

PoHMChain: A Sustainable Cryptocurrency Platform with Proof of Human Mobility Consensus Algorithm

D.M.H.P. Dissanayake
M.S.W. Salgado
S.N.S. Wickramasinghe
2025



PoHMChain: A Sustainable Cryptocurrency Platform with Proof of Human Mobility Consensus Algorithm

D.M.H.P.Dissanayake
Index No: 20000431

M.S.W.Salgado
Index No: 20001541

S.N.S.Wickramasinghe
Index No: 20002122

Supervisor: **Prof. Kasun De Zoysa**

June 2025


Submitted in partial fulfillment of the requirements of the
B.Sc.(Honours) Bachelor of Science in Software Engineering
Final Year Project



Declaration

I certify that this dissertation does not incorporate, without acknowledgment, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

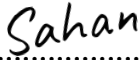
Candidate Name: D.M.H.P. Dissanayake

Signature of Candidate 

Date: 28/06/2025

I certify that this dissertation does not incorporate without acknowledgment, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.


Candidate Name: M.S.W. Salgado

Signature of Candidate 

Date: 28/06/2025

I certify that this dissertation does not incorporate, without acknowledgment, any material previously submitted for a degree or diploma in any university and to the best of my knowledge and belief, it does not contain any material previously published or written by another person or myself except where due reference is made in the text. I also hereby give consent for my dissertation, if accepted, be made available for photocopying and for interlibrary loans, and for the title and abstract to be made available to outside organizations.

Candidate Name: S.N.S Wickramasinghe

Signature of Candidate 

Date: 28/06/2025

This is to certify that this dissertation is based on the work of

Mr. D.M.H.P. Dissanayake

Mr. M.S.W. Salgado

Mr. S.N.S Wickramasinghe

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Supervisor's Name: Prof. Kasun De Zoysa

.....

Signature of Supervisor

.....

Date

Co-Supervisor's Name: Dr. Asanka Sayakkara



.....

Signature of Co-Supervisor

2025-06-29

.....

Date

Abstract

Along with the positive impact, blockchain technology also has negative impacts on the environment, climate, and energy consumption. Algorithms like Proof of Work was tend to consume a great amount of energy, hence alternative methods such as Proof of Stake, Proof of Space have come into play.

This dissertation focused on the newly introduced human mobility-based consensus algorithm known as Proof of Human Mobility and its practical implementation. This sustainable, proof-based, leader selection algorithm uses human mobility as the trust factor. A desktop application was implemented for blockchain connectivity, along with a mobile application, which is used to generate tickets proving human mobility. Furthermore research and evaluations were carried out in the following areas. In regard to choosing a better short-range communication method, the practical limitations of the integration of Bluetooth, Near Field Communication, and Wi-Fi Direct is discussed, and an alternative method to address those issues is given for the implementation. To make the Proof of Human Mobility (PoHM) more scalable evaluation were carried out on 8 different scenarios and calculated the average decreasing percentage and transaction per second metrics. Brotli compression with a validator group mechanism had the lowest average decreasing percentage of 26% which showcased the best strategy among all scenarios to work with, when the number of nodes increases. To identify the optimal sensor set to detect fraudulent movements evaluation were conducted on 10 sensor combinations using Random Forest Classifier with captured smartphone sensor data. The GPS + Accelerometer combination achieved 91.73% accuracy, selected as the optimal set for its higher accuracy and lower power consumption.

Acknowledgement

We extend our heartfelt gratitude to our supervisors, Prof. Kasun De Zoysa and Dr. Asanaka Sayakkara, for their invaluable guidance, support, and advice throughout this project. Despite their busy schedules, they have been instrumental in keeping us on the right track and arranging necessary resources for our work.

We would also like to thank Dr. Randil Pushpananda and Ms. Amali Perera, the coordinators of the SCS4223 Final Year Project in Software Engineering at the University of Colombo School of Computing, for their unwavering support and guidance.

Our sincere appreciation goes to all the participants who generously contributed their time and expertise to the data collection process. Their input was crucial to the success of this research. Last but not least, we are grateful to our parents and colleagues for their continuous support, encouragement, and motivation. Their unwavering belief in us has been a driving force behind our progress.

Table of Contents

| | |
|---|------------|
| List of Figures | vii |
| List of Tables | ix |
| List of Abbreviations | ix |
| Chapter 1 - Introduction | 1 |
| Background Knowledge | 1 |
| Project Aim | 1 |
| Research Gap & Research Questions | 2 |
| Objectives | 3 |
| Scope of the Project | 3 |
| Chapter 2 - Literature Review | 5 |
| Background | 5 |
| Proof of Human Mobility | 5 |
| Initial Block | 6 |
| Leader Selection | 7 |
| Forks/Sub-networks | 8 |
| Evaluation of PoHM | 9 |
| Chapter 3 - Research Methodology | 11 |
| Software Development Perspective | 11 |
| Research Perspective | 11 |
| Chapter 4 - Project Design | 15 |
| Functional Requirements | 15 |
| Non-functional Requirements | 15 |
| System Architecture | 16 |
| System Modeling | 17 |

| | |
|--|-----------|
| Chapter 5 - Implementation | 24 |
| Blockchain Node Implementation | 24 |
| Blockchain Peer-to-Peer Network Implementation | 30 |
| Leader Generation Implementation | 31 |
| Block Transferring Implementation | 32 |
| Block Compression Implementation | 34 |
| Validator Group Implementation | 34 |
| Location Generation Implementation | 37 |
| User Verification and Ticket Generation Implementation | 38 |
| Chapter 6 - Evaluation and Results | 39 |
| Strategies to make PoHM more scalable | 39 |
| Short-range Communication Method | 44 |
| Optimal Sensor Set for detecting Fraudulent Movements | 54 |
| Chapter 7 - Conclusion | 66 |
| Conclusion about the Research Problem | 66 |
| Limitations | 66 |
| Future Work | 68 |
| Chapter 8 - Peer Evaluation | 70 |
| References | 72 |

List of Figures

| | | |
|----|---|----|
| 1 | Development Flow | 11 |
| 2 | High Level Architecture | 16 |
| 3 | Use Case Diagram | 17 |
| 4 | Performing Transaction | 18 |
| 5 | Ticket Send/Receive | 19 |
| 6 | Private Key Setup | 19 |
| 7 | Location generation and travel | 20 |
| 8 | Verify Ticket | 20 |
| 9 | Transaction States | 21 |
| 10 | Ticket States | 22 |
| 11 | Location Generation and Travel | 22 |
| 12 | Validate tickets | 23 |
| 13 | Block Transferring | 33 |
| 14 | Block transferring after validator group | 33 |
| 15 | Comparison of TPS in each Mode | 42 |
| 16 | Bluetooth interface | 45 |
| 17 | Bluetooth Discover Comparison | 47 |
| 18 | Discovery plot for each run | 48 |
| 19 | Discovery plot for average devices discovered | 48 |
| 20 | Wi-Fi Direct interface | 49 |
| 21 | Android 8.1 Wif-Direct | 50 |
| 22 | User variability | 51 |
| 23 | Common IT knowledge | 51 |
| 24 | Bluetooth vs Wi-Fi Direct preference | 52 |
| 25 | Screenshots of the mobile app: (a) Interface of the application; (b) Movement type selection screen, with options for walking, running, cycling, and travel by vehicle; (c) Recording interface, displaying derived sensor data and a save button to export the CSV file. . . . | 55 |
| 26 | GPS vs Accelerometer | 56 |

| | | |
|----|--|----|
| 27 | Accelerometer vs Gyroscope | 57 |
| 28 | Accelerometer vs Magnetometer | 57 |
| 29 | Scatter plot of average_gps_speed vs. average_acceleration_magnitude for GPS & Accelerometer sensor combination | 65 |

List of Tables

| | | |
|----|--|----|
| 1 | Number of transfers Before and After Validator Group | 37 |
| 2 | Normal block transferring evaluation | 40 |
| 3 | GZIP Compression for block transfers evaluation | 40 |
| 4 | Snappy Compression for block transfers evaluation | 40 |
| 5 | Brotli Compression for block transfers evaluation | 41 |
| 6 | Validator Group Block transfers evaluation | 41 |
| 7 | Validator Group & GZIP Compression for Block transfers evaluation | 41 |
| 8 | Validator Group & Snappy Compression for Block transfers evaluation | 42 |
| 9 | Validator Group & Brotli Compression for Block transfers evaluation | 42 |
| 10 | Average Power Consumption per hour | 46 |
| 11 | Power Consumption per hour | 46 |
| 12 | Discovery time vs No. of devices discovered | 47 |
| 13 | Average Power Consumption per hour Wi-Fi and Quickshare | 50 |
| 14 | Bluetooth and Wi-Fi Direct - Survey | 52 |
| 15 | Quickshare - survey | 52 |
| 16 | Average of Short-range communication traffic | 53 |
| 17 | Accuracy of Different Sensor Combinations | 60 |
| 18 | Threshold Values for Fraud Detection | 61 |
| 19 | Battery Drain for One Hour of Continuous Sensor Operation at 3.7 V | 63 |
| 20 | Total Battery Drain for One Hour of Continuous Sensor Combina- tions at 3.7 V | 63 |

List of Abbreviations

PoHM Proof of Human Mobility

NFC Near Field Communication

POW Proof of Work

POS Proof of Stake

IB Initial Block

TPS Transactions Per Second

Chapter 1 - Introduction

Background Knowledge

In recent years, the blockchain has become the transforming force behind different industries (Wüst & Gervais 2017). The introduction of Bitcoin (Warmke 2024) has gained great popularity in society, with the promise of decentralization, security, and transactions without third parties. But in recent years, the traditional consensus algorithms (Proof of Work (POW), Proof of Stake (POS)) started to face criticism due to energy consumption, centralization, and not being sustainable.

Therefore, researchers have been exploring new and sustainable consensus algorithms, that can not only address sustainability but also empower the primary strengths of blockchains. One such approach proposed by Kongahage et al. (2022) is the use of human mobility as a trust factor for block creation. Since natural human behavior is used within this algorithm, it might provide health benefits and rewards for being active. Since the Smart Mobility and Mobile Crowd Sensing (MCS) technologies have been developed and enhanced over the past years, these technologies might be useful to track human mobility patterns. As blockchain technology is adopted in different industries, Karger et al. (2021) and Huang et al. (2020) researched how adopting blockchain affects smart mobility and MCS. The research of Kongahage et al. (2022) has discussed a mechanism that involves human mobility and has introduced a novel blockchain algorithm.

Project Aim

- Make blockchain technology sustainable by replacing the POW algorithm
- Make blockchain technology not biased towards wealthier people
- Incentivize people for engaging in physical activities
- Provide a base platform for future researchers who will be involving with PoHM algorithm

Research Gap & Research Questions

Research Gap

PoHM is a novel algorithm introduced to the literature where any real-world implementation has not yet been implemented. So the practicality of the algorithm has a research gap. As a result of novelty, there are several other research gaps, such as finding out the best short-range communication methods for the verification process of the PoHM algorithm, finding different strategies for making a more scalable blockchain technology using PoHM, and discovering methods to detect fraudulent nodes in the network in the ticket generation process. There are few short-range communication methods on a mobile device like Bluetooth and Near Field Communication (NFC). Therefore, determining a better communication method is crucial for a better outcome of the application. As the literature shows, scalability will decrease when the number of nodes in the network increases, resulting in fewer Transactions Per Second (TPS). Therefore, finding ways to make the PoHM more scalable is needed. Some users can use cars or any other vehicles for the part where a typical user is intended to walk or run. So users who use vehicles can gain higher benefits than regular users. Ways to detect these types of fraudulent movements are also required when implementing an unbiased algorithm.

Research Questions

1. Which short-range communication methods can be used for location-based verification in the PoHM consensus algorithm?
2. What strategies can be used to make the PoHM algorithm more scalable?
3. What is the optimal sensor set can be used to detect fraudulent movements in ticket generation process of PoHM?

Objectives

- Implement blockchain technology using PoHM algorithm
- Implement a mobile application and desktop application where users will join the blockchain network and do verifications
- Implementing a crypto wallet where users will be able to send/receive cryptocurrency
- Research and Integration of a better short-range communication method for the ticket-generation Process
- Research and Implementation of a more scalable blockchain application
- Research and Integration of an optimal sensor set to detect fraudulent movements in the network when the ticket generation happens

Scope of the Project

In Scope

- A novel Blockchain technology using PoHM Consensus algorithm
- Mobile application and Desktop application integrated with the implemented blockchain that users will utilize for verification and ticket generation
- Cryptocurrency wallet where users can send/receive the crypto they earned through the block generation process or transaction
- Integration of a better short-range communication method for the ticket-generation process
- Implementation of a more scalable blockchain application
- Integration of optimal sensor combination to the mobile application for detect fraudulent movements in the network when the ticket generation happens

Out Scope

- Connecting the Mobile Application directly to the blockchain network
- Crypto exchanging/trading facilities or usage of other crypto coins within the implemented platform

Chapter 2 - Literature Review

Background

Blockchains have some characteristics that make them unique and secure. The main characteristic is decentralization, which means there is no central authority to govern and control the transactions that happen within the network, hence providing some independence. This helps to be fair to everyone in the network. The consensus algorithm is the heart of blockchains. The process model of the consensus process consists of accountant selection, block addition, and transaction confirmation (Fu et al. 2021). The sustainability of the blockchain also lies in the implementation of the consensus algorithm. Algorithms such as PoW require miners to be involved in a mining process to verify the created blocks, which requires high computational power. These resource-consuming algorithms raise many concerns about their carbon footprint and climate change (Gallersdörfer et al. 2020). Even though the concept of blockchain is popular nowadays, it also faces some critical security issues. Some of them are double spending, denial of service, 51% attack, fork issues, etc (Lin & Liao 2017). Even though algorithms such as PoW, PoS, Proof of Activity exist each of these algorithms consists of a few or several issues related to them.

Proof of Human Mobility

Human mobility means a human moving from one place to another. Thus, capturing human movement means confirming that a person arrived at some location from a another location within a given amount of time. The proposed algorithm leverages human mobility as the trust factor for the block creation. The private key and public key pair are used to identify a node in the network. Kongahage et al. (2022) proposes a community-based verification system which is fair and competitive, with nodes using each other to confirm their existence at the place in according to a predetermined protocol. A node's likelihood of being chosen as the leader mostly depends on its capacity for human mobility. A location is any

easily accessible area close to a node, usually a public area. Every node will have a list of locations, and a single location can be shared by several user nodes. A ticket can be identified as an opportunity to be chosen as a leader. A node has to demonstrate their identity to the network and participate in constant mobility in order to create a ticket. Mobility, or moving between sites, is a continuous activity where a user can run or walk. The likelihood of node being chosen as the leader increases with the number of locations it visits and the tickets it generates.

Initial Block

The relevant node who gets selected as the leader should create the Initial Block (IB) for location generation. A single ticket is generated by an IB, which is made up of a certain number of tickets. To make a single ticket, a collection of tickets is consumed in this instance. But the ticket pool will run out if this loop keeps on, which will put the entire system in a deadlock. To prevent that kind of a scenario, same ticket is picked several times into the IB with an upper bound in order to prevent this scenario. The upper bound prevents the same set of tickets from being chosen every time. The tickets list in IB is hashed using Merkle hashing according to Kongahage et al. (2022). After generating a location, the node should arrive at the generated location, where a new ticket is created when the node's presence is confirmed. The tickets hash (Merkle hash of the tickets) of the IB is hashed in order to get a pseudo-random number N , which is then used to choose a location. Since the value N is arbitrary and unpredictable as a randomly chosen fixed number of tickets will always result in a random and unpredictable hash. Here, the hash function/random number generator uses the tickets hash, $\text{HashMerkle}(\text{tickets})$ as its seed.

$$N = \text{Hash}(\text{Hash}_{\text{Merkle}}(\text{tickets})) \quad (1)$$

Before the user gets to the place, the N is unknown to the other nodes to ensure that a node's location remains private from other users hence important to privacy and security. However, the verifier will confirm that N is randomly

generated using the tickets in the IB during short-range location verification. To select a place from the location list, one can utilize equation 2's random value, which ranges from 0 to the number of locations L . To prevent the same place from being chosen again, the location of the current node is omitted.

$$index = N \bmod (L - 1) \quad (2)$$

As soon as a location is produced, the $hash(N)$ and $hash(locationlist)$ are broadcast to the network. Broadcasting $hash(N)$ is used to confirm the value of N . After the node initializes IB, a $hash(locationlist)$ is transmitted to ensure that the location list is not changed (Kongahage et al. 2022).

The prover and verifier transmit data via a short-range communication channel. Mobile devices may communicate across short distances using a variety of methods, including Near Field Communication (NFC) and Bluetooth.

Leader Selection

As soon as a new block is introduced to the chain, every node participates in the leader selection process and chooses the leader for that block. The leader is chosen based on the available local blockchain ledger. The chosen leader should be unpredictable and random, and is consistent throughout all of the network's nodes. The tickets from the latest m blocks of the blockchain will be chosen by the nodes in order to choose a leader. The implementation may affect the value m . Based on the ticket's timestamp, the chosen tickets from each block are placed in chronological order, starting with the earliest. After a validator initializes a ticket, the timestamp that is being considered here is appended. As seen in equation 3, the random number generator function generates a random number (RN) by utilizing the leader's public key, K_{n-1}^{pub} , as the seed and the preceding block hash, H_{n-1} . The values of H_{n-1} and K_{n-1}^{pub} are random since the RN generation occurs as soon as a new block is added to the chain.

$$RN = hash(K_{n-1}^{pub} + H_{n-1}) \quad (3)$$

$$index = RNmod(L) \quad (4)$$

When choosing the leader for the n th block, an index between 0 and the length of the chosen tickets L may be obtained using equations 4. Using the *index*, a ticket will be selected from the tickets list and the next leader will be K^{pub} , the ticket that was chosen. The leader node itself will discover that it is the next leader as each node in the network goes through this predetermined leader selection procedure. The winning ticket's block will then be selected by the leader from the local block store.

