

Consensus-Based Cooperative Location Proof for Untrusted Devices

K.K.W. Vishwajith



Consensus-Based Cooperative Location Proof for Untrusted Devices

Author

K.K.W. Vishwajith

Index Number : 19001789

Supervisor: Dr. T.N.K. De Zoysa

Co-Supervisor: Dr. A.P. Sayakkara

May 2024

Submitted in partial fulfillment of the requirements of the
B.Sc in Computer Science Final Year Project (SCS4224)



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 consent for my dissertation, if accepted, to be made available for photocopying and inter-library loans, and for the title and abstract to be made available to outside organizations.

Candidate Name:


.....
Signature of Candidate


Date: 10/01/2024

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

K.K.W. Vishwajith

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

Principle/Co- Supervisor's Name: Dr. Asanka P. Sayakkara / co-supervisor


.....
Signature of Supervisor

Date: 2024-04-22

List of Figures

1	GSM tower acting as a proximity detector of Cellular Devices.	6
2	RSSI propagation method	7
3	Angulation	8
4	Summary of Localization Techniques that are used for localization. (Zhou et al. 2018)	9
5	Highlevel Architecture of the protocol	10
6	Angle on Arrival (AoA)	11
7	Angle of Departure (AoD)	11
8	Node-to-Node workflow.	12
9	Untrusted Node beacon	12
10	Witness Node beacon	13
11	Witness Node to overlay network	13
12	Overall protocol	16
13	Random Scenario Creator	17
14	OMnet++ Protocol Simulation	19
15	OMnet++ Protocol Simulation	21
16	Unlocated Node to Witness Node (AVISPA)	23
17	Witness Node to Witness Node (AVISPA)	24
18	With Fewer Nodes picture 1.	25
19	With Fewer Nodes picture 2.	25
20	Distance vs Packet Lost	26
21	TxPower over Range	27
22	OMnet++ Mobility Test	28

Acknowledgment

I would like to express my sincere gratitude to my supervisor, Dr. Kasun De Zoysa and my co-supervisor Dr. Asanka Sayakkara for their continued support of my study for the whole year, their patience, motivation, and immense knowledge. Their guidance allowed me to improve my thinking, writing, and presentation skills. Besides my supervisors, I would like to thank Dr. Manjusri Wickramasinghe for supporting my literature review. Also, my sincere thanks go to my batch-mates who helped to succeed in this project by allowing me to discuss problems that arose while continuing this research. Last but not least, I would like to thank my family members who supported me in all other ways during my hard times.

Abstract

Modern Location-based Services depend on the user's honesty to provide the benefits. Since the location is determined mainly using the user devices, adversarial actors abuse these services for their intent. To address this problem we propose a location-proof method that'll help Location-based Services to serve genuine users of the service positioned in a particular position at a given time. The proposed solution verifies user location by nearby communication method giving proofs of location tied to a boundary provided by the transmission channel. With the range of 30 meters to 50 meters per location-proof, We also address the issues created by having this kind of location method like privacy and security. Verified by Automated Validation of Internet Security Protocols and Applications (AVISPA), we prove that attacks on the protocol can be easily mitigated. We show that the proposed solution is cost-effective since it does not need additional infrastructure and modification to the hardware.

List of Acronyms

Acronyms

AP Access Points. 5

GPS Global Positioning System. 4, 5

IMU Inertial Measurement Unit. 2, 6

IoT Internet-of-Things. 1

LBS Location-Based Services. 1, 9, 10, 14, 15

NLOS Non-Line-of-Sight. 1

RSS Received Signal Strength. 6

RSSI Received Signal Strength Indicator. 2

TDOA Time Difference of Arrival. 1

TOA Time of Arrival. 1, 2

TOF Time of Flight. 1, 2

VANET Vehicular Ad Hoc Network. 4

WSN Wireless Sensor Networks. 1

Contents

List of Figures	ii
List of Acronyms	v
Table of Contents	vii
1 Introduction and Background	1
1.1 Problem Statement	2
1.2 Research Questions	2
1.3 Research Aim and Objectives	2
1.4 Objectives	3
1.5 Scope	3
1.6 Contributions	3
2 Literature Review	4
2.1 Cooperative Methods of Location-Proof	4
2.1.1 Consensus-Based Location-Proof	4
2.2 Alternative Methods of Localization other than GPS	5
2.2.1 Technologies of Localization	6
2.2.2 Summary of Localization Techniques	6
3 Methodology and Research Design	9
3.1 Design Outline	9
3.2 Selecting a Communication Method for Nearby Communication	10
3.2.1 Importance of Bluetooth or Bluetooth Low Energy	10
3.2.2 Methods of Localization via Bluetooth	10
3.2.3 Design Decisions	11
3.3 Protocol Overview	11
3.3.1 Mobile Node-to-Node	12
3.3.2 Design Decisions	13
3.3.3 Location Proof on Decentralized Network	14
3.3.4 Certificate Authority on Decentralized Network	14
3.3.5 Rewards on Decentralized Network	15
3.3.6 Escrow on Decentralized Network	15
3.3.7 Design Decisions	15
3.4 Conclusion	15
4 Implementation	16
4.1 Introduction	16
4.2 Implementing a Testbed for Mobile Nodes	16
4.2.1 MATLAB	16
4.2.2 OMNet++	18
4.3 Implementing an Overlay Network	22
4.3.1 Design Decisions	22
4.4 Implementing Formal Verification of the Protocol	22
5 Results and Evaluation	23

5.1	Privacy and Security Validation	23
5.2	Simulation Implementation	24
5.2.1	Minimum and optimum number of nearby nodes as witnesses to correctly prove a location claim	24
5.2.2	Impact on distance for signal propagation over nearby nodes . . .	25
5.2.3	Impact on Transmitting Power over Range	26
5.2.4	Effective distance on the location claim	27
5.2.5	Impact on the mobility of nodes to the Location claim	27
5.2.6	Effective Throughput of location verification tokens	28
6	Conclusion	28
A	Mobile Node TestBed	29
A.1	MATLAB	29
A.1.1	Generating Random Senarios	29
A.1.2	Analysing Path Loss and Distance vs Transmission Power	32
A.1.3	Statistics on Distance vs Packet Loss	35
A.2	OMNet++	36
A.2.1	Witness.cc: Code for witness	36
A.2.2	Unlocated.cc: Code for Unlocated Node	38
A.2.3	Code for Simulation	39
B	Overlay Network	41
B.1	Location-proof Contract	41
	References	44

1 Introduction and Background

Location proofs are essential in the current technological world with the usage of high-performing mobile computing and Location-Based Services (LBS). The dependence on smartphones and smart devices has been increasingly growing and it has provoked the LBS technology sector to come up with reliable ways to secure the information given by these user devices. LBS is useful for many related areas such as ad hoc networks, robotics, Wireless Sensor Networks (WSN), Navigation and Tracking, Internet-of-Things (IoT), military, and aviation.

Location proof usually depends on the information given by the user device to the Location-Based Services (LBS). Localization is the academic research field of the methods of identifying the position of a device. Two domains of localization have been considered in the literature:

- Outdoor localization
- Indoor localization

Outdoor localization refers to the process of determining an object or a person's position in an open environment. Outdoor localization can be traditionally done via GPS, as every new smartphone, vehicle, and even a tiny Internet-of-Things (IoT) device can be equipped to have a GPS receiver. But GPS can be troublesome when there are *interference, multi-path, and ionospheric scintillation* (Hegarty and Kaplan 2005). Thus making GPS unreliable even in urban outdoors, where the GPS receiver is in a Non-Line-of-Sight (NLOS) situation. As an alternative to GPS, other techniques like Proximity Detection, RSSI radio propagation, Angulation, and Time-Based techniques such as Time of Arrival (TOA), Time Difference of Arrival (TDOA), Time of Flight (TOF) and Dead Reckoning. The accuracy of those techniques can be improved by using a hybrid of either technique and using machine learning (Acar et al. 2022, Yadav, Sharma, and Rishiwal 2022).

When it comes to indoor localization the *interference* and *multipath* problems become worse. GPS requires at least four beacons to triangulate a position. The Indoor environment has obstructions to the signals such as concrete and glass. Signals would have problems. Plenty of alternatives exist with or without extra infrastructure for accurate localization. Which we will discuss in the literature review.

Both localization techniques depend on the user's device to attest that location proof is genuine. This is a disadvantage for Location-Based Services as the adversaries can abuse the service and take advantage of it. We have found that using natural human mobility in an urban, suburban area can assist as a location proof for a user device by testifying that the user has been in a location at a certain time. Using this location claim with additional witnesses as proof, LBS can provide services for the intended user with confidence that was not available previously.

There are Location-Based Services (LBS) that use User localization for providing services, such as Social Networks (Bao et al. 2015), Block-chain Applications (P 2021, Wu et al. 2020, Amoretti et al. 2016). Civil GPS is unencrypted, leading to fraud, spoofing, and jamming Scoles 2018. In a situation where the devices and the infrastructure are tampered with by malicious actors, falsified positions could be determined by the system. This need for a localization method that can determine an object, or a person's position with reasonable accuracy without having to trust the device and infrastructure is the main

goal of this proposal.

1.1 Problem Statement

Location Based Services are important in both industrial services and public usage. Users get personalized experiences based on their location, helping them to discover nearby points of interest, and navigate unfamiliar territories.

Getting a location by trusting a smartphone device can lead to a major security flaw in the system, because GPS signals can be spoofed, and used for malicious intent by the adversaries. (Songala et al. 2020, Papadimitratos and Jovanovic 2008, Psiaki and Humphreys 2016, Nighswander et al. 2012)

The unreliability of GPS on smartphones can be problematic for applications that rely on the accuracy and validity of positioning. Examples of such applications include Digital Twins (Tao et al. 2019), where the user has a virtual equivalent in a physical world (Keqin Shi, Qian, and Yu 2022), or a location-based access control for sensitive data (Cleeff, Pieters, and Wieringa 2010, Androulaki et al. 2014). The same problem affects the Proof-of-Location-based blockchain technologies such as *FOAM*¹, *SIORKA*², *Platin*³, where the technology itself has to trust the location given to it by the host device.

1.2 Research Questions

Q1: How does device mobility affect the accuracy and performance of a cooperative location-proof infrastructure?

Hypothesis: As the target device or the user moves some other nodes in the network will still be within the range and can come to a consensus by communicating with each other on an ad hoc network.

Q2: How to design a cooperative location-proof method that makes use of existing technologies of smartphones that can withstand device mobility?

Hypothesis: Smartphones have various sensors (i.e. Inertial Measurement Unit (IMU)) and transceivers (Wi-Fi, GSM, LTE). From these technologies and by incorporating Received Signal Strength Indicator (RSSI), time-based techniques such as Time of Arrival (TOA) and Time of Flight (TOF), smartphones are capable of determining their position. By utilizing alternative localization methods of the neighbor nodes, the target position can be localized. Furthermore by using nearby communication technologies like Wi-Fi and Bluetooth (Ippisch, Sati, and Graffi 2017), the adjacent devices can serve as witnesses to prove the location of a smartphone.

1.3 Research Aim and Objectives

The research aims to implement a cooperative localization method that incorporates mobile agents, that come to a consensus on a targeted device's position in a cooperative manner.

¹<https://foam.space/>

²<https://web.archive.org/web/20171113232142/http://sikorka.io/>

³<http://web.archive.org/web/20181201025913/https://platin.io/>

1.4 Objectives

- Study how a cooperative location-proof would work where the nodes are mobile.
- Understand the current alternative localization methods for mobile nodes.
- Implement a cooperative location-proof protocol for mobile nodes.
- Study and implement countermeasures for byzantine tolerance for such cooperative location-proof.
- Analyse the security and privacy of the nodes' location-proof.
- Evaluate the performance of such implementation.

1.5 Scope

In Scope

- Develop a communication protocol for smartphone devices to come to a consensus on a device's location-proof.
- Analyse the security of such protocols such as users' privacy.

Out Scope

- Improvements to existing Cooperative Localization Protocols.

1.6 Contributions

Presented Research in the thesis makes the following contributions,

- Blockchain-based Cooperative Location-proof method for generating location claims for user devices.
- Privacy-aware, decentralized trust for the users, thus this method does not depend on a centralized service such as a Certificate Authority.
- Solution incentivizes users to participate for a reward, but also takes into account possible abuse of the method.
- Tests the security and privacy of the solution with formal verification.

2 Literature Review

To create an accurate measurement of the location-proof of a device, we should look into how traditionally location is gathered using portable devices, such as smartphones. Global Positioning System (GPS) is popular and widely used for location tracking because even low-end devices have inbuilt GPS receivers.

However, GPS needs a clear line-of-sight to the satellites with fewer nearby disturbing objects, such as buildings or trees, while having good weather conditions (Cui and Ge 2001). If the user continues to use GPS to locate the position, it could drain the battery faster (Gaonkar and Choudhury 2007). Furthermore, GPS can be spoofed (Oligeri et al. 2019), on a device level and a regional level (due to weather conditions or planned signal inference).

2.1 Cooperative Methods of Location-Proof

Most of the research is done on the topic of indoor localization because of the limitations of GPS. However, having computation on a single device and relying on output is less realistic in the need for high-accuracy localization. Ideally in a GPS-free localization, there have to be self-location-aware nodes (such as LTE base stations) that cover the area with either high power or high-density deployment (Shen, Wymeersch, and Win 2010). In a real-world scenario, this is not practical.

Cooperative localization can help to improve the accuracy of a single node's location by communicating with the surrounding nodes. Rohani et al. 2015 does this localization adjustment by using the built-in GPS and estimating other vehicles using other methods in a Vehicular Ad Hoc Network (VANET). Yin et al. 2020 uses federated learning to create a framework for cooperative localization. The authors have also created a low-sampling GPS to create a less-resourced incentive method of localization (J. Qi et al. 2019).

Cooperative Location-Proof is the method of obtaining a location-proof with the help of nearby nodes of the network. Javali et al. 2016 uses a centralized architecture to store and verify the location proofs, and utilizes channel state information used in WiFi to decide the location proximity. In the works of Talasila, Curtmola, and Borcea 2012, they use Centralized Certificate authority to manage the trust of the network and use a single certificate of trust unique to each node, making it possible to track a particular device/person utilizing the location claims. Couderc and Maurel 2019, Pham et al. 2016, W. Luo and Hengartner 2010 use the additional infrastructure as Javali et al. 2016, making these implementations costly for commercial implementation. 1 summarizes the advantages and disadvantages of related work.

2.1.1 Consensus-Based Location-Proof

In cooperative localization, nodes learn their position or state of the entire network cooperatively. In a consensus cooperative location-proof, the nearby nodes help the needy in their localization process, by either passing on information or helping estimate the position. This area is still in development (Olfati-Saber, Fax, and Murray 2007, Soatti et al. 2017). Usage of this kind of localization method can create an accurate position without trusting the device itself because the device's location comes from the consensus

Paper	Features	Advantages	Disadvantages
Javali et al. 2016	2 Step Location-proof Generation and Verification	Nontransferable Location-Claim	Centralized AP infrastructure for verification
W. Luo and Hengartner 2010	Depends on a Trusted Third party Infrastructure	Uses Public Key Infrastructure	Centralized AP CA for proofs
Couderc and Maurel 2019	Similar to Wardriving WiFi APs	Uses already deployed Infrastructure	Has to Preprocess an new environment to gather new data points.
Zhu and Cao 2011	Nearby Communication Vouching	Uses already deployed Infrastructure Privacy-aware	Communication overhead
Davis, H. Chen, and Franklin 2012	Privacy Aware scheme	Uses PKI	Requires a secured PKI
Reza Nosouhi et al. 2018	Privacy Aware scheme	No Databases	Weak Prover-Witness verification
Gambs et al. 2014	Zero Knowledge Proofs	Privacy Aware	Weak Prover-Witness verification

Table 1: Location-proof work with Cooperative Localization

of the nearby network. Another usage of Consensus in the location is that it can be used in storing the location claims and act as a verification platform. In our research, we also use consensus as a decentralized Certificate Authority.

