UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Eduardo Ribas Brito

Towards Decentralized Proof-of-Location

Master's Thesis (30 ECTS)

Supervisor: Ulrich Norbisrath, PhD

Tartu 2023

Towards Decentralized Proof-of-Location

Abstract:

Location-based services have become ubiquitous in today's society, and their integration with various applications and technologies has shaped our interaction with the physical world. The current state of location-based systems is very far from ensuring integrity of the location data, especially in trustless environments, with no individual reliability guarantees. A paradigm shift is needed, in order to provide security against geo-tampering or location spoofing. To address such requirements, digital and verifiable Proof-of-Location systems may help on materializing the idea of location-based authentication or authorization, in adversarial environments. Such systems allow for a vast range of applications in the fields of smart cities, augmented democracy, digital integrity, liability, and internet transparency. In this thesis, we present a novel approach to the problem, dissecting the Proof-of-Location systems' paradigm and building upon existing work, to further prototype the path towards fully decentralized Proof-of-Location. Making use of mesh network technologies and permissionless consensus mechanisms, we specify a new protocol, and implement and evaluate a proof-of-concept, showcasing the generation of complete, verifiable, and spatio-temporally sound location proofs.

Keywords: location-based services, proof-of-location, mesh networks, permissionless consensus, blockchain, smart contracts

CERCS: P170 - Computer science, numerical analysis, systems, control

Suunas detsentraliseeritud asukoha tõendamisele

Lühikokkuvõte:

Asukohapõhised teenused on tänapäeva ühiskonnas üldlevinud ning nende integreeritavus erinevate rakenduste ja tehnoloogiatega on kujundanud meie suhtlust füüsilise maailmaga. Asukohapõhiste süsteemide praegune seisund ei taga kaugeltki asukohateabe terviklikkust, eriti vaenulikes keskkondades, kus puuduvad individuaalsed usaldusväärsuse tagatised. On vaja paradigma muutust, et tagada turvalisus geograafilise manipuleerimise või asukoha võltsimise vastu. Sellistele nõuetele vastamiseks võivad digitaalsed ja tõendatavad asukoha tõendamise süsteemid aidata realiseerida asukohapõhist autentimist või autoriseerimist vaenulikes keskkondades. Sellised süsteemid leiaksid laialdaselt rakendust nutikate linnade, laiendatud demokraatia, digitaalse terviklikkuse, vastutuse ja interneti läbipaistvuse valdkondades. Selles lõputöös tutvustame probleemile uudset lähenemist, uurides asukoha tõestamise süsteemide paradigmat ja arendades olemasolevat tööd, et liikuda täielikult detsentraliseeritud asukohatõenduse poole. Kasutades võrgusilma tehnoloogiad ja lubadeta konsensusmehhanisme, määratleme uue protokolli ning rakendame ja hindame kontseptsiooni tõestust, mis tutvustab täielike, kontrollitavate ning ruumilis-ajaliselt kõlav asukoha tõendite genereerimist.

Võtmesõnad: asukohapõhised teenused, asukohatõend, võrgusilma võrk, lubadeta konsensus, plokiahel, targad lepingud

CERCS: P170 - Arvutiteadus, arvanalüüs, süsteemid, kontroll

Contents

1 Introduction

Throughout history, humans have sought to accurately locate themselves, within the vastness of space and time. From mapping their surroundings and establishing their borders, to navigating the seas, with the help of stars, humans have always had an intrinsic need for fundamentally solving the localization problem, asserting their physical existence in the world. This need has been adaptively met through various means, including the use of maps, compasses, and astronomical observations. Today, with the advent of technology, we have access to advanced methods of precise localization, like the well-known GPS-based systems. Applications that rely on such systems operate founded in the expectation that the involved parties are sufficiently honest to be correctly synchronized in space and time, having their individual receivers relatively calibrated, and marking the correct value in their internal clocks. In this thesis, we will address the design challenges of trustless systems that aim at ensuring integrity of the location data, examining the rise of location-based authentication or authorization and the need for tamper-proof, correct, and spatio-temporally sound location information. In GPS-based systems, strategic interactions between rational agents often end up supporting this implicit trust. One party provides a location-based service, and another party makes use of it for its individual benefit, by providing an allegedly non-tampered time-conscious piece of location claim. This interaction appears to be one of a cooperative strategy, that can be observed in most mapping platforms, navigation systems, or mobility and ride-hailing apps. If driven by the reasoning goal of extracting correct information from the interacting system, users are logically motivated to calibrate their devices and report an accurate location. The services, with the higher goal of not losing users, due to malfunctioning or inaccuracy, are thus motivated to provide maximized quality when consuming the location data.

This paradigm is now ubiquitous, but it may be structurally unsuitable for other scenarios. Our work stresses those situations that, therefore and inversely, fundamentally require verifiable Proof-of-Location to assert a particular state or derive a conclusion. The trust levels required, in these scenarios, to testify to one's location claims ought as well to be crucially measurable. The concept of location-based authentication or authorization in adversarial environments that rely on information gathered in a trustless setup eventually materializes into services requiring, for instance, a digital certificate as proof that a given user is within a particular geographical area, to enable certain functionalities or assert liability. Applications of such services include customer reward systems for physical stores, location-authenticated business review platforms, location-restricted web content delivery, or voter's physical presence attestation. The present surge of highly realistic generative AI tools has also made the case for the need to certify the physical originality of digital content, by photographers, reporters, and other content creators. Another use case is the verification of the provision of services, such as the delivery or supply of goods, by third-party providers. Security against geo-tampering or

location spoofing in a relatively trustless environment is needed to achieve high levels of integrity. In consequence, enforcing and contributing to tamper-proof, correct, and censorship resistant location information, in today's chaotically data driven world, may preemptively demand for novel efforts.

The basic infrastructural concept of a Proof-of-Location system is somewhat understood, and theoretical or experimental solutions have been delivered throughout the years. These solutions have evolved parallel with their trust assumptions, beginning with a fully trusted setup and progressively shifting towards modern requirements for operational decentralization, with an inevitable distribution of resources, power, and profit. Recent attempts contemplate the need for a permissionless means of reaching consensus between a quorum of witnesses, which can attest to one's presence at a given point in space and at a given moment in time. These concepts take shape with a combination of tools: wireless technologies as short-ranged message-exchanging means, cryptographic protocols as confidentiality, integrity, or authentication enablers, and distributed ledgers as publicly trusted record keepers. The quest for a solution that could make this kind of location-based services as prevalent and ubiquitous shall aim to address a set of design challenges. These challenges are, among others, the solution's flexibility and deployability, preferably by making use of existing infrastructure, and the solution's security and privacy, obeying the modern cryptographic standards and requirements, to guarantee envisioned levels of integrity and resiliency to attacks. This thesis, aiming to address these matters, delivers the following contributions:

1. It provides an overview of the Proof-of-Location systems' paradigm, including the underlying premises and the strategic interactions between rational agents, along with a review of the state of the art in the field. The review is categorized in terms of trust levels, from fully trusted to permissionless environments, and, consequently, in terms of infrastructure, from centralized to decentralized systems.

2. It also attempts at the design of a novel Proof-of-Location protocol and the implementation of a proof-of-concept that can be deployed in a permissionless manner, using existing technology. The work is specifically based on permissionless consensus mechanisms and the use of routing protocols, for mobile ad hoc mesh networks. The goal is to set up a mesh network of witnesses that can collectively agree to assert one's presence in a given geographical area.

The structure of the work is as follows. In Chapter 2, an introduction to the underlying concepts, hypotheses, and applications is provided, together with an outline of the technology involved in the practical implementation. Chapter 3 examines similar work, classified according to different trust levels. In Chapter 4, a general overview of the requirements for the proposed solution is given. Chapter 5 details the architecture's design, implementation, and evaluation. Finally, Chapter 6 presents the conclusion and recommendations for future work.

2 Background

This chapter introduces not only the underlying concepts that sustain the work, but also the technology involved in implementing the proposed practical solution. First, in Section 2.1, we state and define the Proof-of-Location problem, its participants, the most common threat models, and a non-exhaustive list of some application scenarios. Section 2.2 reviews the concept of Mesh Networks and related routing protocols for establishing nearby witnessing. Lastly, Section 2.3 introduces the permissionless consensus problem and its role in achieving agreement for obtaining a location proof in a trustless environment.

2.1 Proof-of-Location

The problem of attesting to one's location is a fundamental act of metaphysical reasoning that happens everywhere, at every moment. Unconsciously and unwittingly, we do claim to be somewhere at an indiscriminate point in time, and we do expect others to believe in us. However, this act is grounded on informal and implicit levels of trust that are not often explicitly asserted, as liability is usually not categorically assigned. When it does happen, trust is usually delegated to a third party or distributed between multiple parties, that may be able to testify to one's presence, synchronously, at the very same location. The act of witnessing is, therefore, a regular yet fundamental part of our interactions with physical reality. When we do claim our presence at an event, assert our location to a service provider, or even state our alibi to authorities, as defence in a criminal charge, a protocol for location attestation is implicitly followed. Some may require a physical interaction of any kind, while others may find digitalized and infrastructural means to gather the required location proof [1].

A digital Proof-of-Location can then be defined as an electronic certificate that attests one's relative position in both space and time [2]. The relativity of the attestation is, nevertheless, a non-trivial matter. It is, in fact, a complex and multi-faceted process that requires the simultaneous existence of various untrusted or semi-trusted parties, especially in an environment with no individual honesty guarantees. According to Nasrulin et al. [3], a Proof-of-Location protocol may be considered secure if complete, spatio-temporally sound, and non-transferable. Consequently, the system that materially backs the implementation of such a protocol is expected to provide fault-tolerance, reliability, and availability guarantees. More advanced protocols may also explore the possibility of providing privacy and anonymity assurances [4], as well as the possibility of being used in a fully trustless environment [2]. Chapter 3 will later and deeply explore the particularities of some of these solutions. Following is the conceptualization of the common entities of a Proof-of-Location protocol plus an attempt of a formal and general definition of the problem.

2.1.1 Parties Involved

The general act of witnessing alludes to the simultaneous spatio-temporal existence of a set of entities with distinct roles. The majority of the protocols convey a clear contrast between these roles, highlighting the relative dynamism that differentiates those entities.

Concisely and in concrete terms, these location-proof arrangements expect the existence of a *prover* that engages in any communication protocol with nearby participants, the *witnesses*, with the goal of gathering a verifiable Proof-of-Location claim, to be later presented to a *verifier*, therefore convincing it of one's existence within a geographical area, at a given moment [5] (see Figure 1).