By combining a set of transactions from the transaction pool with the Merkle hash of the transactions $h_{\text{Merkle}}(\text{transactions})$, the leader completes the block.

The hash from the previous block, *previousHash*, is appended and serves as the next block's pointer. The block finalization time is represented by the addition of a timestamp. At last, the hash value is updated. The leader broadcasts the completed block to the network, where every node verifies the block and any available transactions (a process that other blockchains often follow). The new block will be added to their local blockchains after successful verification (Kongahage et al. 2022).

Forks/Sub-networks

When sub-networks split off and operate independently on their own blockchains, forks may happen. The chains will become inconsistent as a result of this. In a scenario where there are partitions, the partition with fewer nodes will produce fewer tickets than the partition with more nodes. As a result, the partition with fewer nodes will have to wait longer to get the amount of tickets required to start a block. When compared to the blockchain of the division with a higher number of nodes, this will lower the block generation pace and shorten the blockchain's length. The blockchain ledger with the longest length (maximum height) is the one chosen according to the Fork selection rule. As soon as the pace at which criminal users create tickets exceeds that of honest users, this system will become insecure. These users could make a fork with the longest possible length in such

a scenario.

Evaluation of PoHM

Kongahage et al. (2022) implemented a simulation from p5.js to simulate human mobility in an area of $800m \times 800m$ for 30 minutes.

The assumptions made by Kongahage et al. (2022) should be analyzed since these assumptions greatly affect practical implementation.

1. Only direct paths exist between locations
2. Data exchange time between the prover and verifier is negligible
3. Average human speed: $2ms^{-1}$
4. No nodes are dropped or added
5. All locations are common to all nodes

Kongahage et al. (2022) was able to implement a simple yet limited implementation of the PoHM algorithm with an increasing number of locations. For each number of locations, Kongahage et al. (2022) has taken an increasing number of nodes and calculated the average number of tickets. Kongahage et al. (2022) has observed that when the number of nodes increases, the number of tickets generated increases. Furthermore, Kongahage et al. (2022) observed that when the number of locations increases, the number of tickets will decrease. Also, two deadlock conditions were introduced.

1. $numberofnodes \leq numberoflocations$: nodes may have to wait indefinitely for another node to visit the same location.
2. Common Location's number of users: If the location is shared by many users, there is a high chance of being verified, and if a location is shared by only a single user, that node will wait indefinitely.

Kongahage et al. (2022) implemented a Naive coin implementation by replacing the available PoW with PoHM. Based on the implementation Kongahage et al.

(2022) discovered that if there are n nodes in the network, then the message-passing complexity is $O(n^2)$. Kongahage et al. (2022) guaranteed that all nodes maintain identical ledgers by outputting the results of the hash values of the first 50 blocks of 10 nodes, hence Consistency guaranteed. Kongahage et al. (2022) depicts that the block interval and size affect the throughput / TPS. Also, TPS reduces as the number of nodes increases.

Kongahage et al. (2022) has described how the agreement, termination, and validity properties are maintained. Since honest nodes select the same leader consistently, the agreement property is maintained. Since the PoHM always decides the next leader when the block is added and algorithm termination happens with a final decision, the termination property is maintained. A mining node always generates the appended block, and it is verified by the network and secured by signatures, hence validity property is maintained.

Chapter 3 - Research Methodology

Design Science Research (DSR) methodology was utilized to develop this product-based research project. The initial problem to solve was identified as lack of practical implementation for PoHM algorithm. Then the new platform was implemented, followed by a thorough evaluation to confirm that the product solved the initial problem. After the evaluations, improvements were made to improve the solution.

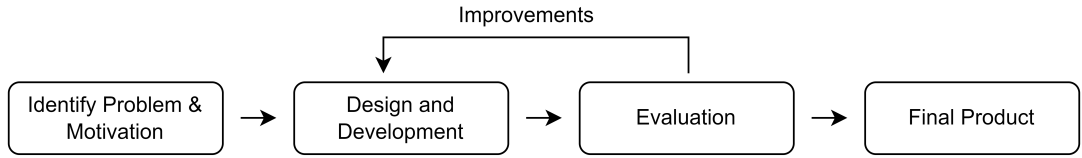


Figure 1: Development Flow

Software Development Perspective

According to the algorithm introduced by Kongahage et al. (2022), the initial version of the blockchain was developed within a mobile application and a desktop application. With the use of readings taken from the evaluations, parallel research components were added to the system, and further evaluations were performed in the upcoming versions. Finally, the selected versions were merged to produce the final product.

Research Perspective

Q1: What short-range communication methods can be used for location-based verification in the PoHM consensus algorithm?

Different short-range communication mechanisms, such as Bluetooth, NFC and Wi-Fi Direct exist. Primarily, it is necessary to identify the most popular and most common short-range communication mechanisms that are available on mobile phones to make them more fair and more applicable to the nodes and users

on the network and to keep them active. Then each of them was evaluated using a few criteria.

- Discovery speed of device detection
- Device detectability and prerequisites for being discoverable
- Power consumption of mobile phone for each short-range method
- Availability of each short-range communication method in single area
- User preference of short-range communication method
- Find the most suitable short-range communication method to implement

Q2: What strategies can be used to make the PoHM algorithm more scalable?

According to the initial evaluations performed in the PoHM algorithm, it demonstrated that in a condition where the transactions per block and number of evaluated blocks are constant, it takes longer to reach the finality of the blockchain when the number of nodes in the network increases. Therefore, the TPS decreased. To address this issue of scalability, this research worked around two strategies, which are

1. Using a selected validator group's acceptance when new blocks are appended to the network instead of the total network's acceptance.
2. Using suitable compression methods when blocks and related data are transferred.

The validator group-based validations discussed in delegated POS in Saad & Radzi (2020) and the analysis of the Tendermint blockchain network in Cason et al. (2021) shows how singling out a selected set of network participants affect the increase in scalability. Proposed stake and voting-based validator group selection methods in the literature could be adapted to the PoHM algorithm if the number of tickets generated by a network user is considered. Therefore, adapting such a group-based validator mechanism was evaluated and performed. After adopting

the PoHM algorithm with the mentioned validator mechanism, scalability metrics like TPS and decreasing percentages were calculated to find the effectiveness of this adaptation.

Block transferring speed in the network also results in the finality time of a new block being appended to the blockchain. Therefore, using techniques like compression, the size of the payload can be reduced, so that the bandwidth needed for transferring will be reduced, which results in faster block transfers. Compression methods like GZIP discussed in Rauschert et al. (2004), Brotli introduced in Alakuijala et al. (2018), and Snappy discussed in Lu & Hua (2019) result in different speeds of compression. Therefore, the effect of using compression methods in block transferring was compared using the mentioned compression methods. Resulting time difference it makes to the network when reaching the finality condition was evaluated.

Q3: What is the optimal sensor set can be used to detect fraudulent movements in ticket generation process of PoHM?

In the PoHM consensus algorithm, it's really important to detect fraudulent nodes, especially when they're providing their location and activity. During the ticket generation process, detecting and preventing fraudulent node movements are especially important when users may pretend to be running or walking but are instead traveling by vehicle or bicycle. The most suitable method for detecting fraudulent nodes involves analyzing three distinct approaches, a manual plotting method using sensor measurements dataset, a machine learning model-based evaluation of sensor combinations, and a threshold-based way to detect fraudulent movement in the system. These approaches utilize measurements from GPS (average speed), accelerometer (movement intensity), gyroscope (rotational speed), and magnetometer (magnetic field variance) to differentiate between legitimate human movements (walking, running) and fraudulent movements (cycling, vehicle travel). The manual plotting approach using manually collected dataset explored visual patterns in sensor data, the machine learning method Random

Forest identified the optimal sensor combination based on classification accuracy, and the threshold method set statistical boundaries (mean \pm standard deviation) for legitimate movements. These methods were chosen for their ability to process real-time sensor data from a mobile application, making them effective for fraud detection in PoHM.

Chapter 4 - Project Design

Functional Requirements

- Synchronize with the blockchain network and have the most updated local copy each time
- Perform transactions through the wallet
- View transaction history
- View added tickets to the network
- Send tickets generated from the mobile application to the blockchain node
- Download a shareable Private key file from the node for initial mobile setup
- Generate locations to travel for generating tickets
- Human mobility verification with another user for ticket generation

Non-functional Requirements

- **Security:** A Blockchain application should have security as a core requirement
- **Scalability:** Application should be scalable when the number of nodes increases
- **Decentralization:** Application should be decentralized without the need of being managed by a central entity

However, due to the trilemma of blockchain systems described in Werth et al. (2023) shows that all three aspects above are not satisfiable at a time, because improving one property will reduce effects of another property.

System Architecture

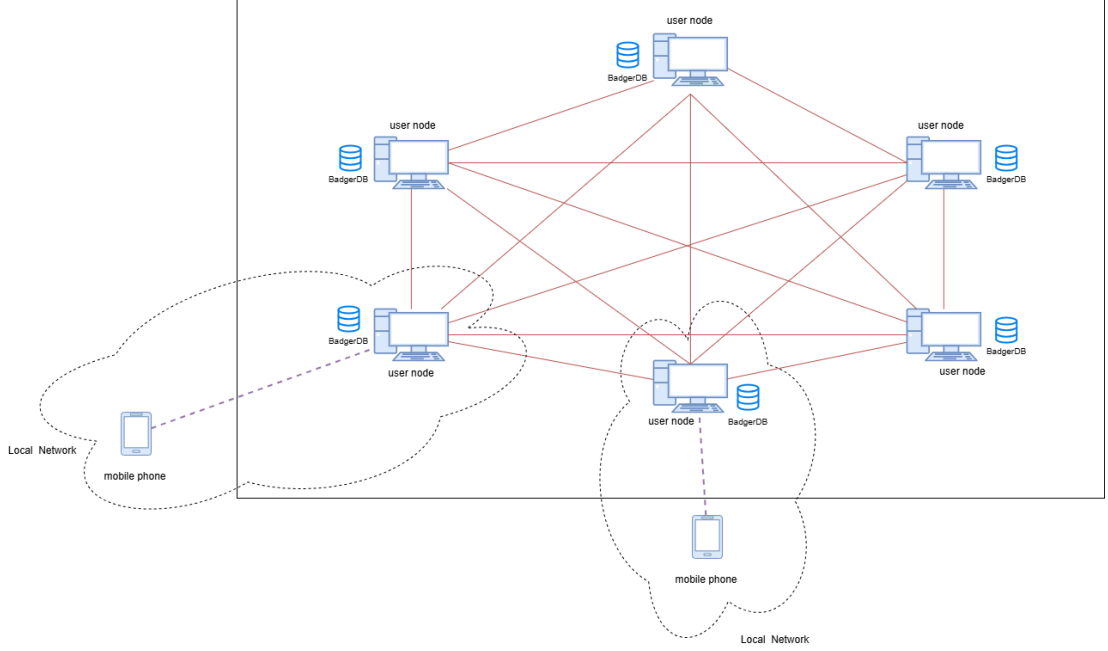


Figure 2: High Level Architecture

Each node has a BadgerDB instance to store the blockchain-related information. Nodes are connected using a p2p network. Initial node advertisement was made through the default bootstrap nodes provided by Libp2p. For relaying purposes, when nodes are behind a NAT, an EC2 instance was configured. Mobile application and node communicate and share tickets through a local web server opened on the node.

System Modeling

Use Case Diagram

The figure 3 depicts the main use cases of the PoHM implementation. The sytem is used by the 2 main actors, who are mobile application users and blockchain node users.

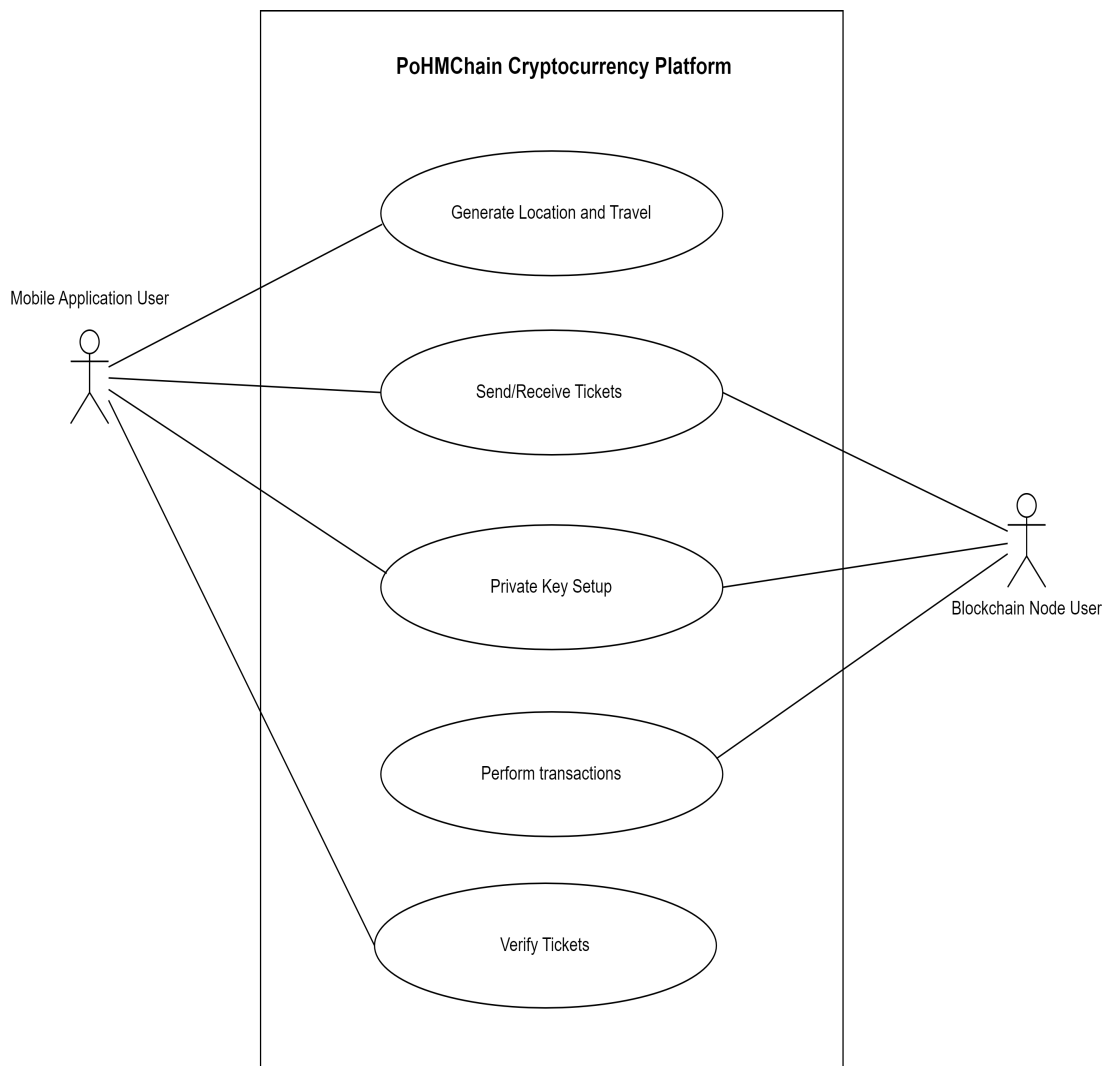


Figure 3: Use Case Diagram

Activity Diagrams

Figure 4 shows how a user would send a transaction through the blockchain node and how the blockchain nodes process a transaction after receiving it before adding it to the block.

