency in consensus algorithm in blockchain, businesses using IoT technology are facing performance issues in adopting blockchain. A number of studies address the obstacles for transaction processing performance and system scalability, mostly based on a public blockchain. However, they still involve centralised components, and thus fail to fully utilise decentralisation. Therefore, a private blockchain-based IoT data integration platform is proposed to achieve data integrity and system scalability. Along with the lightweight IoT gateway instead of any other additional middleware, the process and the system configuration are streamlined. By using Hyperledger Fabric, the design is validated and it is shown that the proposed architecture outperforms other conventional model in IoT data processing.**

***Index Terms*—Private blockchains, Internet of Things, IoT, scalability, data integrity, Industry 4.0.**

I. Introduction

Technology solutions in the era of Industry 4.0 are geared towards industrial automation [1]. When it comes to Internet of Things (IoT) environments in particular, data volume, fault tolerance, and generation frequency are key considerations. Full automation can only be achieved when each process is seamlessly integrated and any single component does not affect the service downtime. Various forms of process automation in manufacturing are accelerated by IoT. The core part of IoT is the processing of a huge volume of data gathered from IoT sensors to obtain useful information. It is the foundation for the autonomous control of equipment for production using IoT data without human intervention. Consequently, the importance of data integrity increases significantly in IoT.

IoT data have become ubiquitous in diverse environments nowadays, collected by a vast range of IoT devices (e.g. sensors and actuators) and sent over communication links to backend legacy systems in manufacturing, with cloud and distributed computing being the most commonly used system architecture for processing the IoT data [2]. However, the centralized architecture of these approaches make them vulnerable to security problems and single point of failure. Blockchain [3] has been identified as an appropriate technology for process automation along with high availability. However, blockchain has its own weaknesses in performance and scalability mainly caused by the consensus algorithm. These weaknesses hinder blockchain adoption in IoT environments as data are generated continuously and voluminously by IoT devices. Despite numerous studies on performance improvements, they do not fully utilise blockchain features as these approaches include additional centralised components.

In this paper, we propose a private blockchain network platform for an IoT-based facility management system in the smart factory environment that reinforces data integrity and ensures seamless processing under IoT-specific constraints. This platform does not include any centralized components and assures high transaction processing performance. The rest of the paper is organized as follows: in the next section, we provide an overview of key blockchain features and limitations of conventional blockchain architectures, as well as, related efforts to address these issues. Following which, we present the design of our approach and discuss how we validated our design to achieve the desired goals. Lastly, we conclude with a summary of our contributions and discuss potential future enhancements.

II. Related Work

A significant amount of research has emerged in recent years on the adoption of blockchain technology in IoT applications [4]. Research on blockchain has previously focused more on the public blockchain, but the private blockchain is preferred in IoT environments due to their speed, lower costs and better privacy [5].

Given the limited literature specifically dedicated to private blockchain-based solutions, this section focuses on the related work pertaining to the applicability of blockchains in IoT environments for data integrity, resilience and scalability, irrespective of whether the blockchain is private or public. Further, we organize the discussion into two distinct

[†]C.Y. Kim was supported by a full international Masters by Research scholarship from the Victoria Huawei NZ Research Programme.

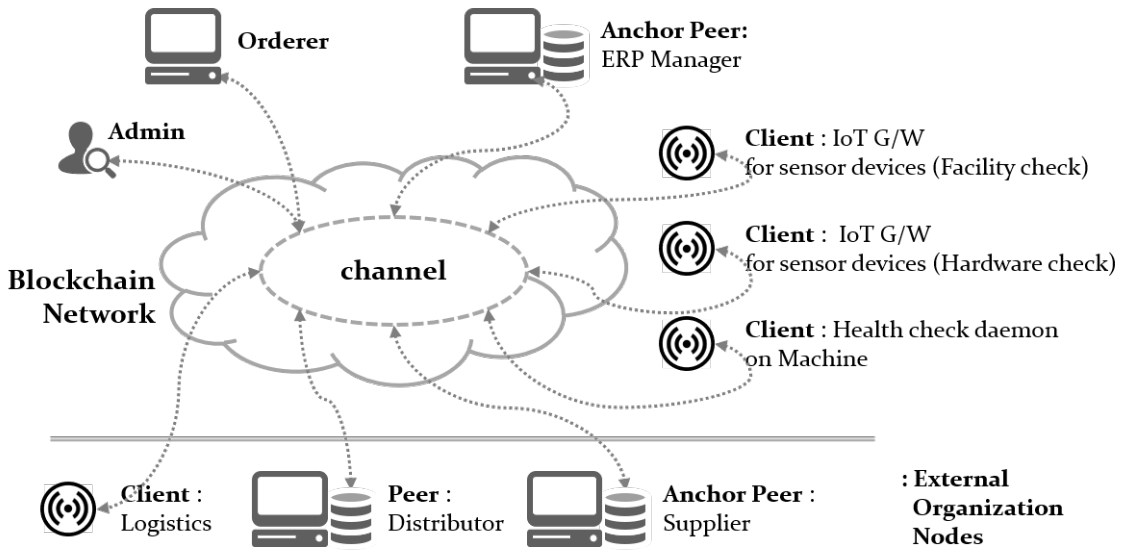
Orderer
Anchor Peer: ERP Manager
Admin
Client : IoT G/W for sensor devices (Facility check)
Client : IoT G/W for sensor devices (Hardware check)
Client : Health check daemon on Machine
Blockchain Network
channel
Client : Logistics
Peer : Distributor
Anchor Peer : Supplier
: External Organization Nodes

Fig. 1. Enterprise blockchain system architecture

categories: a) Cloud integrated blockchain solutions and b) Distributed file system integrated blockchain solutions.

A. Cloud integrated blockchain solutions

Leveraging the Internet, cloud services have gained widespread adoption for processing and storing vast volumes of IoT data. Nevertheless, relying solely on cloud can expose the hosted services to inherent vulnerabilities of the Internet and may not ensure data integrity and availability, which is a crucial requirement for IoT applications [6]. To address this challenge, a cloud blockchain integration approach is many a times utilized, where cloud facilitates distributed storage for IoT applications, while blockchain ensures data integrity and guards against data tampering.