2.2 Alternative Methods of Localization other than GPS

In this section, we discover alternative technologies for localization without relying on GPS. While these methods are not as accurate as GPS, the discovery of the technologies helps in emergencies.

Sensor-based Localization: By leveraging built-in wireless transceivers as sensors in the mobile phone, such as WiFi and Bluetooth, and also by augmenting certain sensors such as accelerometer and gyroscope, one can create an algorithm that can localize the person carrying the device.

For example Y.-C. Cheng et al. 2005 uses war-driving (a method of driving around a certain area while capturing nearby WiFi access points, with a GPS-enabled device to map the location of the WiFi Access Points (AP)). The data of the war driving is used to create a map with the locations of the WiFi APs. Also, they have used the signal strengths of these WiFi APs to predict the locations. Their research concludes that using WiFi localization in an urban area gives a much more accurate location than when using it in a suburban area. Research by LaMarca et al. 2005 uses not just WiFi APs, but a collection of WiFi APs, GSM cell towers, and Fixed Bluetooth devices to pinpoint the location. Using these hybrid methods they have shown much more accurate localization on mobile devices.

The research by Abdelnasser et al. 2016, utilizes inertial sensors (i.e., accelerometer, compass, and gyroscope) along with the help of WiFi APs and cell tower information, to create a framework to detect the user’s pose and special locations where the user is subjected have a special pattern, e.g., climbing stairs. Aly, Basalamah, and Youssef 2015 uses inertial sensors to predict the lane of a moving vehicle on a highway. Aly and Youssef 2013 uses the same low-energy sensors to detect outdoor locations by collecting

data and locations that affect the inertial sensors, such as going inside tunnels, moving over bumps, going up a bridge, and even potholes. By crowd-sourcing such data and using them to predict the location, they have shown that it is possible to save energy usage by up to 300%.

Cellular-based Localization: Received Signal Strength (RSS) based or Cell-ID-based localization is an alternative method. Cell-ID uses the strongest cell tower location as the user's location (Shokry, Torki, and Youssef 2018). Similarly, RSS-based localization uses the signal strengths received by the mobile device. In Paek, K.-H. Kim, et al. 2011 uses a Cell-ID-based method to locate the position of the user. The research uses a sequence-matching technique, where the user's movement from point A to point B causing transitions between the cell towers, is matched with possible locations within the sequence of frequently traveled routes.

The benefit of using the RSS-based method is that the required initial infrastructure is already laid down by telecommunication service providers. After an initial data gathering of the area, RSS fingerprinting has to be done to get an accurate call back to the location. Paek, J. Kim, and Govindan 2010 uses RSS assistance when GPS is less reliable. Ibrahim and Youssef 2011 uses probabilistic models to further improve the accuracy of the model.

Finally, many technologies and techniques are used for localization as an alternative. Most prominent technologies include Bluetooth (Toyama et al. 2021), UWB (B. Li, Zhao, and Sandoval 2020), RFID (Zhang and al. 2019), ZigBee (C. H. Cheng and Syu 2021, Zhen et al. 2020), Wi-Fi (Gentner et al. 2020), LTE (Abdallah, Shamaei, and Kassas 2019, Gertzell and al. 2020), 5G, Inertial Measurement Unit (IMU) (H. Luo and al. 2021), and Frequency Modulation (FM) (Du et al. 2020). The techniques of localization using the above-mentioned technologies include calculations based on RSSI, AoA, ToA, and TDoA. Finally, to increase the accuracy Machine Learning techniques can be used with Path-Loss Modeling, Fingerprinting, and Proximity Analysis (Moradbeikie et al. 2021).

2.2.1 Technologies of Localization

2.2.2 Summary of Localization Techniques

Proximity Detection Proximity Detection uses detectors in known locations, and when the target device is inside the proximity detector, it will be considered as the device is within the general vicinity of the detector. This approach is widely used in GSM localization methods (DEMİR and ÖZTEKİN 2021).

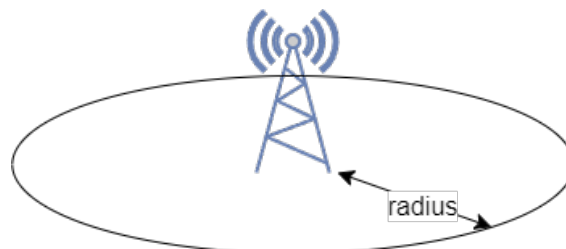


Figure 1: GSM tower acting as a proximity detector of Cellular Devices.

RSSI based RSSI-based techniques have two methods of approach in localization.

Signals	Technique	Range	Positoning error/m	Complexity	Robustness	Cost	Infrastructure
WiFi	Ranging Fingerprinting	Indoor	1–7	Medium	Depends on the positioning algorithm and fingerprinting database	Medium	WiFi infrastructure
GSM	Ranging Fingerprinting	Indoor/Outdoor	5	Medium	Depends on the positioning algorithm and fingerprinting database	Medium	Base station
Bluetooth	Ranging Fingerprinting	Indoor	2–5	Medium	Performance is sensitive to obstacles	Medium	Bluetooth tags
FM broadcast	Ranging Fingerprinting	Indoor/Outdoor	≤ 2	Medium	Depends on the positioning algorithm and fingerprinting database	Medium	Base station
Acoustic	Ranging Fingerprinting	Indoor/Outdoor	0.4	Low	Performance is sensitive to noise	Low	Acoustic sensors
Geo magnetic	Ranging Fingerprinting	Indoor	≤ 1	Medium	Performance is sensitive to metallic objects	Medium	Magnetic compass
Visible light	Ranging	Room	≤ 0.35	Low	Restriction on the number of LEDs	Low	LED lighting
Image	Image matching	Room	1–4	High	Performance is sensitive to scale, rotation, and illumination	High	Infrastructure free
Motion	Dead reckoning	Indoor	2–6	Low	Restriction on the number of landmarks	Low	Infrastructure free

Table 2: Summary of Wireless Technologies that are used for localization. Zhou et al. 2018

1. Fingerprint-based method - By collecting signal information of a position beforehand and later matching it with online measurements. There have been many researches done using WiFi RSSI data since it is easier to gather WiFi RSSI data using a smartphone. And with machine learning techniques the devices can recall the position approximately using the fingerprint data(Ssekidde et al. 2021).
2. RSSI radio propagation method - Using the known location of the base stations, the strength of the receiving signal, and the prior calculated path loss factor a device can calculate the distance between a Base station and itself. (Sohan et al. 2019)

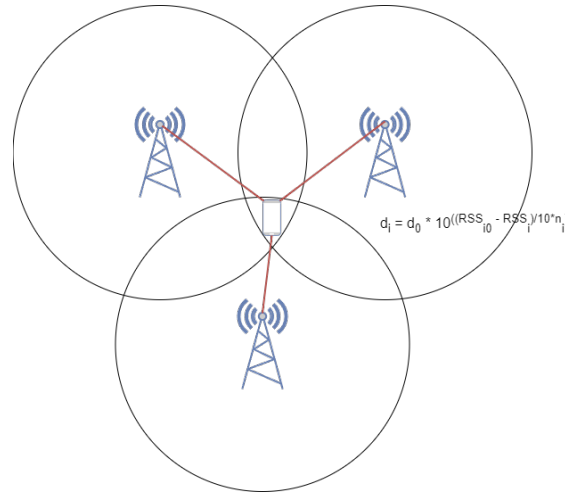


Figure 2: RSSI propagation method

Angle based A directional technique where base stations that are aligned with the north pole calculate the position of the device using triangulation (“Advances in VLSI, Communication, and Signal Processing” 2020).

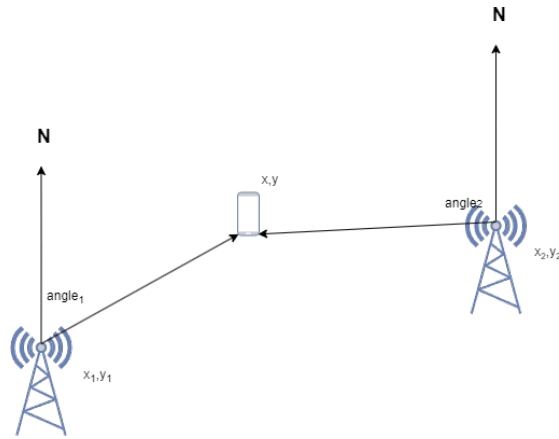


Figure 3: Angulation

Time-based Time-based techniques measure the distance between a target device and a base station using the signal propagation time. There are several subtechniques involved in the time-based method.

1. Time of Arrival (TOA)
2. Round Trip Time (RTT)
3. Time Difference of Arrival (TDOA)

Dead Reckoning Dead reckoning technique can be used alongside Smartphones IMU (Inertial Measurement Unit) that calculates device-specific force, angular rate, and sometimes the orientation of the body, using a combination of accelerometers, and gyroscopes (W. Li et al. 2021). It can estimate the position with step-length of a person by using the accelerometer and headed angle using a gyroscope.

Vision Based Useful in localizing in an indoor scenario where the already extracted information from images and videos can be matched with online information from a camera or any other vision sensor. Recently this technique has been used combined with deep learning/ machine learning techniques (Pino et al. 2019).

Summary of localization techniques

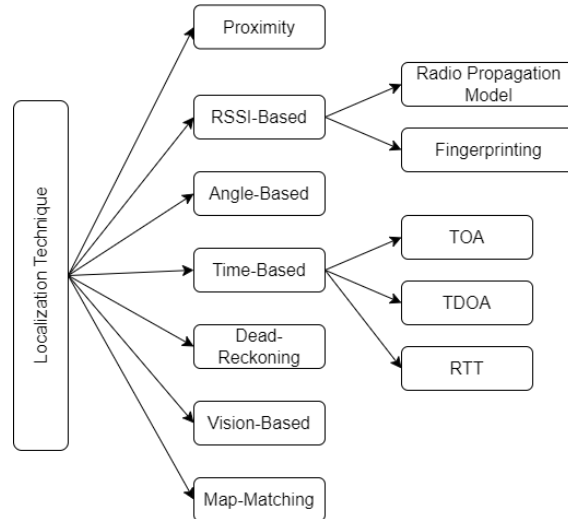


Figure 4: Summary of Localization Techniques that are used for localization. (Zhou et al. 2018)

3 Methodology and Research Design

The methodology used in the research will be a combination of both experimental and build methods. This research study will find solutions to the research questions then implementation will be made as a proof of concept. Therefore a mix of experimental and build methodologies is essential in this research process.

First, we developed a protocol for secure location proof. In addition to that we implemented the needed smart contracts for the location-claims and rewards given to the network. We also evaluated the performance of the network in a simulated environment created in both MATLAB and OMNET++. Finally, we checked it for Privacy and Security using AVISPA Project (*The AVISPA Project 2017*).

As mentioned in the problem statement, the objective of this research is to have a method of proving the location of a device along with time that can be verifiable. This can be archived by using cooperative localization focusing on the location proof.

Indoor localization usually relies on using dedicated or external infrastructure such as beacons (Kriz, Maly, and Kozel 2016) / WiFi access points to support the localization of a device. It is much easier in commercial deployment of such indoor localization as they can invest in external devices that can be used as fingerprinting or ranging mechanisms (LaMarca et al. 2005, Zhou et al. 2018).

Outdoor localization on the other hand uses infrastructure such as GSM, and WiFi access points to aid in the localization process. Also in outdoor localization, users can crowd-source the data to have the ability to localize the devices on their own.

3.1 Design Outline

The proposed System architecture has 4 types of entities as displayed in the 5:

- **User:** The client device that needs a location-proof to be presented to a Location-Based Services (LBS).

- **Witness:** Smart device that generates a proof using a nearby communication method to gain a reward.
- **Certificate Authority:** Smart Contract that is deployed to provide pseudo-anonymity to the network nodes by generating decentralized certificates.
- **Rewards:** Smart Contract that will reward a certain amount of tokens/coins to the participating users to incentivize them to use the system.
- **Claims:** Smart Contract that validates the location-proofs given by witnesses and gives a unique token to be presented to the Location-Based Services (LBS).

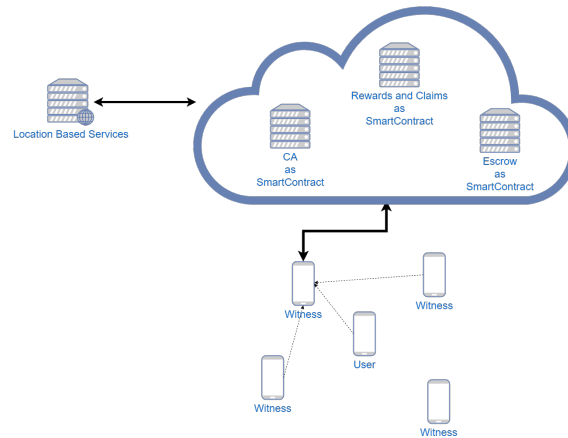


Figure 5: Highlevel Architecture of the protocol

3.2 Selecting a Communication Method for Nearby Communication

Using cooperative localization means passing messages among the network to gain knowledge about the target device and the nearby network. Higher cross-messaging between devices will lead to battery drain in most communication technologies like LTE, 3G, and WiFi. For this research we have focused on Bluetooth because it has low energy consumption on devices and is available on every smartphone in the market, making it ideal for cooperative localization communication among smartphones.

3.2.1 Importance of Bluetooth or Bluetooth Low Energy

As explained above, the main reason for choosing Bluetooth is the wide availability among devices and the low energy consumption. The recent advances in Bluetooth technology also make it compelling to use for research. The Bluetooth 4.0 version carried the ability to use the RSSI techniques to localize a device (H. Qi et al. 2021, Einavipour and Javidan 2021). In Version 5.1 upwards however can localize a device using AoA or AoD techniques (*Bluetooth Direction Finding: A Technical Overview — Bluetooth® Technology Website — bluetooth.com* 2019).

3.2.2 Methods of Localization via Bluetooth

From Bluetooth 5.1 onwards, there are 2 localization methods from the specification itself (*How AoA and AoD Changed the Direction of Bluetooth Location Services — Bluetooth Technology Website — bluetooth.com* 2023).