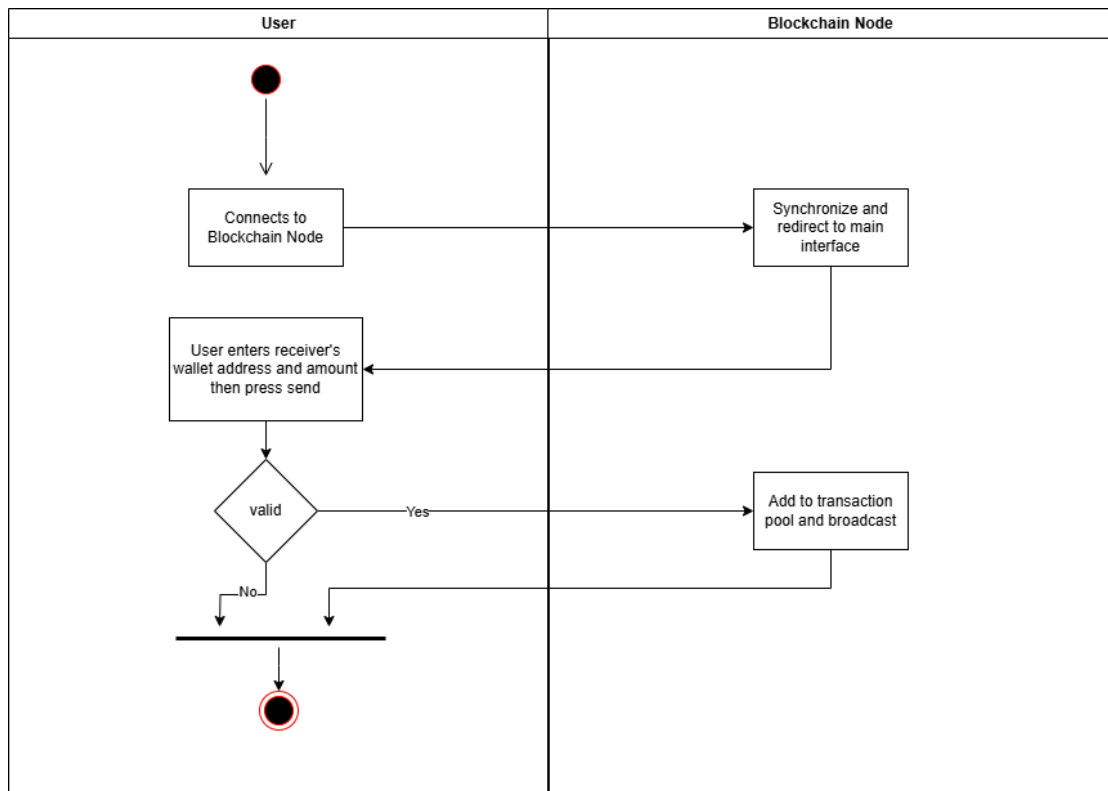


Figure 4: Performing Transaction

Figure 5 represents the flow of transferring tickets from mobile to the desktop environment. Since the tickets are collected with the mobile device, users should initiate this process to make the tickets accessible and available for the blockchain to perform.

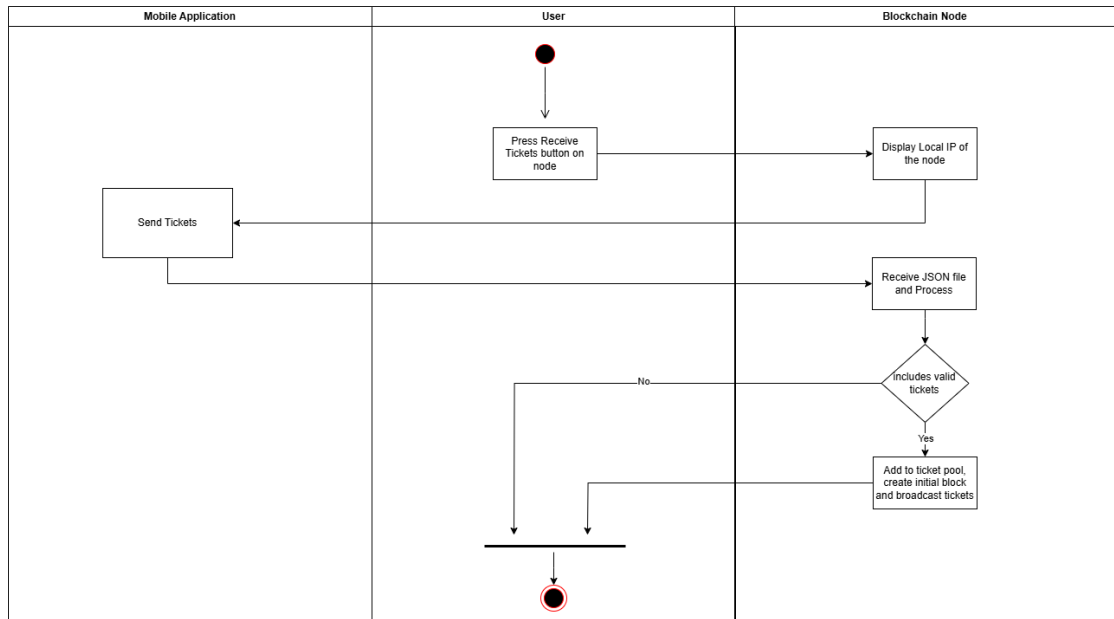


Figure 5: Ticket Send/Receive

Figure 6 represents the setup process of the primary key generated in the desktop application into the mobile.

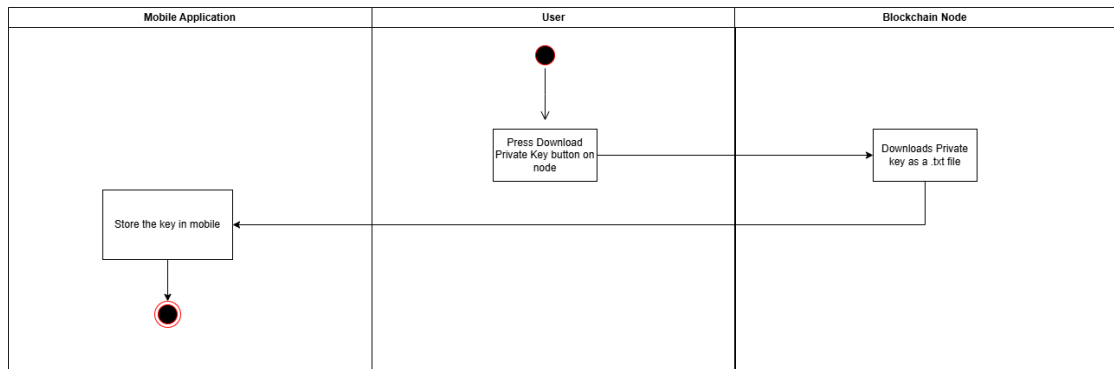


Figure 6: Private Key Setup

Figure 7 depicts how the travel process works on the user side.

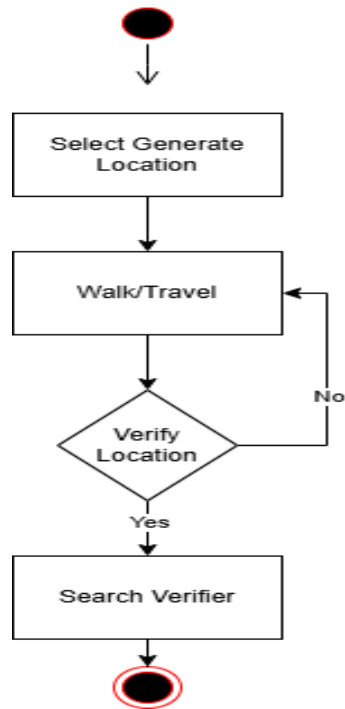


Figure 7: Location generation and travel

Figure 8 depicts how the verification process happens in the ticket generation.

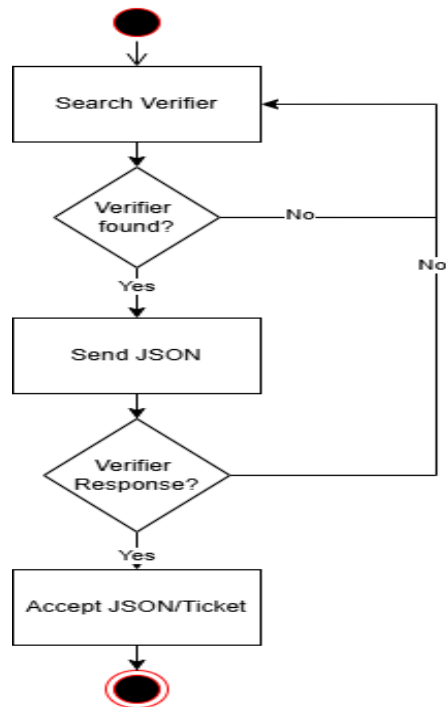


Figure 8: Verify Ticket

State Diagrams

Figure 9 shows how the transaction goes through different states when the transaction is performed or transaction is received from broadcast.

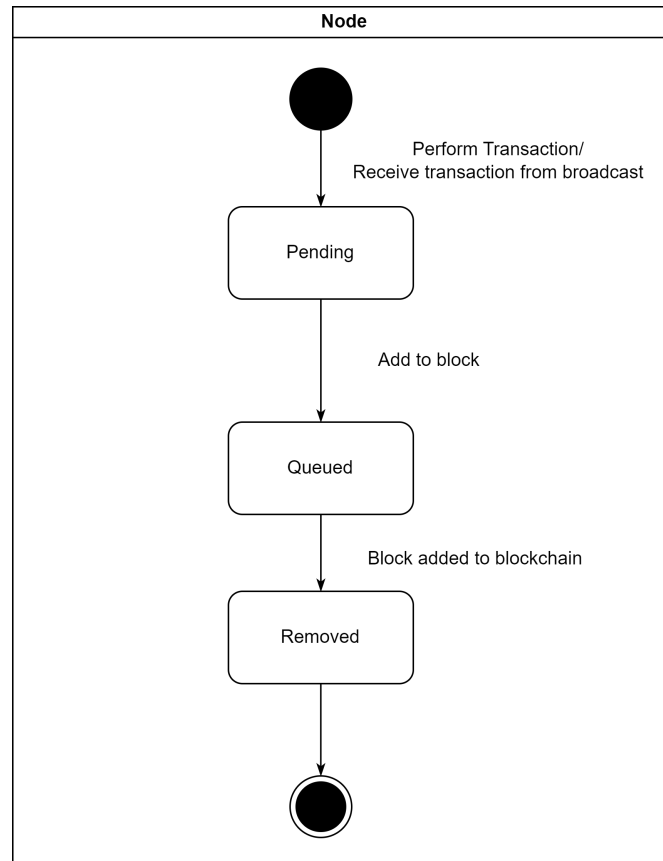


Figure 9: Transaction States

Figure 10 shows how the ticket goes through different states when the ticket is processed from JSON or ticket is received from broadcast.

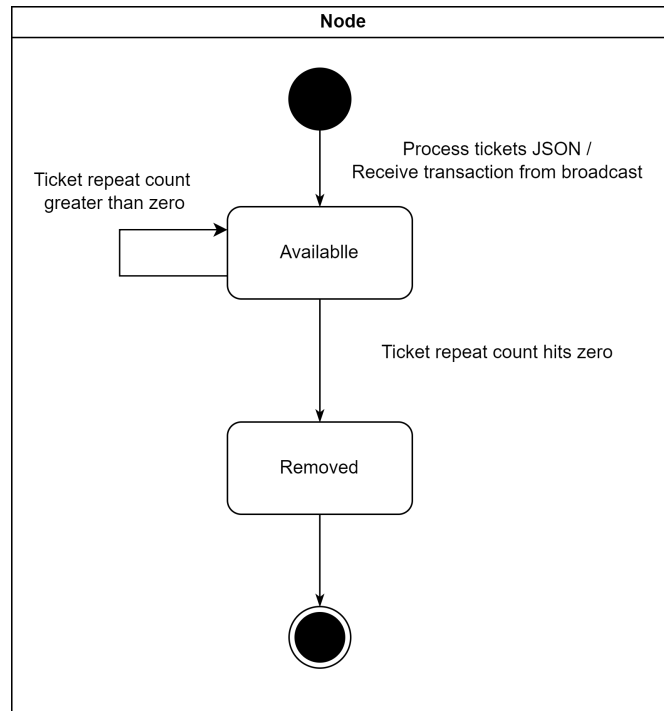


Figure 10: Ticket States

Figure 11 shows how the location generation process goes through different states until it reaches the verified state.

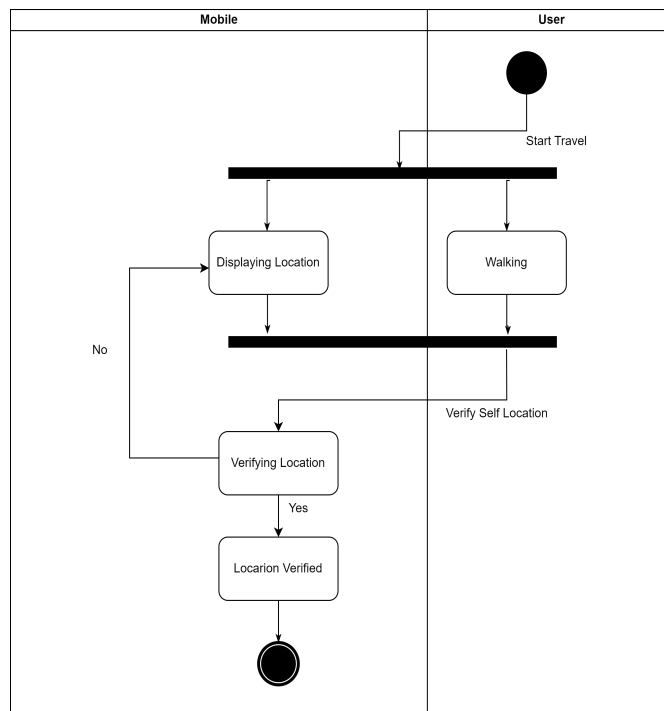


Figure 11: Location Generation and Travel

Figure 12 shows how the state transitions occur when two users validate for ticket generation.

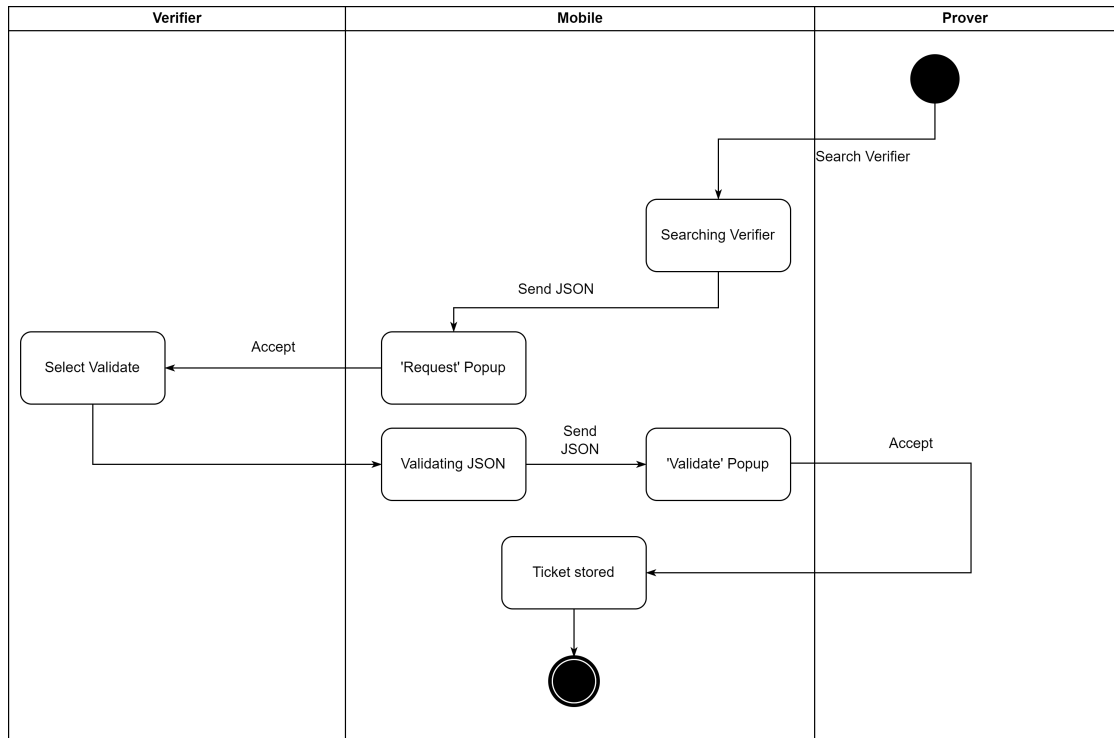


Figure 12: Validate tickets

Chapter 5 - Implementation

Blockchain Node Implementation

The implemented blockchain node consists of multiple significant components that are blocks, transactions, transaction pool, unspent transaction set, tickets, ticket pool, ticket server, and wallet.

Blockchain is initially created with a genesis block. Every node is initiated with the same genesis block to keep the starting point of the blockchain consistent across the network. A Blockchain Block has the following structure:

| | | | |
|-------|------------------------|--------------------|---|
| Block | Timestamp | : int64, | } |
| | Hash | : []byte, | |
| | PrevHash | : []byte, | |
| | Transactions | : []*Transaction, | |
| | Tickets | : []*Ticket, | |
| | MerkleRootTransactions | : []byte, | |
| | MerkleRootTickets | : []byte, | |
| | PublicKeyLeader | : []byte, | |
| | Height | : int64 | |

A timestamp is used to track when the block is added to the blockchain network. Hash is created by taking **SHA256** hashing all the content of the block itself. PrevHash is the hash of the previous block in the blockchain. Transactions are either **Coinbase transactions** or transactions performed by node users. A Coinbase transaction is a reward transaction given by the network to the leader of a particular block for the effort to creating tickets. Our implementation consists of **TRANSACTIONS_PER_BLOCK** set to 5, which means that a block consists of a maximum of 5 transactions, and the coinbase transaction reward is set at 100.00 coins. Tickets are either the **GenesisTickets** defined or user-generated tickets that were uploaded to the network through the ticket server. Merkle Roots

of both transactions and tickets are taken after constructing a Merkle tree to ensure the integrity of the block sent through the network, as even a small change to the ticket or transactions could change those Merkle root values, which results in a change of the hash of that block. This Hash value is reconstructed and compared with the assigned Hash at every node before they are added to their local blockchains. The PublicKeyLeader is the public key of the block leader who proposed that block. Height in each block represents the chain height at a certain moment. The Genesis block will have a height of 1, and it will increment after each block added. This height is also used for chain updates and synchronization between the network.