Liu *et al.* [7] addressed IoT data integrity verification in cloud environments and proposed a blockchain based data integrity service framework for IoT data stored through a cloud storage service. They made a number of assumptions such as 51% attack is rarely possible and blockchain consensus can be reached within a short time which may not be currently viable in all kinds of networks.

Liang *et al.* [8] proposed a blockchain-based IoT data integration model for communication between drones, a good example of wireless IoT devices and control systems for data assurance and resilience. Cloud computing was used to store and audit the data that was communicated between drones and the control system, while blockchain networks were used to ensure the integrity of the data.

Xiang *et al.* [9] proposed a blockchain-based decentralized authentication and access control protocol that integrates with edge and cloud computing to enhance e-health systems. They employed a practical Byzantine fault-tolerant (PBFT) negotiation mechanism to improve the negotiation process. However they validated their approach using a set of BAN rules, without any significant empirical evaluations that are essential for testing scalability.

Pon and V [10] proposed SECure LearningChain (SEC-LearningChain), a design that combines blockchain technology, machine learning (ML), and cloud computing primitives to facilitate secure data transactions in a Peer-to-Peer network and efficient data sharing services.

Li *et al.* [11] proposed a blockchain-assisted secure storage scheme for managing logistics records within a smart logistics system, which is backed by IoT devices. Further they employed edge computing to alleviate the burden on the cloud server.

While the above approaches may ensure data integrity, the deployment of both blockchain and the cloud can lead to unnecessary data replication, which may result in data processing latency. Such architectures are not able to fully facilitate decentralization as the cloud system and other centralized controlling systems are vulnerable to a single point of failure.

B. Distributed file system integrated blockchain solutions

To overcome the challenges posed by cloud integrated blockchain models, recent research has focused on creating purely distributed solutions utilizing distributed file systems, that ensure data integrity in IoT platforms.

Hang and Kim [12] proposed an IoT blockchain platform to preserve IoT sensing data integrity. They used permissioned private blockchain to validate the effectiveness and resilience of their concept. However, they also deployed an additional server to control IoT devices and to send transactions, negating the benefits of the decentralised architecture.

Oktian *et al.* [13] proposed a general-purpose framework, SIGNORA, for provisioning dataflow integrity in an untrusted data pipeline. The proposed approach is versatile and applicable across a wide range of data payload sizes. It however suffers from performance and scalability issues as the payload size and the number of chains increase.

Hang*et al.* [14] designed a blockchain based platform for fish farmers to provide them with secure storage capabilities and to safeguard agriculture data from any unauthorized tampering. Although a proof-of-concept was demonstrated, the simulation environment used for testing had limitations and may not be entirely reliable for addressing scalability-related issues.

Sharma *et al.* [15] proposed a blockchain-based IoT architecture incorporating the Identity-Based Encryption (IBE) algorithm, to enhance the security of healthcare data. The model however suffers from high computation time and significant processing overheads.

Kumar *et al.* [16] proposed an integration of deep learning with blockchain architecture to identify intrusions more accurately in the IoT-enabled healthcare system network. Using their proposed estimable Proof of Work (ePoW) consensus algorithm, the detection achieves high accuracy percentage. However, the work does not provide analysis on the proposed ePoW in terms of computational power consumption and energy efficiency which are salient weaknesses of PoW that are not suitable for low-powered IoT devices.

Dorri *et al.* [17] proposed a lightweight blockchain processing by releasing the memory burden of nodes based on the cache system. It enables IoT users and service providers to put an expiry time for the old transactions which could optimize the memory usage in large-scale IoT networks. Although memory usage has shown to be decreased by about 25%, however, the trade-off is that the optimization may be suitably deployed for limited types of non-crucial applications that need not any future referencing or auditing at all. This work still needs to consider an IoT-specific model that is highly scalable for wider adoption.

In this work we propose a novel blockchain platform tailored for IoT environments that is capable of providing enhanced data integrity and scalability while minimizing computation costs, outperforming existing solutions in the field.

III. Blockchain-based Network Design

We now present the design of our enterprise blockchain system for IoT-based production line in a manufacturing industry.

A. System Architecture

The system architecture is as shown in Fig. 1. It includes IoT devices to detect environmental state information and external stakeholders such as hardware parts suppliers, distributors, and logistics companies. Generic application of the blockchain to the public is mostly permissionless, but in many IoT enterprises, devices are deployed by administrators. Therefore, it is reasonable that permissioned private network with an administrator node is considered for our proposed blockchain-based IoT network. All the entities interact with each other and share information on the blockchain.

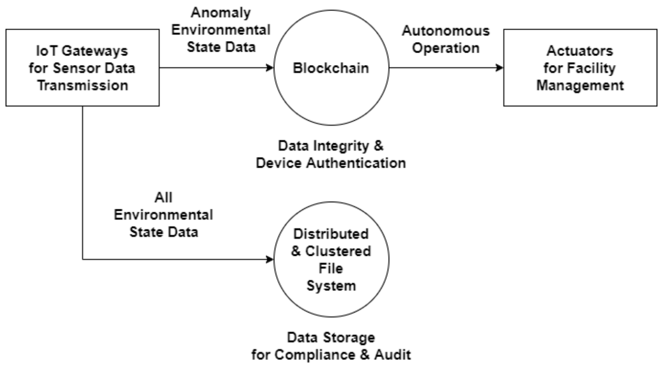
IoT Gateways for Sensor Data Transmission
Anomaly Environmental State Data
Blockchain
Autonomous Operation
Actuators for Facility Management
Data Integrity & Device Authentication
All Environmental State Data
Distributed & Clustered File System
Data Storage for Compliance & Audit

Fig. 2. Workflow of the proposed architecture

- A peer is a basic entity that participates in a blockchain. Among peers, there exist special nodes, viz., anchor peers, endorsers and orderers. Anchor peers are literally a sort of representatives that propagate information of each peer, such that at least one anchor peer exists in every organisation. Endorsers perform transaction verification whereas orderers execute transactions sequentially.
- Except the orderers, all the peers on the blockchain share the ledger database. They replicate the transactions and configuration data locally.
- An entity is designated as a client if it performs transactions with the network but does not join the network. Clients are IoT devices such as IoT sensors, IoT gateways, actuators, and other devices with mobility to send data frequently. IoT sensors acquire data related to environmental states and send them to IoT gateways. Then, the IoT gateways submit transactions to the blockchain, the results of which invoke actuators' action.
- Since it is a private blockchain, administrative tasks are limited, such as, initial network configuration, membership management, and network channel update.