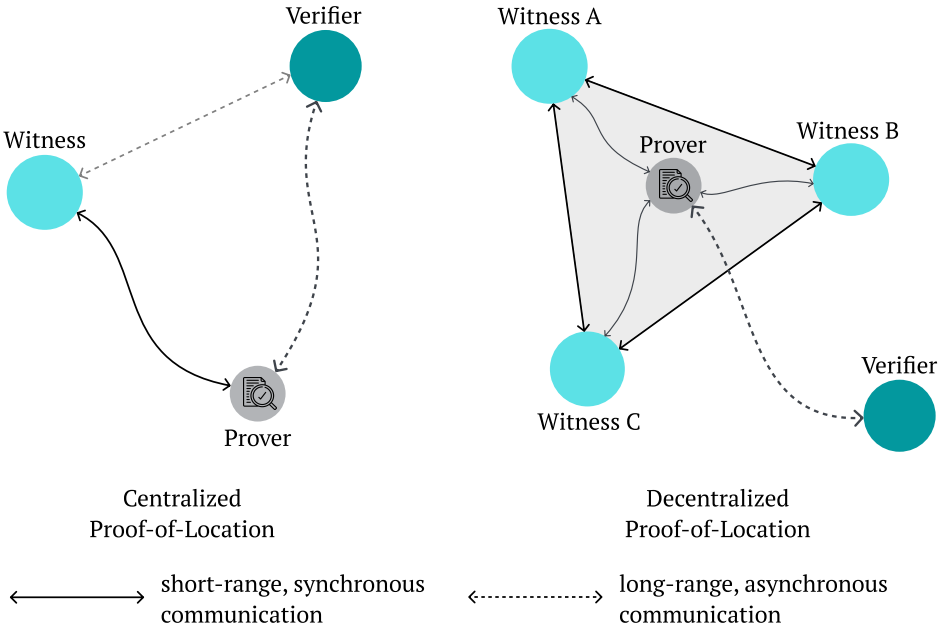


Figure 1. The entities involved in a Proof-of-Location protocol. The left side of the figure represents the typical arrangements of a centralized protocol, where the witness and the verifier may establish some bond between each other. The right side shows the configuration of a decentralized and trustless solution, where a quorum of witnesses attests the prover's location.

Prover. A prover is a dynamic entity, both in movement and availability terms. It is expected to be able to communicate with the witnesses, to gather a proof of its location, and to be later able to provide a location claim to the verifier. Communication with nearby witnesses is thought to happen via any short-ranged message transmission means.

Provers are also expected to be associated with a verifiable but desirably private identity, often as a pseudonym.

Witness. A witness is an entity that is expected to be able to communicate with the prover via the same short-ranged communication channel and to provide it with a verifiable piece of location attestation. These parties are envisioned to statically or dynamically exist around the prover's location, during the protocol process, and to maintain, in the most recent decentralized protocols, a relatively stable neighbouring list of nearby peers. Witnesses are also expected to be fictitiously identified, usually by a pseudonym.

Verifier. A verifier is an external entity that is able to receive a location claim from a prover and verify its validity. Although possible and predicted for trusted setups, in a trustless environment and with the general assurances of a permissionless protocol, verifiers shall not have the need to communicate directly with the witnesses. Verifiers' identity is also of no specific importance for the protocol, since the interaction with the prover is usually asynchronous and external to the witnessing process.

Inspired by [3, 5], we now introduce a substantially formal but general definition of the Proof-of-Location problem, along with some of its desirable properties:

Definition 1 (Proof-of-Location). A Proof-of-Location is a verifiable digital certificate that attests the presence of a prover ρ at location l and time t.

Definition 2 (Completeness). A Proof-of-Location is complete if the prover ρ is attested at location l and time t, by a set of witnesses $\omega \in W$.

Definition 3 (Spatio-temporal Soundness). A Proof-of-Location is spatio-temporally sound if it is generally hard for the prover ρ to obtain, forge, or modify a complete Proof-of-Location, if not physically present at location l and time t.

Definition 4 (Non-transferability). A Proof-of-Location is non-transferable if valid only for the prover ρ that obtained it.

Definition 5 (Correctness). A complete Proof-of-Location, generated by an honest prover ρ, in cooperation with honest witnesses $\omega \in W$, must always be accepted by an honest verifier ν.

Additional properties may be protocol specific, but the above definitions are generally considered to be the most common and desirable properties of a Proof-of-Location protocol. Further formalizations can be found in the works dissected in Chapter 3.

2.1.2 Common Threat Models

Like with any technology that involves the collection and processing of sensitive and tamper-prone location data, Proof-of-Location systems must be designed and implemented with a keen awareness of the threat landscape. The threat models of these systems are very often intricately multisided, encompassing a diverse range of actors, motives, and attack vectors. In this context, it is crucial to understand not only the technical mechanisms of Proof-of-Location systems, but also the broader factors that shape their security and privacy risks.

Some common scenarios that may affect the security of Proof-of-Location systems are, for instance, malicious provers that may attempt to forge location claims, or witnesses that may attempt to collude with other entities to falsify the information. Adversary efforts may also be observed in the form of baleful provers, or witnesses, that may try to respectively impersonate other peers. Sybil attacks are too on the horizon of possible threats, often employed to disrupt the operation of the system by flooding it with fake participants [3]. Other works have also considered semi-honest adversaries that, despite following the protocol rules, may try to learn additional information from the messages exchanged [5]. These and other attack vectors are further dissected in Chapter 3, with reference to the multiple solutions that attempt at being shielded from these malicious situations.

2.1.3 Application Scenarios

The concept of verifiable and digital Proof-of-Location has a wide range of applications, as deconstructed by Sariou and Alec, in [6]. For instance, in customer reward systems, Proof-of-Location can be used to provide incentives to customers who visit physical stores, offering rewards and loyalty programs to verified visitors. In location-authenticated business review systems, Proof-of-Location can be used to verify the authenticity of customer reviews. By requiring customers to verify their physical presence at the business location, businesses can prevent fake reviews and ensure that only genuine customer feedback is posted. In location-restricted web content delivery, Proof-of-Location can be used to limit access to online content based on the user's physical location. For example, a video streaming service may use verifiable Proof-of-Location to prevent users from accessing content outside legally allowed regions or countries. In voter's physical presence verification, a Proof-of-Location protocol can be set to prevent voter fraud, by verifying that voters are physically present at the polling stations. Remote working and residency verification, for tax purposes, fit too in the realm of possible

applications. Pournaras [7] goes beyond these specific cases and theorizes about an augmented democracy approach to smart city development. In the author's hypothesis, Proof-of-Location is used to create a transparent and participatory decision-making process by enabling citizens to verify their physical presence at public meetings and events. Proof-of-Location may also play a role in the smart mobility infrastructure, helping in verifying and optimizing the use and cost of public transportation.

Tailored to the reality that wraps the writing of this thesis, this problem becomes increasingly relevant with the advent of Artificial Intelligence (AI) tools, capable of forging highly realistic content at unprecedented scales. To combat such surge, photographers and reporters may make use of Proof-of-Location to certify the physical originality of their photos and videos, while journalists may use it to verify the authenticity of their sources. Another use case is the verification of the provision of services, such as the delivery or supply of goods, by third-party providers. For instance, an Internet Service Provider (ISP) may use Proof-of-Location to attest the physical deployment and provision of connectivity to a group of customers in a particular area. This can be extended to attest, as well, the quality of the service and to prevent the ISP from charging for services that are not provided, increasing overall trust and transparency. For the first use case, let's imagine a reporter who arrives at an accident site, in a busy city centre. The reporter takes out his camera and captures photos and videos of the crash. As he records, several bystanders notice and approach him. Since they all simultaneously witness the reporter's presence at the accident site, they are able to attest his physical presence at the location and time of the event, certifying the authenticity of the content that would feature later in the news. In the second use case, a group of customers has been waiting for their ISP to provide connectivity to their neighbourhood. The regulators demanded proof of the ISP's service provision, after it claimed to have provided the necessary infrastructure. To generate such proof, customers are asked to coordinate their efforts and attest the service in their neighbourhood. They all synchronously connect their devices to the infrastructure and attest the ISP's connectivity, proving it met the expected standards. They all proceed to sign the attestation and submit the proof to the regulatory authorities. These two examples are depicted in Figure 2 and will feature in the following chapters, bridging the specification of the Proof-of-Location protocol with real world use cases.

Targeting different trust levels, additional application scenarios can be derived from the above, while many others are yet to be discovered. Given all this, digital and verifiable Proof-of-Location has a wide range of applications that may disruptively benefit individuals, businesses, and the society as a whole. Chapter 3, afterwards, gives a more nuanced overview of both the evolution of the protocols and specific use cases that these multiple solutions aim at covering. The next section will introduce the technologies that are set to enable short-range communication between the entities of a Proof-of-Location protocol.

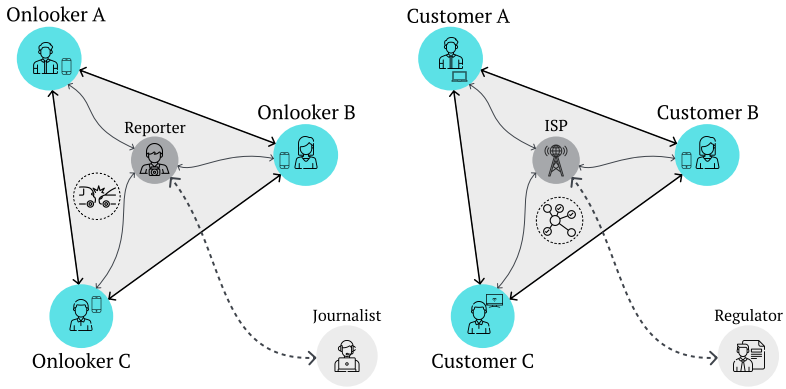


Figure 2. Examples of Proof-of-Location applications.

2.2 Mesh Networks

The envisioned fourth industrial revolution has set the track for modern advancements in achieving a global web of pervasive connectivity between all sorts of machines [8, 9]. New means of radio and wireless communication have been pushing for the technological heterogeneity of protocols, architectures, devices, and consequent performance levels, in order to find their design suitability for different coverage or range scenarios, transmission or bandwidth rates [10]. Additionally, requirements for more complex, adaptable, and resilient topologies have captured broad interest, in both academic and industry domains.