Overall Genesis block consist of zero transactions, genesis tickets, **Genesis-PrevHash** which is []byte("genesis_prev_hash") **GenesisPublicKey** which is []byte("genesis_public_key"), timestamp of 1, height of 1 and Merkle roots of both transactions and tickets. Genesis tickets are defined as below:

```
[
    {
        Timestamp:          1,
        ID:                  [ ]byte("gensis-ticket1"),
        BlockID:             [ ]byte("genesis-block1"),
        PublicKeyProver:     config.FIRST_LEADER_PUBLIC_KEY,
        PublicKeyVerifier:    [ ]byte("gensis-publicKeyVerifier1"),
        RepeatCount:         config.TicketRepeatCount,
    },
    {
        Timestamp:          2,
        ID:                  [ ]byte("gensis-ticket2"),
        BlockID:             [ ]byte("genesis-block2"),
        PublicKeyProver:     config.FIRST_LEADER_PUBLIC_KEY,
        PublicKeyVerifier:    [ ]byte("gensis-publicKeyVerifier2"),
```

```

RepeatCount:      config.TicketRepeatCount,
},
{
Timestamp:        3,
ID:                [ ]byte("gensis-ticket3"),
BlockID:           [ ]byte("gensis-block3"),
PublicKeyProver:   config.FIRST_LEADER_PUBLIC_KEY,
PublicKeyVerifier: [ ]byte("gensis-publicKeyVerifier3"),
RepeatCount:      config.TicketRepeatCount,
},
]

```

The initial **TicketRepeatCount** of each ticket is defined as 50, which means a single ticket can be used a maximum of 50 times when creation of initial blocks for tickets. About **FIRST_LEADER_PUBLIC_KEY** will be discussed later in the leader generation implementation.

Transaction structure is defined as follows :

$$\text{Transaction} \left\{ \begin{array}{ll} \text{ID} & : []\text{byte}, \\ \text{Inputs} & : []\text{TxInput}, \\ \text{Outputs} & : []\text{TxOutput} \end{array} \right\}$$

$$\text{TxInput} \left\{ \begin{array}{ll} \text{ID} & : []\text{byte}, \\ \text{Out} & : \text{int}, \\ \text{Signature} & : []\text{byte}, \\ \text{PubKey} & : []\text{byte} \end{array} \right\}$$

$$\text{TxOutput} \left\{ \begin{array}{ll} \text{Value} & : \text{float64}, \\ \text{PubKeyHash} & : []\text{byte} \end{array} \right\}$$

Transactions consist of 3 main parts. Transaction ID taken by **SHA256** hashing the transaction inputs and outputs. Transaction inputs are always formed from the previous transaction outputs of that specific user. Unspent transaction set (UTXOSet) manages all transactions of the chain and handles the discovery of spendable transaction outputs for a specific user. When a transaction initiates, a set of outputs of previous transactions addressed to the current node which the combined value of which is equal to or greater than the sending amount, will be picked. ID in transaction input means the transaction ID of those picked transactions. Out is the index of those transactions in the transaction set. The signature is taken by signing the transaction with the Elliptic Curve Digital Signature (ECDSA) algorithm. PubKey is the sender's wallet public key. Transaction output consists of a maximum of 2 entries. One is the sending value with the receiver's wallet address. This is mandatory. The second transaction output is addressed to self with the remaining value after the sending value. A Coinbase transaction includes a single transaction input with an empty ID and Pubkey, the out index is -1 since the transaction is created and awarded by the network at that moment.

Transaction output consists of a single entry as well, including the minting reward amount and the receiver's wallet address.

Once a user performs a transaction through the node's wallet, the transaction will be formed and sent to their own transaction pool first, then it will be broadcast to other nodes, where they upload their own transaction pools with this transaction as well. The transaction pool stores transactions until they are added to a block. Within that pool transactions go through multiple states defined in 9. A transaction is selected to process on a block based on the amount of that transaction. High-value transactions will get priority in getting selected and included in a block sooner.

Ticket structure is defined as follows:

$$\text{Ticket} \left\{ \begin{array}{ll} \text{Timestamp} & : \text{int64}, \\ \text{ID} & : []\text{byte}, \\ \text{BlockID} & : []\text{byte}, \\ \text{PublicKeyProver} & : []\text{byte}, \\ \text{PublicKeyVerifier} & : []\text{byte}, \\ \text{RepeatCount} & : \text{int}, \\ \text{PeerID} & : \text{peer.ID} \end{array} \right\}$$

The timestamp of the ticket is used in creating initial blocks and also in the new validator group creation logic. As **TICKETS_PER_BLOCK** threshold set to 3, each node will be created only when 3 or more ticket with repeat count not equal to zero of them. When tickets are used, the most recent repeat count non-zero tickets will be chosen for processing. ID is for uniquely identify a ticket. BlockID is the hash of the initial block associated with that ticket. This BlockID is used in the leader generation process, which will be discussed leader generation implementation section. PublicKeyProver is the public key of the one who proved is Human mobility through the mobile. PublicKeyVerifier is the public key of the one who verified, before mentioned prover's human mobility. As multiple tickets

are used in the creation of an initial block, which in turn creates a single new ticket, make tickets decay faster; a repeat count is given for each ticket, same as Kongahage et al. (2022) proposed. The PeerID of the prover is also stored in a ticket, which is used in the validator group implementation process.

The ticket server is implemented directly to handle ticket JSON files receiving from the mobile application. The connection happens through a local network on port 8081, where both the blockchain node and mobile application connect to the same network via the IP displayed on the blockchain node. Once connected, sent JSON files, including ticket lists, will be transferred to the node's UserTickets directory. A file watcher will be placed on this directory, so whenever there is any new file addition, this will be triggered. Once a certain file is processed, the SHA256 hash of that file's content and a processed flag will be stored in the DB so that the file won't be processed again. When the file is processing for each ticket, an initial block will be created with including the most recent 3 tickets in the block. The very first user-generated tickets will be associated with the initial block created from genesis tickets, which are defined before. Every node other than the defined first leader node can generate user-generated tickets for the pool until there are at least 3 user-generated tickets in the ticket pool. Once that condition is satisfied, the defined first leader can create the block as well as contribute to generating tickets by proving human mobility. Every other elected leader can participate in the network without any restriction. The created initial block will be saved locally for the process of leader election in the future. Then the ticket's BlockID will be assigned with the initial block's hash, and the ticket will be added to their own pool as well as broadcasted to others, so other nodes in the network also add this ticket to their local pools. Also, only tickets belong to the node itself will be permitted for this processing. In ticket pools, tickets go through a couple of states, which are mentioned in 10.

In the blockchain's main interface it includes a wallet functionality where users can add a wallet address and a valid amount and perform a transaction. Once they initiate the transaction it will process as mentioned above. Also, the blockchain

node will create 2 files which are `.networkConfig` and `.walletData` file to persist the wallet information and peer connectivity information. If any malicious edits are made on this, it will recreate these files, resulting in losing the previous account. Finally there will be 2 goroutines running on each node to make sure any new transaction or ticket added to the network is updated in its local pool as well.

Blockchain Peer-to-Peer Network Implementation

In the startup, each node creates a libp2p host using the `.networkConfig` file created. Then, a new distributed hash table will be created, and each node will make a connection with the default bootstrap nodes. After that, each node will advertise a rendezvous point called **PoHMChain** to the routing discovery and will look for peers who have advertised the same rendezvous point before and will try to make a connection with them. If a direct connection is possible, they will successfully make the connections; if not, in a case where NAT issues persist, a relay node, which is an AWS EC2 instance, will be utilized for relaying the connection between those nodes.

After a successful connection each node will subscribe to 3 pubsub topics, which are "general-channel", "transactions-channel", and "tickets-channel". The general channel topic is used mainly for block transfers, while the transactions channel and the tickets channel respectively used for transaction and ticket sharing between subscribed nodes. If a node is selected as a validator then they will self-subscribe to a new topic called "validator-channel," where all the validator group-related communication happens. Nodes will unsubscribe themselves if they are previously subscribed to the validator channel but not selected as a validator for the current validator routine. There will be two concurrent go-routings running on each node called `ManageEvents` and `HandleEvent`, where `ManageEvents` will be looking for any messages published on the pubsub channel and triggering relevant functionalities to update our node itself, and `HandleEvent` will make sure that whatever a node does locally, like adding tickets, performing transactions, adding blocks to be broadcasted to all the nodes subscribed to the above channel, making

it consistent across all the nodes in the network.

Leader Generation Implementation

After adding a new block to the chain every node will perform the leader generation process for selecting the next leader who proposes the block. Initially it is checked that what is the block height of that moment, and if it is 1, which means only the genesis block is present, that block is picked. If it has more than one but less than or equal to 3, only the blocks other than the genesis block will be picked. If there are more than 3 blocks in the chain then the most recent 3 blocks will be picked. This count is defined in the **LEADER_ELECTION_BLOCK_COUNT** variable. Then the tickets of those selected blocks will be taken and ordered from the latest to the oldest. Then, as Kongahage et al. (2022) proposed, a random number **RN** is taken by SHA256 hashing of the combination of the previous block's leader's public key and the previous block's hash. From that hash, an unsigned integer will be derived. Then an index will be derived from that random number by taking the modulus of that random number by the length of the selected ticket list. Using that index on the tickets list, a ticket will be picked, then it will check for the validator group requirement, which will be discussed in the validator group implementation section. As every nodes perform this process simultaneously, the node that added that ticket will get to know that he is selected as the next leader, therefore he will proceed with the block broadcasting process. If a selected leader couldn't broadcast the block to the network within 10 minutes time, using the above selected ticket, a new random number will be generated by taking the SHA256 hashing of the current random number. Therefore, after 10 minutes, if a block has not proposed a new leader will be elected to continue the process. As the leader is selected from the previous block in the scenario where only the genesis block is available, the PublicKeyProver value defined on the genesis tickets will be selected as the first leader of the network. In this scenario, as there are no previous blocks to take the hash and public key of the previous leader, the hash will be assigned with the genesis previous hash defined on the genesis block, and

the previous block leader's public key will be the same as the defined first leader's public key.

When broadcasting the block to the network, if that the first block after the genesis block, a defined first leader will try to create an initial block. To create this initial block, there should be at least 3 other user-generated tickets in the pool as discussed in the blockchain node implementation section. If there are 3 or more user-generated tickets leader will use them and create an initial block. Finally, after adding other info such as the final hash, public key of the leader itself, and any available transactions block will be broadcasted to the network. If this is not the block right after the genesis block, then whoever the selected leader will get the initial block stored locally when that ticket is added to the network and add the relevant information mentioned above, and broadcast that block to the network. After a successful broadcast that block will be deleted from the local initial block store.

Block Transferring Implementation

Below is the 3-phase block transferring implemented. In the standard scenario, according to 13, once the leader broadcast the block it will be sent to all the nodes via the general pubsub channel. If there's a need for a compression method, then the block will be compressed before sending. If only the leader is available in the network, no transferring will happen. Once other nodes receives the block, they will decompress it if it is compressed in the first place, then verify the block's validity by comparing heights and reconstructing hashes of the block information received. If not verified, each node will perform block synchronization to make sure they are updated with the current blockchain. If verified then the block will be stored in the pendingBlocks list and will send a valid vote back to the leader. Leader, on the other hand, will wait for $2/3$ votes to confirm the validity and agreement toward the new block addition. Once received, he will add the block to his own chain and send the final commit message to other nodes. So

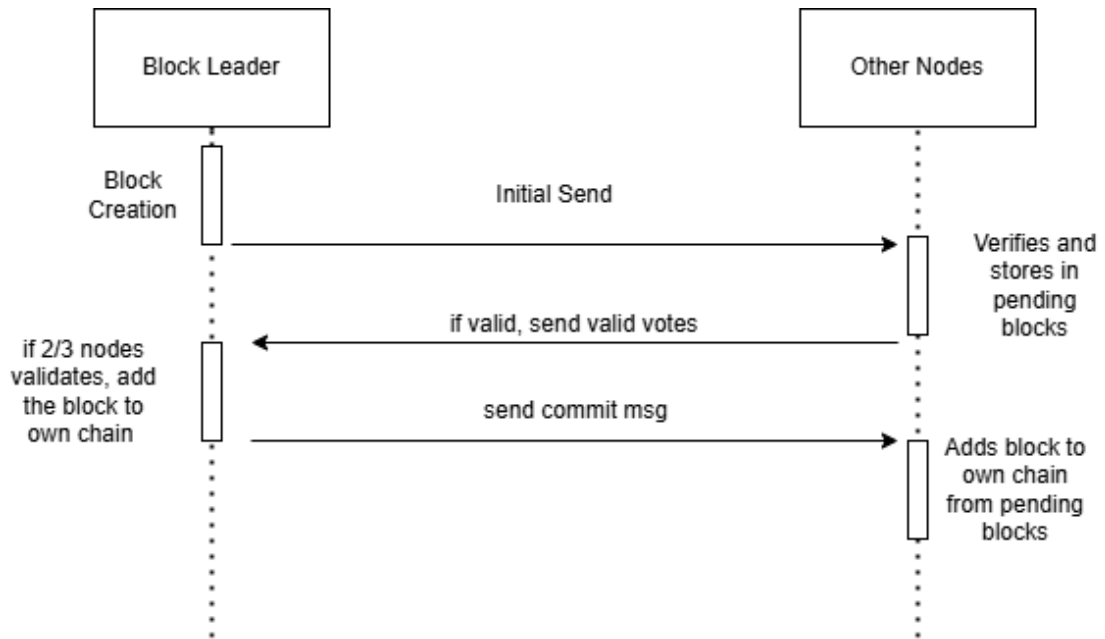


Figure 13: Block Transferring

once other nodes receive this commit message, they will pick that block from the pendingBlocks list and add it to their local chains. After this the next leader election process will start.

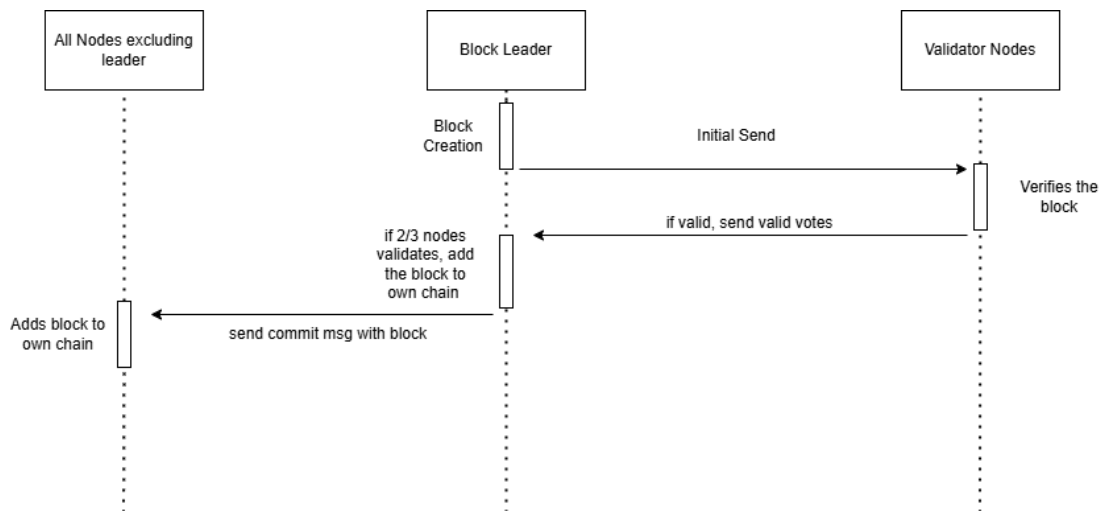


Figure 14: Block transferring after validator group

In a scenario where validator group concepts are used as shown in 14, first the block will be sent to the validator group to confirm their validity. Once the leader gets 2/3 validity agreement from the validator group nodes, he will add the blockchain to his own local chain and broadcast the block back to every other

nodes including validator nodes, as well with the commit message. Once this commit message is received, every node will add the block to their chains with the trust of the validator group's decision.

Block Compression Implementation

For evaluation purposes, to check the effects of compression and decompression blocks on transfer time, GZIP, Brotli, and Snappy compression techniques are utilized. As mentioned in the above section, these compressions and decompressions are only applied for blocks when they are transferred to other nodes through the channel, and once a node receives a compressed block, it need to decompress it to extract block data. However, the final implementation consists of only the Brotli compression technique, which was evaluated as successful in making PoHM more scalable.