B. Workflow

The workflow of the proposed system focuses on sensor data processing to control corresponding actuators in a production line, as depicted in Fig. 2.

1. IoT sensors gather environmental data, such as, temperature, humidity, power levels, water volume, vibration, and dust density. They are continuously transmitted to geographically nearest IoT gateways or tightly coupled IoT gateways.
2. IoT gateways collect and filter all the data from sensors, and forward appropriate information in order to control facilities for machinery. IoT gateways are authenticated by Certificate Authority (CA) using X.509 certificate issued when they are enrolled to the network, so that the transactions from the gateways with the certificate can be authorised to process transactions.
3. IoT gateways upload all the data to distributed and clustered peer-to-peer storage systems to support the accounting compliance and audit trail of an organisation.

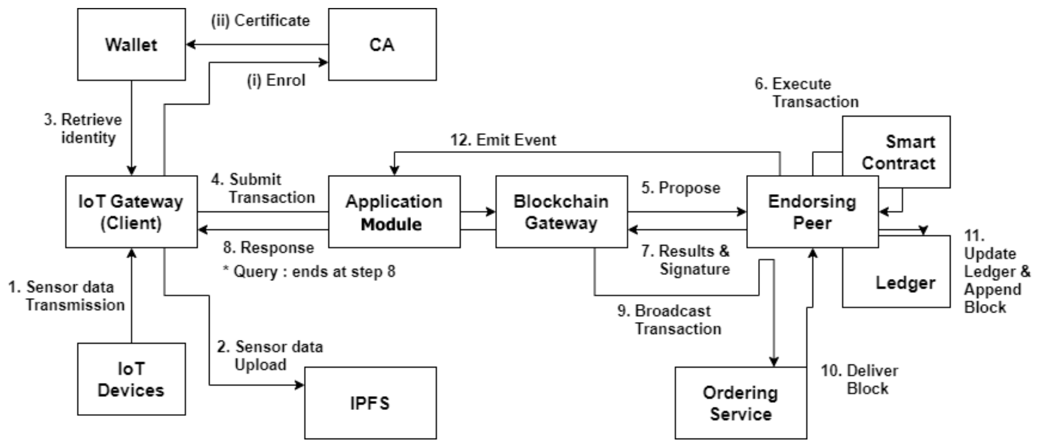

Fig. 3. Transaction flow

4. IoT gateways also concurrently transmit sensed data to the blockchain. The data are filtered by gateways according to predefined thresholds. By filtering the data, the workload on blockchain can be enormously eased. Only anomaly data are processed further onto blockchain. The data selection criteria can be changed to meet ongoing requirements.
5. Once a transaction from the gateway has successfully completed, the operational status of an associated actuator will be updated on the blockchain and the result will be sent to the actuator application.
6. Lastly, facility management systems take proper actions to cope with environmental variation, based on the result from the blockchain. The facilities are operated by actuators, which are another form of IoT devices. Applications on the actuators listen to the transaction results from the blockchain at all times and respond promptly.
7. Data from gateways are processed and integrated onto the blockchain, so that the transaction history may be traced back in the future while immutability and transparency are ensured.

C. Model System Design

To demonstrate the viability of the proposed architecture, we developed a proof-of-concept (PoC) system which comprises four key aspects. Firstly, the transaction flow more specifically shows which component executes what tasks in sequence. In the proposed flow, essential system components for the PoC are illustrated. Secondly, a physical system architecture for the PoC is outlined. In this subsection, how to deploy each component physically is shown inclusive of applied technologies. Thirdly, processes that take place in the smart contract on blockchain for the PoC is explained. This subsection deals with the algorithm in the smart contract. Lastly, client application and its integration with blockchain are described. There exist two types of clients for the PoC. One is the IoT gateway and the other is the actuator. These clients are not participants of blockchain, so that they should interface with blockchain in a certain way.

1) Transaction Flow: Figure 3 depicts the workflow in detail by describing each task on each component with the executing orders.

It is assumed that every IoT gateway is enrolled with CA servers, as shown in Fig. 3 step (i). For this, an internal CA server is used. For the enrollment, the CA server provides with X.509 certificate, cf: Fig. 3 step (ii), which is stored in the file system on each gateway in the form of cryptographic public and private keys, in this architecture, a digital wallet. For the processing of a transaction, it flows as follows:

1. IoT sensors transmit data periodically to IoT gateways.
2. IoT gateways upload the sensed data to the InterPlanetary File System (IPFS)[1] for compliance and audit purposes.
3. When an IoT gateway detects abnormal or unusual environmental data, for instance, when the temperature goes over the predefined upper threshold, the IoT gateway triggers the relevant transaction which is defined in a smart contract on the blockchain; to invoke the transaction, the IoT gateway retrieves the certificate from its digital wallet to be authorised by the blockchain.
4. The IoT Gateway, as a client on the blockchain, then submits the transaction using the deployed Application Module.
5. The Application Module sends a message for the transaction to endorsing peers defined in the blockchain gateway API's connection profile through the gateway.
6. The endorsing peers execute the transaction using the deployed smart contracts on the blockchain but the data are not updated on the state database or the blockchain.
7. Based on the execution in the endorsing peers, they send back the results and if the transaction is validated, including their signature along with the results, to the Application Module.
8. The Application Module forwards the results from the endorsing peers to the IoT gateway. In case of data query