The development of new hardware, protocols, and applications started gaining momentum and branched their way forward to support the popularisation of Wireless Mesh Networks (WMNs). In mesh topologies (see Figure 3), network nodes are directly and dynamically connected in a non-hierarchical way. This trait eventually allows for many-to-many communications between the devices, to efficiently route data from a generic source to a generic destination. The infrastructure nodes that make up the mesh are expected to dynamically self-organize and configure themselves, resulting in beneficial distributed effects on the overall fault tolerance, ease of deployment, and workload allocation [9, 10]. WMNs follow these principles with the particularity of being made up of radio nodes that communicate via any sort of wireless means.

Some of the most common technologies that have been, throughout the years, ported to WMNs are Bluetooth, LoRa and IEEE 802.11. The first two are prominent solutions for the extremities of the mesh networking spectrum, with Bluetooth under the short-range realm of Personal Area Networks (PANs), and LoRa under the Low Power, Wide Area (LPWA) scenario. Downsides of these technologies are, respectively, the limiting coverage range for one-hop neighborhoods, or the low bandwidth rates [9]. Hence, IEEE 802.11 became the most flexible and widely used technology, being the basis of the Wi-Fi standard, which, at the beginning of the last decade, saw an amendment that

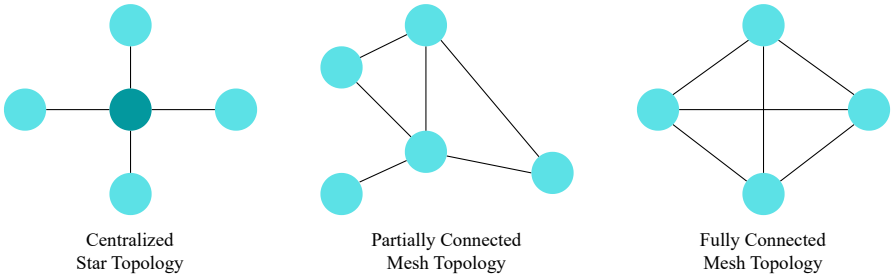


Figure 3. Examples of different topologies of a computer network. Starting on the left, the star topology shows all the nodes connected to a central hub. To the right, examples of mesh topologies depict the direct and decentralized connections between the nodes.

mainly targeted mesh networks — the IEEE 802.11s WLAN Mesh Standard [11]. The novelty came with the introduction of routing mechanisms operating at the ISO/OSI Layer 2, allowing for compatible information delivery in the layers above. The dynamic establishment of a topology for IEEE 802.11s-based mesh networks relies on the phased transmission of beacon messages that allow for the discovery, synchronization, and maintenance of the links between the peers. IEEE 802.11s has a default routing protocol, the Hybrid Wireless Mesh Protocol (HWMP), which is based on a series of flooding procedures for both proactive and reactive path finding and selection [12]. However, this protocol is not strictly enforced by the standard and has been replaced by other more popular solutions. One notable example is the Better Approach To Mobile Ad-hoc Networks (B.A.T.M.A.N.) routing protocol.

This thesis will explore the concept of mesh networks and their potential for serving as the infrastructural topology that enables the relatively short-ranged exchange of messages between the participants of a Proof-of-Location protocol. The following sections will present the B.A.T.M.A.N. routing protocol, OpenWrt, and other relevant tools that will be later used to implement the proof-of-concept.

2.2.1 B.A.T.M.A.N. Routing Protocol

The Better Approach To Mobile Ad-hoc Networks (B.A.T.M.A.N.)[1] is a proactive routing protocol for WMNs, operating at the data link layer instead of the network layer, asserting the reliability of radio links using routing metrics and a distance-vector approach [13]. Its newer version, *batman-adv*, has gained traction and popularity and eventually made itself available in the Linux kernel.

Route discovery is preemptively replaced with neighbour discovery, and each infrastructural node is instructed to calculate its potential best next-hop, significantly reducing

[1] `http://www.open-mesh.org/`

the overhead of requiring each peer to be aware of the whole network topology. Its version V introduced a throughput metric to evaluate the links' quality and routing choices, replacing version IV packet-loss metric, deemed unsuitable for larger network sizes [13].

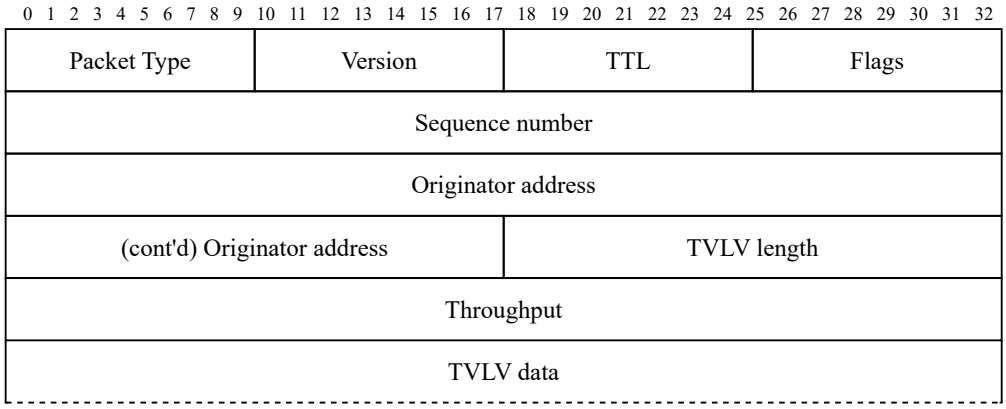


Figure 4. OriGinator Message version 2 (OGMv2) packet format [9, 14]. These messages are broadcasted with a collision avoidance delay mechanism defaulting to 1 second. The packets contain, among other fields, the originator's MAC address and throughput metric values, measured in units of 100 kbit/s.

The discovery of neighbouring nodes is accomplished with the capture of broadcasted OriGinator Messages (OGMv2, see Figure 4), that feature a collision avoidance delay mechanism, the detection of new or duplicate messages, and other fields for throughput measurement and gateway discovery. The Echo Location Protocol (ELP) handles the received messages and ranks the discovered neighbours. The OGM flooding protocol, on the other hand, enables mesh routing procedures that, simultaneously but independently, allow for estimating the quality of the individual links [9]. Additionally, the protocol facilitates OGM aggregations as an effort to reduce the overhead of sending many short-sized frames. Nevertheless, there is still a quest for optimizations that would allow for more efficient use of multiple interfaces. An implementation of a subset of the Internet Control Message Protocol (ICMP) is also made available, allowing, for instance, the use of the *ping* command to test the connectivity between nodes [13].

Building on the previous, the B.A.T.M.A.N. routing protocol has been, through multiple initiatives, successfully blended into the OpenWrt project, which will also be employed in the proof-of-concept. The following section will present OpenWrt and other relatable tools.

2.2.2 OpenWrt, QEMU, and Raspberry Pis

The OpenWrt project[2] is a Linux distribution for embedded devices, which, in the context of this thesis, will serve as the host operating system for running the proof-of-concept solution. The project is based on the Linux kernel, encapsulating several of its libraries and packages, and is designed to be used on resource-constrained devices. OpenWrt features not only a writable root filesystem and automated build tools with integrated cross-compiler toolchain, but also a package management system that allows for the installation of additional software. The project also provides extensive configuration options for networking capabilities, which includes enabling mesh networking support through the B.A.T.M.A.N. routing protocol.

To facilitate the development and testing of the proof-of-concept, the QEMU[3] emulator will be used. QEMU is a generic and open-source machine virtualizer that, through its versatile set of features, allows for the full-system emulation of a wide range of hardware and software. The emulator will run the OpenWrt generated images and spawn multiple virtual machines. These machines will simulate the various protocol participants by establishing, with the help of the network emulation tools, a fully connected mesh network. The intention is to ease and accelerate the development process by allowing for testing the proof-of-concept in a controlled environment, without the management, maintenance, and deployment hustle of physical devices. Later, the solution is planned to be deployed on a set of Raspberry Pis[4], the most widely used single-board computers for developing IoT solutions. The implementation journey will be documented in Chapter 5.

2.3 Permissionless Consensus

Long has been the time when consensus was still on the verge of being considered such a fundamental problem of distributed systems. Generally defined by Lamport et al. [15, 16], consensus means reaching an agreement between multiple parties in the potential presence of faulty individuals. As per multi-agent systems, interacting over computer networks, consensus is thought to be the result of a coordination effort, that eventually leads the parties to agree on some value at a given moment. However, the evolution of the consensus problem has been invariably limited by a set of strong assumptions. The well-known Byzantine-Fault-Tolerant multiparty consensus systems, that have been designed over the years, are usually meant to work only with a set of known participants, being them faulty or not [17].

The other side of the coin is the permissionless consensus challenge, consisting of achieving agreement in an environment where the parties are unknown and untrusted [18, 19]. The relative openness and lack of any kind of central authority are other intrinsic

[2] https://openwrt.org/
[3] https://www.qemu.org/
[4] https://www.raspberrypi.org/

particularities of this type of networks, which inevitably adds complexity to the problem. The participants are not only unknown and untrusted but can also join or leave the network at any time, freely choosing if they care to participate in the consensus protocol. Nevertheless, the problem of permissionless consensus is still seen as a special case of the general consensus definition, but under more meticulous trust assumptions.

Further in this thesis, we will evaluate the different high-level Proof-of-Location protocols and draw a parallel between the evolution of their trust levels and the ultimate need for a low-level permissionless consensus algorithm that allows for establishing decentralized and time-conscious agreement, in an eventual trustless setup, between the multiple witnesses. The next subsections will briefly review some of the most relevant aspects and proof units that give practicality to the roots of the permissionless consensus problem.

2.3.1 Proof-of-X

The solution is, nonetheless, unsettled and the scientific community has been reasoning about the need for permissionless consensus when there are already well known and established consensus protocols that work in trusted environments [17, 20]. However, even those protocols have their own limitations, not only in terms of trust, fault-tolerance, centrality, permissions, or bottlenecks, but also in terms of scalability [20], despite assuring deterministic finality [21]. The need for permissionless consensus is then justified by the fact that permissioned protocols are not compatible with the requirements of the new generation of distributed systems, especially in the context of Blockchain networks. These requirements include dealing with today's sparse networks of anonymously and dynamically participating devices, without interrupting consensus and while battling the disruption of the system, typically by subverting it with many pseudo-entities — the so-called Sybil attacks [22, 23]. Fundamentally, the permissionless consensus problem is the need for a consensus protocol that can be run in a distributed and decentralized environment, where the participants are unknown and untrusted, and where the network is bigger, sparser and unpredictably less reliable.