Validator Group Implementation

The validator group selection process is introduced and implemented as follows :

```
1
2 If totalNodes < 4
3     Return "Error: Minimum 4 nodes required to form a
      validator group"
4
5 If previousValidators is empty
6     sortedNonLeaderTickets <- Filter out selected next
      leader's tickets from ticketPool and sort by latest
      to earliest
7     groupedNodesWithTicketCounts <- take the 50 or fewer
      latest tickets from sortedNonLeaderTickets and group
      them per node
8     selectedNodes <- Take the highest ticket count generated
      2 nodes from groupedNodesWithTicketCounts
9     newValidators <- selectedNodes
```

```

10     Return newValidators
11 Else
12     X <- Size of previousValidators
13     Y <- Ceiling(2/3 * X) // Keep 2/3 of previous
        validators
14     Z <- totalNodes - Y // Remaining nodes
15     A <- Floor(1/2 * Z) // Half of remaining nodes
16
17     leaderInPrevGroupFlag <- if selected leader in previous
        validator group
18     previousValidators <- remove the selected leader node if
        available in previousValidators
19     shuffledValidators <- Shuffle previousValidators
20     retainedValidators <- Take top Y (or Y - 1 if
        leaderInPrevGroupFlag true) nodes from
        shuffledValidators and remove other nodes from
        previousValidators
21
22     sortedNonLeaderTickets <- Filter out selected next
        leader's tickets from ticketPool and sort by latest
        to earliest
23     nonValidatorTickets <- Filter out retainedValidators'
        tickets from ticketPool
24     groupedNodesWithTicketCounts <- take the 50 or fewer
        latest tickets from nonValidatorTickets and group
        them per node
25
26     If A >= Y
27         nextGroupCount <- Y + Y - 1
28         If leaderInPrevGroupFlag
29             nodesToAdd <- Y
30         Else

```

```

31         nodesToAdd <- Y - 1
32     Else
33         nextGroupCount <- Y + A
34         If leaderInPrevGroupFlag
35             nodesToAdd <- A + 1
36         Else
37             nodesToAdd <- A
38         EndIf
39
40         newNodes <- Take top nodesToAdd nodes from
41                     groupedNodesWithTicketCounts
42         newValidators <- retainedValidators + newNodes
43     Return newValidators
44 EndIf
End

```

This made every subsequent validator group consist of more than or equal to half of the previous validator group nodes. It was assumed that,

1. The nodes that will be eliminated from the previous validator group and the newly selected nodes for a validator group are online when the next validator group creation happens, as every node independently performs this algorithm.
2. The number of nodes participating in the network remains relatively stable, such that no significant increase or decrease in the number of nodes will occur between the creation of any two consecutive blocks. Because such a condition could make the PoHM algorithm more centralized towards a specific set of nodes and on the other hand validator group is a subset of all nodes therefore the validator group node count can't exceed the total node count.

Table 1 above table shows how the validator group creation algorithm works for 10 nodes. The block transfer slightly decreases when the total number of nodes

| Total Number of Nodes | Initial Block sends before the validator group | Initial Block sends after the validator group |
|-----------------------|--|---|
| 1 | - | - |
| 2 | 1 | 1 |
| 3 | 2 | 2 |
| 4 | 3 | 2 |
| 5 | 4 | 3 |
| 6 | 5 | 4 |
| 7 | 6 | 5 |
| 8 | 7 | 6 |
| 9 | 8 | 6 |
| 10 | 9 | 7 |

Table 1: Number of transfers Before and After Validator Group

increases.

Location Generation Implementation

The mobile application contains a list of locations with the latitude and longitude of the respective location. When user initiate a walk/travel, a location from the list will be picked according to a special number N . A special ticket(.json) is used for the generation of the number N . When a new travel starts, the timestamp of the special ticket will be changed and the ticket will be transformed into a JSON string to create a hash value. This hash value then will be divided by "Number of Locations - 1" amount to get a random value between 0 to Number of locations.

$$N = (Hash(Ticket)) \bmod (L - 1)$$

The location value related to the N -valued index will be picked for the next location and user has to travel to that location. The user then starts the verification process. To initiate the verification process, the user should find another user at the location with the PoHM mobile application. Both of these users act as prover and verifier for each of them.

User Verification and Ticket Generation Implementation

The prover will send following data to the verifier via a short-range communication method.

$$\text{Req}_{\text{prover} \rightarrow \text{verifier}} \left\{ \begin{array}{ll} \text{TimestampofSpecialTicket} & : \text{int}, \\ \text{N} & : \text{int}, \\ \text{PublicKeyProver} & : \text{String} \end{array} \right\}_{\text{PrivateKeyProver}}$$

Then the verifier examines the data and verifies it by comparing the value of N_{Verifier} with N_{prover} . Then the verifier will modify its special ticket's value by replacing $\text{Timestamp}_{\text{Verifier}}$ with $\text{Timestamp}_{\text{Prover}}$. Then the special ticket converts in to JSON string then the hash value will be created. Then the N value of prover will be verified. After the verification, the verifier sends the ticket data of prover as follows.

$$\text{Res}_{\text{verifier} \rightarrow \text{prover}} \left\{ \begin{array}{ll} \text{TicketID} & : \text{String}, \\ \text{Timestamp} & : \text{int}, \\ \text{PublicKeyProver} & : \text{String}, \\ \text{PublicKeyVerifier} & : \text{String} \end{array} \right\}_{\text{PrivateKeyVerifier}}$$

Received ticket data from the prover side then will be extracted and the ticket of the prover will be created inside the inner storage of the mobile application.

Chapter 6 - Evaluation and Results

Strategies to make PoHM more scalable

To evaluate the two strategies for making PoHM algorithm more scalable working version of the blockchain node was utilized. However to make this evaluation deterministic, leader election algorithms or any validation logics for blocks, transactions, or tickets were not used. A selected node was given the option to send blocks by reading a block JSON file having 50 sample blocks, each having 5 transactions per block.

Once the sending was initiated, the selected leader send the blocks to all the other nodes in below mentioned scenarios, starting from 4 nodes all the way up to 10 nodes. Other nodes were listening for block receiving from the leader node through the channel and acted upon it by either decompressing and providing valid votes, or simply providing votes, or adding the blocks after a commit message from the leader. The three phase block trasfering discussed in above sections are exactly used according to each scenario below. Also the evaluation was carried out by connecting all the node to a similar network to avoid latencies that might occur through NAT traversals or such. Through the experiment, a time measurement was taken from when the leader node sends the initial send to other nodes until the commit message is sent for the block by him. Then this time measurement was used to calculate the TPS in each scenario using the below formula given by Kongahage et al. (2022):

$$\text{Throughput} \left(\frac{\text{tx}}{\text{sec}} \right) = \frac{\text{tx}}{\text{block}} * \frac{\text{blocks}}{\text{ft} - \text{it}} \quad (5)$$

The time measured was equal to ft-it in the above formula. Then the decreasing percentage is also calculated by using the below formula to check what rate the TPS is affected when adding a new node to the system.

$$\text{Percentage Decrease} = \frac{\text{TPS}_{\text{previous}} - \text{TPS}_{\text{current}}}{\text{TPS}_{\text{previous}}} \times 100 \quad (6)$$

Measured Results

Scenario 01: Normal block transferring

| No of Nodes | Time Taken to finalize (Sec) | TPS | Decreasing % |
|-------------|------------------------------|-------------|--------------|
| 4 | 1.5271 | 163.7089909 | |
| 5 | 8.2831 | 30.18193671 | 81.5636658 |
| 6 | 11.8381 | 21.11825377 | 30.0301569 |
| 7 | 10.9827 | 22.76307283 | 0 |
| 8 | 15.1923 | 16.45570453 | 27.7087735 |
| 9 | 18.101 | 13.81139164 | 16.0692779 |
| 10 | 22.4253 | 11.14812288 | 19.2831311 |

Table 2: Normal block transferring evaluation

Scenario 02: GZIP Compression for block transfers

| No of Nodes | Time Taken to finalize (Sec) | TPS | Decreasing % |
|-------------|------------------------------|-------------|--------------|
| 4 | 2.0713 | 120.6971467 | |
| 5 | 9.6076 | 26.02106666 | 78.4410259 |
| 6 | 12.6229 | 19.80527454 | 23.8875377 |
| 7 | 16.9928 | 14.71211337 | 25.7161857 |
| 8 | 14.6555 | 17.05844222 | 0 |
| 9 | 18.0041 | 13.88572603 | 18.5990969 |
| 10 | 25.158 | 9.937196915 | 28.4358852 |

Table 3: GZIP Compression for block transfers evaluation

Scenario 03: Snappy Compression for block transfers

| No of Nodes | Time Taken to finalize (Sec) | TPS | Decreasing % |
|-------------|------------------------------|-------------|--------------|
| 4 | 2.596 | 96.30200308 | |
| 5 | 8.9481 | 27.93889206 | 70.9882545 |
| 6 | 12.8898 | 19.39518069 | 30.5799935 |
| 7 | 11.8689 | 21.06345154 | 0 |
| 8 | 14.4715 | 17.27533428 | 17.984314 |
| 9 | 18.9467 | 13.19490993 | 23.6199444 |
| 10 | 23.4617 | 10.65566434 | 19.2441298 |

Table 4: Snappy Compression for block transfers evaluation

Scenario 04: Brotli Compression for block transfers

| No of Nodes | Time Taken to finalize (Sec) | TPS | Decreasing % |
|-------------|------------------------------|-------------|--------------|
| 4 | 3.3459 | 74.71831196 | |
| 5 | 9.7917 | 25.53182798 | 65.8292227 |
| 6 | 13.2329 | 18.8923063 | 26.0048818 |
| 7 | 12.584 | 19.86649714 | 0 |
| 8 | 15.7774 | 15.84544982 | 20.2403438 |
| 9 | 19.579 | 12.76878288 | 19.416722 |
| 10 | 23.4736 | 10.65026242 | 16.5914048 |

Table 5: Brotli Compression for block transfers evaluation

Scenario 05: Validator Group Block transfers

| No of Nodes | Time Taken to finalize (Sec) | TPS | Decreasing % |
|-------------|------------------------------|-------------|--------------|
| 4 | 4.152 | 60.21194605 | |
| 5 | 11.1317 | 22.45838461 | 62.7011148 |
| 6 | 16.2652 | 15.3702383 | .31.5612473 |
| 7 | 23.2602 | 10.74797293 | 30.0728283 |
| 8 | 15.6093 | 16.01609297 | 0 |
| 9 | 19.9121 | 12.55518002 | 21.6089714 |
| 10 | 27.6142 | 9.05331315 | 27.8918093 |

Table 6: Validator Group Block transfers evaluation

Scenario 06: Validator Group & GZIP Compression for block transfers

| No of Nodes | Time Taken to finalize (Sec) | TPS | Decreasing % |
|-------------|------------------------------|-------------|--------------|
| 4 | 4.6781 | 53.44049935 | |
| 5 | 10.4358 | 23.95599762 | 55.172579 |
| 6 | 17.0888 | 14.62946491 | 38.931932 |
| 7 | 24.5941 | 10.16503958 | 30.5166686 |
| 8 | 17.2937 | 14.45613142 | 0 |
| 9 | 21.3406 | 11.71475966 | 18.9633843 |
| 10 | 27.8041 | 8.991479674 | 23.2465715 |

Table 7: Validator Group & GZIP Compression for Block transfers evaluation

Scenario 07: Validator Group & Snappy Compression for block transfers

| No of Nodes | Time Taken to finalize (Sec) | TPS | Decreasing % |
|-------------|------------------------------|-------------|--------------|
| 4 | 5.9345 | 42.12654815 | |
| 5 | 11.424 | 21.8837535 | 48.0523459 |
| 6 | 18.3771 | 13.60388745 | 37.8356759 |
| 7 | 27.3281 | 9.148092989 | 32.7538321 |
| 8 | 17.1934 | 14.5404632 | 0 |
| 9 | 20.6475 | 12.10800339 | 16.7289018 |
| 10 | 28.9412 | 8.638204359 | 28.6570702 |

Table 8: Validator Group & Snappy Compression for Block transfers evaluation

Scenario 08: Validator Group & Brotli Compression for block transfers

| No of Nodes | Time Taken to finalize (Sec) | TPS | Decreasing % |
|-------------|------------------------------|-------------|--------------|
| 4 | 9.6836 | 25.81684498 | |
| 5 | 11.9496 | 20.92120238 | 18.9629778 |
| 6 | 20.4813 | 12.20625644 | 41.6560472 |
| 7 | 26.102 | 9.577810129 | 21.533599 |
| 8 | 17.925 | 13.94700139 | 0 |
| 9 | 22.6544 | 11.03538385 | 20.8762978 |
| 10 | 31.2071 | 8.010997497 | 27.4062633 |

Table 9: Validator Group & Brotli Compression for Block transfers evaluation

Comparison of outcomes plotted together:

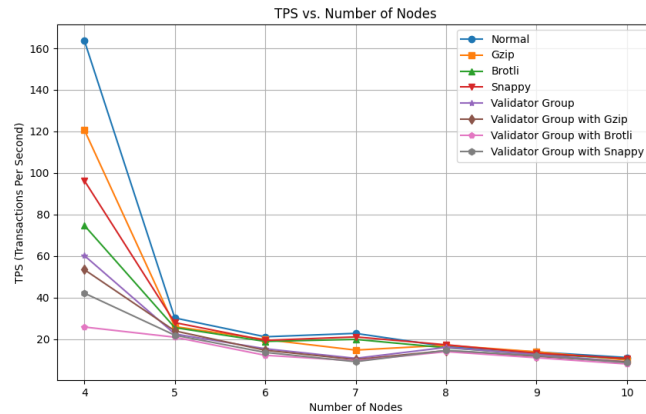


Figure 15: Comparison of TPS in each Mode

Result Interpretation

Initially, in each scenario, TPS had a higher value when less nodes, but when the number of nodes increased, TPS values got reduced to a similar value range. There are some outliers measurements showing the TPS increased when adding nodes due to temporary fluctuations in network conditions. Also, within the 10 nodes normal block transferring method still had greater TPS compared to others. But to make sure if this persist or not when the number of nodes increases higher than that, the rounded-up average decreasing percentages for each scenario is taken.

1. Normal block transferring - 35%
2. GZIP Compression for block transfers - 35%
3. Snappy Compression for block transfers - 32%
4. Brotli Compression for block transfers - 30%
5. Validator Group Block transfers - 35%
6. Validator Group & GZIP Compression for block transfers - 33%
7. Validator Group & Snappy Compression for block transfers - 33%
8. Validator Group & Brotli Compression for block transfers - 26%

Above calculations show that the validator group mechanism with the Brotli compression method has the lowest decreasing percentage, which means that when the number of nodes increases, with this method implemented, TPS will have the lowest effect. Therefore Brotli compression with the validator group has identified as the most suitable strategy to work with when making the PoHM algorithm more scalable. Finalized implementation was developed using this method, integrated.

Short-range Communication Method

This section focuses primarily on the analysis and selection of the most suitable and favorable short-range communication technology for integration into the mobile application. The possible technologies under consideration are Near Field Communication (NFC), Bluetooth, and Wi-Fi Direct. In subsequent sections, these technologies will be assessed based on their social adoption, technical feasibility, and advantages and disadvantages of the practical implementation.

Near Field Communication

Near Field Communication (NFC) is a wireless, short-range communication method, which typically requires a distance of 4cm. NFC facilitates sharing of small payloads of data. NFC can work in 3 different modes.

- **Card emulation Mode :** allows the mobile device to read and write passive NFC tags and stickers.
- **Reader/Writer Mode :** allows the mobile device itself to act as an NFC card.
- **Peer-to-peer Mode :** allows direct communication between two NFC-enabled mobile devices

The **peer-to-peer** mode will be most suitable out of 3 modes of NFC. But the Android development allows only following 2 modes.

- Card emulation Mode
- Reader/Writer Mode

If implemented using either of above mentioned modes, users might need to close contact with each other to verify themselves. This requirement compromises the user anonymity, as physical presence of the user may reveal personal identities and location. Furthermore, instead of implementing a complex solution like NFC for the verification mechanism, an alternative approach using QR codes

could be considered, since QR code-based verification offers a simpler, effective method to authenticate users while mitigating some of the complexities of NFC implementation.

Bluetooth

Bluetooth is a popular wireless, short-range communication technology, which enables devices to exchange data over small distances. Bluetooth operates on the 2.4 GHz frequency band and supports low-power and secure communication. Several points need to be considered when implementing Bluetooth feature on mobile phones. Device discovery is an essential feature in mobile devices, enabling verification process. Users might be able to find specific mobile phone with Bluetooth enabled within a given range. However an issue arises around the detectability of mobile phones. Despite having Bluetooth turned on, a mobile device may not get detected.

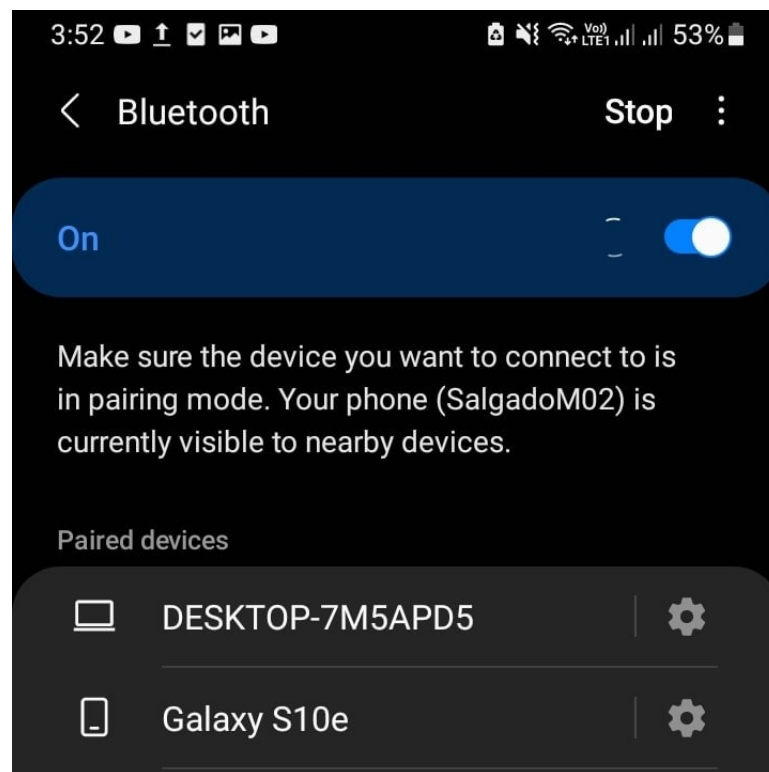


Figure 16: Bluetooth interface

As seen in the above image, the mobile phone needs to be explicitly set to

discoverable mode by keeping the relevant Bluetooth specific interface opened. Additionally, the detectability is time-limited, meaning the device will only be visible for limited time. Therefore, users might lose the window of opportunity if they did not turn on discoverable mode promptly.