[1]https://ipfs.tech/

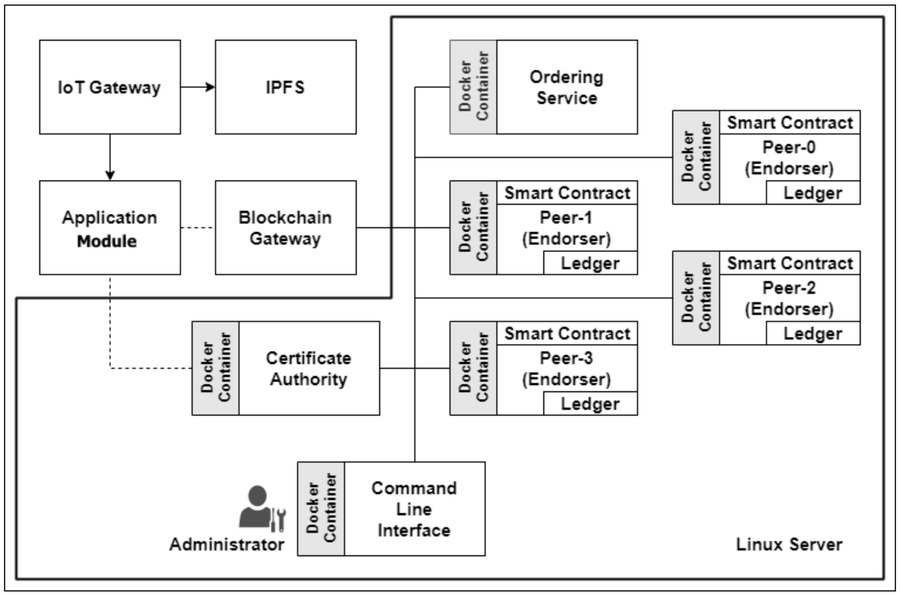

Fig. 4. Proof-of-Concept (PoC) system configuration

rather than the update in general usages of the blockchain, the process will end at this step.

9. Once the Application Module has received the results and the signature, it broadcasts the transaction to be ordered.
10. After the transaction has been ordered and a block has been assembled by the ordering service, the block is sent to committing peers.
11. The committing peers update the state database and append the new block to the blockchain.
12. Finally, the committing peers notify the Application Module that the transaction has been processed.

2) Physical System Configuration: As shown in Figure 4, the Linux server represents the whole blockchain-based system of an organisation based on the proposed architecture. The whole system is streamlined with only essential components. For the purpose of validation, IoT sensor data generation is simulated as the design focuses on the data integration on blockchain rather than on how to gather sensed data from sensors. It is assumed that all the data from IoT sensors are transferred to IoT gateways.

The IoT gateway uploads all the sensed data to a public decentralised peer-to-peer (P2P) file system, namely, IPFS. With minimum criteria, the IoT gateway separates the data that need to be processed further on blockchain from the ordinary data which are discarded. It should be noted that all the sensor data are already stored in IPFS even though the normal data with the usual state are discarded in the IoT gateway.

The most popular enterprise-grade private blockchain, Hyperledger Fabric[2] is adopted to verify the proposed design. As Hyperledger Fabric is available as Docker[3] images, Docker containers host each system component with relevant software libraries respectively. Each Docker container is a set of running software that includes application codes and all the dependencies, so that developed applications can be deployed quickly and reliably to other platform regardless of the environments where the applications should run on. It is regarded to be an enhanced and lightweight virtualisation concept. The set of packages is distributed in the form of Docker images. On top of the Docker container images, platform-dependent executable binaries from Hyperledger Git repository will generate initial materials, such as channel artifacts for blockchain network credentials and the genesis block at which blocks begin to be appended.

For the consensus on transactions, there must be an ordering service. Although there will be multiple orderers in practice, the PoC deploys one orderer to model its ordering operation. In the case of peer nodes, however, multiple nodes are running on discrete Docker containers. The peer node represents a medium on blockchain processing IoT sensor data that are transmitted from IoT gateways. In order to receive data from a considerable number of IoT gateways located in each sector, multiple peer nodes exist. These peer nodes are the participants of the blockchain while IoT devices are not. Peers elected as an endorser among all verify transactions and decide whether to acknowledge them or not. The multiple peers also enable load balancing and fault tolerant processing.

Each peer hosts the ledger replica and smart contracts. As the state database, a dedicated NoSQL key-value store structure database (DB) will run on each peer node. The block of the network is stored on the local file system of each peer node, which is a disk volume of each container. For the certificates to authenticate each IoT gateway on the network, this is done by a central authority (CA) system provided by Hyperledger running on an independent container. Administrative tasks on the system console will be

[2]https://www.hyperledger.org/use/fabric
[3]https://www.docker.com/

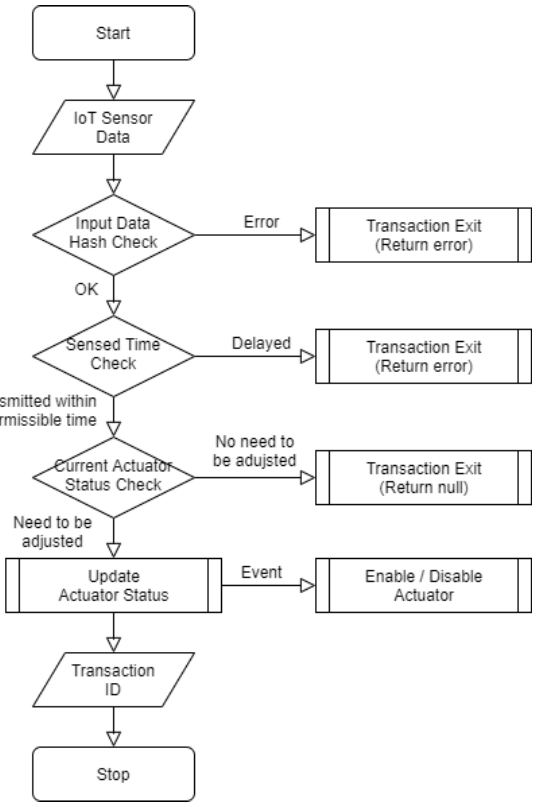
Start
IoT Sensor Data
Input Data Hash Check
Error
Transaction Exit (Return error)
OK
Sensed Time Check
Delayed
Transaction Exit (Return error)
Transmitted within permissible time
Current Actuator Status Check
No need to be adujsted
Transaction Exit (Return null)
Need to be adjusted
Update Actuator Status
Event
Enable / Disable Actuator
Transaction ID
Stop

Fig. 5. IoT sensor data processing algorithm flow chart

done through the command line interface container. The tasks involve the network configuration setting, system operations, such as manual update, and tracing logs.