1. Angle on Arrival (AoA) - uses a single antenna transmitter with a receiver with multiple antenna arrays.

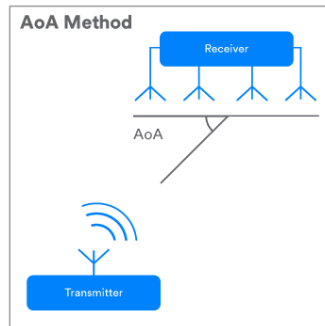


Figure 6: Angle on Arrival (AoA)

2. Angle of Departure (AoD) - uses a single antenna receiver with a transmitter with multiple antenna arrays.

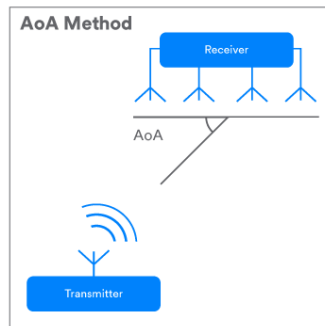


Figure 7: Angle of Departure (AoD)

3.2.3 Design Decisions

Even though Bluetooth 5.1 has already implemented technology to pinpoint the location of a device using AoA or AoD, practical usage scenarios need specialized hardware that requires multiple antennas (2 or more) to accurately localize a user device. Since the research is targeting a location proof on a much larger audience with regular off-the-shelf hardware, we decided that it's not feasible to rely on newer technology and be backward compatible.

3.3 Protocol Overview

Protocol is divided into two components.

1. Nearby Communication of the nodes - This will help to verify the claims of an untrusted device that it is situated in a (x,y) location
2. Location proof on a decentralized network - If the claim is verifiable with the help of nearby devices, a verified claim should be recorded somewhere that is decentralized achieving consensus among peers on the network.
3. Rewarding scheme to incentivize nodes in the network to participate in the service.

Let us discuss the overview of the protocol for each part.

3.3.1 Mobile Node-to-Node

In this section, the cooperative communication between the nodes and the untrusted device is explained.

There are 2 actors in this part of the protocol,

1. Untrusted node - The node that needs to verify its location by the network.
2. Witness nodes - Nodes that are in the vicinity of the Untrusted node.

The workflow will be as follows,

- The untrusted node will send a Bluetooth beacon that needs to be verified. For a certain time, this beacon will be broadcast in the Bluetooth mesh network.
- The receiving Witness nodes gather the untrusted nodes' beacon, and broadcast its witness ID for a certain time.
- The Witness nodes gather both the nearby witness node transmitted beacons and the untrusted nodes' initial beacons, package all of them, and send them over to the decentralized overlay network for processing.

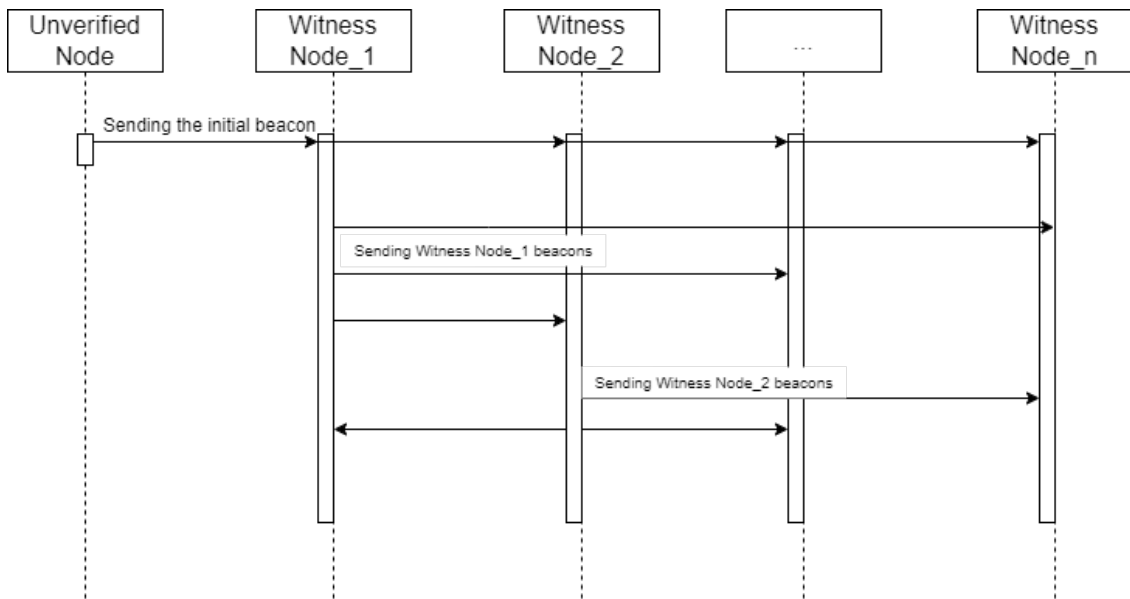


Figure 8: Node-to-Node workflow.

The beacon of the untrusted node will send out the following details packaged inside,

$$\left\{ \begin{array}{l} NodeID/UserID \\ \langle latitude, longitude \rangle \\ timestamp \\ ServiceID \\ Signature(NodeID) \end{array} \right\}$$

Figure 9: Untrusted Node beacon

NodeID will be a pseudonymous ID for giving the user node privacy for the location proof (L. Chen et al. 2017). This prevents a third party tracing back location claims to build a profile of a user. Longitude and Latitude will be given by the users GPS module, it is untrusted information as a default only to be verified by both witnesses and overlay network. Unix Timestamp will be there to aid in as a aid in location-proof. ServiceID is the service that user has to show the location claim once its verified. This ties each location claim to a service. Finally signature of the User given as a authentication method of the beacon.

The beacon of the witness node will have the same information, beacon with its nodeID. Signed by the witness for authentication.

$$\left\{ \begin{array}{l} \textit{NodeID/WitnessID} \\ \langle \textit{latitude, longitude} \rangle \\ \textit{untrusted_NodeID} \\ \textit{timestamp} \\ \textit{ServiceID} \\ \textit{Signature(NodeID)} \end{array} \right\}$$

Figure 10: Witness Node beacon

The witness beacon will consist of the information about,

1. Untrusted NodeID - Who requested the location claim?
2. ServiceID - What service this location claim will be presented?

The final package sent by the witness nodes to the overlay network will contain the following details. $\{\textit{Set_of_NearbyNodeIDs}\}$ will contain the information from the beacons of nearby nodes.

$$\left\{ \begin{array}{l} \textit{NodeID/WitnessID} \\ \langle \textit{latitude, longitude} \rangle \\ \{\textit{Set_of_NearbyNodeIDs}\} \\ \textit{untrusted_NodeID} \\ \textit{untrusted_payload} \\ \textit{timestamp} \\ \textit{ServiceID} \end{array} \right\}$$

Figure 11: Witness Node to overlay network

Algorithm for creating the $\{\textit{Set_of_NearbyNodeIDs}\}$

3.3.2 Design Decisions

When each node is registering a decentralized Certificate Authority will issue the node Node a set of randomly generated public key, private key pairs. These keys will be used to Sign the beacons transmitted over the network to ensure authenticity and can be used to

Algorithm 1 Creating $\{\text{Set_of_NearbyNodeIDs}\}$

```
WaitTime  $\leftarrow N$ 
Set_of_NearbyNodeIDs  $\leftarrow \{\}$ 
while WaitTime  $\neq 0$  do
    ReceivingWitnessBeacon  $\leftarrow \text{WitnessBeacon}$   $\triangleright$  Recieve a witness beacon
    Set_of_NearbyNodeIDs  $\cup \text{ReceivingWitnessBeacon}$ 
    WaitTime  $\leftarrow N - 1$ 
end while
```

encrypt sensitive data. Beacons tied with the serviceID guarantee issuance of a location claim that is uniquely generated for the Location-Based Services which it can trust.

3.3.3 Location Proof on Decentralized Network

This part of the protocol deals with the consensus of the protocol, after successful execution the untrusted node would have a verified token issued by the Decentralized network, that verifies the nodes' presence for a certain location at a particular time. After receiving the packages from the nodes, it processes them for the issuing of the verification token.

The workflow will be as follows,

- Gather the packages of Witness nodes
- Get the $\{\text{Set_of_NearbyNodeIDs}\}$, take the intersection of the individual sets. If there are no empty set with a qualified number of nodes (predetermined by the network), proceed.
- Check for the nearness of the location of the nodes in the set generated by step 2. If there is an anomaly, return, otherwise proceed.
- Issue a verification token for the location proof for the untrusted node.

Algorithm 2 Decentralized Network Procedure: Creating the set of nearby nodes

Require: *ReievedWitnessBeacons*

Ensure: $\text{Size}(\text{ReievedWitnessBeacons}) \geq 3$ \triangleright Due to byzantine fault tolerance, we need at least 3

$N \leftarrow \text{Size}(\text{ReievedWitnessBeacons})$

Set_of_NearbyNodes $\leftarrow \{\}$

for *int* $i=N$; $i > 0$; $i--$ **do**

$\text{Set_of_NearbyNodes} \leftarrow \text{Set_of_NearbyNodes} \cap \text{ReievedWitnessBeacons}[i].\text{Set_of_NearbyNodeID}$

end for

3.3.4 Certificate Authority on Decentralized Network

One of the drawbacks in the current literature is that they rely on a centralized architecture for managing trust (Javali et al. 2016, W. Luo and Hengartner 2010, Pham et al. 2016). This introduces a single point of failure for confidence and creates an unbalanced nature to the method proposed. We have proposed a decentralized Certificate Authority for managing trust and issuing pseudo-anonymous keys to the nodes of the network.

Algorithm 3 Decentralized Network Procedure: Checking for location anomalies

Require: *Set_of_NearbyNodes*

Ensure: $Size(Set_of_NearbyNodes) \geq 0$

$N \leftarrow Size(Set_of_NearbyNodes)$

$UntrustedNode \leftarrow Set_of_NearbyNodes.UntrustedNode$

$VerifiedNodes \leftarrow 0$

for int $i=N$; $i > 0$; $i--$ **do**

if **CheckforPositionAnomalies**($UntrustedNode.location, Set_of_NearbyNodes[i].location$)
 then

$VerifiedNodes++$

Continue

else

Break

end if

end for

IF(VerifiedNodes > 3)...Continue to Issue the verification token

3.3.5 Rewards on Decentralized Network

Rewards are given to the participating witness nodes who successfully create a location-proof request to the decentralized service. They will be given a token that can be spendable just as a coin on a regular decentralized currency.

3.3.6 Escrow on Decentralized Network

To reduce the abuse of the network by Sybil attack with multiple identities of witnesses and user nodes (Douceur 2002), collaborate to harvest the rewards given out to the legitimate location-proofs. By using an escrow method we force the witnesses to only get paid when a user device redeems the location-claim to a Location-Based Services. This criteria prevents above mentioned attack vector from an adversary.

3.3.7 Design Decisions

The rewards given to the witnesses were regular tokens/coins on a decentralized platform. The location claims are generated unique and non-transferable by design. This decision makes the user tied to redeeming the location-based service.

3.4 Conclusion

The overall protocol is illustrated below as in figure 12.

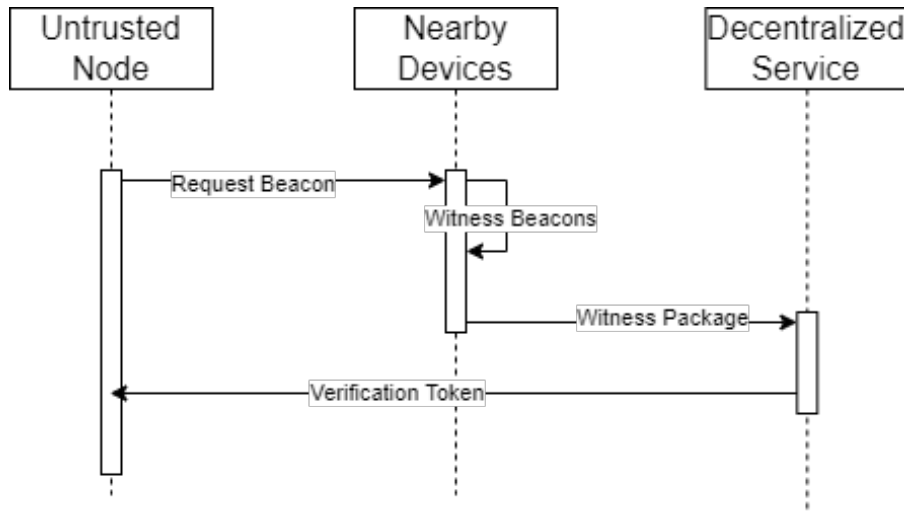


Figure 12: Overall protocol

4 Implementation

4.1 Introduction

The main part of the research will be stimulating the protocol and measuring its accuracy and the throughput of it. In this section, the tools used in the simulation will be discussed. Results will be discussed in the *Results and Evaluation section*. We will discuss the protocol in two parts,

1. Mobile Nodes
2. Decentralized Overlay Network

4.2 Implementing a Testbed for Mobile Nodes

A viable protocol simulation testbed was created to evaluate the protocols' performance and reliability. There are 2 testbeds designed to evaluate the protocol,

1. MATLAB (Inc. 2022)
2. Omnet++ (*OMNeT++ Discrete Event Simulator* — omnetpp.org n.d.)

4.2.1 MATLAB

For the first test MATLAB was to create random scenarios with mobile devices that have Bluetooth Low Energy enabled. The simulation was intended to visualize the connections between a possible range and number of nodes in a simulated scenario that a protocol device can connect.

Listing 1: Implementation of random scenarios in MATLAB

```

selectRunType = 4;
disp(selectRunType);
if(selectRunType == 1)
    totalNodes = 4;
  
```

```

fprintf('Total nodes = %d\n',totalNodes);
meshNodesPositions = [50 50; 40 40; 60 60; 50 60; 80 90];
disp(meshNodesPositions);
elseif(selectRunType == 2)
totalNodes = 6;
fprintf('Total nodes = %d\n',totalNodes);
meshNodesPositions = [50 10; 45 30; 60 60; 50 60; 25 10; 10 20; 70 80];
disp(meshNodesPositions);
elseif(selectRunType == 3)
totalNodes = 6;
fprintf('Total nodes = %d\n',totalNodes);
meshNodesPositions = [0 0; 45 30; 60 60; 50 60; 25 10; 10 20; 70 80];
disp(meshNodesPositions);
else
totalNodes = 25;
fprintf('Total nodes = %d\n',totalNodes);
meshNodesPositions = randi([0,100],totalNodes+1,2);
disp(meshNodesPositions);
end

```

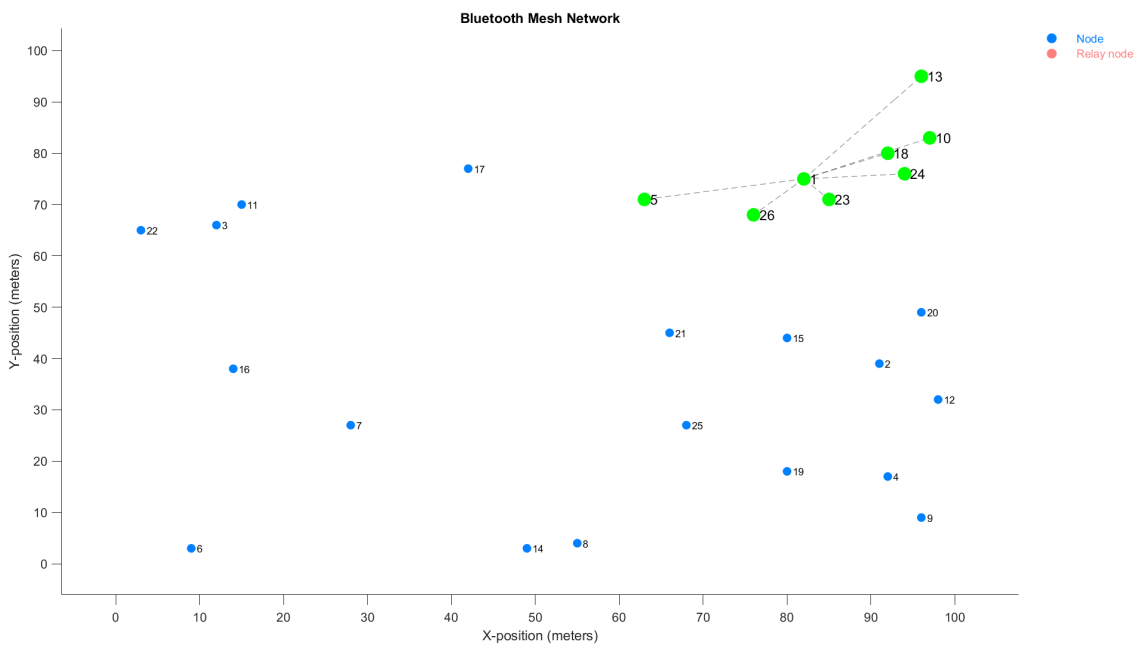


Figure 13: Random Scenario Creator

In each scenario, a node is chosen as the unlocated node and it tries to send packets via Bluetooth Low Energy, with simulated path loss models and inference. The reachable nodes are green as shown in figure 13.