Technically, permissionless environments allow for larger networks that depict lower connectivity between the participants. Operationally, everything is expected to happen in an asynchronous or partially synchronous fashion, and the number of transactions is predicted to be smaller than in the permissioned counterparts. Participation is free, and the governance is not centralized, but rather distributed and public. The identity of the participants is secured or semi-secured as it often relies on pseudonymity for protecting the nodes' identity, enabling, at the same time, full transparency concerning the rest of the network's content and operation [24]. Expectedly, the goal of permissionless consensus, as for any consensus protocol, is to reach agreement on a single value, or a set of values. However, due to the nature of the protocols, the values that are agreed upon end up establishing the serialization of the transactions, and so establishing time

consciousness and total order of the events [22].

Also described by Xiao et al. [23], very concisely, the way to achieve an operating protocol, as seen in the mainstream blockchain networks, is by first generating the agreeable value, in this particular case, a block and its proof. Next is the phase of proposing and disseminating the information to the network, followed by the eventual validation and acceptance of the block by the majority of the nodes. This is the approximate moment of probabilistic finality, when consensus is ultimately reached (see Figure 5). During the whole process, a fair and somewhat predictable incentive mechanism is also needed, that rewards participants for their honest effort in reaching consensus, and punishes the ones that are not behaving correctly. These incentives are of major importance in this very context of permissionless consensus, and all these building phases form the basis of the inner functioning of Bitcoin itself [18], replicated with some variations in other networks [19, 23]. The following section is a short introduction to some relevant proof units that feature in the most popular blockchain systems.

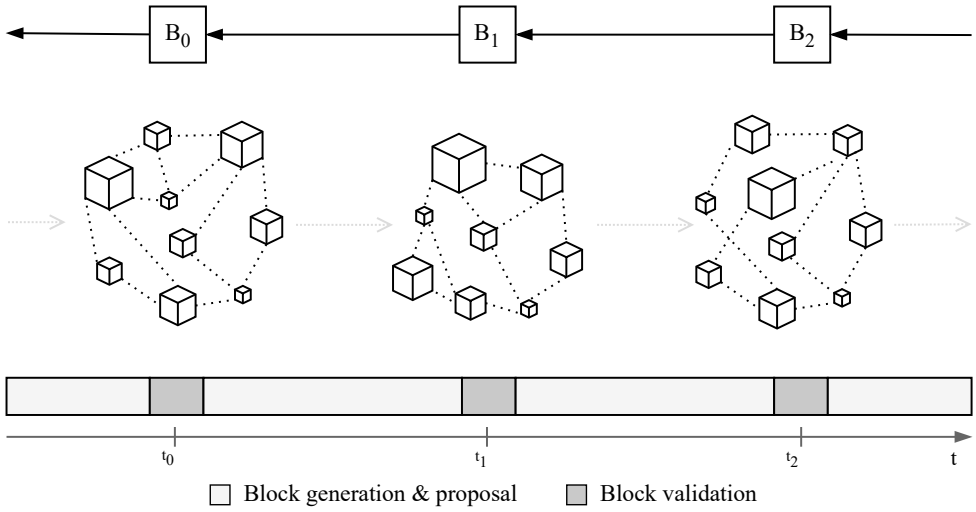


Figure 5. An illustration of the permissionless consensus building phases. From the bottom to the top, the asymmetric arrow of time discretizes the block generation and proposal phases, followed by the block validation, along with frequent network topology changes and the consequent time conscious serialization of the blocks.

2.3.2 Proof-of-Work and Proof-of-Stake

Without discrediting the previous attempts, the first practical permissionless consensus algorithm was proposed by Nakamoto in [18]. It is a Proof-of-Work consensus protocol that resembles a replicated state machine where the independent participants reach

agreement not only about transactional values, but also about their order — naturally forming the underlying structure of what is now known as a blockchain. The focus shifted for decentralized systems and after Proof-of-Work many other consensus mechanisms have been proposed, relying on different consensus units.

In the classical Nakamoto consensus protocol, the generation of a block, to be proposed for further network agreement, complies with the unit of computational work needed to create, or rather find, a verifiable proof of the effort spent on assembling the block [18]. This essentially requires brute forcing the search for a cryptographic hash value for the aggregation of the block information with a nonce. This value has to satisfy a difficulty threshold (see Procedure 1), which gets adjusted dynamically over time, to maintain the network overall requirement for the block generation interval [22, 23].

Procedure 1: BlockGeneration

Input: Transaction Merkel Tree Root, Hash of the last Block, Timestamp, Other.
Result: new $Block$.

$BlockHeader \leftarrow$ Transaction Merkle Tree Root
    | Hash of the last Block
    | Timestamp
    | Other;

`// the preceding zero bits in` $target$ `depict the mining difficulty`
while $Hash(BlockHeader \mid nonce) \geq target$ **do**
    Increment $nonce$;

`// append transactional data`
return new $Block$;

One can then exercise the reasoning line and extrapolate the previous block generation mechanism to a *Proof-of-Something* pseudo-random competition in which an entity in possession of a higher amount of a certain resource, either computational power, or stake, or certain currency, or, for instance, a higher amount of storage space, guarantees a higher probability of leading the block generation and proposal, consequently winning the acceptance by the majority. This is the essence of Proof-of-Stake, as a derivative of the Proof-of-Work mechanism. Here, stake is a traceable and verifiable amount of a certain unit, token or currency, that is owned by a certain entity who wishes to participate in the consensus protocol. The stake works as a form of collateral that is used to guarantee everyone's honesty, in an attempt to reduce the Sybil attack likelihood. And, respectively as in Proof-of-Work with computational power, the higher the stake, the higher the probability of leading the block generation and proposal.

Idealized and inspired by Proof-of-Stake, extending or adapting Proof-of-Work became a popular trend in the blockchain community. The main idea is to replace the

computational power with some other resource, that is more scarce, or more valuable, or more verifiable, or more traceable, to combine multiple resources, or even to add extra requirements to pure Proof-of-Work [23]. Not that every one of the options has a considerable potential for entirely solving the permissionless consensus problem, but each one of them may tackle different use cases where consensus needs to be reached, and where different resources are available to make the agreement happen [25, 26]. Nonetheless, the design of these consensus mechanisms shall aim for a protocolar choice between a set of properties that form a trilemma: security, scalability, and decentralization. Briefly put, relaxing the security requirements may allow for more scalability, both of which, consequently, have hands tied with decentralization. These trade-offs are of practical consideration when defining the network goals and use cases [23]. Further dissection of various classes of Proof-of-Stake based protocols, diverging alternatives to the classic Nakamoto consensus, and comparisons between them can be found in [22, 23, 25–27].

With all the above in mind, we will proceed to review some of the proposed Proof-of-Location solutions, discriminated by trust levels. Aiming at achieving spatio-temporal agreement among the witnesses, we will reason about the applicability of one of these permissionless consensus protocols, in the context of a fully decentralized and trustless environment.

3 Related Work

This chapter presents a description of the current state of the Proof-of-Location problem, spanning the spectrum of its trust levels, from fully trusted to permissionless environments. Furthermore, it encompasses an assessment of the typical infrastructural scenarios, detailing the progressive shift from centralized to decentralized systems. The organization of the chapter is as follows. Section 3.1 outlines the starting point in a trusted and centralized setting. Section 3.2 details the progressive shift towards distributed and decentralized protocols. Finally, Section 3.3 presents the most recent developments in the Proof-of-Location problem, which ultimately target permissionless and fully trustless setups.

3.1 Trusted and Centralized Architectures

The establishment of not just the concept, but also the need for a new kind of systems that, in simple terms, would allow for attesting and prove some device's location, dates back to the early beginnings of this century.

Waters and Felten, in [28], attempt at pioneering the design of a location-proving system by proximity that simultaneously ensures integrity and privacy. The system model assumes a fully trusted setup, fundamentally composed by two entities, a *verifier* and a *device*. The latter is implicitly thought to be managed by an untrusted *user*, but hypothesised and expected to be tamper-resistant, and thus, trusted by the verifier. The motivation behind this scenario is oriented towards practical situations in which, for instance, trusted parties lend their equipment to users and want to verify, or monitor, the equipment's location, such that it remains inside some pre-established location boundaries. The authors explicitly mention the lending of computers by universities and the wish that those devices to not leave the campuses. Home arrest monitoring systems have, as well, the need for ensuring that the ankle device, and so the person in charge, does not escape a certain location.

Faced with the design and coverage unadaptability of GPS-based location systems, which do not structurally aim at serving as Proof-of-Location enablers, the authors identify the need for small wireless networks, covering a relatively short-ranged area, via a *location manager* that acts as an access point. Figure 6 illustrates the multiple entities of the protocol. The device is set to accept input from an untrusted source, the user, containing the identity of a nearby location manager. After receiving this input, the device will try to prove it is physically close to the designated location manager. Aiming at a more secure and precise positioning system, the device may conduct three simultaneous proofs of proximity with three location managers, to determine its relative location. These location managers are either distinctively trusted, or set up by the verifier [28]. Round-trip and signal propagation latency are the metrics used, respectively, for determining the proximity of the device to the location manager and for protecting

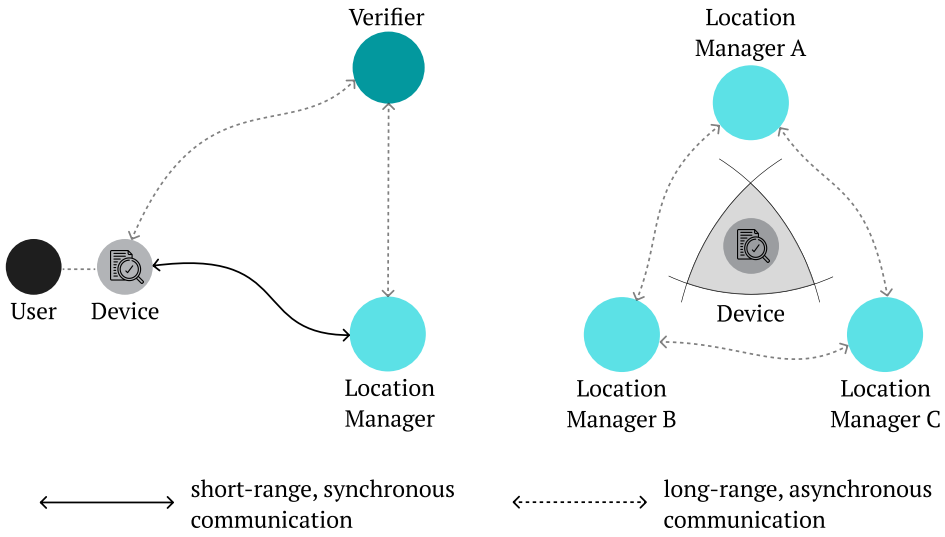


Figure 6. Waters and Felten's trusted and centralized Proof-of-Location system [28].

against proxy attacks — when a proxy device is placed near the location manager and serves as signal repeater for the original device that is somewhere else, outside the coverage area. Concretely, the work targets Wireless LAN network operators and their existing access points' infrastructure, to serve as location managers. A Public Key Infrastructure (PKI) is also proposed in order to delegate the atomic responsibility of authenticating and managing their identities to a trusted third party. Finally, the authors set down the seeds for extending their proximity proof system to a secure and moderately accurate positioning proof mechanism, with the possibility of using multiple location managers and a triangulation algorithm.

