Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

# Decentralized and collaborative approach to mobile crowdsensing by implementing continuous feedback between the nodes

K.R. Jansi *, M. Arulprakash

*Assistant Professor(s), School of Computing, SRM Institute of Science and Technology, Kattankulathur,Chengalpattu -603203, India*

A R T I C L E I N F O

*Article history:*
Received 26 June 2022
Revised 18 October 2022
Accepted 1 December 2022
Available online 15 December 2022

*Keywords:*
Kafka
Smart contract
Miners
Feedback
Peer to peer
Proof of work

A B S T R A C T

Mobile Crowd Sensing (MCS) is an effective way to tackle day-to-day life problems nowadays, from Google Maps to Uber cabs. It is an effective and efficient way of data collection, processing and delivery of results to the users, from the users and for the users. All the entities that make up a Mobile crowd sensing system be it a worker, requester or miner can be represented as a node. All the nodes interact together to make a trustworthy system where there is a limited or no scope of deception. The traditional approach to crowd sensing gives little importance to communication between the nodes and in most cases, it assumes the communication happens through a secure channel and ignores the potential hazards. We propose a novel blockchain-based decentralized and collaborative approach to communicate between the nodes which would help workers/users to interact in real-time and solve conflicts should there be any. To decentralize we use the blockchain technology to overcome single point of failure which is a major issue, as mentioned peer-to-peer link between nodes is more expensive only but it improvise the security and privacy concerns by using the consensus mechanism. With the aim of improving the traditional communication technology between the nodes, we propose a mechanism based on Kafka, blockchain and RSA encryption for continuous exchange of data between the nodes which would help to enhance the security and efficiency of the system.

## 1. Introduction

Mobile crowd sensing has revolutionized the tasks like traffic analysis, weather reporting and would continue to do so as mobile devices nowadays are coming with powerful configurations and ample sensors which are used to complete the tasks. There is no need to deploy dedicated Internet of Things (IOT) devices in the Mobile crowd sensing system. Typically, there are three components in any MCS system. They are: -.

1. Requesters
2. Workers
3. Service Provider

The requester requests for some work to be done or data to be collected from a service provider which then selects a worker for the task to be carried out. The result after being worked upon by the worker is sent back to the provider which is forwarded ahead to the users/requester. At the end or on completion of the task a suitable reward is provided to the worker based on the evaluation by the service provider [1,2].

For our work we would refer to all the entities (requesters, workers and service providers) as nodes in the system. MCS systems seem to be an ideal way to tackle problems such as searching for needed information globally which will be helpful for any business decisions or for any individual. It is good to hear; what the crowd has to say and offer, or even by corporates to break down their problem into chunks and distribute it among people and reward them suitably which would no Though widely used t be

* Corresponding author.
*E-mail address:* jansik@srmist.edu.in (K.R. Jansi).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

https://doi.org/10.1016/j.eij.2022.12.004
1110-8665/© 2023 THE AUTHORS. Published by Elsevier BV on behalf of Faculty of Computers and Artificial Intelligence, Cairo University.

possible without large public involvement. But, still it is not a bed of roses, it has its fair share of problems too.

### *1.1. Problems in traditional approach*

Though widely used, the current MCS system is far from being ideal. As in the examples stated above the presence of a central authority handling the entire operations often results in the other actors involved being exploited or not having suitable redress mechanisms. Privacy of all the entities like workers of the task, Requester of the task is gaining a lot of traction nowadays which is often not taken seriously. And this is a major concern. Even after the claims of having suitable encryption mechanisms by the service providers, often there are instances of data leaks. Since the code is not open and all the dependence lies on a single entity, these problems are bound to happen.

The concept of decentralization has found its way into many serious spheres of life like blockchain being used in finance (Bitcoin, Ethereum), elections, etc. The future of MCS could very well be based upon this.

### *1.2. Challenges of the traditional mobile crowd sensing*

In the traditional system, there is a lack of communication and cooperation between the workers working on the assigned task and also between the requester and the service provider. The evaluation of the task completed by the workers is done at the end after fully completing it. And if the work is deemed unsatisfactory by the requester the reward to be given to the worker is forfeited or there is a demand to do the work again. This also leads to the wastage of workers' efforts or if the work done is satisfactory up to that point, if not, the work can be discarded at that moment itself. We aim to bridge this gap of continuous communication [2].

### *1.3. Challenges of the traditional mobile crowd sensing:*

- Scalability: A large group of users is required to be the part of a MCS system to be really effective in making them scalable for large-scale scenarios like say thousands of motorists driving on a road just like traffic indications of Google Maps.
- Malicious Workers: As MCS is an open system, malicious workers can join any MCS task and also insert false data in the process which in turn can harm the privacy and security of the users of the system severely [3].
- Budget Constraints: Generally speaking, the task of data collection done by most of the users in MCS systems is using day-to-day devices such as mobile phones which have average quality sensors present and lack any specialized and good quality sensor equipment. Even the actors involved may not possess the required expertise. Hence, the data quality cannot be ensured properly or guaranteed [12].
- Unreliability: Workers differ from each other on several factors which makes the system unreliable. So, the results obtained from the workers cannot be considered equally accurate.
- Incentive Mechanism: It is challenging to find an appropriate incentive mechanism that encourages users to participate in the system.
- Redundant Data: As it involves a large number of individual workers or contributors it may lead to data redundancy.
  - Here, we propose a fully distributed and decentralized system for MCS which provides good security and most importantly a mechanism to provide an atmosphere of trust to all the entities involved.
  - This product implements mainly the functions of providing feedback through continuous communication between nodes on the work being carried out, blockchain management, providing a mechanism to make the incentive system fairer to both the parties - requestors as well as workers.

### *1.4. Problem statement*

Where so many parties are involved such as in mobile crowd sensing systems, problems due to conflict of interest are bound to arise. Things get more problematic due to the centralized nature of the system where some parties may feel the service provider is biased towards someone. Therefore, such a system where one does not feel cheated and has a say in the working is much needed.

### *1.5. Motivation*

- To address the problem of centralization, there is a chance of single point of failure which leads to complete crash of the entire system.
- To improve the communication between the nodes so that there is reduction in conflicts and fairness is ensured in rewarding.
- To make communication between the nodes secure and private, implementing RSA encryption which is considered one of the strongest encryption techniques available.

System model

The proposed system consists of three types of nodes:

1. Requesters: who raise a request for some task to be done
2. Workers: who do the work as requested by the requesters
3. Miners: who examine workers' results and act as an over watch in case of conflicts

The traditional system does not support direct communication between workers and requesters. All the communication takes place through the service providers. The requesters submit the request for the task to be done to service providers and as a result the service providers pick the workers based on trust score and other parameters.

The workers continue to work on the task given and can only interact with the service providers throughout the process. The results obtained after the successful completion of tasks by the assigned workers can be seen by the requesters only after that result mentioned has been passed on to the service provider who further will pass it down to the requesters.

If the requester is not satisfied with the result of the task, there is a fresh request for the task by the requester changing a few of the requirements to get the desired result. This clearly results in wastage of workers' efforts and time. This might also give rise to the expenditure occurring in rewarding the workers as the same task has to be carried by the workers again and again.

To eliminate these problems, we propose a direct communication channel between workers and requesters through which workers can update the requesters on their progress on the task given. If the requester is not satisfied with the result or wants to provide additional parameters for the task to be done, it can easily do that by broadcasting the message to all the workers working on the particular task.

This can be achieved if there is some way in which the entities involved can have a secure and private channel providing continuous communication with each other. The nodes can collaborate and exchange feedback throughout the process which would increase the efficiency and quality of the result.