Listing 2: Generating Traffic Between the nodes

```

for nodeId = 2:totalNodes+1
    traffic = networkTrafficOnOff(DataRate=1,PacketSize=10,GeneratePacket=true);

```

```

addTrafficSource(meshNodes{1},traffic, ...
    SourceAddress=meshNodes{1}.MeshConfig.ElementAddress, ...
    DestinationAddress=meshNodes{nodeIdx}.MeshConfig.ElementAddress, ...
    TTL=3);
end

```

Another important part of the research is determining the optimal range of the protocol. By simulating the transmission power vs the effective distance range for the Bluetooth Low Energy packets.

Listing 3: Simulating outdoor environment for measuring maximum distance

```

global cfgRange
cfgRange = bluetoothRangeConfig

%set the environment outdoor
cfgRange.Environment = "Outdoor"

%set the transmitter power
cfgRange.TransmitterPower = -20

global cfgPathLoss
cfgPathLoss = bluetoothPathLossConfig;
cfgPathLoss.Environment = "Outdoor";
cfgPathLoss.TransmitterAntennaHeight = 1;
cfgPathLoss.ReceiverAntennaHeight = 1;

```

Considering the packet loss of the network in distance, the robustness of the protocol can be evaluated. The effective range of the Bluetooth protocol is 100-150m (*How AoA and AoD Changed the Direction of Bluetooth Location Services — Bluetooth Technology Website — bluetooth.com 2023*). The simulation was done to confirm this information by generating traffic from one node to another while increasing the distance between them.

Listing 4: Getting statistics from nodes while increasing the distance

```

destinationStats = statistics(destinationNode)
sourceStats.Network.TransmittedMessages
destinationStats.Network.ReceivedMessages
destinationStats.Network.AcceptedMessages
destinationStats.Network.DroppedMessages
destinationStats.Network

```

4.2.2 OMNet++

For additional verifications of the protocol, a testbed in OMNet++ was created. In this, a simple application for the protocol was created with the protocol messages. As shown in the 14 a network of devices was created for the simulation and programmed to communicate with each other as per the protocol specification.

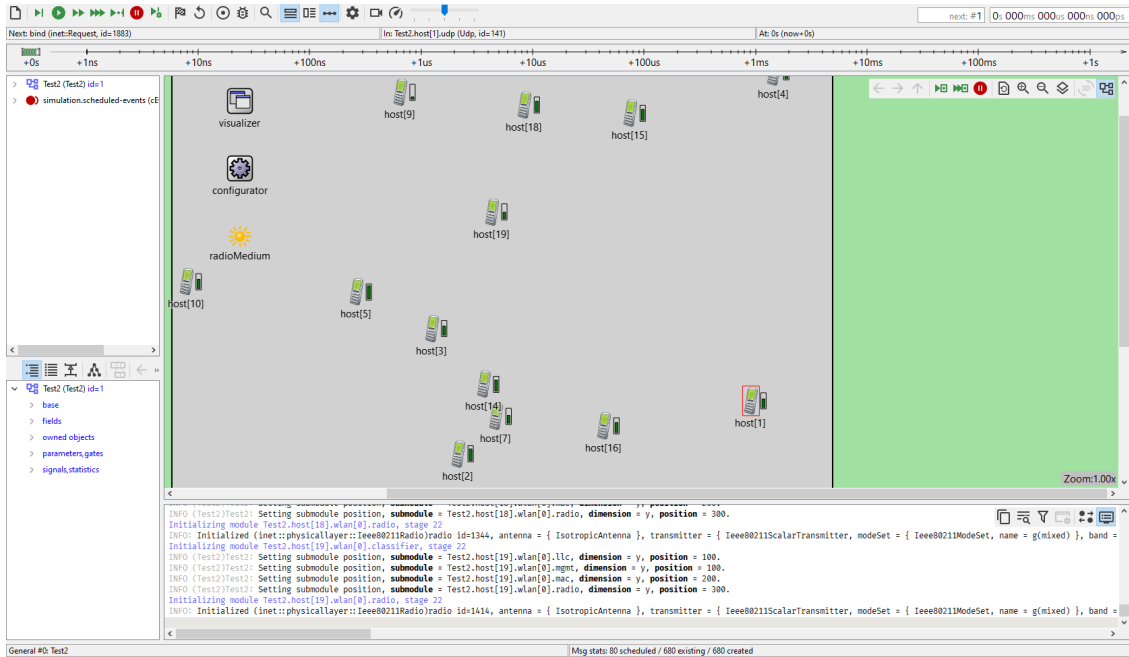


Figure 14: OMnet++ Protocol Simulation

Listing 5: Message implementation with Coordinations and NodeIDs

```

class LPNet extends inet::FieldsChunk {
    int nodeID;
    int type; //0-> unlocated 1-> witness 2->witness reply
    inet::Coord position;
}

```

By using the protocol message given above in the listing, we created applications for the unlocated nodes and the witnesses, depending on the packet they receive or send. For the unlocated nodes, when a location-proof is needed they send out a packet with their location included, broadcasted to the network. Then if the witness device receives it, as the protocol is designed the witness will transmit another message with its own location and nodeID. Witness nodes are listening to each other on Witness packets that are transmitted during this period to reply with Witness Reply. That will finalize the steps in protocol specifications for the mobile node network. The below listings show the packet generation and processing on each witness and unlocated device.

Listing 6: Message creation on Unlocated Device with Location Added

```

void Unlocated::sendPacket() {
    EV_INFO << "Sending Unlocated" << inet::endl;
    // Your custom packet sending logic
    // UdpBasicApp::sendPacket(); // Optionally call base implementation
    const auto &reply = makeShared<LPNet>();
    Coord pos = getHostPosition(); // Assuming getHostPosition() is implemented
    reply->setType(1);
    reply->setPosition(pos);
}

```

```

EV_INFO << "Unlocated■Details:" << reply->getType() <<"■position:"<<
    ↪ reply->getPosition() << inet::endl;

std::ostringstream str;
str << packetName << "-" << numSent;
Packet *packet = new Packet(str.str().c_str());
if (dontFragment)
    packet->addTag<FragmentationReq>()->setDontFragment(true);
const auto &payload = makeShared<ApplicationPacket>();
payload->setChunkLength(B(par("messageLength")));
payload->setSequenceNumber(numSent);
payload->addTag<CreationTimeTag>()->setCreationTime(simTime());
packet->insertAtBack(payload);
//our data
packet->insertAtBack(reply);
L3Address destAddr = chooseDestAddr();
emit(packetSentSignal, packet);
socket.sendTo(packet, destAddr, destPort);
numSent++;
}

```

Listing 7: Message Processing of Witness Nodes with Reply message creation

```

void Witness::processPacket(Packet *pk) {
    // Extract custom packet data
    int replyNeeded = 0;
    auto customPacket = pk->peekData<LPNet>();
    int type = customPacket->getType();
    Coord pos = customPacket->getPosition();

    EV_INFO << "Received■packet:■" << pk->getName() << ",■length:■"
        << pk->getByteLength() << "■bytes." << endl;
    EV_INFO << "Received■packet■Data:■" << type << ",■length:■" << pos
        << "■bytes." << endl;
    const auto &reply = makeShared<LPNet>();

    // Re-serialize the modified packet and send it
    char msgName[40];
    sprintf(msgName, "Processed-%s", pk->getName());
    auto newPacket = new Packet(msgName);
    // Process packet based on type 0 - unlocated 1-> witness 2 -> witness reply
    switch (type) {
    case 0:
        // Update position data and send back as WITNESS
        pos = getHostPosition(); // Assuming getHostPosition() is implemented
        reply->setType(1);
        reply->setPosition(pos);
        replyNeeded = 1;
    }
}

```

```

EV_INFO << "Sending■Witness■Packet." << endl;

break;
case 1:
// Reply with WITNESSREPLY
reply->setType(2);
reply->setPosition(pos);
replyNeeded = 1;
EV_INFO << "Sending■Witness■Reply■Packet." << endl;
break;
case 2:
// Process WITNESSREPLY

break;
}

if (replyNeeded == 1) {
newPacket->insertAtBack(reply);
L3Address destAddr = chooseDestAddr();
socket.sendTo(newPacket, destAddr, destPort);
}
}
}

```

With the simulation, we can see the protocol works as intended in the OMNet++ GUI.

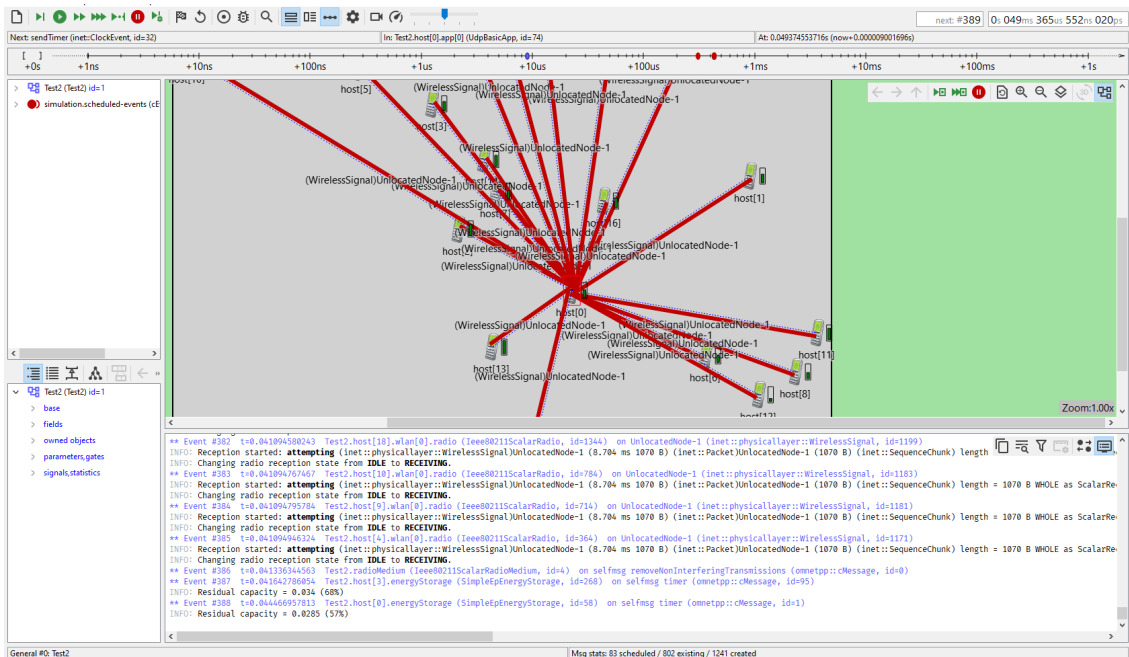


Figure 15: OMnet++ Protocol Simulation

4.3 Implementing an Overlay Network

Decentralized nature was a crucial part of the protocol design from the beginning. Implementation of the Decentralized network directly effects the viability of the protocol. Hence special care was taken in designing the decentralized network.

4.3.1 Design Decisions

To prevent the location-proof stored on the decentralized network to be used by a user other than the one that originally claimed, implementation on Ethereum ERC721 was used. This is a non-fungible token meaning that it was a unique token for the network. With the embedding information about the nodeID and the serviceID we assume that a duplication of a token is not possible. The location-proof token is made transferable only to a Location-Based Service, making them fool-proof method for location verification.

Listing 8: Implementation of the Location-Proof

```
constructor(
    address _erc20Token,
    uint256 _rewardAmount,
    address _escrowAddress
) ERC721("LPCoin", "LPC") {
    require(
        _erc20Token != address(0),
        "ERC20 token address cannot be zero"
    );
    require(_escrowAddress != address(0), "Escrow address cannot be zero");
    erc20Reward = IERC20(_erc20Token);
    rewardAmount = _rewardAmount;
    ercTokenAddress = _erc20Token;
    escrowAddress = _escrowAddress;
}

function mint(
    uint256 tokenId,
    address WitnessNodeId,
    uint256 timestamp, //unix timestamp
    string memory location, // {long, lat}
    address[] memory NearbyNodeIds,
    address unlocatedId
)
```

The contract of location-proof also gives out rewards to the witness nodes who participate, intensifying the participation of the protocol.

4.4 Implementing Formal Verification of the Protocol

Using the AVISPA Protocol Verification Tool (*The AVISPA Project 2017*), the location-proof protocol was verified for security and privacy aspects. The main factors which verified with the AVISPA were,

1. Authentication - The messages were signed using the sender to protect the non-repudiation of the protocol.
2. Secrecy - Intruder cannot eavesdrop data that are sent through the protocol.

Listing 9: Unlocated Node to Witness message

```

messages
1. UnlocatedNode -> Witness : UnlocatedNode,{Payload}Ku'
...
goal
Witness authenticates UnlocatedNode on Payload;
secrecy_of Payload [UnlocatedNode,Witness];

```

As shown in the listing the protocol messages were tested with formal verification for vulnerabilities and strengths. Discussions on the implementation results will continue to next section of the thesis.

5 Results and Evaluation

5.1 Privacy and Security Validation

With the help of the security evaluation tool AVISPA, we have determined that the protocol is secured with the secrecy of the information and authentication. With additional Attributes like timestamp and ServiceID added to the location claim the adversaries only have limited number of attack vectors to the protocol that is resource-intensive for them to even try.