The Proof-of-Location track was inevitably unfogged with this primordial work and location proofs were soon to be fully demystified and categorically defined. Sariou and Alec, in [6], concisely introduce the primitive concepts around Proof-of-Location and some desired properties of an inherently secure system. However, the key contribution of their work was the delineation of a set of example applications that would benefit from Proof-of-Location protocols. These include, but are not limited to, customer reward systems for physical stores, location-authenticated business review systems, location-restricted web content delivery, voter's physical presence verification, among many others. A more thorough depiction of such application scenarios was left in Section 2.1.3.

Further protocols took inspiration from this groundwork and started shaping the landscape. Graham and Gray, in [29], propose a Proof-of-Location scheme called SLVPGP that removes the need for the location manager to be trusted by the central verifier, delegating the trust to tamper-resistant modules. VeriPlace, by Luo and Hengartner [1], is a

complex and expensive privacy-aware location proof architecture that distributes responsibility among three types of trusted entities, taking the first step at avoiding dedicated tamper-resistant hardware. It specifically targeted the integration with Yelp[5], a public crowd-sourced reviews system for businesses. Another worth mentioning piece of work is from Javali et al. [30], still in a centralized and trusted stand, that adds robustness to the previous protocols by simplifying, in theoretical and practical terms, with trusted and existing Wi-Fi infrastructure, the Proof-of-Location generation process. Finally, the work of Akand et al. [31] is a more recent solidification attempt in the design of centralized but provably secure Proof-of-Location systems that protect against geo-tampering attacks.

The next section will report the emergence of the first relatively distributed Proof-of-Location protocols, taking a step further in the direction of fully decentralized and trustless systems.

3.2 Progressively Distributed and Decentralized Protocols

VeriPlace had already profiled and templated an inherently distributed architecture with built-in privacy awareness, taking a first infrastructural step towards defending against proxy attacks, without the need for trusted hardware [1]. The whole setup is especially tangled and consequently resourceful for the levels of trust it assumes, but it definitely settled the ground for the next generation of Proof-of-Location schemes.

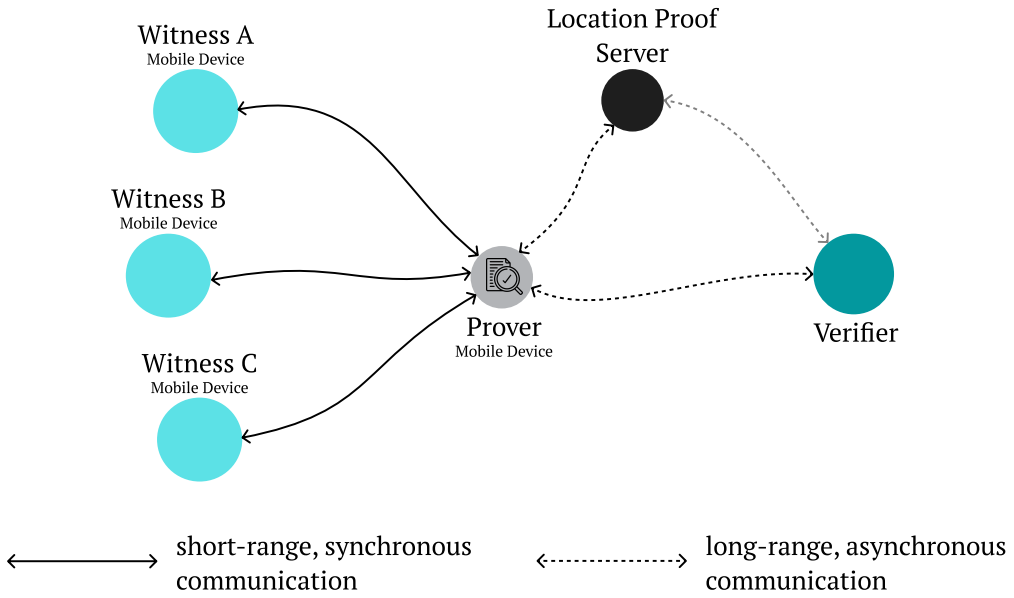


Figure 7. The main arrangements of the APPLAUS protocol, by Zhu and Cao [32].

[5] https://www.yelp.com

The following evolutionary stage of these protocols aims at flexing and distributing trust, resources, power, and responsibility, with the hope of achieving more resilient, fault-tolerant, and scalable systems. APPLAUS, by Zhu and Cao [32], delivers one of the first distributed protocols that combines the location proof and location privacy problems. It uses Bluetooth enabled mobile devices that communicate with nearby participants, during the proof generation process. The protocol asserts certain bond levels between the *prover*, *verifier*, and *witnesses*, all of them known to a trusted Certificate Authority (CA), disregarding, on the other hand, the need for a fully trusted location proof server to store the historic location records. In Figure 7, the essential configuration of the protocol is displayed, envisioning the prover to communicate individually, via Bluetooth, with nearby witnesses. Each witness should agree on providing a location proof, upon the prover's request, to be submitted later to an untrusted location proof server. This server will store the location proof historic records, to be queried by the verifier, in order to assert a prover's location within a specific time period. The prover, witnesses, and verifier are all assumed to be trusted by each other, leaving out the location server. The claim is that, by statistically changing the pseudonyms for each device and by following a user-centric privacy model, the protocol can effectively generate privacy preserving location proofs and store them in a trustless manner. STAMP [33] and PROPS [34] are two contemporaneous works that take the same witnessing approach as APPLAUS, but follow the path of convincing the verifier by presenting several shares of a composite location proof, based on group signatures. Both of them try to more profoundly tackle the prover's and witnesses' privacy concerns, but may admittedly fail at preventing collusion scenarios between them. Gambs et al. argue that the reliance on a trusted third party may be an unavoidable requirement, even if against the authors' principles of location sovereignty, especially when one wants to entirely prevent unbounded collusion attacks [34]. SPARSE, by Nosouhi et al. [35], avoids the typical distance-bounding mechanism and the witness picking process by the prover, as done in the previous works, with the goal of protecting against those collusion attempts, at best, in relatively crowded and decentralized witnessing situations.

At this point, all these schemes have assumed the common goal of protecting the identity of the parties involved in the proof generation process, but they have not yet tackled the additional problem of keeping the location information proportionally private from whoever needs to verify it. Dupin et al. [5] theoretically propose a Secure Multi-Party Computation (SMPC) based protocol that is provably resilient against any semi-honest participant. Their solution could still benefit from the classical distance-bounding mechanisms [5], but it is highly resourceful and practically infeasible, since it relies on expensive and complex cryptographic primitives and assumes directional antennas [36].

On the horizon of these solutions was still the high level need for detaching Proof-of-Location protocols from any kind of trusted central authority, for both identity and information management. This goal has naturally met, along the way, Blockchain

technology. The next section will finally present the most recent developments in achieving decentralized, trustless, and infrastructure-independent Proof-of-Location schemes.

3.3 Fully Trustless Environments

Inspired by the solution proposed by Zhu and Cao [32], Amoretti et al. [2] dive into the definition of a novel decentralized and infrastructure-independent approach that allies together short-ranged communication technology and Blockchain-based storage and information verification. The authors propose the establishment of a distributed overlay network of linked nodes that, at the same time, wirelessly provide or request location proofs from nearby nodes, and verify or store propagated proofs, via any typical lower-level blockchain protocolar agreement, achieving, thus, permissionless consensus. Their solution is claimed to be one of the very first at protecting against the main location-based-systems' attacks, with the help of a fully decentralized and blockchain inspired peer-to-peer scheme, assuring both integrity and user privacy. Real-world performance evaluation and the possibility for integrating higher-level incentive mechanisms were set as future work prospects. Both Amoretti et al. [2] and Nasrulin et al. [3] contemporaneous works illustrate practical constructs that take advantage of the tamper and censorship resistant nature of blockchain technology. The latter tries as well to formalize the main security and spatio-temporal requirements that such a decentralized Proof-of-Location protocol shall present, as seen in Section 2.1, ending up implementing a proof-of-concept, based on a permissioned blockchain framework, to specifically solve the challenges related to supply chain tracking.

Further efforts that build upon the above-mentioned solutions are the ones proposed by Wu et al. [37] and Nosouhi et al. [38]. The first follows the path of Amoretti et al. [2] and tries to enable, on top of it, user-defined hierarchical privacy protection, with the help of Zero-Knowledge proofs. The proposed protocol finds a bridge between the typical Proof-of-Location set of entities and the usual Zero-Knowledge proof participants. The suggested *zk-PoL* protocol aims at allowing the prover to convince the verifier that one was at a specific location, at a certain point in time, but with a granular privacy preserving disclosure of the location proof details. The obvious motivation of the mechanism is to solve spam, traceability, and privacy concerns of publicly storing raw location information, especially within decentralized and public ledgers. Therefore, the scheme is, to a great degree, centred in the privacy assurances and not in the infrastructural aspects of the potential decentralization that it is built upon. Nevertheless, it sets a promising starting point for the introduction of privacy preserving technology in the realms of trustless Proof-of-Location protocols. Optimizations and faster proof mechanisms are kept in the outlook and waiting to be explored. Nosouhi et al. [38] stress out a different proximity checking mechanism, to protect against the still unsolved prover and witnesses collusions, while committing, as well, to privacy preserving location proof generation

and storage, using public and decentralized blockchain technology. Their work has also an original integration of an incentive mechanism that rewards collaborative participants, in order to more strongly prevent the main known attacks. This sets an unprecedented track for the incorporation of these Proof-of-Location protocols into the digital and decentralized economy that already runs, via Smart Contracts, on blockchain networks like Ethereum [19, 38].