3) Transaction Processing on Blockchain: Figure 5 shows the actuator control procedure in the smart contract for PoC by using state update function with IoT sensor data as an input.

1. IoT gateway sends the sensor data to blockchain by a client application through the application module and blockchain gateway. Then, the data are processed by a smart contract which basically checks input variables.
2. Input data integrity is validated by SHA256 hash function. The hash value is appended to the data which is composed of IoT gateway ID, actuator ID, sensed timestamp, and sensed data signal. In the smart contract, the hash value is calculated again with only data part and compared with the hash in the received data packet. If both values are equal, the next step will be executed, or the transaction is revoked otherwise.
3. Next, the time at which sensor data were acquired, denoted as "sensed time" in Fig. 5, is assessed. Delayed data with time difference beyond permissible time tolerance compared with the current system time are discarded. This returns an error and the transaction ends. Outdated data do not need to be processed any further since they are no more valid and the actuator should only react to the current environment.
4. After the above data validation, the smart contract retrieves the current actuator status. Based on the prescribed minimum and maximum threshold values, it determines whether the actuator should be activated/enabled or not. Since smart contract can only return the transaction ID as a successful result of a transaction, it triggers an event to associated application clients or actuators for further processing. On the other hand, if no action is needed, the actuator can also be deactivated/disabled.

4) Client Application Integration: Clients in the PoC are IoT gateways and actuators. IoT gateways transmit sensed environmental data to both IPFS and blockchain. Actuators are the controllers in a facility, e.g. factory, acting appropriately according to the result of blockchain processing. They are not the participating peer nodes on blockchain, and require an interface method in order to interact with the blockchain.

A client node is registered and enrolled with a CA a priori, giving it the credentials to access the blockchain and submit transactions with an authorised signature. These credentials consist of a certificate, and a pair of public and private keys in a cryptographic form, stored in a digital wallet.

The digital wallet can be implemented using several methods, such as, local filesystem, in-memory, hardware security module (HSM), and local database system. The local filesystem is chosen for the PoC as it is easy-to-use and can be mounted anywhere on the network.

Hyperledger Fabric provides a Software Development Kit (SDK) for building client applications that interact with blockchain [18]. The blockchain gateway provided by Hyperledger Fabric is an API module that enables client applications to interact with blockchain. Once the client applications have been connected with the gateway, the gateway manages all the subsequent interactions on behalf of the client applications based on the predefined configuration.

IV. Validation

Blockchain has been adopted in many industries for the past few years, most adoptions have faced challenges with performance and scalability, with inherent drawbacks such as high latency, high configuration complexity, etc. The blockchain concept is based on decentralization, where all participants (peer nodes) are expected to process transactions instead of just one entity as in centralized systems. While deploying more computing resources is expected to improve the performance in traditional distributed systems, blockchains have a critical component, the consensus procedure, which require peer nodes to come to a consensus on each transaction. In this case, more resources do not necessarily guarantee better performance.

The testbed set up for the validation is based on the configuration shown in Figure 4. The Linux server is an Intel i7-6700 @ 3.40Ghz CPU and 8GB main memory. The operating system is Ubuntu 18.04.3 LTS. For the private blockchain package, Hyperledger Fabric 1.4.4 LTS is used. For the IoT gateway, which emulates a client application, a Raspberry Pi 3 is used. It is widely used as an IoT device since it is lightweight and versatile [19].

TABLE I
HARDWARE RESOURCE CONSUMPTION PER NODE CONTAINER (2 PEERS)

COMPONENT	MEM (Max. MB)	MEM (Avg. MB)	CPU (Max. %)	CPU (Avg. %)
ORDERER	17.70	14.09	15.03	6.98
PEER-1	135.30	108.64	27.14	14.95
PEER-2	136.50	119.86	27.39	15.23
SUM (MEM)	289.50	242.59		
MAX (CPU)			27.39	15.23

TABLE II
HARDWARE RESOURCE CONSUMPTION PER NODE CONTAINER (4 PEERS)

COMPONENT	MEM (Max. MB)	MEM (Avg. MB)	CPU (Max. %)	CPU (Avg. %)
ORDERER	18.60	14.46	14.33	6.67
PEER-1	129.00	113.21	22.00	12.88
PEER-2	127.00	103.47	22.13	12.88
PEER-3	139.00	109.67	21.82	12.73
PEER-4	128.90	121.09	22.15	12.96
SUM (MEM)	542.50	461.90		
MAX (CPU)			22.15	12.96

TABLE III
HARDWARE RESOURCE CONSUMPTION PER NODE CONTAINER (6 PEERS)

COMPONENT	MEM (Max. MB)	MEM (Avg. MB)	CPU (Max. %)	CPU (Avg. %)
ORDERER	19.60	15.58	15.41	7.24
PEER-1	148.30	111.82	19.60	11.85
PEER-2	145.40	109.56	20.32	11.99
PEER-3	145.60	111.67	19.05	11.82
PEER-4	148.10	115.94	18.74	10.63
PEER-5	143.50	122.92	19.09	10.79
PEER-6	139.30	122.06	19.31	10.43
SUM (MEM)	889.80	709.55		
MAX (CPU)			20.32	11.99

In IoT environments, hundreds or thousands of devices are connected to the network, and peer nodes should have enough capacity to handle as many devices as possible. Otherwise, the alternative is to deploy more peer nodes on the network. However, the more IoT devices a peer node covers, the higher the risk of failure in that group of devices becomes. On the other hand, adding more peer nodes cause the performance problem as mentioned above. Therefore, it is crucial to balance the workload distribution. Our functional tests have verified the proposed architecture. In this section, we focus on the test that assess the performance (latency) and scalability.

A. System Resource Utilisation

In the experiments, Docker containers for each blockchain node have been implemented on a single machine as they represent virtually independent servers with many advantages such as readiness to deployment without adjustment of computational environments. In practice, those containers can be distributed to multiple servers or each container can be placed on a dedicated server separately. Containers also can be replaced by the native installation without using virtualisation techniques. The deployment method and the resource assignment rely on business requirements and transaction volume in enterprise. To size and dimension hardware, basic measurement of resource consumption in blockchain is essential. Therefore, it is worthwhile to find hardware resource utilisation for blockchain configuration.