The miners can oversee the process and can interfere as and when required. The miners would have information of the task allotted to the workers and can detect if there is some malicious attempt to sabotage the process. If it detects some malicious activities, it can discard the process and can initiate the process from the start.

Our primary focus revolves around communication between the worker nodes. The requesters and miners are part of the system and the same architecture can be used for them but for the implementation we will keep worker nodes at the center.

Our overall contribution for our research work summarized as follows:

- The nodes involved keep exchanging messages with each other and keep updating their progress to each other.
- The overall system is more collaborative where workers can work without interference from other nodes.
- The wastage of workers' efforts is minimized which could lead to better efficiency and quality of result.
- The reduction in expenditure by the requestors and reduced scope for ambiguity in case of conflicts.

## 2. Related work

Dapeng Wu [4] finds shortcomings in MCS systems in terms of security and privacy. To tackle the issues mentioned he develops a system which provides dynamic credibility to the participants of the system using which one can evaluate how much trust could be put on to that particular participant. Overall it provides a good mechanism for crowd sensing systems but does not talk much about how the incentives for the work done would be rolled out. This paper [5] tries to make the already existing MCS systems present in day-to-day services like ride hailing services more efficient by tackling problems such as free riding and false reporting. It also addresses that the privacy of the worker must be preserved even when collecting the incentives. Here, the architecture diagrams were clear and of great help but not much details were given on the actual implementation of the system. The concept of MCS is applied on to vehicular networks [6] which act as ad-hoc networks for which the basic security is provided by public key infrastructure-based authentication protocols, several models for this approach along with in depth theoretical knowledge is provided here but the focus here is on vehicular devices rather than mobile devices carried in hands. Participation based sensing systems [14] make today's systems such as recommendation systems for say movies, TV shows, etc. They provide a sensing pattern to collect and interpret information from the environment. The major concern for such systems is privacy as the users have to shell out their private information to get recommendations. Here again only the concern of privacy is discussed while achieving a particular task involving private data of users but there is no reward or incentive to be paid out to workers in such a system. The problem in this paper [2] states is that nowadays there are lots of tasks being solved through human intelligence by the use of crowdsourcing systems but the majority of them are centralized and also suffer from a single point of failure. In this paper a decentralized approach to a crowdsourcing system has been provided as well as performance and security issues like DDoS attacks discussed and ways to prevent them given. A novel approach [1] for block generation which is essential to develop for a decentralized platform working as a means of data collection and its processing. This solves the issue of fork and centralization. A trust system is also given which can help identify malicious entities and prevent them from harming the MCS system as a whole. Architectural diagrams along with mathematical and graphical representations for complete understanding and implementation are given.

## 3. Architecture diagrams of each phases

i. First pair describes how communication happens between the nodes.

ii. Second pair illustrates in what form of communication (form of blocks) (Fig. 1. Fig. 2. Fig. 3. Fig. 4).

### 3.1. Kafka

Kafka, developed by the Apache foundation is an open source, distributed streaming platform that allows for the development of real-time event-driven applications. It allows the development of applications that continuously produce and consume streams of data records. Kafka is a distributed platform which runs as a cluster that can span multiple servers or data centers. Other servers run Kafka Connect to continuously import and export data as event streams to integrate Kafka with your existing systems such as relational databases as well as other Kafka clusters. To let you implement mission-critical use cases, a Kafka cluster is highly scalable and fault-tolerant: if any of its servers fails, the other servers will take over their work to ensure continuous operations without any data loss. [7,10].

Kafka is built upon four core APIs:

1. Producer API: allows applications to produce (or make) the streams of data

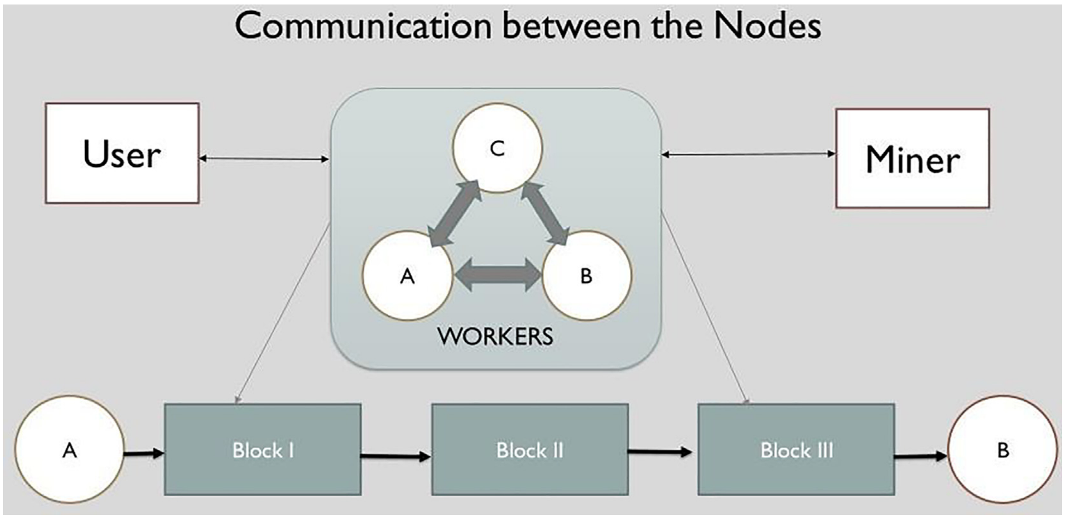


**Fig. 1.** Overview of communication between the nodes.

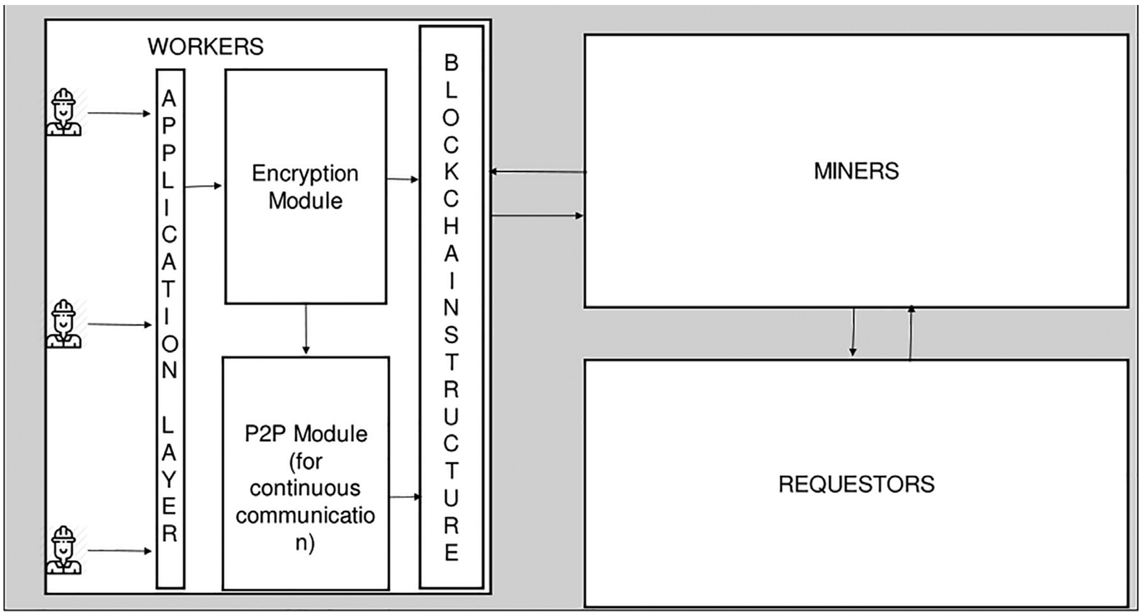


**Fig. 2.** Introduction of encryption layer and P2P channel for communication between the entities.

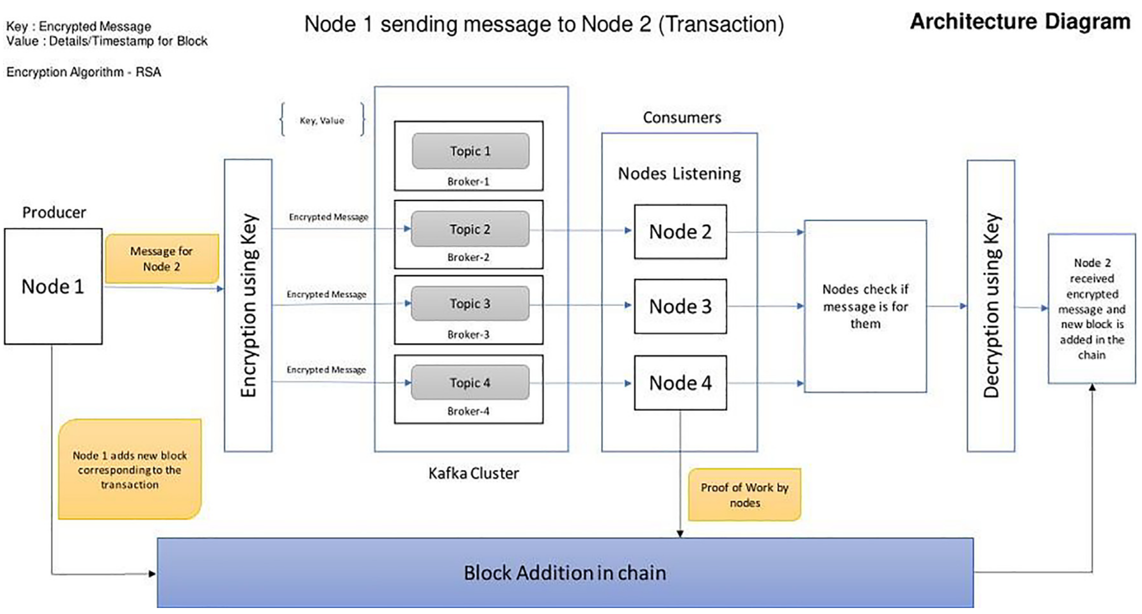


**Fig. 3.** Architecture diagram depicting how the messages are sent from one node to other.

2. Consumer API: subscribes to one or more topics (an ordered list of events) and listens to the data.
3. Streams API: analyzes, aggregates, or transforms the data consumed from topics in real time and produces the resulting streams to a topic.
4. Connector API: allows an integration service written once to be used in their own clusters to get the data source by just configuring it [7].

### 3.2. Peer-to-Peer network

The nodes in the system come together to form a peer-to-peer network. This network is the backbone of the proposed system. This eliminates the need of a central server in the system thus making it decentralized and distributed. The peer-to-peer network is formed using features and capabilities of Apache Kafka.

Each and Every node in the system behaves like a producer and a consumer. A producer writes the messages to the topic whereas a consumer consumes the messages from the topic. When sending a message node acts as a producer and sends the message to a topic. To receive the message a node acts as a consumer and consumes the message from topic.

There is a topic created for every node in the system. The topics can be distributed in different brokers who are coordinated by the zookeeper in Kafka. Also, to boost performance, different partitions of the same topic can be used by the nodes working on the same task or in the same area.

In Kafka, the messages are sent in Key/Value pairs. The nodes in the proposed system send messages as a key in encrypted form and

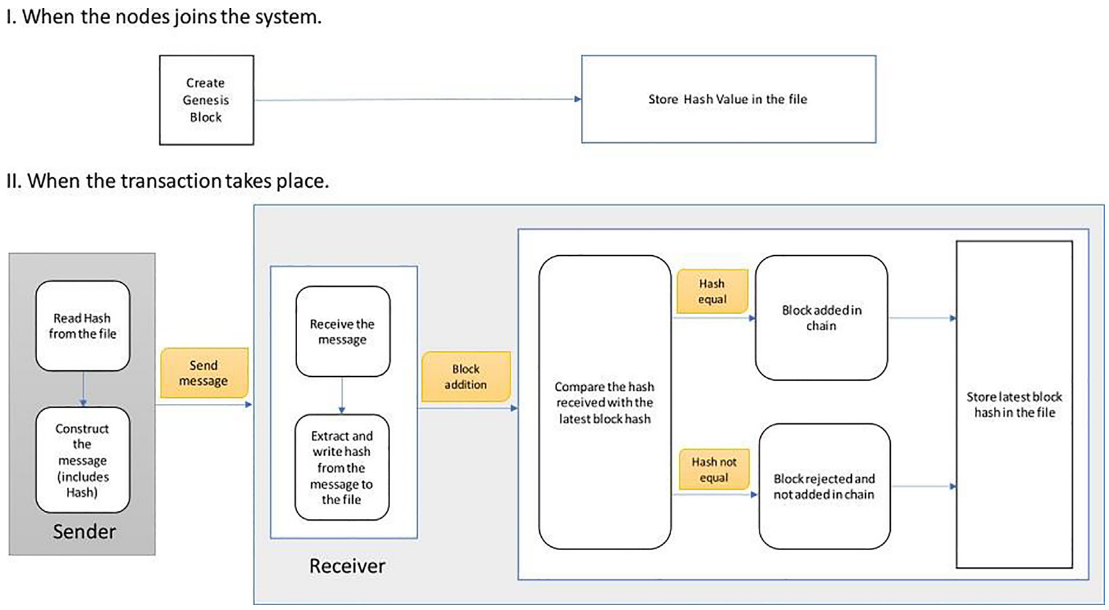

I. When the nodes joins the system.
Create Genesis Block
Store Hash Value in the file
II. When the transaction takes place.
Read Hash from the file
Construct the message (includes Hash)
Sender
Send message
Receive the message
Extract and write hash from the message to the file
Receiver
Block addition
Compare the hash received with the latest block hash
Hash equal
Hash not equal
Block added in chain
Block rejected and not added in chain
Store latest block hash in the file

**Fig. 4.** Addition of block corresponding to a particular message to the blockchain.

the details to add the block are sent as a value which follows Avro Schema. Apache Kafka presently supports three schema formats viz. Json, Avro and Protobuf for key or value. The messages are stored in a topic and can be read by the nodes anytime as offset associated with the topic is stored/tracked.

## 3.3. Encryption

Encryption is the process of converting the original information into an encoded form which cannot be simply interpreted by any other. It requires some decryption technique used to extract the original information from the encoded one. For our work we use the Rivest–Shamir–Adleman (RSA) encryption algorithm, which is an asymmetric cryptographic algorithm. This algorithm requires a public key and a private key to be present and uses both of them in the process.

The message to be sent is firstly encrypted by the public key of the receiver and then sent. The receiver then decrypts the encrypted message using its private key. In our work streams of data are broadcasted to all the nodes by the sender (producer) and each of the receiver nodes (consumer) checks if the message was meant for them or not. If yes, then that consumer tries to decrypt the message and displays it. When a message is sent it is visible to all the nodes in encrypted form in the form of byte arrays and the nodes can access that value to display it or anything else.

RSA is an asymmetric cryptographic technique for encryption and decryption of messages. Being an asymmetric technique there are two keys, namely public key (used for encryption and can be known to every-one) and private key (used for decryption and should be known to the message receiver only). A popular encryption program PGP (Pretty Good Privacy) generally used for encrypting emails is also based on RSA. In our case we would be sending messages from one node to another and hence to promise confidentiality an asymmetric approach has been chosen [11].