Minding all the above, Pournaras [7] proposes the complementing concept of Proof-of-Witness-Presence as a key element in an augmented democracy approach to smart city development. This concept involves validating the accuracy of data collected through participatory crowd-sensing, by requiring physical presence at locations of interest. The author argues that this approach can foster greater citizen engagement and participation in public spaces, and can be incentivized through blockchain consensus and a crypto-economic design. Acknowledging the limitations of current localization methods, such as GPS, it is suggested the need for more advanced and secure location certificates, based on complex social proofs. The Proof-of-Witness-Presence model envisioned by Pournaras may rely on token curated registries and a supplemental fully trustless Proof-of-Location protocol that, for instance, FOAM[6] tries to deliver. The next paragraph will be fully dedicated to this last piece of work.

The theorization of the singularity of a four-dimensional manifold, combining the three dimensions of space with the asymmetric arrow of time, has fundamentally shaped humankind's understanding of physical reality. The establishment of an absolute quorum over space and time is the ultimate goal that has driven forward the development of modern globalization, by the way we coordinate and synchronize our existence. Since the institution of the canonical time, sailing through the acknowledgment of the Longitude problem, to finally setting up intercontinental time synchronizers, the absence, or maintenance impossibilities of a truly global and absolute clock is still a major drawback in the production of correct, tamper-resistant, and spatio-temporally sound location information. Asserting the fundamentality of time synchronization, FOAM leverages Einstein's relativity hypothesis to create a new means for measuring space and time, for cartography and map making [39]. Their protocol, combined with their attempt at standardizing location data, is a totally new conceptual way of achieving decentralized, privacy preserving, highly accurate, censorship resistant, verifiable, and secure Proof-of-Location. *Zone Anchors* and *Zone Authorities* form a dynamic and decentralized network of radio beacons and gateways that reach consensus over the precise time of their clocks, establishing zone-relative clock synchronization. This allows for the formation of time conscious zones of witnesses that can simultaneously determine spacial arrangements and provide presence claims. Figure 8 depicts the expectation that Zone Anchors or Zone Authorities dynamically synchronize their internal clocks, establishing a smart contract's enforced physical coverage zone that offers trustless, but spatio-temporally sound

[6] `https://foam.space/`

location services. Zones may provide precise, verifiable, and secure Proof-of-Location claims, with the precision determined by possible triangulation mechanisms. Combined with token curated registries and crypto-economic incentives, for the maintenance and growth of the decentralized infrastructure, FOAM ultimately aims at creating a global consensus-driven map of the world. Their hopes are on Low Power Wide Area Network (LPWAN) radio technology, for the communication means, and on Ethereum-based Smart Contracts, for decentralized verification, consumption, and incentivization of the protocol operations over the location data [40].

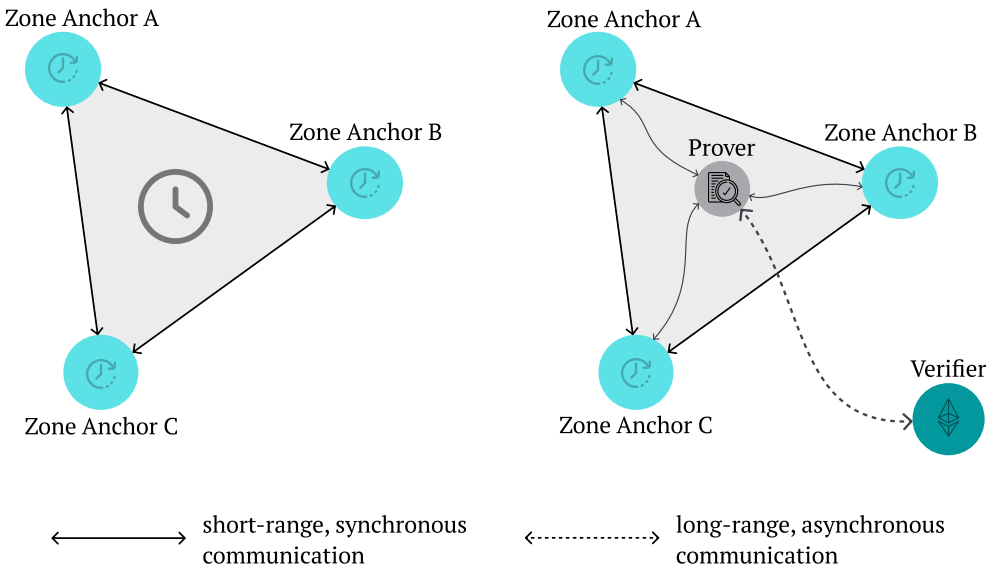


Figure 8. The FOAM protocol for dynamic and decentralized Proof-of-Location [40].

The FOAM protocol is the ultimate inspiration for the work developed further in this thesis. The processes of zone establishment, spatio-temporal synchronization, and decentralized witnessing consensus will be explored as in FOAM, but taking advantage of WiFi-based mesh networking for the short-range exchange of information, as introduced in Sections 2.2 and 2.3.

4 Protocol Fundamentals

This chapter outlines the key requirements and long-term goals of the proposed Proof-of-Location protocol. It also provides an overview of the thesis' objectives, while aiming at contextualizing the subsequent technical work.

4.1 Overview

The general approach to the design of Proof-of-Location protocols has been mainly focused on the proof generation process, as seen in the multiple examples dissected in Chapter 3. Advancements made towards more distributed and decentralized solutions have highlighted the need for a comprehensive and detailed description of the protocol's entire range of requirements. To achieve an operable system that meets the demands of real-world applications, a phased strategy with a keen awareness of the intrinsic details, at every stage of the solution, is essential for providing a complete and coherent picture of the protocol's design. Therefore, the design of a Proof-of-Location protocol starts with an infrastructural foundation, and ends with a complete system — aiming to achieve the goal of proving one's absolute location.

The following sections will guide the reader through the multiple steps of the protocol's design. This journey, depicted in Figure 9, starts with the understructure of the system, powered by a dynamic and non-hierarchic Mesh Network topology. This topology should enable the network agents to communicate with each other in a peer-to-peer, short-ranged, and conveniently wireless fashion. The next step entails the nodes' neighbourhood establishment, eased by lower-layer routing protocols, leading to the eventual creation of fully connected zones of neighbours. Each node, however, may simultaneously belong to multiple zones, with the processes of zone affinity, zone switching, zone expanding, and, consequently, the overall configuration of the mesh topology being dictated by protocolar arrangements, or even, application level incentives. The theoretical aim is at achieving a latticework of space and time, with zone-relative clock precision. Therefore, the next step is to establish, or derive, spatio-temporal zone synchronization. Space synchrony is achieved with the assumptions regarding the short-ranged communication means. Time synchrony requires a clock synchronization mechanism, which may simultaneously allow for zone-relative event serialization, via a Turing-complete strongly consistent consensus-based system. Nonetheless, the main aim is to achieve zone-relative time consciousness, to finally enable spatio-temporal soundness [3].

In the following sections, we provide a more detailed analysis of these multiple steps, but will point only, in practical terms, to a subset of the entire problem. The steps that precede the zone discovery and zone affinity procedures, as well as the ones that succeed the goal of achieving relative Proof-of-Location are either abstracted, explicitly assumed, or left for future work.

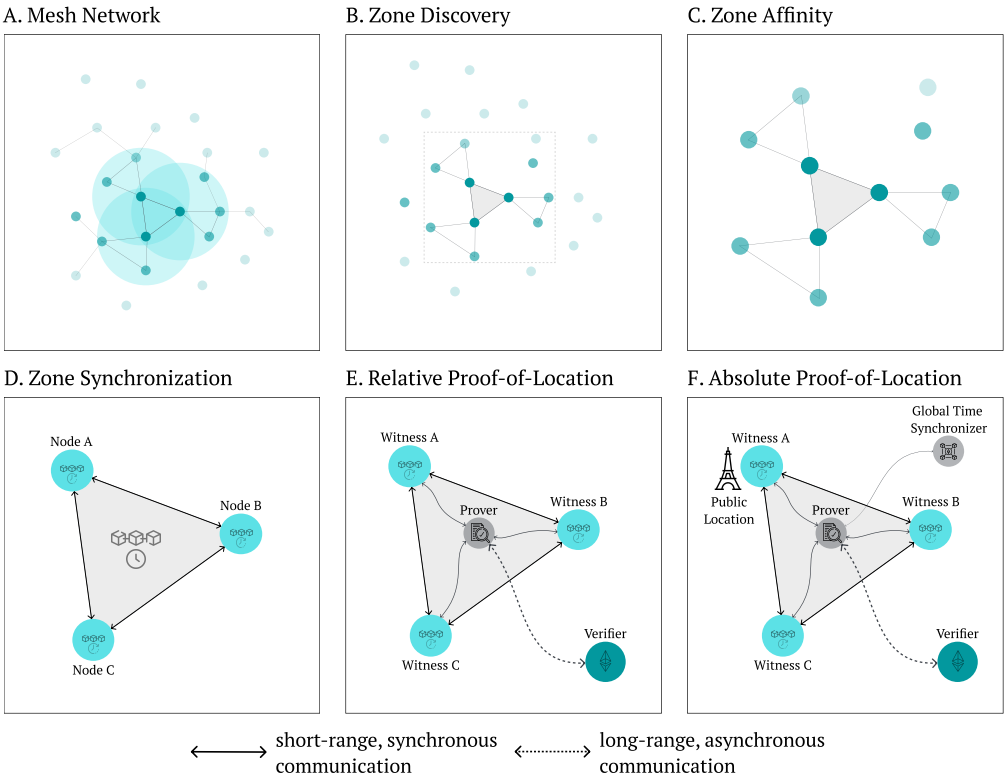


Figure 9. A discretization attempt to capture the multiple steps of the protocol design, from a dynamic mesh topology, towards the ultimate goal of achieving Absolute Proof-of-Location.

4.2 Dynamic and Non-Hierarchic Mesh Networks

Dynamic and non-hierarchic mesh networks are a type of network architecture that allows for the creation of ad hoc networks in which nodes can communicate with each other without the need for a central coordinating device. This type of network is characterized by its ability to self-organize and dynamically adapt to both changes in the environment and in the overall topology. As presented in Section 2.2, one of the key features that materializes the concept of dynamic and non-hierarchic mesh networks is the existence and implementation of lower-layer routing protocols to facilitate the peer-to-peer communication between the nodes.