Tables I, II, and III show computation resource (CPU) and main memory consumption per a Docker container which hosts each blockchain component node. The tests with two, four, and six peer nodes with LevelDB were experimented, each of which used one ordering service and one CA host using state database update transaction. The host machine is equipped with i7-6700 CPU @ 3.40GHz and 8GB main memory.

When a chaincode (or "smart contract") is instantiated on a blockchain, it is launched by a new container for one endorsing peer node, and if a transaction defined in the chaincode is submitted, new containers are created dedicated to each endorsing peer node. These containers for chaincode instantiation consume very little system resources, and can be ignored in the light of main components. The CA node which controls membership and command line interaction node for administrators can also be neglected in terms of resource consumption as they are used only for a certain purpose occasionally. Therefore, only the usages of each peer node and ordering service node are investigated.

However, in practice, there are a few more things to be considered. If CouchDB[4] is used, it runs on a discrete container, so that additional resources should be taken into account for the ledger database implementation. The role of each peer node influences the total resource consumption. Whether a peer is an endorser or not, or which endorsement policy is applied to the network affects the total performance, as it determines the utilisation of the peer node.

All the three scenarios of different number of peer nodes showed average memory consumption of 115 Mbytes and around 15% CPU utilisation. Memory usage is related to the application, which is a container in the experiment. When the application starts, it reserves memory for use, which generally is similar across all the cases.

In the case of CPU usage when the number of peers is smaller, there is a slightly higher utilization, however, as the number of peers increases, the CPU usage improves. It is associated with the endorsement policy. It is set to 'OR' among endorsers in the experiment based on the scenario, so that the transaction will be processed in one peer node while all the peer nodes interact with each other for the service discovery. Thus, all the submitted transactions are eventually spread out evenly to all peer nodes.

In contrast, in the case of 'AND' policy, since all the endorsing peer nodes should process the proposed transaction and verify it, the CPU usage in each peer node becomes higher. Hence, in this case, it is evident that the more peers exists,

[4]https://couchdb.apache.org/

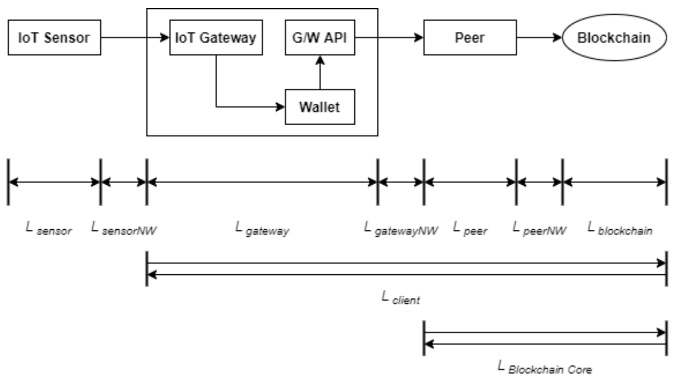

Fig. 6. Latency measurement in components

the worse the overall performance becomes as an inherent downside of decentralisation.

In the case of six peers, as shown in Table III, each peer node consumes up to 150 Mbytes of memory at its maximum usage, which sums up to nearly 1 GBytes in total, whereas around 700 Mbytes in total is used on average. In the case of CPU utilisation, we show the maximum value among the different peers as the utilisation expressed as a percentage cannot exceed 100%. Besides, since each peer node may discretely reach its own maximum CPU usage apart from others' behaviour.

Nevertheless, during the peak period of a particular peer node, it may utilise CPU resources intensively, such that other nodes are vulnerable to the shortage of the resources under multi-threading environments. This increases the CPU wait time, which in turn raises the likelihood of the contention for CPU. Therefore, it is still crucial to consider each node's maximum usage for predictive monitoring. Furthermore, if every node happens to utilise the CPU at its maximum at the same time by any chance, the system will undergo serious delay in processing any transactions. In particular, the maximum CPU usage will be an important factor in hardware sizing.

On the other hand, in the case of memory usage, the accumulated value is correct compared with CPU usage. It is simply because the resource utilisation methods are different. In terms of memory usage, applications allocate physical memory addresses when they start to run. In the experiment, the applications are the peer node containers. The applications are always up and running, so that regardless of user transactions, the reserved memory areas remain exclusively assigned.

Thus, under the similar environments with such hardware resources as the experiment, it is recommended that fewer than six peers are to be implemented on a single host. Otherwise, an additional hardware resource should be deployed.

B. Latency

The time required to execute a transaction is related to the size of blockchain, and the number of peer nodes in the blockchain. If the number of participants grows, the latency for reaching consensus will increase proportionally. Scaling up the resources on a peer node will accelerate the processing, but adding peers to a blockchain deteriorates the performance.

The overall latency comprises the components as shown in Fig. 6. The total response time L_{client} is composed of the latency from a client application through blockchain processing and back to the application as shown in Eqn. (1).

$$\begin{aligned} L_{\text{client}} = \ & (L_{\text{gateway}} + L_{\text{gatewayNW}} + L_{\text{peer}} + L_{\text{peerNW}} \\ & + L_{\text{blockchain}}) + (L_{\text{blockchain}} + L_{\text{peerNW}} \\ & + L_{\text{peer}} + L_{\text{gatewayNW}} + L_{\text{gateway}}) \end{aligned} \tag{1}$$

The latency in blockchain core section $L_{\text{Blockchain Core}}$ is determined as Eqn. (2).

$$\begin{aligned} L_{\text{Blockchain Core}} = \ & (L_{\text{peer}} + L_{\text{peerNW}} + L_{\text{blockchain}}) \\ & + (L_{\text{blockchain}} + L_{\text{peerNW}} + L_{\text{peer}}) \end{aligned} \tag{2}$$

Special attention is paid to L_{gateway} (latency in the client node) and $L_{\text{Blockchain Core}}$ (latency in the blockchain) in the measurements as they are affected by the proposed architecture.