The power consumption when Bluetooth is enabled is examined using three devices with different battery capacity. The average battery drain per hour in 25°C is depicted in following table 13. For this experiment, devices were kept idle and display always turned on. It is important to know that the discharge rate may depends on environmental factors and internal battery usage.

| Mobile Phone | Battery Capacity | Bluetooth |
|-----------------|------------------|-----------|
| Samsung S10e | 3100 mAh | 4% |
| Samsung M02 | 5000 mAh | 2% |
| Galaxy J7 Prime | 3300 mAh | 4.6% |

Table 10: Average Power Consumption per hour

| Mobile Phone | Battery Capacity | Bluetooth | Wi-Fi Direct | Quickshare |
|-----------------|------------------|-----------|--------------|------------|
| Samsung S10e | 3100 mAh | 4% | 5% | 5.67% |
| Samsung M02 | 5000 mAh | 2% | 2.67% | 3% |
| Galaxy J7 Prime | 3300 mAh | 4.6% | 6% | 6.33% |

Table 11: Power Consumption per hour

Another examination has done for collecting device discovery time of Bluetooth enabled devices. The important thing discovered was inconsistencies in discovery time.

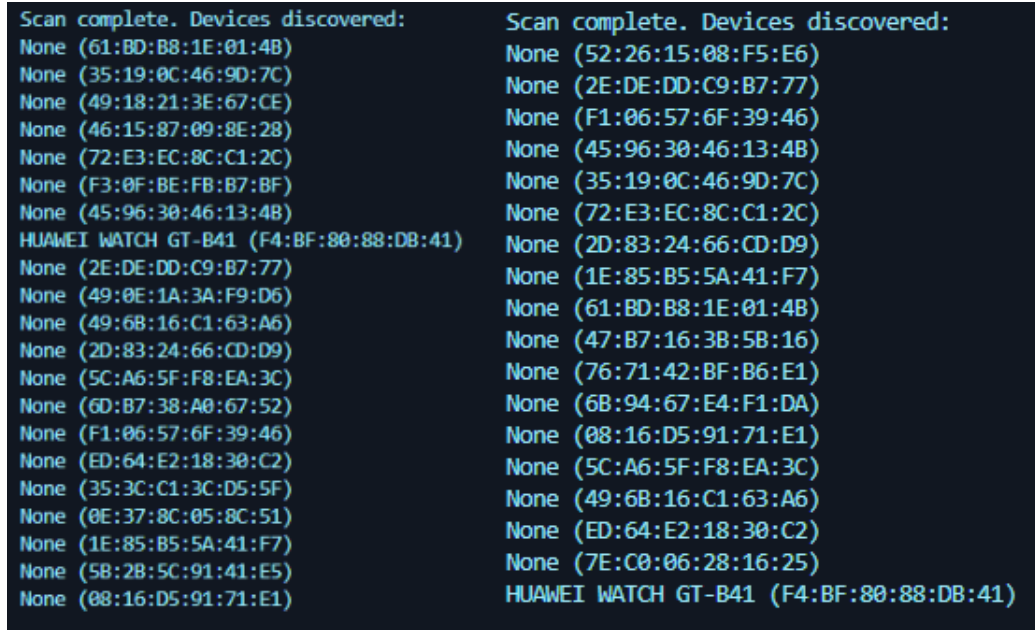


Figure 17: Bluetooth Discover Comparison

As seen in the figure 17, it is possible to see that HUAWEI WATCH GT-B41 has an inconsistency in the discovery time. The table below contains the discovery time and the related number of devices discovered. It is visible that number of devices detected depends on the time allocated for discovery.

| Discovery Time | Run 1 | Run 2 | Run3 | Run 4 | Run 5 | Average |
|----------------|-------|-------|------|-------|-------|---------|
| 2s | 9 | 11 | 8 | 12 | 11 | 10 |
| 4s | 12 | 15 | 14 | 13 | 16 | 14 |
| 8s | 17 | 19 | 17 | 19 | 16 | 17 |
| 16s | 23 | 18 | 23 | 24 | 24 | 22 |
| 32s | 28 | 20 | 19 | 20 | 22 | 21 |
| 60s | 35 | 40 | 26 | 30 | 34 | 33 |

Table 12: Discovery time vs No. of devices discovered

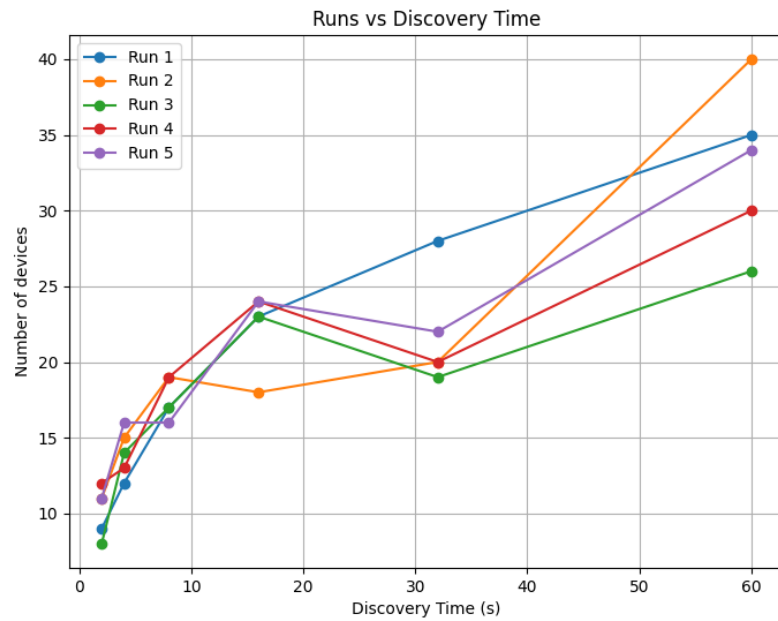


Figure 18: Discovery plot for each run

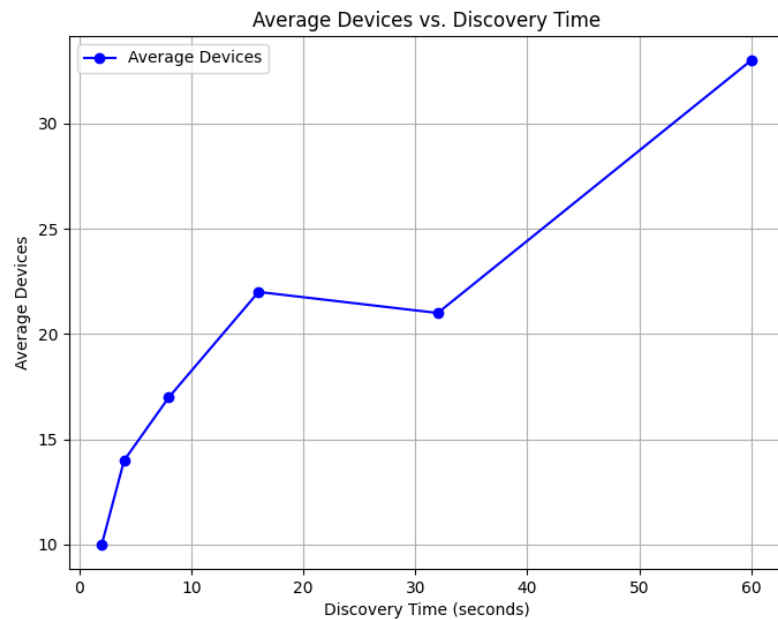


Figure 19: Discovery plot for average devices discovered

Wi-Fi Direct

Wi-Fi Direct is a short-range communication method, which allows mobile devices to connect directly without the requirements of access points or routers. It offers

a reliable, high-speed data transfer, utilizing the same Wi-Fi standards. Similarly to Bluetooth, Wi-Fi Direct also has the same issue of detectability. To be discovered, the mobile device need to be on the Wi-Fi Direct specific interface to be discovered by others.



Figure 20: Wi-Fi Direct interface

Also, newer Android devices lack the ability of share files using Wi-Fi Direct directly among mobile phones. The 'send' functionality was removed from newer Android versions, with the new OneUI 3.1. Therefore, newer Android devices should utilize the Quickshare or Nearbyshare feature available on mobile as a replacement.

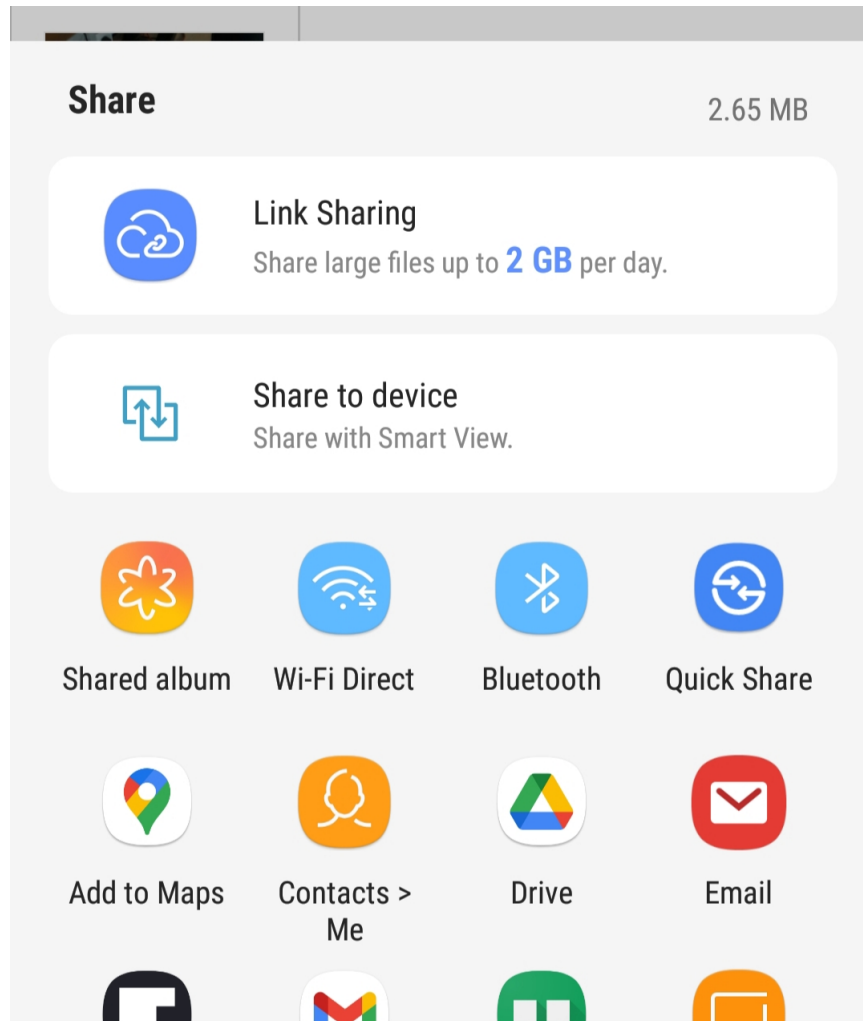


Figure 21: Android 8.1 Wif-Direct

The power consumption per hour when Wi-Fi Direct is enabled is shown below.

| Mobile Phone | Battery Capacity | Wi-Fi Direct | Quickshare |
|-----------------|------------------|--------------|------------|
| Samsung S10e | 3100 mAh | 5% | 5.67% |
| Samsung M02 | 5000 mAh | 2.67% | 3% |
| Galaxy J7 Prime | 3300 mAh | 6% | 6.33% |

Table 13: Average Power Consumption per hour Wi-Fi and Quickshare

User Survey

To gather comprehensive and representative data in various age groups, a carefully designed survey was conducted. The survey aimed at participants from under 30

years to over 50 years of age, ensuring a broad spectrum of perspectives. To maintain balance and minimize bias in data collection, the sample was equally distributed between participants under 30 years and those over 30 years.

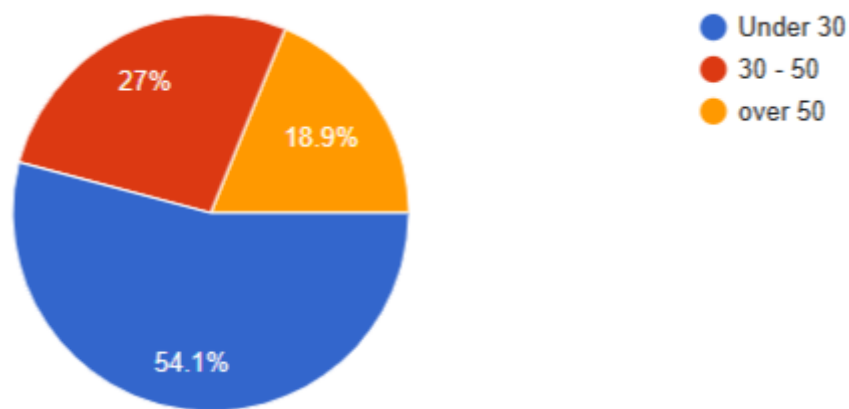


Figure 22: User variability

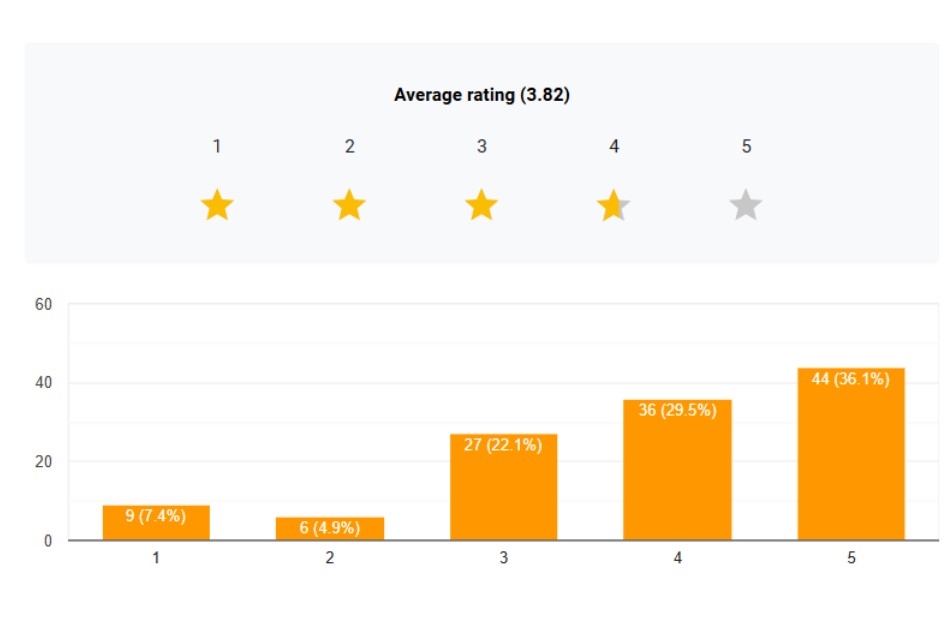


Figure 23: Common IT knowledge

The survey specifically explored participants' knowledge and usage of a particular short-range communication method. The following sections will present a detailed discussion of the survey questions and the corresponding responses for each short-range communication method, offering insights of familiarity among

the respondents.

| Question | Positive perc. | Negative perc. |
|------------------------|----------------|----------------|
| Bluetooth Knowledge | 96.7 | 3.3 |
| Bluetooth Usage | 94.3 | 5.7 |
| Wi-Fi Direct Knowledge | 86.1 | 13.9 |
| Wi-Fi Direct Usage | 73 | 27 |

Table 14: Bluetooth and Wi-Fi Direct - Survey

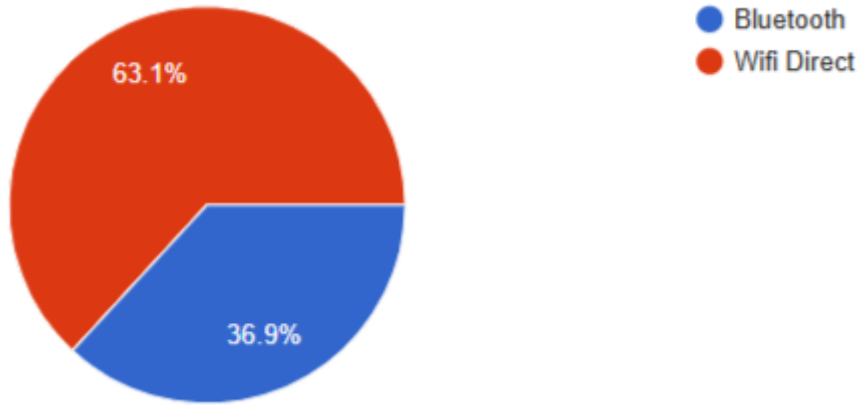


Figure 24: Bluetooth vs Wi-Fi Direct preference

Results of the survey indicates that, despite Bluetooth being more common among the participants, 63.1% of the participants prefer Wi-Fi Direct over Bluetooth. 106 participants out of 122 participants provided us with detailed responses explaining why their preference for one over another. Also, these responses highlight the knowledge of short-range methods among participants. The responses of the participants can be accessed [[here](#)].