Mesh networks, as expected, rely first on the physical layer, according to the standards of computer networking, which is responsible for transmitting raw bit stream data over the physical medium, copper wire, optical fibre, or wireless frequencies, for example. This layer defines the physical characteristics of the data transmission, such as voltage levels, data rates, and the physical connectors and media used for communication. Its

main function is to provide a reliable and efficient transmission of bits between devices, without any regard made to the higher-layer protocols and their associated data. The physical layer is responsible for encoding and decoding data into a format that can be transmitted over the network, while also detecting and correcting errors that occur during transmission [41]. It also defines the substrate physical topology of the network, which describes the arrangement of the physical components such as devices, cables, and other network equipment[7], as depicted in Figure 10. To then enable efficient and coordinated communication, there is a need for routing protocols that determine the best path for data packets to travel through.

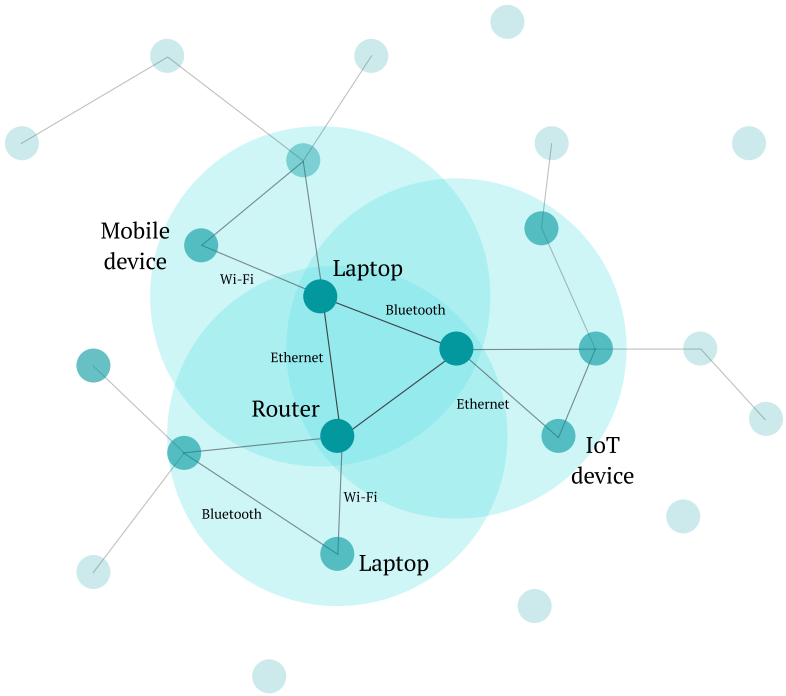


Figure 10. The heterogeneity of a mesh network and the substrate diversity of its physical topology — the devices, their connections, and their physical arrangements.

These routing protocols are expected to operate, not at the network layer, but instead at the data link layer, which is responsible for handling the transmission of data frames over the physical layer. Their main goal is to determine the best path for data frames to travel, based on metrics gathered from lower-level physical information, as, for instance, signal strength and link stability metrics [42]. These protocols can support neighbourhood discovery and the ranking of neighbours [43], and thus, potentially enable the processes of

[7]The description of the physical layer is a combined effort of the author and ChatGPT (11.04.2023), which helped to highlight and summarize the role of this layer, during data transmission.

zone discovery and zone affinity management, as illustrated in Figure 11. Neighbourhood discovery is the process of physically discovering neighbouring nodes within the mesh network [14]. Neighbour ranking is the process of determining the quality and reliability of each neighbouring link [13]. By measuring, understanding, and ranking the quality and reliability of the data links, and by instructing nodes to independently calculate their best next-hops, a routing protocol can establish neighbourhoods and determine the affinity of nodes within these coverage localities. This information can be primarily used to optimize communication paths, reduce congestion, and increase the overall efficiency of the network.

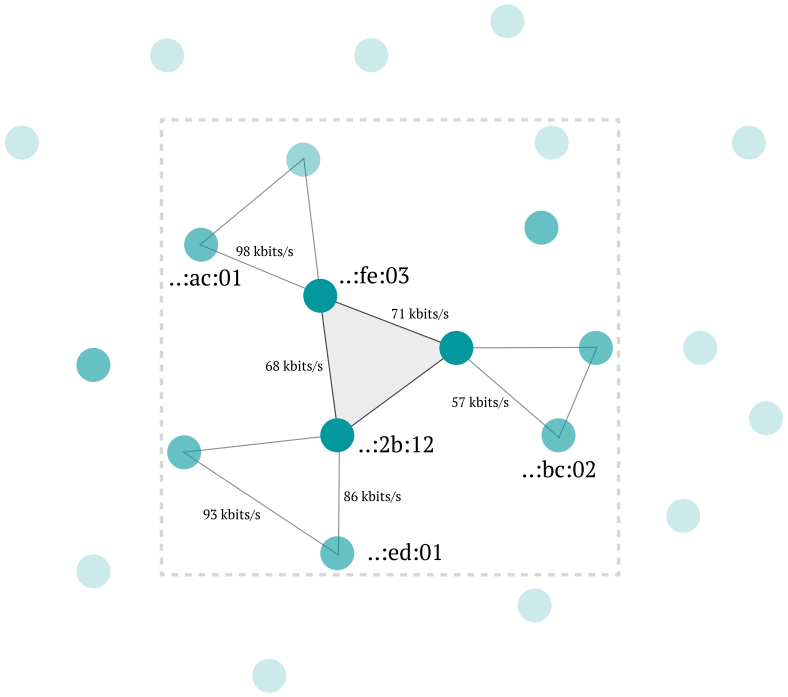


Figure 11. The routing protocol's neighbourhood discovery and ranking processes, using the MAC sublayer and link quality metrics.

The second usage of such accomplishment is to finally enable the targeted discovery of zones within the mesh network. Zones can be viewed as strongly connected sets of neighbours, that, in consequence, are one-hop away from each other. The process of zone establishment is facilitated by the routing protocol, but not strictly enforced, since the protocol solely enables the discovery of potential neighbours and the ranking of their links. The final decision of which nodes are to be grouped together into a zone is left to the nodes themselves, which can then use this information to establish their own zone affinity. Zone affinity is the process of determining the likelihood of nodes communicating and establishing zones with other nodes. The motivation may also be

extrinsic to the process of zone discovery or establishment. An identity management protocol, for instance, can be used to determine the zone affinity of nodes, based on their identity and their individual wishes to communicate with other nodes. Incentives of higher degree can be used to motivate nodes to communicate, to establish zones with other, relatively specific, sets of nodes, and hence, to collaborate in the next step of providing zone-relative location services. In the second scenario provided in Figure 2, customers may be incentivized to form a zone, in order to attest the ISP's connectivity provision in their neighbourhood, with the identities of all the involved parties being known to the government regulators. The customers, who happen to be neighbours of each other, have the common wish of being provided connectivity, or seeing the ISP's service level agreements enforced in their neighbourhood. Hence, they proceed to coordinate their efforts and synchronously connect their devices, forming a zone and providing Proof-of-Location services to the ISP. The ISP, in turn, is incentivized to meet the demand in this particular neighbourhood and, at the same time, prove to the regulators and all the other parties that the agreements are being met. The FOAM protocol, on this matter, envisions the reliance on token-curated registries to provide decentralized identity management. It may also rely on crypto-economic incentives to motivate nodes to collaborate and, together, establish and maintain coverage zones, for the higher purpose of providing Proof-of-Location capabilities [40]. This thesis' main focus, with regard to the proof-of-concept implementation, is abstracted from the whole process of zone discovery and zone affinity management, and assumes that a zone has been already agreed to be established by some out-of-band process. Nonetheless, these aspects are still to be targeted for future work, as they are essential for the overall success of the protocol.

After the establishment of operational zones and the affinity filtering potentially happening at the data link layer, the typical Internet Protocol or TCP/IP suite can be used to enable end-to-end data communication for application-specific purposes. The TCP/IP suite consists of a set of protocols that operate at the network and transport layers, providing end-to-end communication services for applications running on different nodes [41]. At the network layer, the Internet Protocol (IP) is used to route data packets within and between the zones, based on their IP addresses. IP is a connectionless protocol that operates independently of the underlying physical and data link layers, allowing it to be used with a variety of network technologies. At the transport layer, protocols such as TCP and UDP can be used to enable end-to-end data communication between application instances. TCP is a reliable and connection-oriented protocol that provides features such as flow control, error detection, and congestion avoidance to ensure that data is transmitted reliably and efficiently between applications. UDP, on the other hand, is a connectionless protocol that provides a lightweight alternative to TCP, suitable for applications that require low-latency communication, or do not require reliability guarantees at the transport layer [41]. The choice between the two may be based on

the application requirements, the network topology, and the available resources, but the overall conclusion is that, after enabling network layer capabilities, any typical Internet service can be provided to the end-users, sustained by the underlying mesh network[8]. Additionally, as pictured in Figure 12, by subnetting and assigning a unique range of IP addresses to each zone, nodes can communicate with each other, in the same zone, and with nodes in other zones using IP-based protocols. Subnetting can also provide a range of benefits, such as improved security, better network management, and more efficient address assignment and usage [41].

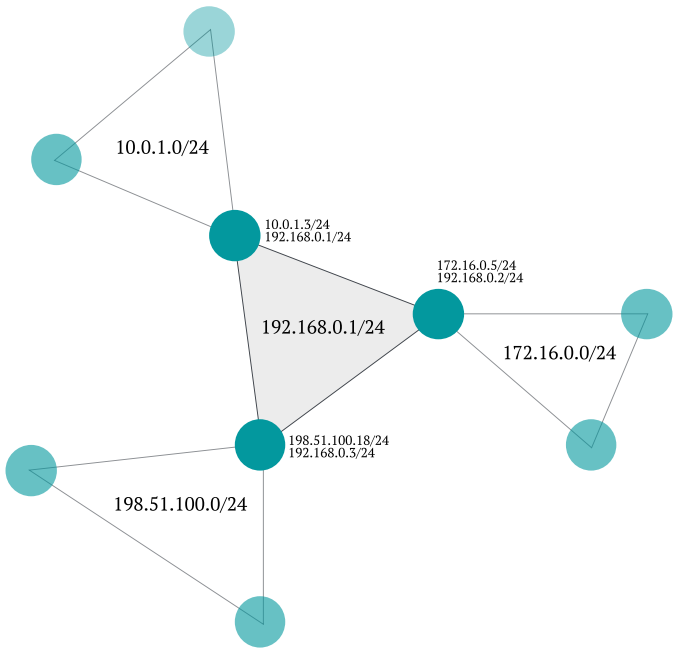


Figure 12. Subnetting and IP-based communication between nodes, within and between zones, after the establishment of their respective zone affinities.