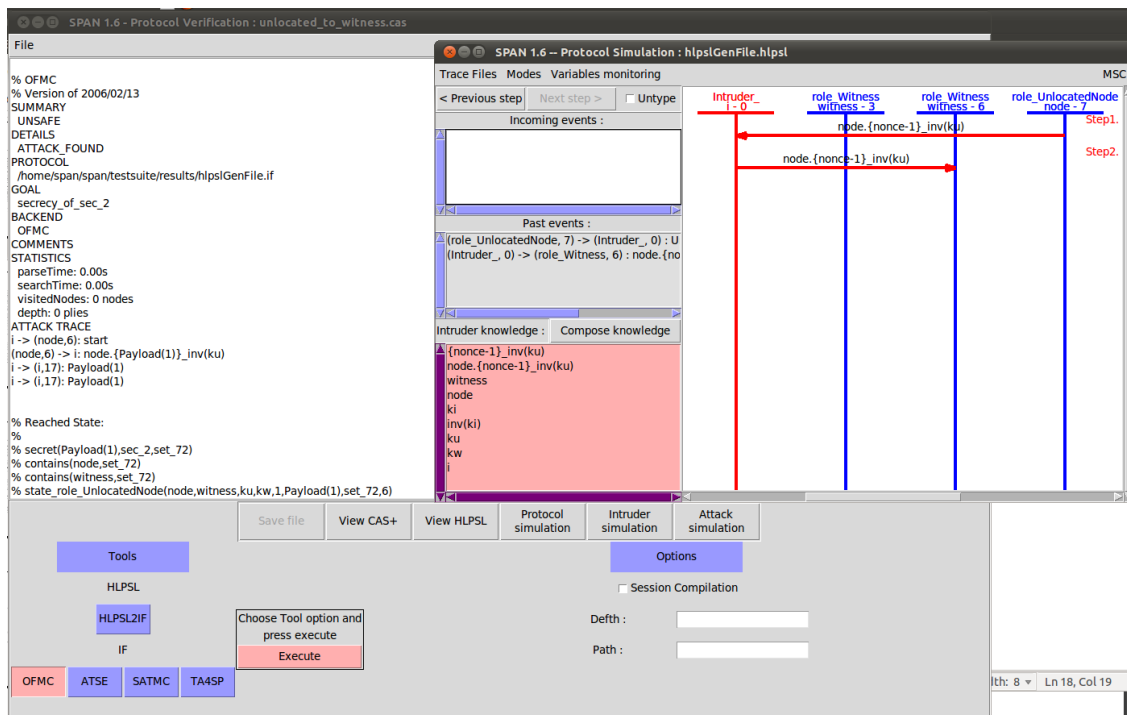


Figure 16: Unlocated Node to Witness Node (AVISPA)

16 shows the verification of unlocated node to witness node communication. It shows that even though the information of the message is in the hands of the intruder (EVE) the information is secured with the use of the decentralized PKI infrastructure. 17 Shows the witness to witness message security.

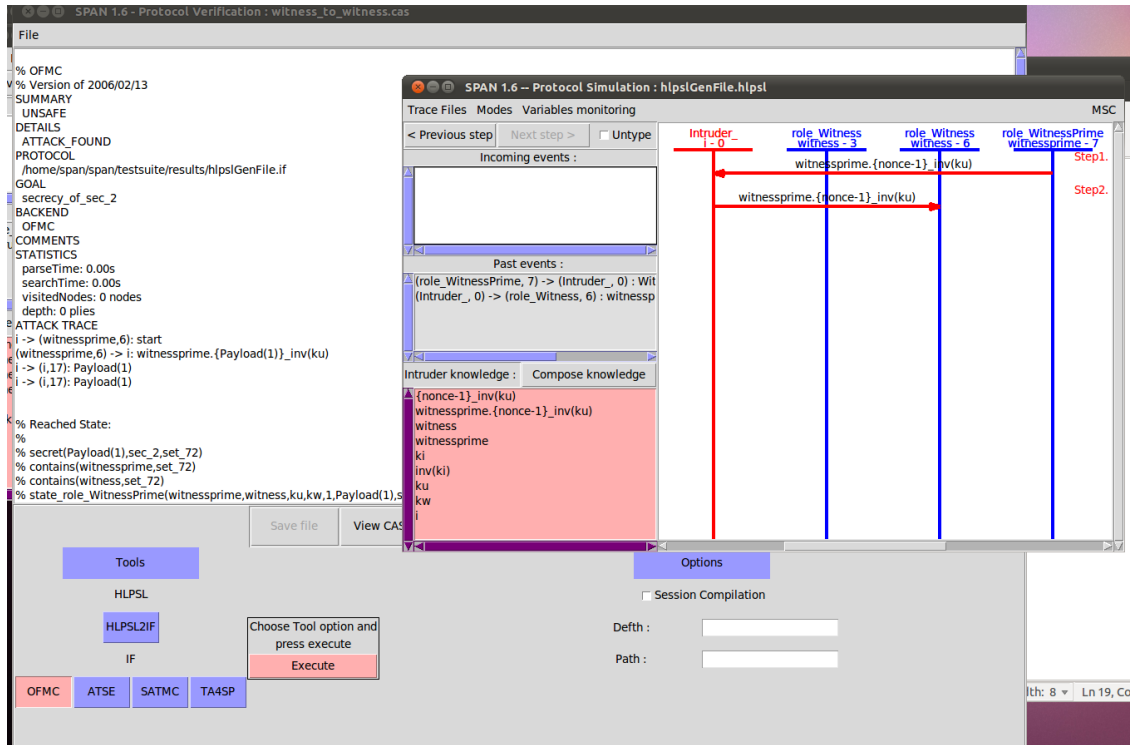


Figure 17: Witness Node to Witness Node (AVISPA)

5.2 Simulation Implementation

The simulation implementation plays a major role in evaluating the protocol of the research. There are multiple factors that can be extracted via a reasonable simulation environment.

5.2.1 Minimum and optimum number of nearby nodes as witnesses to correctly prove a location claim

To achieve byzantine fault tolerance (Xu and Kaizhou Shi 2022), there are a minimum number of trusted nodes within the network to make the protocol work in honesty. Also when there is a fewer number of witness nodes, the protocol may struggle to create verification claims for an untrusted device. With the simulation Results, we gather that at least 3 nodes should be available actively in the network for byzantine fault tolerance and verification.

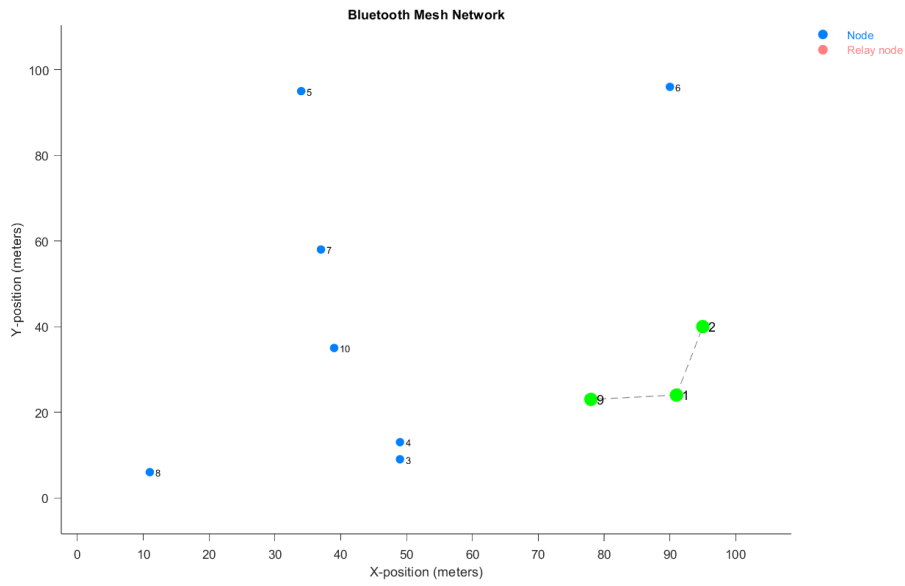


Figure 18: With Fewer Nodes picture 1.

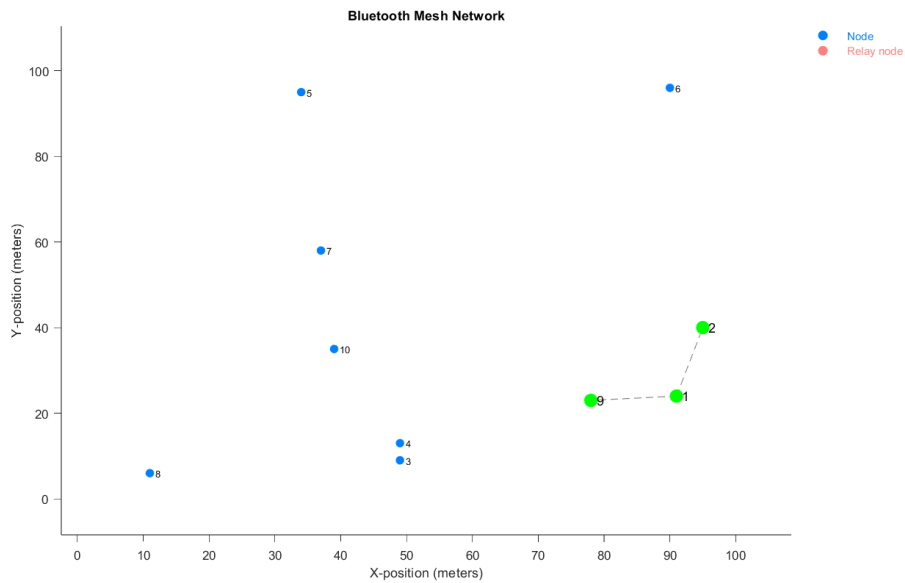


Figure 19: With Fewer Nodes picture 2.

5.2.2 Impact on distance for signal propagation over nearby nodes

The distance is what determines the range of the node. It heavily affects the witness and the untrusted device whom they can communicate with in order to have a verification claim. In the simulation, we gathered data on how distance affects the packets of Bluetooth Low Energy.

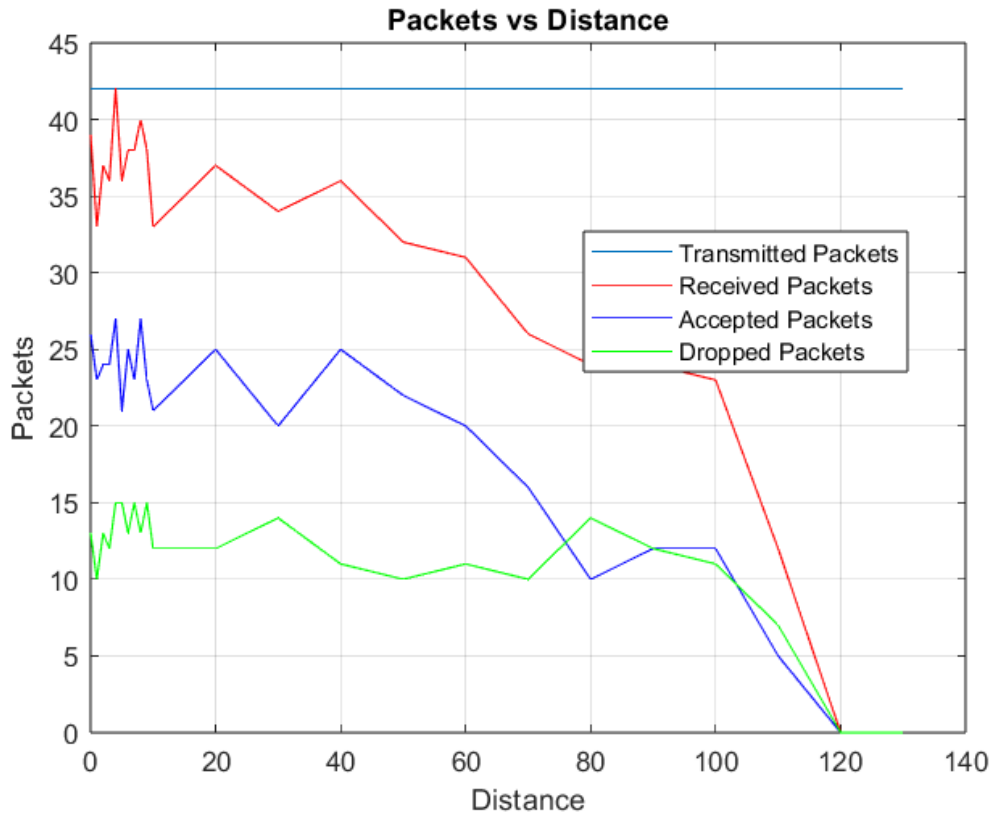


Figure 20: Distance vs Packet Lost

As seen in the 20 the packet rate are reduced drastically if the distance is over 100 meters. This also gives the location verification protocol a physical limitation of boundary over 100 meters.

5.2.3 Impact on Transmitting Power over Range

Another factor of limiting the location-proofs scope is the transmission power of the smart devices (*How AoA and AoD Changed the Direction of Bluetooth Location Services — Bluetooth Technology Website — bluetooth.com 2023*). With the commercial off the shelf devices the effective range of Bluetooth communication will be on a range of 30m to 90m according to simulations done by the research. The ?? shows the appropriate location-proof range to be 30m to 50m for general devices.

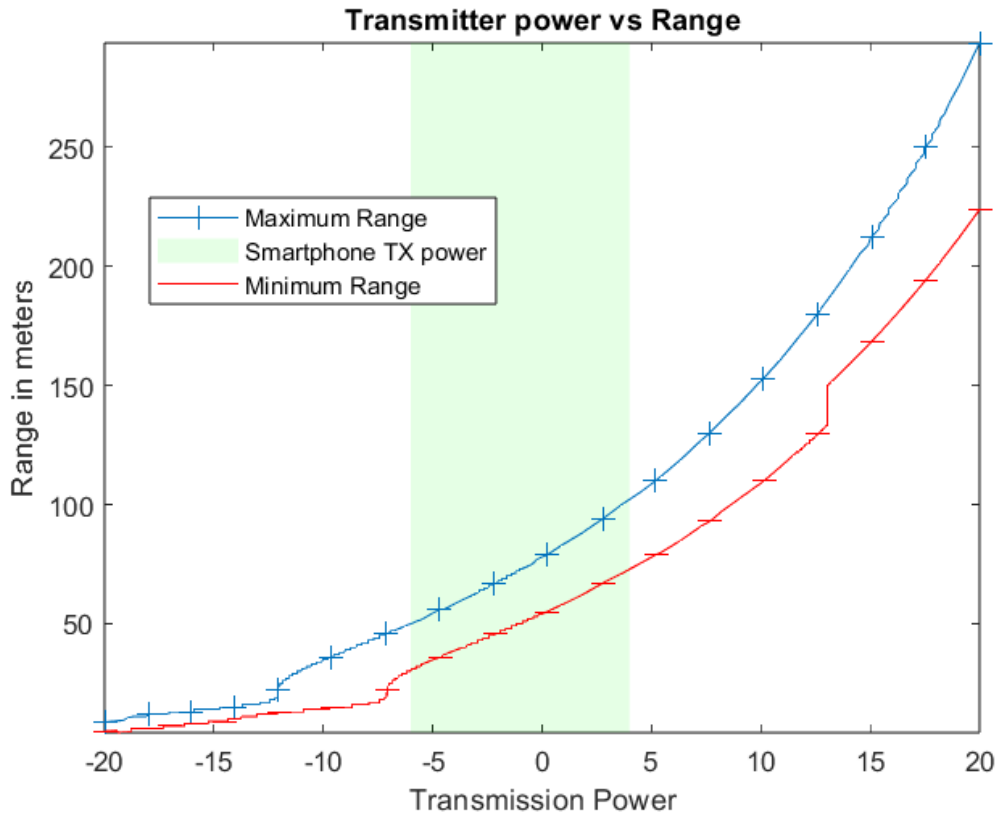


Figure 21: TxPower over Range

5.2.4 Effective distance on the location claim

The verification claim depends on the range of the Bluetooth beacon and nearby node distances. The claim should support a given location of a node and also have a margin of error that the untrusted node can be. With the previous simulation tests we can confirm the range of 30 meters and 50 meters of range given to the location claim.