Quickshare

The final 2 questions asked were the following.

| Question | Positive perc. | Negative perc. |
|----------------------|----------------|----------------|
| Quickshare Knowledge | 73.8 | 26.2 |
| Quickshare Usage | 50.8 | 49.2 |

Table 15: Quickshare - survey

Due to the disadvantages and implementation issues mentioned above, another alternative method was deemed necessary for the short-range communication. Since both Bluetooth and Wi-Fi Direct have their own distinct advantages, utilizing both technologies presents a viable solution. Hence Quickshare feature available on Android was identified as an effective mechanism .

A separate examination was conducted to identify all available short-range services in different locations around Colombo and Moratuwa. All the Bluetooth, Wi-Fi Direct or Quickshare enabled mobile device's presence in respective location is examined. The data sheet can be accessed through the following [\[link\]](#).

| City | Bluetooth | Wi-Fi Direct | Quickshare |
|----------|-----------|--------------|------------|
| Colombo | 12 | 1 | 0 |
| Moratuwa | 9 | 1 | 0 |

Table 16: Average of Short-range communication traffic

According to the data sheet, it is visible that each area consists of multiple devices enabled with Bluetooth and Wi-Fi. If viewed on the user's mobile device, the user may find it difficult to differentiate the specific device for the verification. Also, it is visible that the number of Quickshare enabled devices is near zero as an average. It is possible to utilize this unique opportunity of low prevalence to implement the short-range communication service via Quickshare.

- Number of Quickshare enabled devices are much lower than other services, which reduces the likelihood of device identification conflicts.
- Quickshare does not show common embedded devices which are not applicable for the use case
- Can leverage both Bluetooth and Wi-Fi for short-range communication.
- Devices need not to be in Discovery mode specifically. The user only needs to enable the Quickshare feature and Bluetooth.

Optimal Sensor Set for detecting Fraudulent Movements

Here, it focuses on detecting fraudulent nodes (people) who are cycling or traveling by vehicle, which are prohibited activities in the PoHM context. According to PoHM context, legitimate nodes are restricted to walking or running. The objective here is to identify the most optimal combination of smartphone sensors (GPS, accelerometer, gyroscope, and magnetometer) that achieves high classification accuracy to differentiate fraudulent activities from legitimate activities, while balancing practical constraints such as power consumption. This section outlines a detailed overview of identifying optimal sensor set for detection of fraudulent node, starting with the development of a custom mobile application to collect sensor data, followed by an initial exploratory approach using manual plotting, switching to a machine learning model for sensor selection, derivation of threshold values to spot fraudulent nodes, and the selection of the optimal sensor set based on accuracy and specifications considering power consumption.

Data Collection and Process

To collect real-world sensor data, developed a mobile application to capture measurements from four smartphone sensors: GPS (measuring speed), accelerometer (capturing movement intensity), gyroscope (recording rotational speed), and magnetometer (assessing magnetic field variance). The application was designed to capture data during specific movement activities: walking, running, cycling, and travel by vehicle carried out by the research team and volunteers. The application allowed users to select a movement type after starting the data recording and export those captured data into a CSV file. This ensured a labeled data set in which each data sample was associated with one of the four movement types and sensor measurements.

The following measurements were derived to each row in the file:

- **average_gps_speed:** Average speed from GPS (in meters per second).
- **average_acceleration_magnitude:** Average movement strength from the accelerometer.

- `average_angular_velocity_magnitude`: Average rotation speed from the gyroscope.
- `average_magnetic_field_variance`: How much the magnetic field changes, from the magnetometer.
- `movement_type`: The label (Walking, Running, Cycling, or Travel by Vehicle).

The application averages raw sensor readings into these measurements to reduce noise and improve data quality.

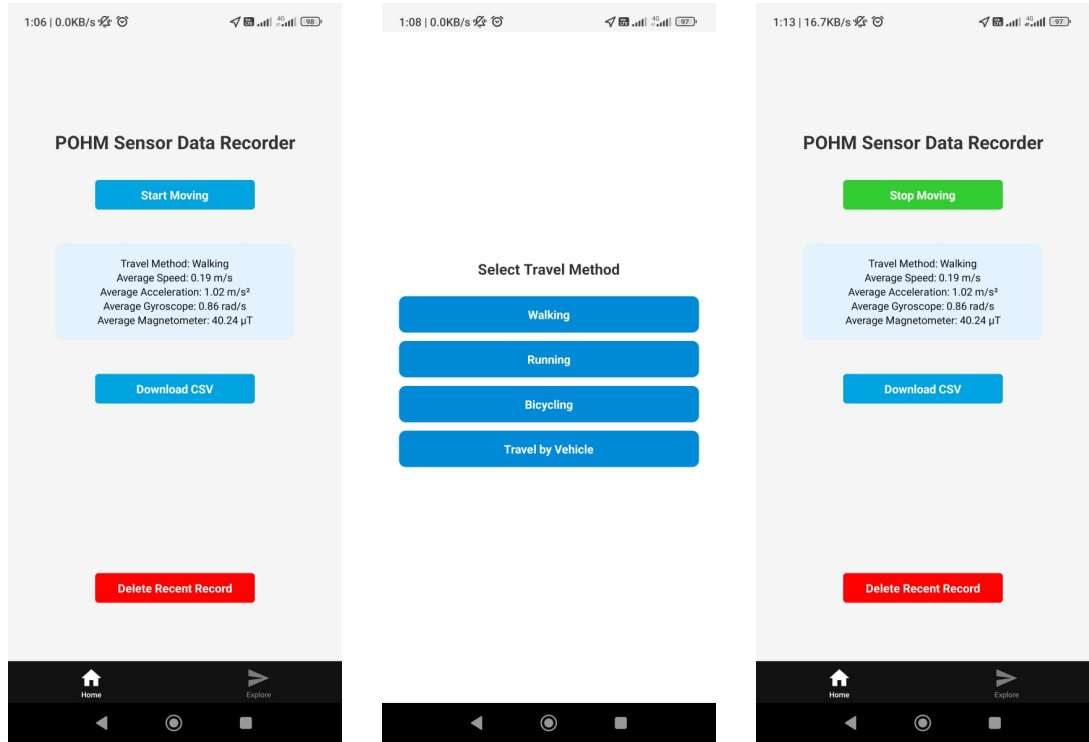


Figure 25: Screenshots of the mobile app: (a) Interface of the application; (b) Movement type selection screen, with options for walking, running, cycling, and travel by vehicle; (c) Recording interface, displaying derived sensor data and a save button to export the CSV file.

Data collection involved research team and volunteers at different ages, approximately 20 to 60 years old, while performing each type of movement in real world scenarios. Walking and running sessions are conducted on outdoor paths, riding bicycles on trails and roads, and perform vehicle travel in motor bicycles,

three wheels and cars on urban roads. Each session lasted approximately 2–10 minutes to capture sufficient data points. The application saved the measurements to a CSV file, with each row containing the four sensor measurements and the movement type label. To ensure data set robustness, data were collected under varied conditions such as different times, weather, and terrains. After removing incomplete entries, the dataset contained approximately 1000 samples. The CSV format facilitated straightforward analysis of the data.

Initial Approach: Manual Plotting and Threshold Derivation

Initially attempted to detect fraudulent nodes by making scatter plots of sensor measurements for each combination and manually searching for thresholds. The idea was that walking and running would form different from cycling and vehicle travel on a graph, allowing thresholds to be set based on cluster boundaries. For each sensor combination such as GPS + Accelerometer, yielding `average_gps_speed` against `average_acceleration_magnitude`, scatter plots were plotted, with points colored by movement type. The goal was to visually identify the ranges for legitimate and fraudulent activities. Here are some plots that get:

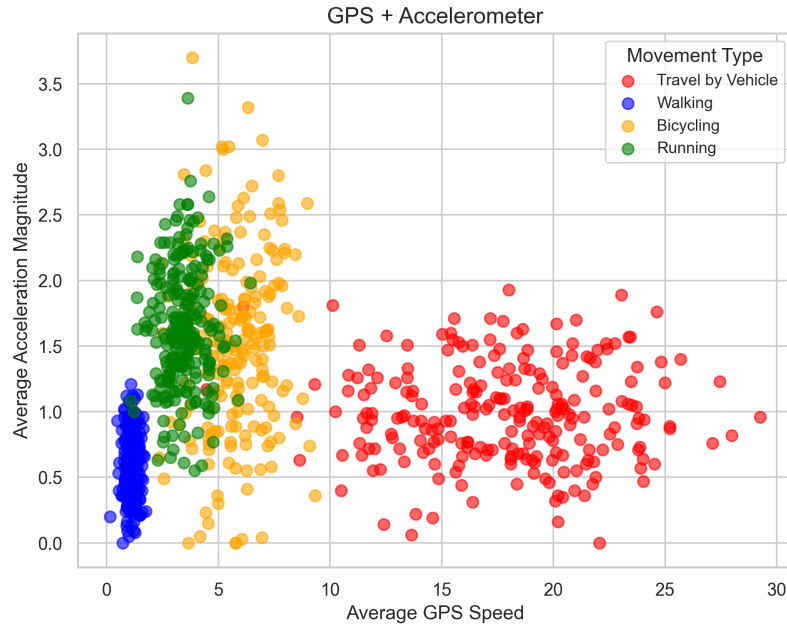


Figure 26: GPS vs Accelerometer

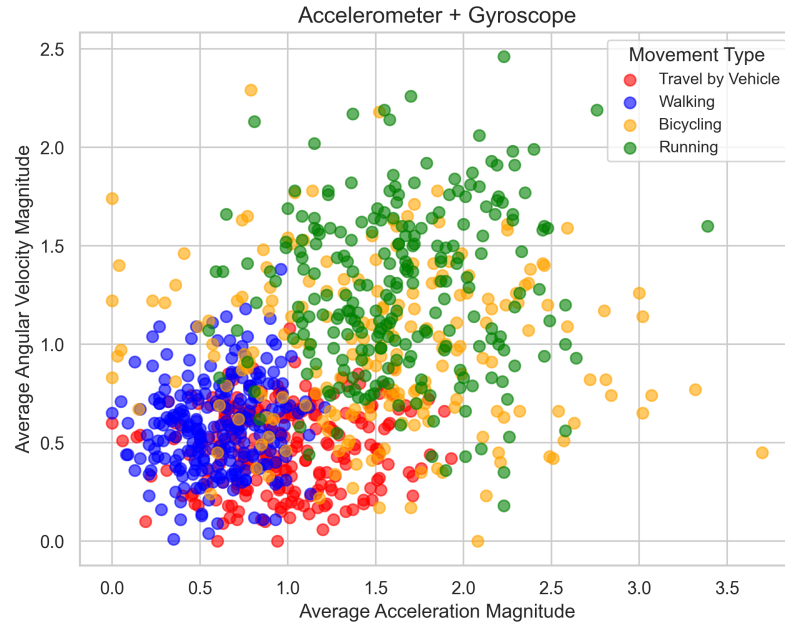


Figure 27: Accelerometer vs Gyroscope

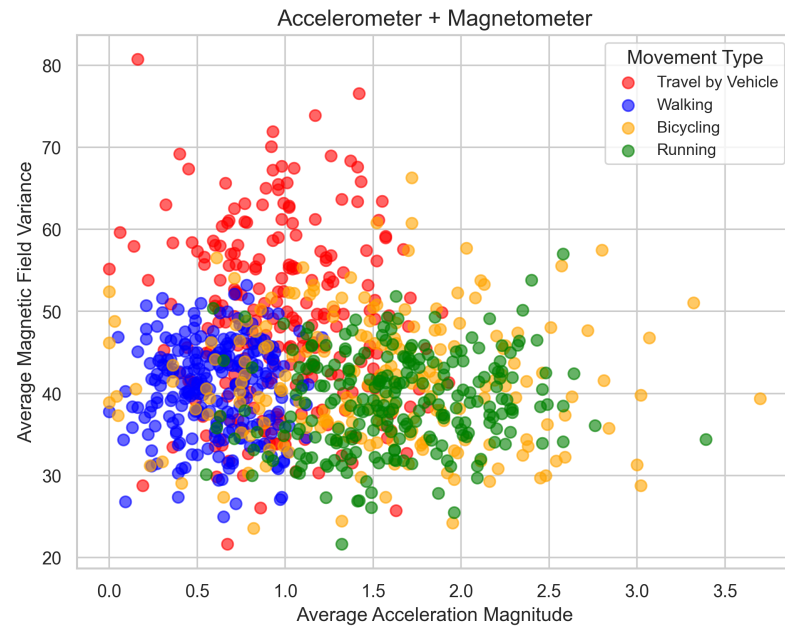


Figure 28: Accelerometer vs Magnetometer

However, this approach faced significant challenges:

1. Combination with more than two sensors, couldn't make simple graphs it produced high dimensional data, making 2D visualization impractical without complex techniques

2. Data points, especially running and cycling often get overlapped, making it hard to separate legitimate (walking, running) from fraudulent (cycling, vehicle) movements.
3. Setting threshold values (like speed ranges) manually based visual inspection of graphs was subjective, risking inconsistent or inaccurate boundaries.
4. Analyzing 10 sensor combinations manually was time consuming and not impractical for iterative analysis.

Recognizing these challenges and limitations led to a shift to a machine learning approach, which offered automated evaluation of sensor combinations and data-driven threshold derivation, overcoming the constraints of manual analysis to get better results.

Machine Learning Approach

The machine learning approach was adopted based on:

- It enabled a systematic evaluation of sensor combinations to identify the most accurate for classifying movement types.
- Predictions were based on statistical patterns rather than subjective interpretation.
- Facilitated the establishment of reliable thresholds for fraud detection.

A Random Forest classifier was selected due to its robustness with noisy data means good at finding patterns in messy data, ability to capture nonlinear relationships, and feature importance insights, which could guide sensor selection.

Data Preprocessing

The data set was preprocessed to ensure compatibility with the ML model:

1. **Handling Missing Data:** Rows with missing values were removed using `df.dropna()`.

2. **Pick Measurements:** For each sensor, the corresponding measurements were extracted based on the mapping:
 - GPS: `average_gps_speed`
 - Accelerometer: `average_acceleration_magnitude`
 - Gyroscope: `average_angular_velocity_magnitude`
 - Magnetometer: `average_magnetic_field_variance`
3. **Normalize Data:** Since measurements had different units, features were standardized using `StandardScaler` to ensure equal weighting.

Model Training and Evaluation

The ML pipeline, implemented in Python with scikit-learn, pandas, and numpy, followed these steps:

1. **Combination Generation:** Created all sets of 2 to 4 sensors producing 10 combinations total.
2. **Split Data:** Split the dataset into 70% for training and 30% for testing, keeping it consistent with `random_state=42` for reproducibility.
3. **Train Model:** Random Forest classifier was trained on the scaled training data to predict movement types.
4. **Evaluation:** Test set accuracy was computed using `accuracy_score`, indicating the proportion of correctly classified samples.

The combinations were ranked according to the following.

- **Primary Criterion:** Highest accuracy with the most correct predictions.
- **Secondary Criterion:** Number of sensors with their specifications considering power consumption.

Results

Table 17 presents the classification accuracies for all combinations of sensors. The

combination of GPS + Accelerometer + Gyroscope achieved the highest accuracy of 92.45%, followed closely by GPS + Accelerometer + Magnetometer (92.09%) and GPS + Accelerometer (91.73%).

| Sensor Combination | Accuracy(%) |
|--|-------------|
| GPS + Accelerometer + Gyroscope | 92.45 |
| GPS + Accelerometer + Magnetometer | 92.09 |
| GPS + Accelerometer + Gyroscope + Magnetometer | 92.09 |
| GPS + Accelerometer | 91.73 |
| GPS + Gyroscope | 91.37 |
| GPS + Gyroscope + Magnetometer | 91.37 |
| GPS + Magnetometer | 90.65 |
| Accelerometer + Gyroscope + Magnetometer | 62.23 |
| Accelerometer + Gyroscope | 51.44 |
| Accelerometer + Magnetometer | 50.36 |
| Gyroscope + Magnetometer | 50.00 |

Table 17: Accuracy of Different Sensor Combinations

Threshold Derivation for Fraud Detection

For detect fraudulent nodes, thresholds must be derived for the identification of measurements characteristic of cycling or vehicle travel. For each feature in the chosen sensor combination, thresholds were calculated for legitimate movement types (walking, running) using mean and standard deviation. Here are the steps:

1. Filtered data by movement type (e.g. walking).
2. The mean (μ) and the standard deviation (σ) were computed for each feature.
3. The thresholds were set as:
 - Minimum: $\mu - \sigma$
 - Maximum: $\mu + \sigma$
4. The derived values were stored for walking and running.

A node is **fraudulent** if its measurements for all features fall **outside the min/max ranges for both walking and running**, implying that they align with cycling or vehicle travel. This approach was chosen for the following reasons:

- Mean ± 1 std covers 68% of data (assuming normality), defining “typical” ranges.
- Clear numerical boundaries enable real-time use.
- Narrow focus on walking and running improves accuracy.