The sensing time L_{sensor} is entirely dependent upon the computational resources of the sensor. Similarly, the latency components L_{sensorNW}, $L_{\text{gatewayNW}}$, and L_{peerNW} correspond to the traversal times through the physical network sections between nodes which are dependent on the network bandwidth and the routing configuration. They are the same for different blockchain-based approaches and, hence, we excluded them in the measurements and focused on the blockchain-related aspects.

In Fig. 6, the actuator, another client in the whole system landscape, is not shown. The actuators access blockchain in the same way as IoT gateways do as they are all external clients of blockchain. The actuator application retrieves locally stored private key to call the blockchain client gateway API to access blockchain. Then, it waits for the event which is emitted from blockchain whenever the ledger update transaction is successfully processed.

Since the actuator is supposed to take immediate action based on the messages from blockchain, an event listening daemon process of the actuator should be constantly run. Otherwise, the actuator is triggered after the event arrives at every time, which causes a significant delay in the processing. Therefore, when the application receives the response from blockchain, it can be regarded that the client in other side, the actuator, has already received the result.

$L_{\text{Blockchain Core}}$, the blockchain transaction processing latency was measured using Hyperledger Caliper and then compared with the total round-trip processing time L_{client} from the client node. This enables the measurement of the latency L_{Gateway} which is highly variable depending on the real environments. This approach is quite significant as the latency of each interval can be analysed respectively. Subsequently, the hotspot can be easily found, and based on the analysis, fine-grained reconfiguration can be achieved.

Fig. 7 shows the average latencies for each case are around 0.1 seconds with a slight variation, whereas the maximum

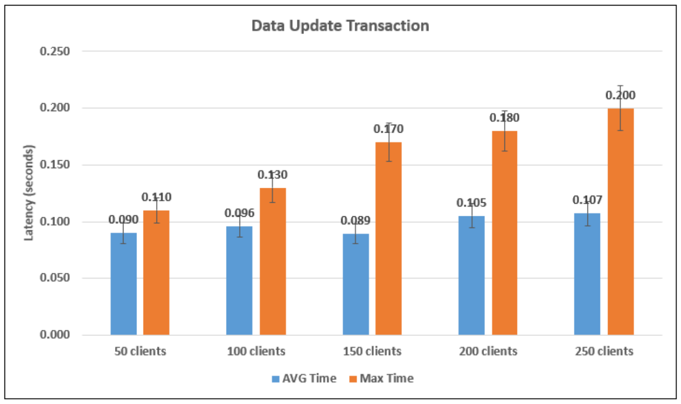

Fig. 7. Update transaction latency in the blockchain segment

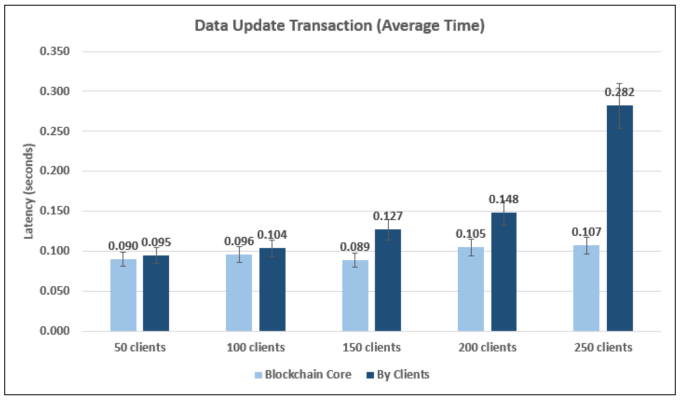

Fig. 8. Average update transaction latency comparison between client and blockchain components

latency values grow as the client number increases. As the number of concurrent transactions increases, the possibility of contention naturally increases, so does the maximum latency. However, these are outliers and have little effect on the average calculation.

Fig. 8 compares $L_{\text{Blockchain Core}}$ and L_{client} as shown in Fig. 6. For the comparison, the average latency in the ledger data update transaction was used. The difference between the left column $L_{\text{Blockchain Core}}$ and the right column L_{client} in each pair becomes the processing time in the client node including the network latency $L_{\text{gatewayNW}}$ between the client node and the blockchain. Since $L_{\text{gatewayNW}}$ is less than 1 millisecond in the simulation environment connected by a wired link, it can be assumed to be negligible, but in practice if the network has very low bandwidth or if a wide area network is used, the latency in this component should be examined carefully so as to avoid undesirable delay.

In scenarios with less than 150 clients, the difference between $L_{\text{Blockchain Core}}$ and L_{client} is small, but in the case of 150 and 200 clients, the gap increases to 0.043 seconds. In the case of 250 clients, the difference becomes more significant. As the number of clients increases, so does the average latency in L_{client}. The increase in L_{client} is larger than in $L_{\text{Blockchain Core}}$ as the network sessions need to be established in the client

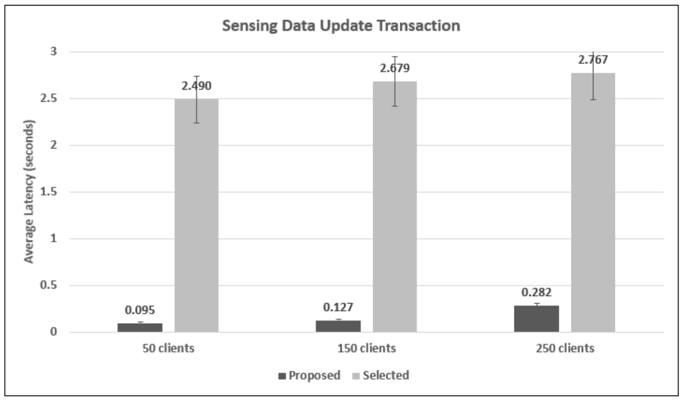

Fig. 9. Average update transaction latency comparison between our **Proposed** system and the **Selected** comparator [12]

node. Since the latency, L_{client} in scenarios with fewer clients is less affected by the delay in network connection, the gap between $L_{\text{Blockchain Core}}$ and L_{client} is small. On the contrary, the larger number of clients makes a big difference in latency caused by the network interface contention. However, the latency $L_{\text{Blockchain Core}}$ in blockchain is stable. With more than 250 clients, L_{client} soared and with 350 clients the simulation environment became inconherent, but $L_{\text{Blockchain Core}}$ continued to show only a little increase in both cases.