5.2.5 Impact on the mobility of nodes to the Location claim

Smartphones with users are always moving in space and time. To effectively use the network, the algorithm has to be modified to accommodate the motion of nodes. The simulation on Omnet++ has proven the usability of the protocol on a mobile setting with enough number of nodes is viable for a location-claim.

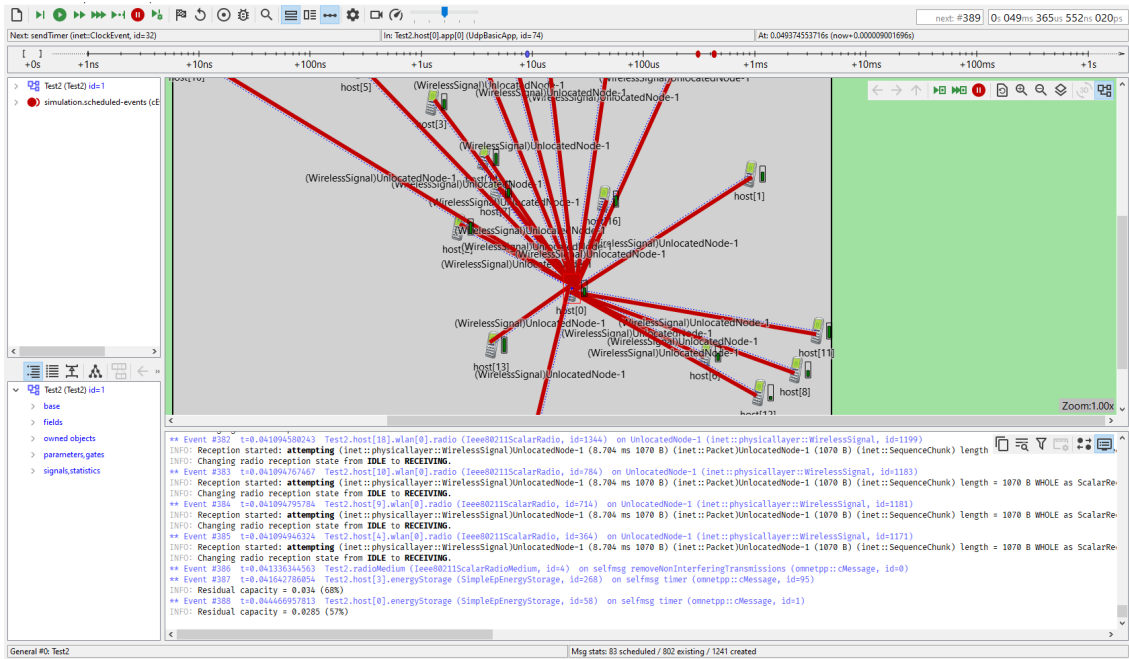


Figure 22: OMnet++ Mobility Test

5.2.6 Effective Throughput of location verification tokens

When the protocol is deployed, one has to measure the rate of verification claims the nodes can handle in any given situation. This factor can depend on the previously mentioned Channel bandwidth and the number of nodes in the network. With the use of the decentralized network the maximum throughput of location-proofs can be limited to 30 location-claims per second (Nasir et al. 2018).

6 Conclusion

In the research we proposed a blockchain-based decentralized design on location-proof method for smartphones for users with mobility. This research has several benefits against the related work in recent literature. We investigated about the practical aspect of such protocol for users location-proof. We have shown that if the user is within the range of 30 meters to 50m meters maximum the location claim for that user is possible. We have implemented the protocol as Decentralized as possible using a Decentralized Certificate authority as well as an Escrow Service for maintaining fair usage of the protocol. We also provide a way for user privacy in location claims giving multiple pseudonymous certificates for users per location claim.

For future work this research can be extended to practical implementation on actual hardware. Also it can be extended to use ultra-low power communication technologies to save more energy and have more range on the location-proof. The research can benefit research on Location based services that can utilize this research as well.

A Mobile Node TestBed

A.1 MATLAB

A.1.1 Generating Random Scenarios

Listing 10: Code for generating random scenarios

```
wirelessnetworkSupportPackageCheck;
%%
% Specify the total number of Bluetooth LE mesh nodes and their respective positions
% in the network. This example creates a 6-node Bluetooth mesh network consisting
% of a relay node, an end node, and two source-destination pairs. For more
  ↪ information
% about the functionalities of these nodes, see <docid:bluetooth_ug#mw_dc4f373f-27
  ↪ a1-4cd9-8d1d-f96d43002c2e
% Bluetooth Mesh Networking>.

% selectRunTypeList = ["Test 1 - 4 Nodes (Manual)","Test 2 - 6 Nodes (Manual)","
  ↪ Test 3 - Auto Generation"];

selectRunType = 4;
disp(selectRunType);
if(selectRunType == 1)
    totalNodes = 4;
    fprintf("Total nodes = %d\n",totalNodes);
    meshNodesPositions = [50 50; 40 40; 60 60; 50 60; 80 90];
    disp(meshNodesPositions);
elseif(selectRunType == 2)
    totalNodes = 6;
    fprintf("Total nodes = %d\n",totalNodes);
    meshNodesPositions = [50 10; 45 30; 60 60; 50 60; 25 10; 10 20; 70 80];
    disp(meshNodesPositions);
elseif(selectRunType == 3)
    totalNodes = 6;
    fprintf("Total nodes = %d\n",totalNodes);
    meshNodesPositions = [0 0; 45 30; 60 60; 50 60; 25 10; 10 20; 70 80];
    disp(meshNodesPositions);
else
    totalNodes = 25;
    fprintf("Total nodes = %d\n",totalNodes);
    meshNodesPositions = randi([0,100],totalNodes+1,2);
    disp(meshNodesPositions);
end

% meshNodesPositions = [15 25; 15 5; 30 15; 45 5; 45 25; 30 30]; % In meters
```

```

%%
% Create the nodes

%%
%
%
% Create the Unlocated Node

unlocatedNode = meshNodesPositions(1,:);
disp(unlocatedNode);
%%
%
%
% Specify the relay node and source–destination node pairs. In this example,
% Node1 and Node2 are source nodes and the corresponding destination nodes are
% Node4 and Node5.

% relayNode = 3;

%create source and destination pairs for all nodes
%
% n = totalNodes;
sourceDestinationNodePairs = ones(totalNodes,2);

for i = 1:totalNodes
    sourceDestinationNodePairs(i,2)= i+1;
end

disp(sourceDestinationNodePairs);

%%
% Set the Bluetooth mesh profile configuration parameters. Create Bluetooth
% mesh nodes with |"broadcaster–observer"| role.

meshNodes = cell(1,totalNodes);
for nodeId = 1:totalNodes+1

    % you don't need relays here

    meshCfg = bluetoothMeshProfileConfig(ElementAddress=dec2hex(nodeId,4));
    % if any(nodeId,relayNode)
    % meshCfg.Relay = true;
    % end
    meshNode = bluetoothLENode("broadcaster–observer", MeshConfig=meshCfg, ...
        Position=[meshNodesPositions(nodeId,:) 0],ReceiverRange=25, ...
        AdvertisingInterval=20e–3, ScanInterval=30e–3);
    meshNodes{nodeId} = meshNode;

```

```

end

disp(meshNodes);
size(meshNodes);
%%
% Add traffic between Node1 – Node4 and Node2 – Node5 source–destination node
% pairs by using the <docid:comm_ref#mw_0b2cd92a–350a–4264–9283–
    ↪ db5da34bd041 networkTrafficOnOff>
% object. The |networkTrafficOnOff| object enables you to create an On–Off
    ↪ application
% traffic pattern.

% % Add traffic between Node1 and Node4
% traffic = networkTrafficOnOff(DataRate=1,PacketSize=15,GeneratePacket=true);
% addTrafficSource(meshNodes{1},traffic, ...
% SourceAddress=meshNodes{1}.MeshConfig.ElementAddress, ...
% DestinationAddress=meshNodes{4}.MeshConfig.ElementAddress, ...
% TTL=3);

% Add traffic between Node1 and all nodes
for nodeId = 2:totalNodes+1

    traffic = networkTrafficOnOff(DataRate=1,PacketSize=10,GeneratePacket=true);
    addTrafficSource(meshNodes{1},traffic, ...
        SourceAddress=meshNodes{1}.MeshConfig.ElementAddress, ...
        DestinationAddress=meshNodes{nodeId}.MeshConfig.ElementAddress, ...
        TTL=3);

end
%%
% Visualize the Bluetooth mesh network by using the |helperBLEMeshVisualizeNetwork
    ↪ |
% helper function. This helper function displays the created Bluetooth mesh network.

meshNetworkGraph = helperBLEMeshVisualizeNetwork(); % Object for Bluetooth
    ↪ mesh network visualization
meshNetworkGraph.NumberOfNodes = totalNodes+1; % Total number of mesh nodes
meshNetworkGraph.NodePositionType = 'UserInput'; % Option to assign node
    ↪ position
meshNetworkGraph.Positions = meshNodesPositions; % List of all node positions
meshNetworkGraph.ReceiverRange = 25; % Reception range of mesh node
meshNetworkGraph.Title = 'Bluetooth■Mesh■Network'; % Title of plot
meshNetworkGraph.SourceDestinationPairs = sourceDestinationNodePairs; % Source–
    ↪ destination node pair
meshNetworkGraph.NodeType = ones(1,totalNodes+1); % State of mesh node
meshNetworkGraph.DisplayProgressBar = false; % Display progress bar
meshNetworkGraph.createNetwork(); % Display mesh network

```


A.1.2 Analysing Path Loss and Distance vs Transmission Power

Listing 11: Code for generating random scenarios

```
%% Bluetooth Graphs
%% Power vs Distance
% Create a Bluetooth range estimation configuration object

global cfgRange
cfgRange = bluetoothRangeConfig

%set the environment outdoor
cfgRange.Environment = "Outdoor"

%set the transmitter power
cfgRange.TransmitterPower = -20

global cfgPathLoss
cfgPathLoss = bluetoothPathLossConfig;
cfgPathLoss.Environment = "Outdoor";
cfgPathLoss.TransmitterAntennaHeight = 1;
cfgPathLoss.ReceiverAntennaHeight = 1;
%%

%%
% Test the config

[rangeBR,pathL,rxPower] = bluetoothRange(cfgRange)
%%
% Get the max distance from the range

rangeMax = max(rangeBR)
%%
% Start the plotting.
%
% Create the funtion to plot. To be used in fplot.

%%
%
% Transmitter power vs Range

%Transmitter power in range -20 to 20
%plot
figure(1);
fplot(@(x) funcRangeMax(x),[-20 20],Marker='+',MarkerSize=8)
hold on
title("Transmitter power vs Range")
ylabel("Range in meters")
```

```

xlabel("Transmission Power")
% color the designated class 2 bluetooth power region
%https://www.lairdconnect.com/support/faqs/what-bluetooth-class
%https://www.researchgate.net/publication/329973291_NIST_Special_Publication_800
    ↪ -121_Revision_2_Guide_to_Bluetooth_Security?_tp=
    ↪ eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6Il9kaXJlY3QiLCJwYWdlIjoX2RpcmVjdCJ9fQ
    ↪
% Max. output power 2.4mW(+4dBm)
% Min. output power 0.25mW(-6dBm), Power control is optional in this bluetooth
    ↪ power class.
% xBox = [-6:4]
% yBox = [0:ylim]
% patch(xBox,yBox,"black","FaceColor","green","FaceAlpha",0.1)
xregion(-6,4,"FaceColor","green","FaceAlpha",0.1)
fplot(@(x) funcRangeMin(x),[-20 20],Marker="_",MarkerSize=8,Color="red")
legend("Maximum Range","Smartphone TX power","Minimum Range",'show','
    ↪ location','best')

hold off

% Minimum Range vs PathLosss

% figure(2);
%
% fplot(@(x) funcPathLoss(x),[-20 20],Marker="_",MarkerSize=8,Color="red")
% hold on
% title("Minimum Range vs PathLoss")
% ylabel("Path Loss")
% xlabel("Range in meters")
%
% % legend("Maximum Range","Smartphone TX power","Minimum Range",'show','
    ↪ location','best')
%
% hold off
% Minimum Range vs Reciever Power

% figure(3);
%
% fplot(@(x) funcRecieverPower(x),[-20 20],Marker="_",MarkerSize=8,Color="red
    ↪ ")
% hold on
% title("Minimum Range vs Reciever Power")
% ylabel("Path Loss")
% xlabel("Range in meters")
%
% % legend("Maximum Range","Smartphone TX power","Minimum Range",'show','
    ↪ location','best')
%

```

```

% hold off

% Distance vs PathLoss

fplot(@(x) funcPathLoss(x),[0 100],Color='red')
hold on
title('Distance vs PathLoss')
ylabel('Path Loss')
xlabel('Distance in meters')

% legend('Maximum Range','Smartphone TX power','Minimum Range','show','
    ↔ location','best')

hold off
%%
% define functions
%
% Maximum Range

function rangeMax = funcRangeMax(x)
    global cfgRange
    cfgRange.TransmitterPower = x;
    rangeBR = bluetoothRange(cfgRange);
    rangeMax = max(rangeBR);
end
%%
% Minimum Range

function rangeMin = funcRangeMin(x)
    global cfgRange
    cfgRange.TransmitterPower = x;
    rangeBR = bluetoothRange(cfgRange);
    rangeMin = min(rangeBR);
end

%%
% PathLoss

function PathLoss = funcPathLoss(x)
    global cfgPathLoss
    PathLoss = bluetoothPathLoss(x,cfgPathLoss);
end

%%
% Reciver Power

function recieverPower = funcRecieverPower(x)

```

```

global cfgRange
cfgRange.TransmitterPower = x;
[rangeBR,pathL,rxPower] = bluetoothRange(cfgRange);
recieverPower = rxPower;
end

%%
%
```