## 3.4. Generation of keys:

Two prime numbers p and q and taken since computationally it is very difficult to factor the product of two large prime numbers. [11] Then n is calculated as the product of p & q. $\phi(n)$ is calculated as the product of $(p - 1)$ and $(q-1)$. Another variable e is assumed any integral value between 1 and $\phi(n)$. Now we get the public key using the values of n and e.

The encrypted data becomes equal to $(\text{data})^e$ mod n.

For private key we required another value $d = (k * \phi(n) + 1)/e$ for some integer value of k. Decrypted data = $(\text{encrypted data})^d$ mod n [13].

## 3.5. Blockchain

A blockchain is a platform used for doing computation and other information technology tasks which is decentralized in nature and instead of depending upon one single authority to make all the decisions there are many authoritative domains available who are capable of doing the same and need not trust each other in taking part in any rational decision-making process. Here the chain of blocks is used to maintain and store the record of messages exchanged between the nodes. Each of the nodes here contains a chain of blocks where all the message records are stored. Since each node is having their own copy, in case some malicious block is added by a 3rd party or by a node itself to cheat or gain unfair advantage the others will get to know. This will be useful to solve any conflicts that may arise using the consensus mechanism[15].

When a message is sent by a node (producer) it contains the details which will be used to generate a block for the same. A new block added will contain the hash of its previous block. And this addition is done only after it satisfies the proof of work done. Since each of the nodes will be maintaining its own copy and the Kafka servers are distributed across regions so there will not be a single point of failure and neither will a malicious entity be able to alter any blocks which would get accepted by the others.

## 3.6. Block structure

The blockchain system consists of several blocks which are connected or linked together in such a way that each block stores the hash value or identifier of the previous block thus making a chain. The first block is often called genesis block and has null or zero value stored in place where other blocks store the hash value. Each block, along with its hash value and the previous blocks' hash value, has some data associated with it. To make it more secure the blockchain usually follows a distributed structure where the chain is distributed among several people who form a Peer-to-Peer network. The process of adding a block in the chain is time consuming as it has to be validated by every-one (or majority) in the system. This makes the system difficult to tamper with and contributes to its security.

In our work the concept of blockchain is used to maintain and store the record of messages exchanged between the nodes. Whenever a transaction takes place, it has to be recorded and this is done by adding a block corresponding to that transaction in the blockchain. A node sending a message to another node is an example of a transaction. The nodes in our system form a Peer-to-Peer network and can contribute in the process of block validation participating in Proof of Work (PoW) which refers to a consensus algorithm in blockchain systems. The blocks in our system are used to store and represent the transaction details. All the nodes of the system contain a chain of blocks. Since each and every node has its own copy of the chain it is difficult to tamper and add any malicious block in the system. In case any malicious node joins the network and tries to add a block to cheat or gain unfair advantage other nodes will detect the same. The malicious nodes can be removed and the chain is kept secured and intact. This mechanism would be useful to solve any conflicts that may arise in the process.

Data classes in Kotlin language can be used to generate a block structure or it can be represented by an ArrayList in Kotlin language. The type of the elements that can be present in the ArrayList are represented by a data class which has some data members providing a blueprint for the same. All the participants have their copy of the chain. In normal circumstances, i.e., without any malicious activity taking place, the blockchain present with each and every participant is updated. A new block is added whenever a message is exchanged between the nodes. The message sent by a node who behaves as a producer contains the details which are used to generate a block for the same. The node sending the message has the hash value of the latest block available in its chain.

The hash value of the latest block is included in the details used to create a new block and thus the block added will contain the hash value of its previous block. The block addition is done only after it satisfies the proof of work consensus mechanism. There will not be a single point of failure as each and every node maintains its own copy and the Kafka servers/brokers are distributed across regions. There will be no malicious entity who can alter the blockchain as it would not get accepted by the other nodes or participants.

### 3.7. Security and privacy

The system is made secure using encryption technique which makes sure that messages exchanged within the system cannot be read by someone outside the system as well as by the other nodes that are part of the system. The sender encrypts the message using the receiver's public key which is sent to the Kafka topic as a Key. The receiver consumes the message from the topic and checks if the message is meant for it. If the node detects that message is meant for it then it uses its private key to decrypt the message. The privacy of the nodes belonging to the system is ensured as the nodes participating in the process are validated when they try to create a new block by the nodes already present in the system. If some node which is not supposed to be a part of the system tries to send any messages to other nodes of the system then other nodes detect that the node is not part of the group and it is removed from the system. Thus, the privacy of the nodes is protected[16].

The nodes broadcast the messages only to the nodes who are part of the system. The node who is not part of the system cannot read those broadcasted messages as it needs to subscribe to the topic which it can only do if it is a part of the system. The privacy of the nodes in the system can be further enhanced if there exists a mechanism which does not reveal the identity of the node sending messages to other nodes. This can be done if the details to form a block contains information only about the receiver but not the sender. The implementation of such a mechanism is complex as there is an assumption that each and every node in the system has a pair of keys corresponding to the transaction taking place. For example, if node A sends a message to node B then it is assumed that node A has node B's public key to encrypt the message being sent. So, in the proposed system it is necessary to know who is the sender. The development of such a mechanism would be part of our future research.

### 3.8. Continuous feedback

Continuous feedback and communication is necessary for conflict-free running of the system as well as to provide fair incentives to the workers for their efforts. It increases the collaboration between the worker nodes which would help in increasing the quality and accuracy of the result. The continuous communication mechanism may result in a feedback model where nodes update their progress to other nodes and also evaluate the work done by them in real time. The nodes who behave like a Kafka consumer are always listening for any broadcasts from other nodes right from the beginning. This is achieved by subscribing to the topics on which the messages are produced. The messages are produced when a transaction takes place in a system or a node sends a message to another node. If the work done by some worker does not seem satisfactory, it can be notified right away just after the start of the work. Publish- subscribe model ensures that only a minimal amount of workers' efforts go wasted in the process. In traditional systems, after the complete work is done, it is submitted for review and then decided whether the work is fine and acceptable. But, this is time consuming and also it is unfair to the workers as they are being evaluated after completing the entire work for which the reward deserved may not be paid by the service provider.

This is beneficial for the requestors' side also. There might be numerous instances when some work has already been done by a worker and doing the same work again is wastage of time and effort for other workers. For example, say, there is some sensing to be done in a particular area and a worker did the sensing accurately. It is likely that after some time another worker might go to that area to accomplish the same task. This will result in service provider rewarding both the workers for the same task completed. This leads to an increase in expenditure which is to be borne by requestors. Thus, implementing a mechanism in the system through which nodes can communicate continuously with other nodes can solve these problems and can facilitate task completion in a shorter amount of time than the traditional systems.

### 3.9. Methodologies incorporation with results and discussion:

- A decentralized system is made consisting of nodes (producer and consumer) using Kafka.
- Each node has got some unique ID to be identifiable on the network.
- The nodes (consumers) are always listening to receive any broadcasts.
- To implement continuous feedback, the producer nodes broadcast the messages (key-value pairs) and the consumer nodes keep receiving whatever is broadcasted.
- Only the details required to add a block are visible to all consumer nodes and the message is sent in encrypted form. Only the designated node can interpret the encrypted message.

- The sender encrypts the message with the public key of the receiver. All the listening nodes check if the message is meant for them or not by extracting the receiver's ID. If it is for them then the message is decrypted using the receiver's private key.
- Whenever a message is sent a block should be added in a chain to record that transaction.
- The details to add the block (for ex- hash value of previous block) is sent to every node and all the nodes participate in proof of work (consensus).
- If all the nodes validate that the block to be added is valid by checking the hash value provided with the block within their own chain then the block can be added.