Among related works, the proposed system by Han and Kim [12] is closest to this research. It is based on a private blockchain. The differences are it is equipped with an IoT server for device management and a REST API server for interfaces. In comparison with their work, our proposed architecture is faster up to 26 times in 50-client scenarios and 9.8 times in 250-client scenarios as shown in Fig. 9.

Our proposed streamlined architecture excludes any additional management servers for IoT devices and middleware to relay IoT data. Instead, it only involves IoT gateways, on which client applications and connection modules are implemented. Multiple IoT sensors are coupled with the dedicated IoT gateways by either wired or wireless connection, and IoT gateways transmit the sensed data directly to the blockchain. Our proposed architecture does not contain any centralised system component, so that no single point of failure exists. Therefore, the proposed architecture outperforms the existing study [12] in terms of latency as well as robustness.

V. Conclusions

This paper has proposed a blockchain-based platform to be utilised in an IoT environment to ensure data integrity and scalability while ensuring high system performance. To demonstrate the concept, a facility management system based on IoT sensor data in manufacturing was selected as a use case. Performance evaluation results showed that the mean transaction latency is around 100 msecs. Compared with a similar private blockchain-based approach, our proposed system performs close to 10 times faster in scenarios of 250 clients, and even faster performance with fewer clients.

In our study, we have assumed a generally safe non-hostile environment. One aspect for future research is the robustness of the proposed architecture under possible attack scenarios and what additional defense measures are needed. Another future direction of this work would look into developing a hierarchical and scalable blockchain-based trust management protocol with mobility support in massively distributed IoT systems, where mobile smart objects disseminate trust information on service providers to the blockchain network to ensure greater integrity and scalability.

Acknowledgment

This paper was supported by the Korea Institute of Design Promotion (KIDP) grant funded by the Korea Government (MOE, MOTIE) (Design and Emerging Technology Integrated Education Program for Cultivating Innovative Talents).

References

[1] S. Suuronen, J. Ukko, R. Eskola, R. S. Semken, and H. Rantanen, "A systematic literature review for digital business ecosystems in the manufacturing industry: Prerequisites, challenges, and benefits," *CIRP Journal of Manufacturing Science and Technology*, vol. 37, pp. 414–426, 2022.

[2] J. Mocnej, W. K. G. Seah, A. Pekar, and I. Zolotova, "Decentralised IoT architecture for efficient resources utilisation," *IFAC-PapersOnLine*, vol. 51, no. 6, pp. 168–173, 2018.

[3] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Whitepaper*, 2008.

[4] H.-N. Dai, Z. Zheng, and Y. Zhang, "Blockchain for Internet of Things: A Survey," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8076–8094, 2019.

[5] M. Jo, K. Hu, R. Yu, L. Sun, M. Conti, and Q. Du, "Private Blockchain in Industrial IoT," *IEEE Network*, vol. 34, no. 5, pp. 76–77, 2020.

[6] X. Wang, X. Zha, W. Ni, R. P. Liu, Y. J. Guo, X. Niu, and K. Zheng, "Survey on blockchain for Internet of Things," *Computer Communications*, vol. 136, pp. 10–29, 2019.

[7] B. Liu, X. L. Yu, S. Chen, X. Xu, and L. Zhu, "Blockchain Based Data Integrity Service Framework for IoT Data," in *Proceedings of the IEEE International Conference on Web Services (ICWS)*, pp. 468–475, June 2017.

[8] X. Liang, J. Zhao, S. Shetty, and D. Li, "Towards data assurance and resilience in IoT using blockchain," in *Proceedings of the IEEE Military Communications Conference (MILCOM)*, pp. 261–266, Oct 2017.

[9] X. Xiang, J. Cao, and W. Fan, "Decentralized authentication and access control protocol for blockchain-based e-health systems," *Journal of network and computer applications*, vol. 207, p. 103512, 2022.

[10] P. Pon and K. V, "Blockchain based cloud service security architecture with distributed machine learning for smart device traffic record transaction," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 3, p. e683, 2022.

[11] H. Li, D. Han, and M. Tang, "A privacy-preserving storage scheme for logistics data with assistance of blockchain," *IEEE Internet of Things Journal*, vol. 9, no. 6, pp. 4704–4720, 2021.

[12] L. Hang and D.-H. Kim, "Design and implementation of an integrated IoT blockchain platform for sensing data integrity," *Sensors*, vol. 19, no. 10, p. 2228, 2019.

[13] Y. E. Oktian, S. Heo, and H. Kim, "SIGNORA: A Blockchain-Based Framework for Dataflow Integrity Provisioning in an Untrusted Data Pipeline," *IEEE Access*, vol. 10, pp. 89714–89731, 2022.

[14] L. Hang, I. Ullah, and D.-H. Kim, "A secure fish farm platform based on blockchain for agriculture data integrity," *Computers and Electronics in Agriculture*, vol. 170, p. 105251, 2020.

[15] P. Sharma, N. R. Moparthi, S. Namasudra, V. Shanmuganathan, and C.-H. Hsu, "Blockchain-based IoT architecture to secure healthcare system using identity-based encryption," *Expert Systems*, vol. 39, no. 10, p. e12915, 2022.

[16] P. Kumar, R. Kumar, G. P. Gupta, R. Tripathi, A. Jolfaei, and A. K. M. N. Islam, "A blockchain-orchestrated deep learning approach for secure data transmission in iot-enabled healthcare system," *Journal of Parallel and Distributed Computing*, vol. 172, pp. 69–83, 2023.

[17] A. Dorri, S. S. Kanhere, and R. Jurdak, "MOF-BC: A memory optimized and flexible blockchain for large scale networks," *Future Generation Computer Systems*, vol. 92, pp. 357–373, 2019.

[18] "Hyperledger Fabric SDK for node.js Module." Available from https://hyperledger.github.io/fabric-sdk-node/release-1.4/module-fabric-network.html (Accessed on 01/07/2023).

[19] D. Ryu, "Development of IoT Gateway based on Open Source H/W," *The Journal of the Korea institute of electronic communication sciences*, vol. 10, no. 9, pp. 1065–1070, 2015.