A.1.3 Statistics on Distance vs Packet Loss

Listing 12: Code for generating random scenarios

```

%% Create, Configure and Simulate Bluetooth Mesh Network
wirelessnetworkSupportPackageCheck;
%%
% Create a wireless network simulator.

networkSimulator = wirelessNetworkSimulator.init;
%%
% Create a Bluetooth mesh profile configuration object, specifying the element
% address of the source node.

cfgMeshSource = bluetoothMeshProfileConfig(ElementAddress="0001")
%%
% Create a Bluetooth LE node, specifying the role as |"broadcaster-observer"|.
% Specify the position of the source node. Assign the mesh profile configuration
% to the source node.

sourceNode = bluetoothLENode("broadcaster-observer");
sourceNode.Position = [0 0 0];
sourceNode.MeshConfig = cfgMeshSource;
%%
% Create a Bluetooth mesh profile configuration object, specifying the element
% address of the Bluetooth LE node.

cfgMeshDestination = bluetoothMeshProfileConfig(ElementAddress="0003")
%%
% Create a Bluetooth LE node, specifying the role as |"broadcaster-observer"|.
% Specify the position of the destination node. Assign the mesh profile configuration
% to the destination node.

destinationNode = bluetoothLENode("broadcaster-observer");
destinationNode.Position = [110 0 0];
destinationNode.MeshConfig = cfgMeshDestination;
%%

traffic = networkTrafficOnOff(OnTime=inf, ...
```

```

    DataRate=1, ...
    PacketSize=15, ...
    GeneratePacket=true);
%%
% Add application traffic between the source and destination nodes.

addTrafficSource(sourceNode,traffic, ...
    SourceAddress=cfgMeshSource.ElementAddress, ...
    DestinationAddress=cfgMeshDestination.ElementAddress,TTL=10);
%%
% Create a Bluetooth mesh network consisting of the source node, relay node,
% and destination node.

nodes = {sourceNode,destinationNode};
%%
% Add the mesh nodes to the wireless network simulator.

addNodes(networkSimulator,nodes);
%%
% Set the simulation time and run the simulation.

simulationTime = 5; % In seconds
run(networkSimulator,simulationTime);
%%
% Retrieve application, link layer (LL) , and physical layer (PHY) statistics
% related to the source, relay, and destination nodes by using the
% statistic object function.

% sourceStats = statistics(sourceNode)
destinationStats = statistics(destinationNode)
sourceStats.Network.TransmittedMessages
destinationStats.Network.ReceivedMessages
destinationStats.Network.AcceptedMessages
destinationStats.Network.DroppedMessages
destinationStats.Network
%
```

A.2 OMNet++

A.2.1 Witness.cc: Code for witness

Listing 13: Witness

```

#include "Witness.h"
#include "inet/common/INETDefs.h"
#include "inet/common/packet/Packet.h"
#include "LPNet_m.h"
#include "inet/mobility/contract/IMobility.h"
```

```

#include "inet/common/ModuleAccess.h"
#include "inet/common/lifecycle/NodeStatus.h"

Define_Module(Witness);

void Witness::initialize(int stage) {
    EV << "Init" << inet::endl;
    UdpBasicApp::initialize(stage); // Calling base class initializer
}

void Witness::sendPacket() {
    EV_INFO << "Sending■Witness" << inet::endl;
    // Your custom packet sending logic
    UdpBasicApp::sendPacket(); // Optionally call base implementation
}

void Witness::processPacket(Packet *pk) {
    // Extract custom packet data
    int replyNeeded = 0;
    auto customPacket = pk->peekData<LPNet>();
    int type = customPacket->getType();
    Coord pos = customPacket->getPosition();

    EV_INFO << "Received■packet:■" << pk->getName() << ",■length:■"
        << pk->getByteLength() << "■bytes." << endl;
    EV_INFO << "Received■packet■Data:■" << type << ",■length:■" << pos
        << "■bytes." << endl;
    const auto &reply = makeShared<LPNet>();

    // Re-serialize the modified packet and send it
    char msgName[40];
    sprintf(msgName, "Processed-%s", pk->getName());
    auto newPacket = new Packet(msgName);
    // Process packet based on type 0 - unlocated 1-> witness 2 -> witness reply
    switch (type) {
    case 0:
        // Update position data and send back as WITNESS
        pos = getHostPosition(); // Assuming getHostPosition() is implemented
        reply->setType(1);
        reply->setPosition(pos);
        replyNeeded = 1;
        EV_INFO << "Sending■Witness■Packet." << endl;

        break;
    case 1:
        // Reply with WITNESSREPLY
        reply->setType(2);
        reply->setPosition(pos);
    }
}

```

```

        replyNeeded = 1;
        EV_INFO << "Sending■Witness■Reply■Packet." << endl;
        break;
    case 2:
        // Process WITNESSREPLY

        break;
    }

    if (replyNeeded == 1) {
        newPacket->insertAtBack(reply);
        L3Address destAddr = chooseDestAddr();
        socket.sendTo(newPacket, destAddr, destPort);
    }

// sendPacket(newPacket);
}

Coord Witness::getHostPosition() {
    auto mobility = check_and_cast<IMobility*>(
        getParentModule()->getSubmodule("mobility"));
    return mobility->getCurrentPosition();
}

```

A.2.2 Unlocated.cc: Code for Unlocated Node

Listing 14: Unlocated

```

#include "Unlocated.h"
#include "inet/common/INETDefs.h"
#include "inet/common/packet/Packet.h"
#include "LPNet_m.h"
#include "inet/mobility/contract/IMobility.h"
#include "inet/common/ModuleAccess.h"
#include "inet/common/lifecycle/NodeStatus.h"
#include "inet/applications/base/ApplicationPacket_m.h"
#include "inet/common/TagBase_m.h"
#include "inet/common/TimeTag_m.h"
#include "inet/common/lifecycle/ModuleOperations.h"
#include "inet/networklayer/common/FragmentationTag_m.h"
#include "inet/networklayer/common/L3AddressResolver.h"
#include "inet/transportlayer/contract/udp/UdpControlInfo_m.h"

Define_Module(Unlocated);

void Unlocated::initialize(int stage) {
    EV_INFO << "Init■Unlocated" << inet::endl;

```

```

    UdpBasicApp::initialize(stage); // Calling base class initializer
}

void Unlocated::sendPacket() {
    EV_INFO << "Sending■Unlocated" << inet::endl;
    // Your custom packet sending logic
    // UdpBasicApp::sendPacket(); // Optionally call base implementation
    const auto &reply = makeShared<LPNet>();
    Coord pos = getHostPosition(); // Assuming getHostPosition() is implemented
    reply->setType(1);
    reply->setPosition(pos);

    EV_INFO << "Unlocated■Details:" << reply->getType() <<"■position:"<<
        ↪ reply->getPosition() << inet::endl;

    std::ostringstream str;
    str << packetName << "-" << numSent;
    Packet *packet = new Packet(str.str().c_str());
    if (dontFragment)
        packet->addTag<FragmentationReq>()->setDontFragment(true);
    const auto &payload = makeShared<ApplicationPacket>();
    payload->setChunkLength(B(par("messageLength")));
    payload->setSequenceNumber(numSent);
    payload->addTag<CreationTimeTag>()->setCreationTime(simTime());
    packet->insertAtBack(payload);
    //our data
    packet->insertAtBack(reply);
    L3Address destAddr = chooseDestAddr();
    emit(packetSentSignal, packet);
    socket.sendTo(packet, destAddr, destPort);
    numSent++;
}

Coord Unlocated::getHostPosition()
{
    auto mobility = check_and_cast<IMobility *>(
        getParentModule()->getSubmodule("mobility"));
    return mobility->getCurrentPosition();
}

```

A.2.3 Code for Simulation

Listing 15: Sim.ned

```

package finalfinal.sim1;

import inet.networklayer.configurator.ipv4.Ipv4NetworkConfigurator;

```



```

import inet.node.contract.INetworkNode;
import inet.physicallayer.wireless.common.contract.packetlevel.IRadioMedium;
import inet.visualizer.contract.IIntegratedVisualizer;

network Sim1
{
  parameters:
    @display("bgb=650,500;bgg=100,1,grey95");
    @figure[title](type=label; pos=0,-1; anchor=sw; color=darkblue);

    @figure[rcvdPkText](type=indicatorText; pos=380,20; anchor=w; font=,18;
      ↪ textFormat="packets■received:■%g"; initialValue=0);
    @statistic[packetReceived](source=hostB.app[0].packet+ReceivedhostA.app
      ↪ [0].packetReceived; record=figure(count); targetFigure=rcvdPkText);

  submodules:
    visualizer: <default(firstAvailableOrEmpty("IntegratedCanvasVisualizer"))>
      ↪ like IIntegratedVisualizer if typename != "" {
      @display("p=580,125");
    }
    configurator: Ipv4NetworkConfigurator {
      @display("p=580,200");
    }
    radioMedium: <default("UnitDiskRadioMedium")> like IRadioMedium {
      @display("p=580,275");
    }
    hostA: <default("WirelessHost")> like INetworkNode {
      @display("p=50,325");
    }
    hostB: <default("WirelessHost")> like INetworkNode {
      @display("p=450,325");
    }
}

```

Listing 16: omnetpp.ini

```

[General]
network = Sim1

*.host*.ipv4.arp.typename = "GlobalArp"
*.host*.*.bitrate = 1Mbps

*.hostA.numApps = 2
*.hostA.app[0].typename = "Unlocated"
*.hostA.app[0].destAddresses = "hostB"
*.hostA.app[0].destPort = 5000
*.hostA.app[0].sendInterval = exponential(1s)
*.hostA.app[0].packetName = "UnlocatedRequest"

```

```

*.hostA.app[0].messageLength = 1000B

*.hostA.app[1].typename = "UdpSink"
*.hostA.app[1].localPort = 7000

*.hostB.numApps = 2
*.hostB.app[0].typename = "UdpSink"
*.hostB.app[0].localPort = 5000

*.hostB.app[1].typename = "Witness"
*.hostB.app[1].localPort = 6000
*.hostB.app[1].destAddresses = "hostA"
*.hostB.app[1].packetName = "WitnessPacket"
*.hostB.app[1].sendInterval = exponential(1s)
*.hostB.app[1].destPort = 7000
*.hostB.app[1].messageLength = 1000B

*.host*.wlan[0].typename = "AckingWirelessInterface"
*.host*.wlan[0].mac.useAck = false
*.host*.wlan[0].mac.fullDuplex = false
*.host*.wlan[0].radio.transmitter.communicationRange = 500m
*.host*.wlan[0].radio.receiver.ignoreInterference = true
*.host*.wlan[0].mac.headerLength = 23B

```

B Overlay Network

B.1 Location-proof Contract

Listing 17: LpCoin

```

// SPDX-License-Identifier: MIT
// Compatible with OpenZeppelin Contracts ^5.0.0
pragma solidity ^0.8.16;

import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Burnable.sol";
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";

contract LpCoin is ERC721, ERC721Burnable {
    uint256 private _nextTokenId;
    address private ercTokenAddress;
    //reward coins
    IERC20 public erc20Reward;

    uint256 public rewardAmount;
    uint256 public totalSupply;

    address private escrowAddress;

```

```

// Event to emit when ERC20 reward is given
event RewardGiven(address indexed recipient, uint256 amount);

function safeMint(address to) public {
    uint256 tokenId = _nextTokenId++;
    _safeMint(to, tokenId);
}

struct CoinMetadata {
    address WitnessNodeId;
    uint256 timestamp; //unix timestamp
    string location; // {long, lat}
    address[] NearbyNodeIds;
    address unlocatedId;
}

mapping(uint256 => CoinMetadata) public coinMetadata;

constructor(
    address _erc20Token,
    uint256 _rewardAmount,
    address _escrowAddress
) ERC721("LPCoin", "LPC") {
    require(
        _erc20Token != address(0),
        "ERC20 token address cannot be zero"
    );
    require(_escrowAddress != address(0), "Escrow address cannot be zero");
    erc20Reward = IERC20(_erc20Token);
    rewardAmount = _rewardAmount;
    ercTokenAddress = _erc20Token;
    escrowAddress = _escrowAddress;
}

function mint(
    uint256 tokenId,
    address WitnessNodeId,
    uint256 timestamp, //unix timestamp
    string memory location, // {long, lat}
    address[] memory NearbyNodeIds,
    address unlocatedId
) external {
    _mint(msg.sender, tokenId);
    coinMetadata[tokenId] = CoinMetadata({
        WitnessNodeId: WitnessNodeId,
        timestamp: timestamp,
        location: location,

```

```

        NearbyNodeIds: NearbyNodeIds,
        unlocatedId: unlocatedId
    });

    // Ensure there are enough tokens to reward
    require(
        ERC20Reward.balanceOf(address(this)) >= rewardAmount,
        "Insufficient ERC20 balance in contract"
    );

    // Transfer ERC20 reward to msg.sender
    require(
        ERC20Reward.transferFrom(ercTokenAddress, msg.sender, rewardAmount
            ↪ ),
        "Failed to transfer ERC20 tokens as reward"
    );

    emit RewardGiven(msg.sender, rewardAmount);
}

// Implement other update functions as needed

function getCoinMetadata(
    uint256 tokenId
) external view returns (CoinMetadata memory) {
    require(ownerOf(tokenId) != address(0), "Token does not exist");
    return coinMetadata[tokenId];
}

function burn(uint256 tokenId) external {
    require(ownerOf(tokenId) == msg.sender, "Only owner can burn");
    address rewardee = coinMetadata[tokenId].WitnessNodeId;
    _burn(tokenId);
    Escrow(escrowAddress).release(msg.sender);
}
}

```

References

- Abdallah, A. A., K. Shamaei, and Z. M. Kassas (2019). “Indoor localization with LTE carrier phase measurements and synthetic aperture antenna array”. In: *Proceedings of the 32nd International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS+ 2019*, pp. 2670–2679. DOI: 10.33012/2019.17030.
- Abdelnasser, Heba et al. (2016). “SemanticSLAM: Using Environment Landmarks for Unsupervised Indoor Localization”. In: *IEEE Transactions on Mobile Computing* 15, pp. 1770–1782.
- Acar, Yusuf et al. (2022). “TDOA Based Hybrid Localization Algorithm”. In: *2022 30th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4.
- “Advances in VLSI, Communication, and Signal Processing” (2020). In: *Lecture Notes in Electrical Engineering*. URL: <https://api.semanticscholar.org/CorpusID:240798438>.
- Aly, Heba, Anas Basalamah, and Moustafa Youssef (2015). “LaneQuest: An accurate and energy-efficient lane detection system”. In: *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 163–171.
- Aly, Heba and Moustafa Youssef (2013). “Dejavu: an accurate energy-efficient outdoor localization system”. In: *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*.
- Amoretti, Michele et al. (2016). “Blockchain-Based Proof of Location”. In: *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 146–153.
- Androulaki, Elli et al. (June 2014). “Enforcing Location and Time-Based Access Control on Cloud-Stored Data”. In: *2014 IEEE 34th International Conference on Distributed Computing Systems*. IEEE. DOI: 10.1109/icdcs.2014.71. URL: <http://dx.doi.org/10.1109/ICDCS.2014.71>.
- Bao, Jie et al. (2015). “Recommendations in location-based social networks: a survey”. In: *GeoInformatica* 19, pp. 525–565.
- Bluetooth Direction Finding: A Technical Overview — Bluetooth® Technology Website — bluetooth.com* (2019). <https://www.bluetooth.com/bluetooth-resources/bluetooth-direction-finding/>. [Accessed 02-11-2023].
- Chen, Liang et al. (2017). “Robustness, Security and Privacy in Location-Based Services for Future IoT: A Survey”. In: *IEEE Access* 5, pp. 8956–8977. URL: <https://api.semanticscholar.org/CorpusID:2134031>.
- Cheng, C. H. and S. J. Syu (2021). “Improving area positioning in ZigBee sensor networks using neural network algorithm”. In: *Microsystem Technologies* 27.4, pp. 1419–1428. DOI: 10.1007/s00542-019-04309-2.
- Cheng, Yu-Chung et al. (2005). “Accuracy characterization for metropolitan-scale Wi-Fi localization”. In: *ACM SIGMOBILE International Conference on Mobile Systems, Applications, and Services*.
- Cleeff, Andre van, Wolter Pieters, and Roel Wieringa (Dec. 2010). “Benefits of Location-Based Access Control: A Literature Study”. In: *2010 IEEE/ACM International Conference on Green Computing and Communications and International Conference on Cyber, Physical and Social Computing*. IEEE. DOI: 10.1109/greencom-cpscom.2010.148. URL: <http://dx.doi.org/10.1109/GreenCom-CPSCom.2010.148>.
- Couderc, Paul and Yoann Maurel (Jan. 2019). “Location corroboration using passive observations of IEEE 802.11 Access Points”. In: *2019 16th IEEE Annual Consumer*