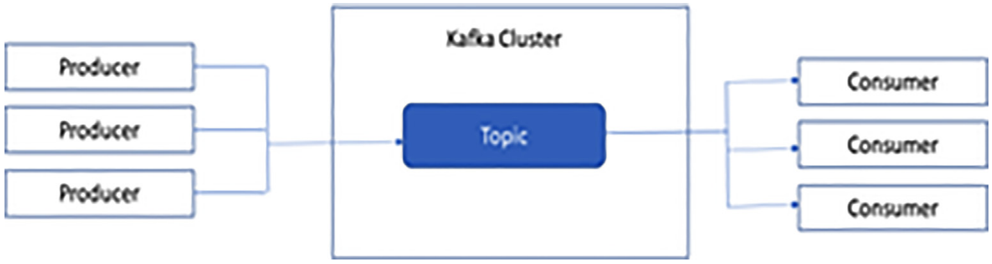
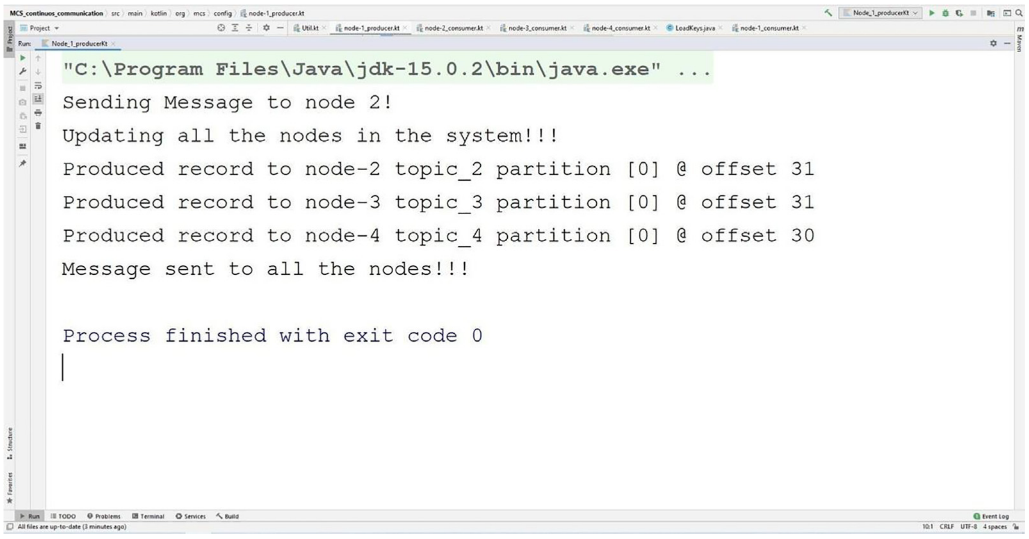
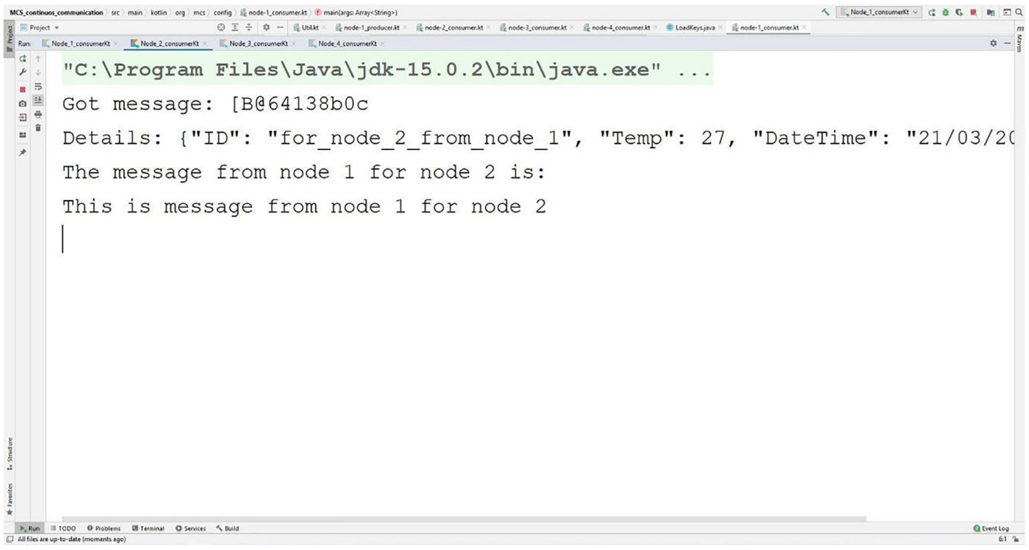


**Fig. 5.** Kafka architecture.



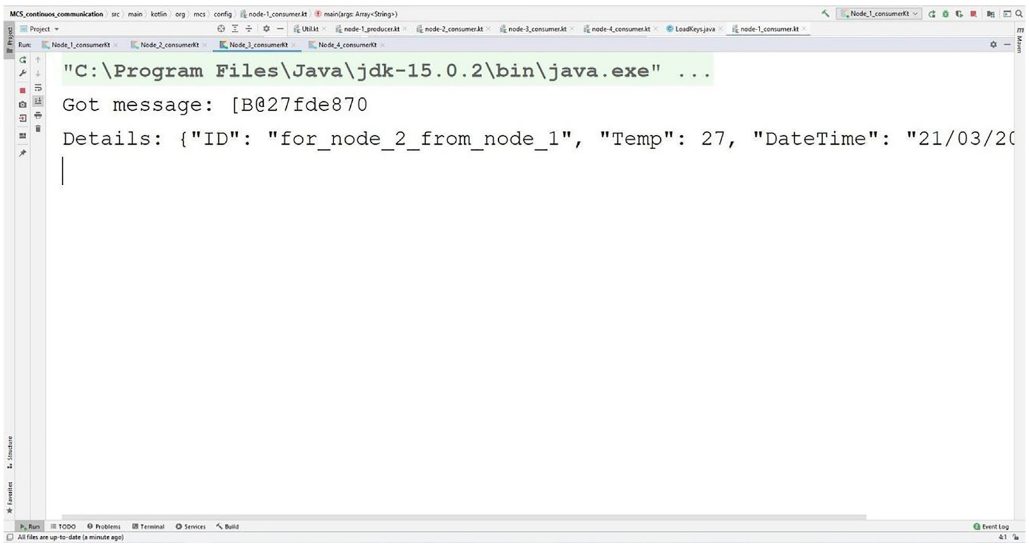
```
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" ...
Sending Message to node 2!
Updating all the nodes in the system!!!
Produced record to node-2 topic_2 partition [0] @ offset 31
Produced record to node-3 topic_3 partition [0] @ offset 31
Produced record to node-4 topic_4 partition [0] @ offset 30
Message sent to all the nodes!!!

Process finished with exit code 0
```