Results

Table 18 presents the derived thresholds for the GPS and Accelerometer combination.

| Feature | Movement Type | Min | Max | Mean | Std |
|--------------------------------|---------------|------|------|------|------|
| average_gps_speed | Walking | 0.87 | 1.35 | 1.11 | 0.2 |
| average_gps_speed | Running | 2.55 | 4.31 | 3.43 | 0.8 |
| average_acceleration_magnitude | Walking | 0.35 | 0.85 | 0.60 | 0.25 |
| average_acceleration_magnitude | Running | 1.17 | 2.10 | 1.64 | 0.47 |

Table 18: Threshold Values for Fraud Detection

Fraud Detection Logic

A node is flagged as fraudulent if, for all features, its values lie outside the walking and running threshold ranges. For example, a **gps_speed** exceeding the maximum for running and an **accelerometer_variance** inconsistent with walking or running indicates cycling or vehicle travel, classifying the node as fraudulent.

Sensor Selection Based on Specifications

Although classification accuracy was the primary criterion, power consumption was a critical factor for mobile applications.

To quantify the power consumption of each sensor on a smartphone, the battery drain was continuously calculated for an hour based on the power consumption values. The calculations assume a typical smartphone battery voltage of 3.7 V and use the formula $I = \frac{P}{V}$, where P is power in milliwatts (mW), V is voltage in volts (V), and I is current in milliamperes (mA). The battery drain in milliampere hours (mAh) is then $\text{Drain} = I \times 1$ for one hour of operation. The following calculations show the battery drain for each sensor.

GPS

Power consumption ranges from 50 to 100 mW Texas Instruments (2020).

- Lower bound (50 mW):

$$I = \frac{50}{3.7} \approx 13.5135 \text{ mA}, \quad \text{Drain} = 13.5135 \text{ mAh}$$

- Upper bound (100 mW):

$$I = \frac{100}{3.7} \approx 27.0270 \text{ mA}, \quad \text{Drain} = 27.0270 \text{ mAh}$$

Accelerometer

Power consumption ranges from 0.1 to 1 mW STMicroelectronics (2021).

- Lower bound (0.1 mW):

$$I = \frac{0.1}{3.7} \approx 0.0270 \text{ mA}, \quad \text{Drain} = 0.0270 \text{ mAh}$$

- Upper bound (1 mW):

$$I = \frac{1}{3.7} \approx 0.2703 \text{ mA}, \quad \text{Drain} = 0.2703 \text{ mAh}$$

Gyroscope

Power consumption ranges from 5 to 10 mW STMicroelectronics (2021).

- Lower bound (5 mW):

$$I = \frac{5}{3.7} \approx 1.3514 \text{ mA}, \quad \text{Drain} = 1.3514 \text{ mAh}$$

- Upper bound (10 mW):

$$I = \frac{10}{3.7} \approx 2.7027 \text{ mA}, \quad \text{Drain} = 2.7027 \text{ mAh}$$

Magnetometer

Power consumption ranges from 1 to 5 mW Honeywell (2017).

- Lower bound (1 mW):

$$I = \frac{1}{3.7} \approx 0.2703 \text{ mA}, \quad \text{Drain} = 0.2703 \text{ mAh}$$

- Upper bound (5 mW):

$$I = \frac{5}{3.7} \approx 1.3514 \text{ mA}, \quad \text{Drain} = 1.3514 \text{ mAh}$$

| Sensor | Battery Drain (mAh/hour) |
|---------------|--------------------------|
| GPS | 13.5–27.0 |
| Accelerometer | 0.03–0.27 |
| Gyroscope | 1.4–2.7 |
| Magnetometer | 0.27–1.4 |

Table 19: Battery Drain for One Hour of Continuous Sensor Operation at 3.7 V

As a summary of battery drain values over one hour of continues, smartphone sensors combinations are as follows:

| Sensor Combination | Total Battery Drain (mAh/hour) |
|--|--------------------------------|
| GPS + Accelerometer + Gyroscope | 14.93–29.97 |
| GPS + Accelerometer + Magnetometer | 13.80–28.67 |
| GPS + Accelerometer + Gyroscope + Magnetometer | 15.20–31.37 |
| GPS + Accelerometer | 13.53–27.27 |
| GPS + Gyroscope | 14.90–29.70 |
| GPS + Gyroscope + Magnetometer | 15.17–31.10 |
| GPS + Magnetometer | 13.77–28.40 |
| Accelerometer + Gyroscope + Magnetometer | 1.70–4.37 |
| Accelerometer + Gyroscope | 1.43–2.97 |
| Accelerometer + Magnetometer | 0.30–1.67 |
| Gyroscope + Magnetometer | 1.67–4.10 |

Table 20: Total Battery Drain for One Hour of Continuous Sensor Combinations at 3.7 V

For combinations with approximate similar accuracies, power consumption served as a tiebreaker to optimize energy efficiency.

Final Selection The GPS + Accelerometer combination was selected as the optimal sensor set, despite the slightly higher accuracy of GPS + Accelerometer + Gyroscope (92.45% versus 91.73%). This decision was based on the following rationale:

1. The combination of GPS + Accelerometer achieved an accuracy of 91.73%, only 0.72% lower than the three sensor combination with highest performance. This marginal difference is unlikely to significantly impact the performance of fraud detection in practical scenarios.
2. Using two sensors instead of three simplifies the implementation, reduces computational overhead, and minimizes power consumption on mobile devices.
3. The GPS + Accelerometer combination drains maximum 27.27 mAh per hour at 3.7 V, compared to maximum 29.97 mAh per hour for GPS + Accelerometer + Gyroscope, due to the gyroscope's additional 2.7 mAh per hour Texas Instruments (2020), STMicroelectronics (2021). Due to high energy demands of GPS, pairing it with the low drain accelerometer optimizes energy efficiency, making GPS + Accelerometer more suitable for mobile use.
4. Both GPS and accelerometer are standard, widely available sensors in smartphones, ensuring broad applicability.

The GPS + Accelerometer combination thus represents the optimal balance of high classification accuracy, minimal sensor usage, and energy efficiency, aligned with the practical requirements of mobile-based fraud detection.

A scatter plot was generated for the validated threshold values of the best sensor combination. Figure 29 shows the plot, with points colored by movement type, highlighting the distinctions between legitimate and fraudulent movements.

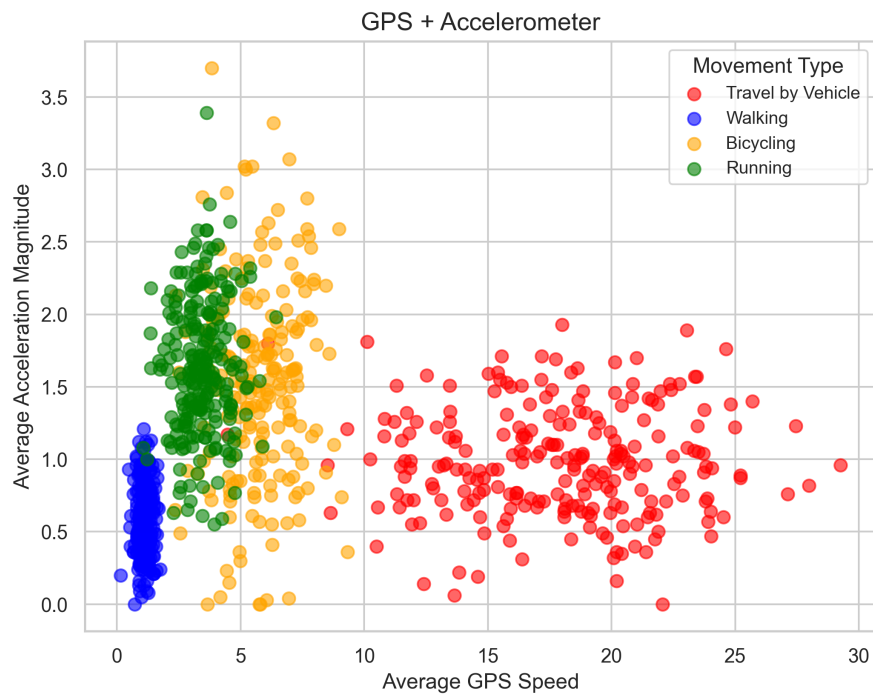


Figure 29: Scatter plot of average_gps_speed vs. average_acceleration_magnitude for GPS & Accelerometer sensor combination

Chapter 7 - Conclusion

Conclusion about the Research Problem

This project was carried out mainly to build a working system for the PoHM algorithm. By tackling multiple practical applicability challenges, we have come up with a working system for PoHM algorithm. In addition to practical implementation, the system was enhanced by applying a better short-range communication method, integrating processes to make the system more scalable when the number of nodes increases and introducing fraudulent movement detection thresholds. For short-range communication method, the Quickshare feature built in Android is used for file transfer between mobile phones, due to issues in Bluetooth and Wi-Fi Direct. Quickshare can be used with both Bluetooth and Wi-Fi; therefore, leverages the positive attributes of both Bluetooth and Wi-Fi. To make the PoHM more scalable, Brotli compression with a validator group mechanism which demonstrated the lowest average decreasing percentage of 26% among others, was utilized for final implementation. To identify the optimal sensor set to detect fraudulent movements in PoHM ticket generation process GPS and Accelerometer combination selected as the optimal set due to its higher accuracy and lower power consumption. And statistical thresholds for selected sensor set were derived to enable effective fraud detection.

In summary, our research has contributed to the advancement of the sustainability of blockchain applications by implementing and enhancing the PoHM algorithm. Moving forward, these findings can be utilized to further enhance the performance, scalability and usability of the PoHM applications.

Limitations

Mobile Application

- The primary limitation of the mobile app is the absence of in-built/ integrated support for Bluetooth or Wi-Fi Direct method. Implementer should carefully consider the Android API version differences and availabilities, and

upgrades in each Android API levels. Short-range communication method like Ultra-Wideband Module (UWB) would be a promising alternative.

- Another limitation is the use of two devices. Mobile phone is needed for mobility tracking and a Laptop or a desktop is needed for the blockchain.
- Users may reside in different locations which are far more apart from others(Overseas, rural areas, etc.). Initially, these users cannot generate tickets in the network.
- Mobile application cannot control file sharing activity through short-range communication. Shared files will be saved into predefined folder(Downloads, Documents) in mobile device and the mobile app has to access the specific folder and modify/replace/delete the file.
- Location latitude and longitude may change depending on signal interference and satellite data acquisition methods. Hence for the same location latitude and longitude may differ slightly.

Blockchain Node

- Current implementation relies on Libp2p default broadcast nodes to discover peers. If at a time those peers are down or struggling with high network traffic, peer discovery will be affected. Better to have self-hosted servers for these peer discovery purposes.
- For network relaying purposes, an EC2 instance was used, and assumed that it is always live. But it is not the actual case, therefore, dedicated relay nodes should be used.
- At least a single node should always be online to keep the blockchain network going correctly with consistent synchronization.
- Scalability evaluation was carried out on a maximum of 10 nodes. Better to perform evaluation on more nodes.

Optimal Sensor Selection

- The data set used of 1000 samples may not capture all movement scenarios, such as different environmental conditions or user groups.
- The mean \pm standard deviation approach may miss edge cases, particularly for overlapping movement patterns such as running and cycling.
- Power consumption values were based on typical specifications, instead of device-specific data.
- Sensor data, especially collected from GPS and magnetometer, is vulnerable to noise from urban interference or magnetic disturbances.
- Variations in smartphone sensor quality may affect data consistency.

Future Work

- Integrated Bluetooth or Wi-Fi Direct service, carefully designed for the use case.
- Integration of UWB short-range communication method.
- Careful handling of odd-number arrival issue occurs in the ticket generation process.
- Implementing a synchronized version of the application where the mobile application and the blockchain node are always connected
- Providing wallet facility to the mobile application.
- Evaluations can be carried out on more than 10 nodes to verify the scalability strategy decision taken.
- Research algorithms to make sure two users have a higher chance of meeting at a generated location.

- Introducing an incentive mechanism to validator group members for their effort in verifying the nodes on behalf of everyone.
- Implement a more controllable short-range communication method.
- Improving the validator group election logic to work around when unstable node counts exist and non-retained/newly selected nodes are not online.
- Gather additional samples across varied demographics, environments, and movement conditions to improve model robustness.
- Investigate other machine learning algorithms, such as neural networks or anomaly detection, to improve classification accuracy and address potential overfitting.

Chapter 8 - Peer Evaluation

In this dissertation, all three team members contributed to finding an approach to achieve the proposed goal outlined in the problem definition. Each member implemented different components of the project and some components were implemented collaboratively by multiple members or the entire team.

1. D.M.H.P. Dissanayake

- Implementing the PoHM consensus protocol
- Implementing the desktop blockchain application and integrating the consensus protocol
- Researching on using compression methods and a validator group mechanism to reduce block transfers to make the PoHM algorithm more scalable
- Integrating and evaluating the above strategies to find the best strategy
- Integrate the chosen Brotli compression with the validator group strategy into the final version of the system

2. M.S.W. Salgado

- Implement the mobile application
- Implement the location generation
- Connect mobile application to the desktop application
- Research quantitatively and qualitatively and identify possible candidates for short-range communication methods and filter them out according to feasibility.
- Test, compare, and find the limitations of the results gathered from surveys and research.
- List down the limitations of the possible short-range methods and find an alternative method to implement

- Integrate the chosen short-range method into the final implementation and make suitable changes to the ticket generation and verification

3. **S.N.S Wickramasinghe**

- Implement the PoHM consensus protocol
- Implement a mobile application to gather sensor data (GPS, accelerometer, gyroscope, magnetometer) and corresponding movement types in real world scenarios, gathering approximately 1000 samples to support robust analysis.
- Implement an initial manual analysis using scatter plots to explore movement patterns
- Evaluate 11 sensor combinations using Random Forest classifier to identify optimal sensor set and derived statistical thresholds for fraud detection from legitimate movement patterns.
- Integrate the chosen sensor combination into the mobile application to identify fraudulent movements in the network when the ticket generation occurs.

References

- Alakuijala, J., Farruggia, A., Ferragina, P., Kliuchnikov, E., Obryk, R., Szabadka, Z. & Vandevenne, L. (2018), ‘Brotli: A general-purpose data compressor’, *ACM Transactions on Information Systems (TOIS)* **37**(1), 1–30.
- Cason, D., Fynn, E., Milosevic, N., Milosevic, Z., Buchman, E. & Pedone, F. (2021), The design, architecture and performance of the tendermint blockchain network, in ‘2021 40th International Symposium on Reliable Distributed Systems (SRDS)’, IEEE, pp. 23–33.
- Fu, X., Wang, H. & Shi, P. (2021), ‘A survey of blockchain consensus algorithms: mechanism, design and applications’, *Science China Information Sciences* **64**.
- Gallersdörfer, U., Klaaßen, L. & Stoll, C. (2020), ‘Energy consumption of cryptocurrencies beyond bitcoin’, *Joule* **4**.
- Honeywell (2017), HMC5883L 3-Axis Digital Compass IC Datasheet, Technical report, Honeywell.
- URL:** <https://www.honeywell.com/content/dam/honeywell/en-us/documents/aerospace/HMC5883L-3-Axis-Digital-Compass-IC-Datasheet.pdf>
- Huang, J., Kong, L., Dai, H. N., Ding, W., Cheng, L., Chen, G., Jin, X. & Zeng, P. (2020), ‘Blockchain-based mobile crowd sensing in industrial systems’, *IEEE Transactions on Industrial Informatics* **16**. MCS;br/¿BMCS;br/¿why MCS bad;br/¿blockchain in MCS.
- Karger, E., Jagals, M. & Ahlemann, F. (2021), ‘Blockchain for smart mobility-literature review and future research agenda’, *Sustainability (Switzerland)* **13**.
- Kongahage, M., de Zoysa, D. & Sayakkara, D. (2022), ‘Proof of human mobility: A novel permissionless consensus algorithm for blockchains, based on human mobility’.

- Lin, I. C. & Liao, T. C. (2017), ‘A survey of blockchain security issues and challenges’, *International Journal of Network Security* **19**. Forks ...
- Lu, L. & Hua, B. (2019), G-match: a fast gpu-friendly data compression algorithm, *in* ‘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)’, IEEE, pp. 788–795.
- Rauschert, P., Klimets, Y., Velten, J. & Kummert, A. (2004), Very fast gzip compression by means of content addressable memories, *in* ‘2004 IEEE Region 10 Conference TENCON 2004.’, Vol. 500, IEEE, pp. 391–394.
- Saad, S. M. S. & Radzi, R. Z. R. M. (2020), ‘Comparative review of the blockchain consensus algorithm between proof of stake (pos) and delegated proof of stake (dpos)’, *International Journal of Innovative Computing* **10**(2).
- STMicroelectronics (2021), LSM6DSOX iNEMO Inertial Module Datasheet, Technical report, STMicroelectronics.
URL: <https://www.st.com/resource/en/datasheet/lsm6dsor.pdf>
- Texas Instruments (2020), WL1835MOD WiLink™ 8 Single-Band Combo Module Datasheet, Technical report, Texas Instruments.
URL: <https://www.ti.com/lit/ds/symlink/wl1835mod.pdf>
- Warmke, C. (2024), ‘What is bitcoin’, *Inquiry* **67**(1), 25–67.
- Werth, J., Berenjestanaki, M. H., Barzegar, H. R., El Ioini, N. & Pahl, C. (2023), ‘A review of blockchain platforms based on the scalability, security and decentralization trilemma.’, *ICEIS (1)* pp. 146–155.
- Wüst, K. & Gervais, A. (2017), ‘Do you need a blockchain?’, *IACR Cryptology ePrint Archive* .