- Communications and Networking Conference (CCNC)*. IEEE. DOI: 10.1109/ccnc.2019.8651873. URL: <http://dx.doi.org/10.1109/CCNC.2019.8651873>.
- Cui, Youjing and Shuzhi Sam Ge (2001). “Autonomous vehicle positioning with GPS in urban canyon environments”. In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164) 2*, 1105–1110 vol.2.
- Davis, Benjamin, Hao Chen, and Matthew Franklin (May 2012). “Privacy-preserving alibi systems”. In: *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*. ASIA CCS '12. ACM. DOI: 10.1145/2414456.2414475. URL: <http://dx.doi.org/10.1145/2414456.2414475>.
- DEMİR, Ercan and Abdulkemir ÖZTEKİN (Apr. 2021). “Automatic Positioning of Mobile Users via GSM Signal Measurements”. In: *Balkan Journal of Electrical and Computer Engineering* 9.2, pp. 152–160. DOI: 10.17694/bajece.852963. URL: <https://doi.org/10.17694/bajece.852963>.
- Douceur, John R. (2002). “The Sybil Attack”. In: *International Workshop on Peer-to-Peer Systems*. URL: <https://api.semanticscholar.org/CorpusID:5310675>.
- Du, C. et al. (2020). “KF-kNN: Lowcost and high-accurate FM-based indoor localization model via fingerprint technology”. In: *IEEE Access* 8, pp. 197523–197531. DOI: 10.1109/ACCESS.2020.3031089.
- Einavipour, Sina and Reza Javidan (Feb. 2021). “An intelligent IoT-based positioning system for theme parks”. In: *The Journal of Supercomputing* 77.9, pp. 9879–9904. DOI: 10.1007/s11227-021-03669-9. URL: <https://doi.org/10.1007/s11227-021-03669-9>.
- Gambis, Sebastien et al. (Oct. 2014). “PROPS: A PRIVACY-Preserving Location Proof System”. In: *2014 IEEE 33rd International Symposium on Reliable Distributed Systems*. IEEE. DOI: 10.1109/srds.2014.37. URL: <http://dx.doi.org/10.1109/SRDS.2014.37>.
- Gaonkar, Shravan and Romit Roy Choudhury (2007). “Micro-Blog: map-casting from mobile phones to virtual sensor maps”. In: *ACM International Conference on Embedded Networked Sensor Systems*.
- Gentner, C. et al. (2020). “WiFi-RTT Indoor Positioning”. In: *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pp. 1029–1035. DOI: 10.1109/PLANS46316.2020.9110232.
- Gertzell, P. and et al. (2020). “5G multi-BS positioning with a single-antenna receiver”. In: *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*. Vol. 2020-Augus, pp. 1–5. DOI: 10.1109/PIMRC48278.2020.9217124.
- Hegarty, Christopher and Elliott Kaplan (2005). URL: https://d1.amobbs.com/bbs_upload782111/files_33/ourdev_584835021W59.pdf.
- How AoA and AoD Changed the Direction of Bluetooth Location Services — Bluetooth Technology Website — bluetooth.com* (2023). <https://www.bluetooth.com/blog/new-aoa-aod-bluetooth-capabilities/>. [Accessed 02-11-2023].
- Ibrahim, Mohamed and Moustafa Youssef (2011). “CellSense: An Accurate Energy-Efficient GSM Positioning System”. In: *IEEE Transactions on Vehicular Technology* 61, pp. 286–296.
- Inc., The MathWorks (2022). *Bluetooth toolbox*. Natick, Massachusetts, United States. URL: <https://www.mathworks.com/>.
- Ippisch, Andre, Salem Sati, and Kalman Graffi (2017). “Device to device communication in mobile Delay Tolerant networks”. In: *2017 IEEE/ACM 21st International Sympo-*

- sium on Distributed Simulation and Real Time Applications (DS-RT)*, pp. 1–8. URL: <https://api.semanticscholar.org/CorpusID:11793948>.
- Javali, Chitra et al. (Nov. 2016). “I Am Alice, I Was in Wonderland: Secure Location Proof Generation and Verification Protocol”. In: *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. IEEE. DOI: 10.1109/lcn.2016.126. URL: <http://dx.doi.org/10.1109/LCN.2016.126>.
- Kriz, Pavel, Filip Maly, and Tomav Kozel (2016). “Improving Indoor Localization Using Bluetooth Low Energy Beacons”. In: *Mob. Inf. Syst.* 2016, 2083094:1–2083094:11. URL: <https://api.semanticscholar.org/CorpusID:29424356>.
- LaMarca, Anthony et al. (2005). “Place Lab: Device Positioning Using Radio Beacons in the Wild”. In: *International Conference on Pervasive Computing*.
- Li, B., K. Zhao, and E. B. Sandoval (2020). “A UWB-Based Indoor Positioning System Employing Neural Networks”. In: *Journal of Geovisualization and Spatial Analysis* 4.2, p. 18. DOI: 10.1007/s41651-020-00059-2.
- Li, Wei et al. (Sept. 2021). “Pedestrian dead reckoning with novel heading estimation under magnetic interference and multiple smartphone postures”. In: *Measurement* 182, p. 109610. DOI: 10.1016/j.measurement.2021.109610. URL: <https://doi.org/10.1016/j.measurement.2021.109610>.
- Luo, H. and et al. (2021). “Integration of GNSS and BLE Technology With Inertial Sensors for Real-Time Positioning in Urban Environments”. In: *IEEE Access* 9, pp. 15744–15763. DOI: 10.1109/ACCESS.2021.3052733.
- Luo, Wanying and Urs Hengartner (Nov. 2010). “VeriPlace: a privacy-aware location proof architecture”. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems. GIS '10*. ACM. DOI: 10.1145/1869790.1869797. URL: <http://dx.doi.org/10.1145/1869790.1869797>.
- Moradbeikie, Azin et al. (2021). “GNSS-Free Outdoor Localization Techniques for Resource-Constrained IoT Architectures: A Literature Review”. In: *Applied Sciences* 11.22, p. 10793. ISSN: 2076-3417. DOI: 10.3390/app112210793. URL: <http://dx.doi.org/10.3390/app112210793>.
- Nasir, Qassim et al. (Sept. 2018). “Performance Analysis of Hyperledger Fabric Platforms”. In: *Security and Communication Networks* 2018, pp. 1–14. DOI: 10.1155/2018/3976093.
- Nighswander, Tyler et al. (2012). “GPS software attacks”. In: *Proceedings of the 2012 ACM conference on Computer and communications security*.
- Olfati-Saber, Reza, J. Alexander Fax, and Richard M. Murray (2007). “Consensus and Cooperation in Networked Multi-Agent Systems”. In: *Proceedings of the IEEE* 95, pp. 215–233.
- Oligeri, Gabriele et al. (2019). “Drive me not: GPS spoofing detection via cellular network: (architectures, models, and experiments)”. In: *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*.
- OMNeT++ Discrete Event Simulator — omnetpp.org (n.d.). <https://omnetpp.org/>. [Accessed 03-11-2023].
- P, Kongahage (2021). “A Novel Consensus Model for Blockchains Based on Physical Fitness”. In:
- Paek, Jeongyeup, Joongheon Kim, and Ramesh Govindan (2010). “Energy-efficient rate-adaptive GPS-based positioning for smartphones”. In: *ACM SIGMOBILE International Conference on Mobile Systems, Applications, and Services*.

- Paek, Jeongyeup, Kyu-Han Kim, et al. (2011). “Energy-efficient positioning for smart-phones using Cell-ID sequence matching”. In: *ACM SIGMOBILE International Conference on Mobile Systems, Applications, and Services*.
- Papadimitratos, Panos and Aleksandar Jovanovic (2008). “GNSS-based Positioning: Attacks and countermeasures”. In: *MILCOM 2008 - 2008 IEEE Military Communications Conference*, pp. 1–7.
- Pham, Anh et al. (Aug. 2016). “SecureRun: Cheat-Proof and Private Summaries for Location-Based Activities”. In: *IEEE Transactions on Mobile Computing* 15.8, pp. 2109–2123. ISSN: 1536-1233. DOI: 10.1109/tmc.2015.2483498. URL: <http://dx.doi.org/10.1109/TMC.2015.2483498>.
- Pino, E. S. et al. (Oct. 2019). “An Indoor Positioning System Using Scene Analysis in IEEE 802.15.4 Networks”. In: *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*. IEEE. DOI: 10.1109/iecon.2019.8926652. URL: <https://doi.org/10.1109/iecon.2019.8926652>.
- Psiaki, Mark L. and Todd E. Humphreys (2016). “GNSS Spoofing and Detection”. In: *Proceedings of the IEEE* 104, pp. 1258–1270.
- Qi, Hongxia et al. (June 2021). “BLE-based floor positioning method for multi-level atrium spatial environments”. In: *Acta Geodaetica et Geophysica* 56.3, pp. 471–499. DOI: 10.1007/s40328-021-00348-2. URL: <https://doi.org/10.1007/s40328-021-00348-2>.
- Qi, Jiahui et al. (2019). “Navigation with low-sampling-rate GPS and smartphone sensors: a data-driven learning-based approach”. In: *IET 8th International Conference on Wireless, Mobile and Multimedia Networks*.
- Reza Nosouhi, Mohammad et al. (Dec. 2018). “SPARSE: Privacy-Aware and Collusion Resistant Location Proof Generation and Verification”. In: *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE. DOI: 10.1109/glocom.2018.8647933. URL: <http://dx.doi.org/10.1109/GLOCOM.2018.8647933>.
- Rohani, Mohsen et al. (2015). “A New Decentralized Bayesian Approach for Cooperative Vehicle Localization Based on Fusion of GPS and VANET Based Inter-Vehicle Distance Measurement”. In: *IEEE Intelligent Transportation Systems Magazine* 7, pp. 85–95.
- Scoles, Sarah (2018). *Spoof, Jam, Destroy: Why We Need a Backup for GPS*. Wired. URL: <https://www.wired.com/story/spoof-jam-destroy-why-we-need-a-backup-for-gps>.
- Shen, Yuan, Henk Wymeersch, and Moe Z. Win (2010). “Fundamental Limits of Wideband Localization— Part II: Cooperative Networks”. In: *IEEE Transactions on Information Theory* 56.10, pp. 4981–5000. DOI: 10.1109/tit.2010.2059720. URL: <https://doi.org/10.1109%5C%2Ftit.2010.2059720>.
- Shi, Keqin, Kun Qian, and Hai Yu (2022). “Visual Human Localization and Safety Monitoring in a Digital Twin of Workspace”. In: *2022 41st Chinese Control Conference (CCC)*, pp. 6117–6121.
- Shokry, Ahmed, Marwan Torki, and Moustafa Youssef (2018). “DeepLoc”. In: *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM. DOI: 10.1145/3274895.3274909. URL: <https://doi.org/10.1145/3274895.3274909>.
- Soatti, Gloria et al. (2017). “Consensus-Based Algorithms for Distributed Network-State Estimation and Localization”. In: *IEEE Transactions on Signal and Information Processing over Networks* 3, pp. 430–444.

- Sohan, Asif Ahmed et al. (Dec. 2019). “Indoor Positioning Techniques using RSSI from Wireless Devices”. In: *2019 22nd International Conference on Computer and Information Technology (ICCIT)*. IEEE. DOI: 10.1109/iccit48885.2019.9038591. URL: <https://doi.org/10.1109/iccit48885.2019.9038591>.
- Songala, Komal Kumar et al. (2020). “Simplistic Spoofing of GPS Enabled Smartphone”. In: *2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)*, pp. 460–463.
- Ssekidde, Paul et al. (Feb. 2021). “Augmented CWT Features for Deep Learning-Based Indoor Localization Using WiFi RSSI Data”. In: *Applied Sciences* 11.4, p. 1806. DOI: 10.3390/app11041806. URL: <https://doi.org/10.3390/app11041806>.
- Talasila, Manoop, Reza Curtmola, and Cristian Borcea (2012). “LINK: Location Verification through Immediate Neighbors Knowledge”. In: *Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer Berlin Heidelberg, pp. 210–223. ISBN: 9783642291548. DOI: 10.1007/978-3-642-29154-8_18. URL: http://dx.doi.org/10.1007/978-3-642-29154-8_18.
- Tao, Fei et al. (2019). “Digital Twin in Industry: State-of-the-Art”. In: *IEEE Transactions on Industrial Informatics* 15, pp. 2405–2415.
- The AVISPA Project* (2017). URL: <https://www.avispa-project.org/>.
- Toyama, A. et al. (2021). “Implementation of an Indoor Position Detecting System Using Mean BLE RSSI for Moving Omnidirectional Access Point Robot”. In: *Springer Cham*, pp. 225–234. DOI: 10.1007/978-3-030-79725-6_22.
- Wu, Wei et al. (2020). “Blockchain Based Zero-Knowledge Proof of Location in IoT”. In: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pp. 1–7.
- Xu, Guangxia and Kaizhou Shi (2022). “An Improved Practical Byzantine Fault Tolerance Consensus Algorithm”. In: *2022 IEEE 4th International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, pp. 1083–1089. URL: <https://api.semanticscholar.org/CorpusID:255188626>.
- Yadav, Preeti, Subhash Chandra Sharma, and Vinay Rishiwal (2022). “Hybrid localization scheme using K-fold optimization with machine learning in WSN”. In: *International Journal of Communication Systems* 35.
- Yin, Feng et al. (2020). “FedLoc: Federated Learning Framework for Data-Driven Cooperative Localization and Location Data Processing”. In: *IEEE Open Journal of Signal Processing* 1, pp. 187–215.
- Zhang, J. and et al. (2019). “Robust RFID Based 6-DoF Localization for Unmanned Aerial Vehicles”. In: *IEEE Access* 7, pp. 77348–77361. DOI: 10.1109/ACCESS.2019.2922211.
- Zhen, J. et al. (2020). “An improved method for indoor positioning based on ZigBee technique”. In: *International Journal of Embedded Systems* 13.3, p. 292. DOI: 10.1504/IJES.2020.10996.
- Zhou, Xiaolei et al. (2018). “From one to crowd: a survey on crowdsourcing-based wireless indoor localization”. In: *Frontiers of Computer Science* 12, pp. 423–450. URL: <https://api.semanticscholar.org/CorpusID:7618954>.
- Zhu, Zhichao and Guohong Cao (Apr. 2011). “APPLAUS: A Privacy-Preserving Location Proof Updating System for location-based services”. In: *2011 Proceedings IEEE INFOCOM*. DOI: 10.1109/infcom.2011.5934991. URL: <http://dx.doi.org/10.1109/INFOCOM.2011.5934991>.