**Fig. 6.** Producer side.



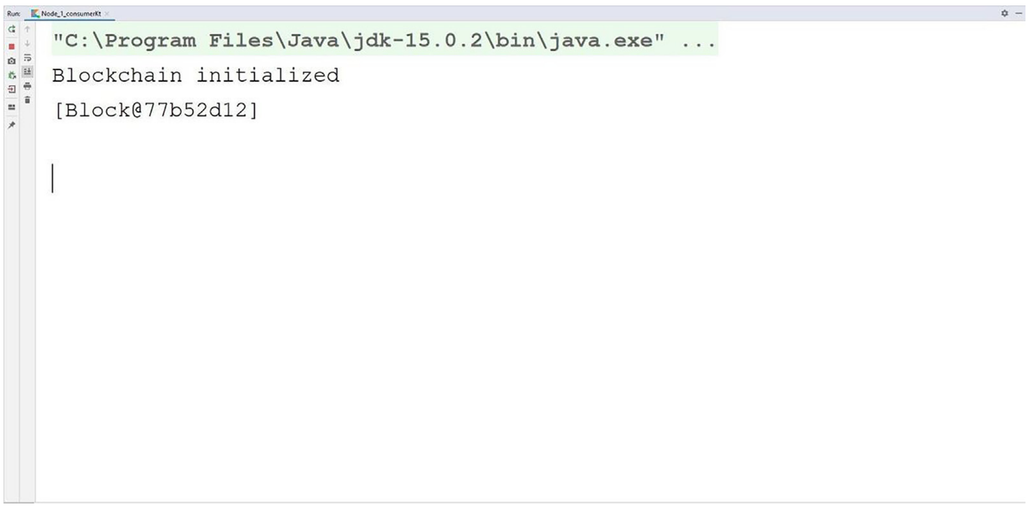
```
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" ...
Got message: [B@64138b0c
Details: {"ID": "for_node_2_from_node_1", "Temp": 27, "DateTime": "21/03/2(
The message from node 1 for node 2 is:
This is message from node 1 for node 2
```

**Fig. 7.** Consumer for whom the message is meant.



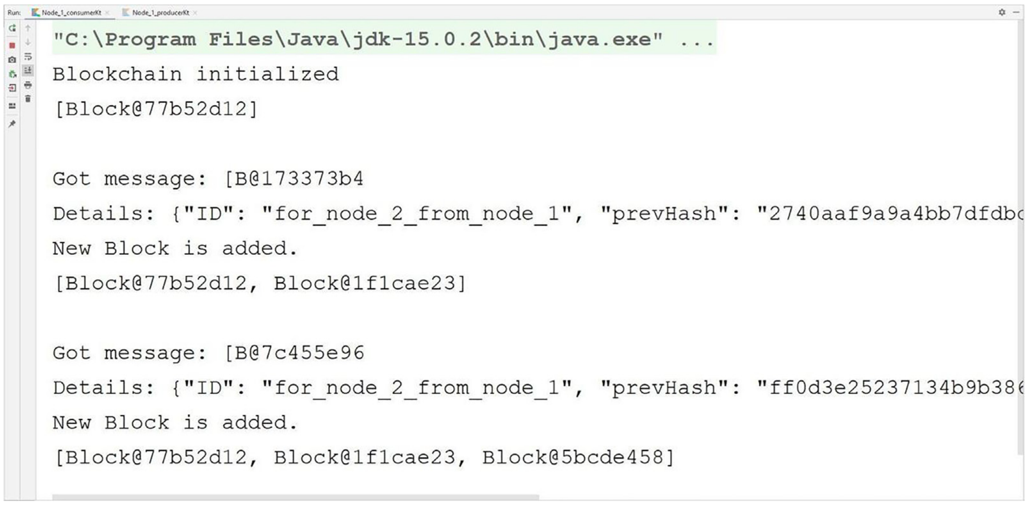
```
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" ...
Got message: [B@27fde870
Details: {"ID": "for_node_2_from_node_1", "Temp": 27, "DateTime": "21/03/2
```

**Fig. 8.** Consumer for whom the message is not meant.



```
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" ...
Blockchain initialized
[Block@77b52d12]
```

**Fig. 9.** Generation of the genesis block in blockchain.



```
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" ...
Blockchain initialized
[Block@77b52d12]

Got message: [B@173373b4
Details: {"ID": "for_node_2_from_node_1", "prevHash": "2740aaf9a9a4bb7dfdb
New Block is added.
[Block@77b52d12, Block@1f1cae23]

Got message: [B@7c455e96
Details: {"ID": "for_node_2_from_node_1", "prevHash": "ff0d3e25237134b9b38
New Block is added.
[Block@77b52d12, Block@1f1cae23, Block@5bcde458]
```

**Fig. 10.** Addition of blocks into the blockchain.

- Thus, the block is added in the chain which represents the transactions (message sent).
- If the validation by the nodes fails then the block is discarded and the authenticity of the chain is preserved

The proposed system uses Kafka for continuous communication. Kafka is an event/message streaming platform which works on publish/subscribe mechanism.

Kafka works by generating and storing events, which are added to the topics. Events are, as the name implies something which happened. A Kafka event or a message comprises of 3 fields and some additional fields:

1. Key: used to send encrypted messages.
2. Value: used to send block details
3. Headers: could be used to send metadata.

Additional fields include timestamp, offset, partition, etc.

Those who write events to kafka topics are referred to as producers and those who listen to (subscribe) these events are referred to as consumers [8,9].

Topics could be thought of as a directory in a file system in which events are present as files. A feature which makes data in Kafka fault tolerant is that topics can be replicated and put across different storage locations which can be used by individual smaller level programs which is easier to work on instead of a single big program. The messages sent to the topics can be consumed in ascending order or descending order i.e. consumers can read latest messages published or earliest messages published on to the topic. The consumers can consume the messages and transform it according to its needs (Fig. 5).

The overall architecture when a node sends a message to another node is depicted in Fig1.2. It consists of several modules comprising:

1. Kafka Module.
2. Encryption Module.
3. Block Module.

The steps followed for message exchange or communication between the nodes are as follows:

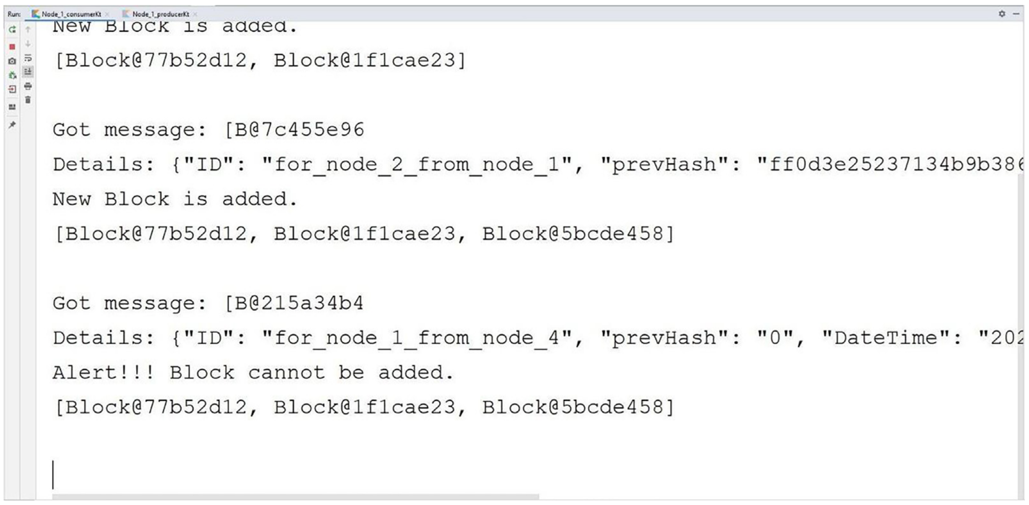
- A decentralized and distributed system is made using Kafka which consists of nodes which behave like producers and consumers

**Table 1**
Shows the time taken by producer nodes to send a message for the 1st time.

| Producer Side – Node sending message to other nodes | | | |
|---|---|---|---|
| S.No | Connect to Cluster(s) | Time to Send(s) | Total Time(s) |
| 1 | 17.99 | 8.76 | 26.75 |
| 2 | 17.07 | 8.88 | 25.95 |
| 3 | 18.16 | 8.55 | 26.71 |
| 4 | 17.56 | 8.47 | 26.03 |
| 5 | 17.49 | 8.64 | 26.13 |
| 6 | 17.05 | 9.78 | 26.83 |
| 7 | 17.25 | 8.46 | 25.71 |
| 8 | 17.99 | 8.53 | 26.52 |
| 9 | 17.26 | 9.51 | 26.77 |
| 10 | 17.45 | 9.26 | 26.71 |
| 11 | 17.63 | 9.1 | 26.73 |
| 12 | 17.54 | 8.95 | 26.49 |
| 13 | 17.88 | 9.05 | 26.93 |
| 14 | 17.99 | 8.89 | 26.88 |
| 15 | 17.77 | 8.78 | 26.55 |
| Average | 17.6053333 | 8.9073333 | 26.2126666 |

**Table 2**
Shows the time taken by consumer nodes in receiving/consuming the messages sent to them.

| Consumer Side –Node receiving message from other nodes | | | |
|---|---|---|---|
| S.No | Time to receive message (s) | Time to add block in chain after message received (s) | Total time (s) |
| 1 | 0.011 | 0.006 | 0.017 |
| 2 | 0.009 | 0.002 | 0.011 |
| 3 | 0.014 | 0.003 | 0.017 |
| 4 | 0.018 | 0.002 | 0.02 |
| 5 | 0.014 | 0.003 | 0.017 |
| 6 | 0.004 | 0.003 | 0.007 |
| 7 | 0.007 | 0.002 | 0.009 |
| 8 | 0.017 | 0.003 | 0.02 |
| 9 | 0.008 | 0.002 | 0.01 |
| 10 | 0.012 | 0.002 | 0.014 |
| 11 | 0.015 | 0.006 | 0.021 |
| 12 | 0.01 | 0.003 | 0.013 |
| 13 | 0.006 | 0.002 | 0.008 |
| 14 | 0.009 | 0.005 | 0.014 |
| 15 | 0.015 | 0.003 | 0.018 |
| Average | 0.0112666 | 0.0031333 | 0.0144 |



```
New Block is added.
[Block@77b52d12, Block@1f1cae23]

Got message: [B@7c455e96
Details: {"ID": "for_node_2_from_node_1", "prevHash": "ff0d3e25237134b9b38
New Block is added.
[Block@77b52d12, Block@1f1cae23, Block@5bcde458]

Got message: [B@215a34b4
Details: {"ID": "for_node_1_from_node_4", "prevHash": "0", "DateTime": "202
Alert!!! Block cannot be added.
[Block@77b52d12, Block@1f1cae23, Block@5bcde458]
```

**Fig. 11.** Alert when a non-authorized block tries to enter.

- Each node has some unique ID to be identifiable on the network.
- The consumer nodes subscribe to the topic and are always listening to receive any messages.
- To implement continuous feedback, the producer nodes broadcast the messages to the topics and the consumer nodes keep receiving the messages.
- The message which is sent as a key is in encrypted form and is visible as a byte array to all the nodes.
- The sender encrypts the message with the public key of the receiver. All the nodes listen to check if the message is meant for them or not by extracting the receiver's ID. If it is for them then the message is decoded using the receiver's private key

In our implementation below node 1 is sending a message to node 2 (Fig. 6) but the message is broadcasted to all the nodes and they are able to generate a block if the message is valid. Node 2 checks if the message is for it by extracting the ID field from the details sent as value and decrypts the message using its private key (Fig. 7) Other nodes can see the message in encrypted form and the details to add the block corresponding to the transaction but are not able to decrypt the message since they do not have the private key for decryption. Thus, they only see the block details and validate if the block should be added. SHA256 algorithm is used for hashing and the value will be added in the block.

The key and value can be modified accordingly to make the system more sophisticated and also to increase the functionality. The key/value pair can be seen here in Fig. 8. This refers to all the nodes who are listening but the message is not meant for them. At the end of the transaction, the block created with the help of the details sent as a value is added to the blockchain if it gets validated by all the nodes in the system. The block is discarded if the validation fails (Fig. 9. Fig. 10. Fig. 11).

Table 1 and Table 2 depicts the time taken by producer nodes to send a message for the 1st time and also the time taken by consumer nodes in receiving/consuming the messages sent to them. Initially, it will connect to the kafka cluster and it will calculate the time to send and receive the messages on both the sides (Producer and consumer side) and also to add block in the chain. Taking the mean of the time taken to send a message to other nodes after getting connected to other cluster we calculated the total time. Connect to cluster means, time taken for a node to get connect with the cluster of nodes.

The performance of the system can be measured with the help of following two graphs:

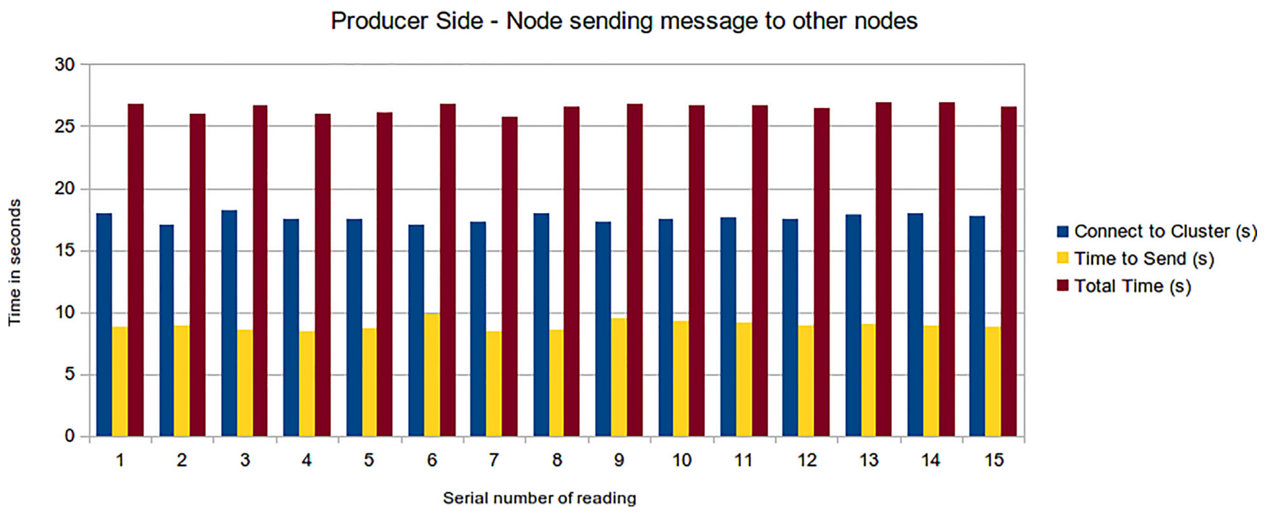


**Fig. 12.** Producer side.

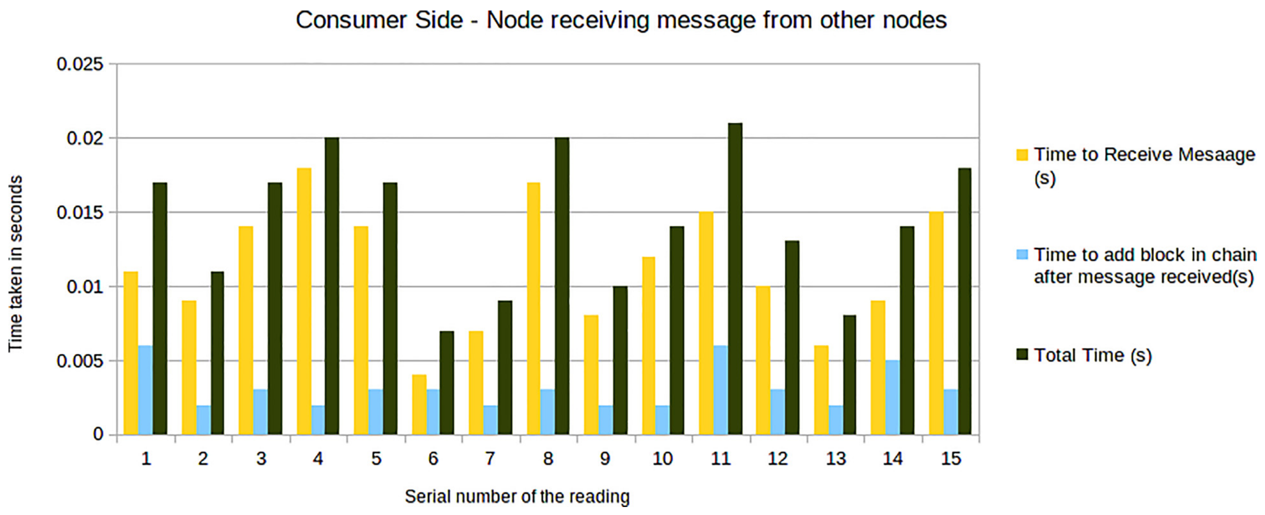


**Fig. 13.** Consumer side.

The Fig. 12 shows the reading taken at the producer side and acts as an indicator showing the time it takes to produce and send the first message. For each of the readings there are three columns: the blue bar represents the time in seconds it takes to connect to the Kafka cluster, the yellow bar represents the time taken to send the message after getting connected to the cluster and the brown bar depicts the sum of the former two. As it can be seen the time taken in each of the readings is near constant (Fig. 13).

Mean time to connect to the cluster Σ (time taken to connect to cluster)/15 = 264.08 s/15.

= 17.605 s.

Taking the mean of the time taken to send a message to other nodes after getting connected to the cluster, we get the value as: Σ (time taken in sending a message after connecting to cluster)/15 = 133.61 s/15 = 8.907 s.

There are some variations when measuring the time taken to receive the messages and add the blocks corresponding to them to the blockchain at the receiver's side as evident from the second chart. Most of the time taken is due to the time taken to receive the message and on the other hand the time taken to add the blocks to the blockchain is pretty less, less than half of the time taken to receive the message in most of the cases.

Now, the mean of time taken to complete the message receiving process = Σ (total time taken in message receiving and block addition)/15 = 0.216 s/15 = 0.0144 s.

## 4. Conclusion

We have presented here a MCS system which is decentralized in nature and supports continuous feedback between the nodes for the work being done which keeps them in loop throughout the process. Decentralization is capable of solving some of the problems faced in traditional approaches and the feedback mechanism implemented in the system aims to solve those problems at a more refined level. This in turn impacts incentive compensation and aims to enhance the reward system prevalent in traditional systems. The focus of this paper is on implementing the continuous feedback mechanism integrating it with an encryption layer to add an extra layer of security. We did not give any details on implementing the reward system but the concepts stated here might be helpful and may contribute to refine the traditional reward systems without any drastic changes. To further enhance this system, we might consider developing a mechanism which makes the system more secure and the information being exchanged more privacy friendly. In current implementation the nodes can see the transaction details and can easily find out who is the sender and the receiver. The identity of the nodes can be kept secret thus protecting their privacy in the system which may consist of thousands of nodes.

## Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

[1] Feng W, Yan Z. MCS-Chain: decentralized and trustworthy mobile crowdsourcing based on blockchain. Futur Gener Comput Syst 2019;95:649–66. doi: https://doi.org/10.1016/j.future.2019.01.036.

[2] Li M, Weng J, Yang A, Lu W, Zhang Y, Hou L, et al. CrowdBC: a blockchain-based decentralized framework for crowdsourcing. IEEE Trans Parallel Distrib Syst 2019;30(6):1251–66. doi: https://doi.org/10.1109/TPDS.2018.2881735.

[3] Feng W, Yan Z, Zhang H, Zeng K, Xiao Y, Hou YT. A survey on security, privacy, and trust in mobile crowdsourcing. IEEE Internet Things J 2018;5(4):2971–92. doi: https://doi.org/10.1109/JIOT.2017.2765699.

[4] Wu D, Fan L, Zhang C, Wang H, Wang R. Dynamical credibility assessment of privacy-preserving strategy for opportunistic mobile crowd sensing. IEEE Access 2018;6:37430–43. doi: https://doi.org/10.1109/ACCESS.2018.2847251.

[5] Wang Y, Li Y, Chi Z, Tong X. The truthful evolution and incentive for large-scale mobile crowd sensing networks. IEEE Access 2018;6:51187–99. doi: https://doi.org/10.1109/ACCESS.2018.2869665.

[6] Lu Z, Liu W, Wang Q, Qu G, Liu Z. A privacy-preserving trust model based on blockchain for VANETs. IEEE Access 2018;6:45655–64. doi: https://doi.org/10.1109/ACCESS.2018.2864189.

[7] Aung T, Min HY, Maw AH. "Coordinate Checkpoint Mechanism on Real-Time Messaging System in Kafka Pipeline Architecture," 2019 International Conference on Advanced Information Technologies (ICAIT), Yangon, Myanmar, 2019, pp. 37-42, DOI: 10.1109/AITC.2019.8921392.

[8] Wu H. "Research Proposal: Reliability Evaluation of the Apache Kafka Streaming System," 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Berlin, Germany, 2019, pp. 112-113, DOI: 10.1109/ISSREW.2019.00055.

[9] Wu H, Shang Z, Wolter K. "Performance Prediction for the Apache Kafka Messaging System," 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), Zhangjiajie, China, 2019, pp. 154-161, DOI: 10.1109/HPCC/SmartCity/DSS.2019.00036.

[10] Wu H, Shang Z, Wolter K. "TRAK: A Testing Tool for Studying the Reliability of Data Delivery in Apache Kafka," 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Berlin, Germany, 2019, pp. 394-397, DOI: 10.1109/ISSREW.2019.00101.

[11] Nisha S, Farik M. RSA public key cryptography algorithm – A review. Int J Sci Technol Res 2017;6(7):187–91.

[12] Zhang X, Yang Z, Sun W, Liu Yu, Tang S, Xing K, et al. Incentives for mobile crowd sensing: a survey. IEEE Commun Surv Tutorials 2015;18:1. doi: https://doi.org/10.1109/COMST.2015.2415528.

[13] Michael Calderbank, The RSA Cryptosystem: History, Algorithm, Primes, University of Chicago, August 20, 2007.

[14] Liu R, Liang J, Gao W, Yu R. Privacy-based recommendation mechanism in mobile participatory sensing systems using crowdsourced users' preferences. Futur Gener Comput Syst 2018;80:76–88.

[15] M. Arulprakash, R. Jebakumar, Towards developing a Block Chain based Advanced Data Security-Reward Model (DSecCS) in mobile crowd sensing networks, Egyptian Informatics Journal, Volume 23, Issue 3, 2022, Pages 405-415, ISSN 1110-8665, https://doi.org/10.1016/j.eij.2022.03.002.

[16] Arulprakash M, Kamal Aditya. and Aishwarya Manisha. "QR-Code scanner based vehicle sharing. ARPN Journal of Engineering and Applied Sciences 2018;13(10):3441–8.