The following section presents the next step towards guaranteeing the infrastructural basis of the proposed Proof-of-Location protocol, detailing not only the need for zone-relative clock synchronization, but also the steps taken to eventually achieve spatio-temporal soundness. It ultimately assumes that the physical, data link, network, and transport layer capabilities have been already established, and that the nodes are able to communicate with each other using the TCP/IP suite and related protocols, on top of the underlying mesh network infrastructure. The verification or enforcement of the usage of short range communication means is still to be researched, as a key aspect to the protocol's foundational security, and soundness guarantees.

[8]The TCP/IP stack description is a combined effort of the author and ChatGPT (13.04.2023). The tool helped to fact-check and clarify the characteristics of the transport layer protocols.

4.3 Turing-Complete Clock Synchronization

Section 3.3 has already outlined the invariable need for clock synchronization, when it comes to reach progressively more accurate, correct, sound, and tamper-proof means of location attestation. What has not been discussed yet is the bridge between the clock synchronization problem and the consensus problem, in the specific context of distributed systems.

This section will briefly discuss how these two fundamental problems are, in fact, intertwined. In simple terms, synchronizing clocks is nothing more profound than reaching agreement about the current time of the internal clocks of a distributed setting of machines. The indiscriminate case of clock synchronization can be stretched to a continued act of counting time at the same pace, as the indiscriminate case of reaching agreement about the current time of the internal clocks can be extended to a continued act of reaching agreement about the current state of the system. The latter is the consensus problem, and the former is a special case of it, the case of time synchronization. As argued in Section 2.3, craving to achieve time synchronization, in not just a distributed setting, but in a fully trustless environment, can be transposed to the problem of achieving permissionless consensus, fulfilling the need for ordering and synchronizing events, at the same pace, in an environment where individual participants are not necessarily trusted. The first example depicted in Figure 2 makes it clear that the eyewitnesses can only attest to the reporter's presence at the accident site if they had been there at the same time and witnessed the same event. However, these entities are not necessarily known to each other and to anyone else around, nor are they necessarily trusted. This implies not trusting that the clocks of the witnesses are correctly adjusted, as well. Therefore, the witnesses need to reach agreement about the current time, in order to be able to collectively and correctly attest to the reporter's presence at the accident site, in a given moment, otherwise they would all have conflicting or different reports. Additional to that, they may also attest to the reporter's course of action, by the means of taking pictures or recording videos, which are also events that may be ordered and synchronized, achieving, thus, permissionless consensus.

In consequence and specifically regarding the Proof-of-Location problem, the need for clock synchronization is not only a matter of achieving more accurate positioning measures, as typically done in GPS-related trilateration systems, but a fundamental matter of achieving complete and sound consensus between the witnesses that are set to attest to one's location. Fundamentally, as defined in Section 2.1, a Proof-of-Location is complete if the attestation is done at location l and time t, by a set of witnesses $w \in W$. To cope, as well, with the proposed property of spatio-temporal soundness, these witnesses should invariably reach consensus. They should not just agree to exist at around the same location l, relative to the precision, reliability, and coverage of their physical communication means, but should also agree on the time t, relative to the precision of their internal clocks. The latter is exactly the clock synchronization problem,

folded into the greater and abstract need for consensus. The FOAM protocol proposes the achievement of such a time agreement with the employment of a self-stabilizing hybrid fault-tolerant clock synchronization protocol [40, 44]. The authors justify the choice highlighting the formal verification methods employed in Malekpour's work. This distributed algorithm is expected to formally work to the extent of the presence of a one third minority of Byzantine nodes [16]. This fact validates the theoretical need for a composition of at least four witnesses for the most basic configuration of a zone, in the FOAM protocol.

This thesis proposes, instead, the empirical experimentation with a permissionless consensus mechanism, in the lines of the ones introduced in Section 2.3. The goal, as illustrated in Figure 13, is equivalent to establishing zone-relative time consciousness, but with the added benefit of providing strongly consistent serialization of transactions and total order of multidimensional events, instead of simply counting time in a unidimensional manner. The technicalities of modern systems that tackle the permissionless consensus problem allow, as well, for stronger guarantees in terms of security, tamper-proof and censorship resistance, as discussed in Section 3.3. The fault-tolerance guarantees of a typical deterministic-finality Byzantine fault-tolerant algorithm are shifted back to a probabilistic finality fault-tolerance threshold, where the probability of a transaction being finalized is directly proportional to the number of nodes that have approved it [23]. This is, arguably, a more desirable property for the considerations of a decentralized Proof-of-Location

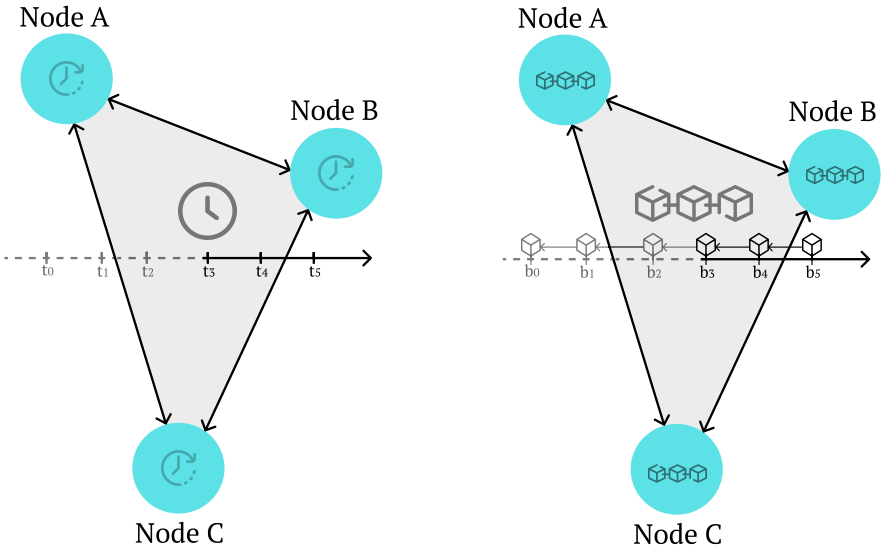


Figure 13. The goal of establishing zone-relative time consciousness, on the left, by the employment of a clock synchronization protocol, and on the right, by the employment of a consensus mechanism. Relative to the precision of both frequencies, the two mechanisms should achieve the same result, with the latter allowing for a strongly consistent serialization of transactions and total order of multidimensional information.

protocol, where the finality of a location certificate is directly proportional to the number of witnesses that have seen the prover, and thus, the number of witnesses that have agreed on the time of the attestation, in the context of trustless environments. The methodical work of formally verifying, measuring, and comparing the two approaches, in the specific context of Proof-of-Location protocols, is also left for future work. This thesis will plainly focus on the experimentation with a consensus mechanism that not only allows for the logical synchronization of clocks, via the regulation of the block interval, an inherent property of such mechanisms, but also for the achievement of a distributed and time-conscious Turing Complete environment, where the execution of more sophisticated logic is structurally enabled, in a decentralized setting. Another point worth mentioning is the extensibility of the proposed way of achieving clock synchronization, which should also be researched further, to assess the need and evaluate the possibility for independently calculating the geometry of the zone, and the practicability of trilateration mechanisms to more accurately determine the prover's exact location [40]. Nevertheless, the theorized approach aligns itself with the spacetime model of a multidimensional manifold and the special relativity idea of a dense latticework of clocks, as depicted in Figure 14. The end goal for the coverage expansion of such a Proof-of-Location protocol is to achieve a global latticework of witnessing zones that provide location services. This resonates with the faraway goal of achieving a secure, verifiable, decentralized, global and consensus-driven map of the world [39, 40].

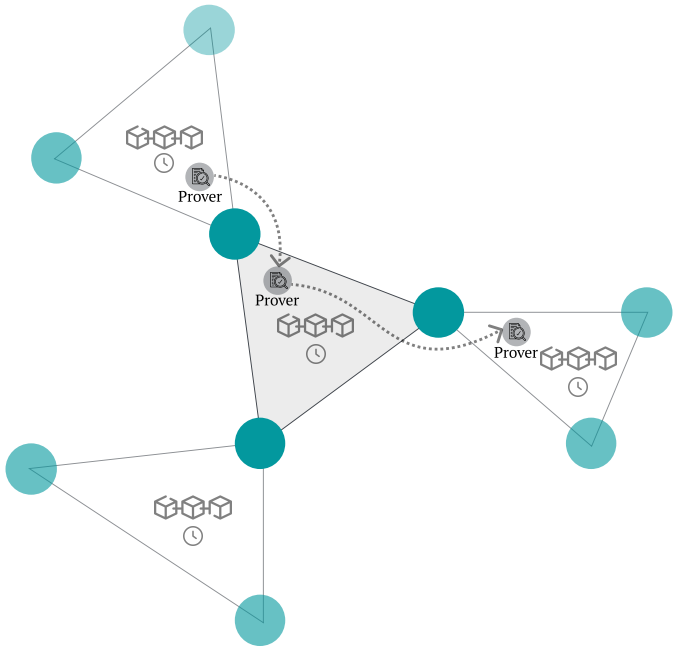


Figure 14. The dense latticework of time-conscious witnessing zones, providing location services. Such a structure would allow for more accurate and verifiable location attestations, for example, regarding moving provers that constantly change their locations.

The potential Turing Completeness property of the envisioned consensus system fundamentally allows for a network of nodes to perform arbitrary computations in a decentralized and fault-tolerant manner. At its core and in more tangible terms, such a system allows for the creation of smart contracts, as self-executing code agreements that are dictated by the terms of the direct consensus between the entities that are involved in the Proof-of-Location protocol. By using a distributed Turing-Complete system, like Ethereum [19], these smart contracts can be deployed across the network, and their execution can be verified by all entities, ensuring that the terms of the contract are met. This allows for the simple transactional case of registering a valid Proof-of-Location claim by the prover, which will be covered in this thesis, and also for progressively more complex location claims. Pictured in Figure 15, a Turing Complete system could, for example, enable a multi-prover configuration, or enforce a set of arbitrary terms dictated by the verifier, the witnesses, or by zone-specific requirements, among all other endless computable possibilities that one may envision.

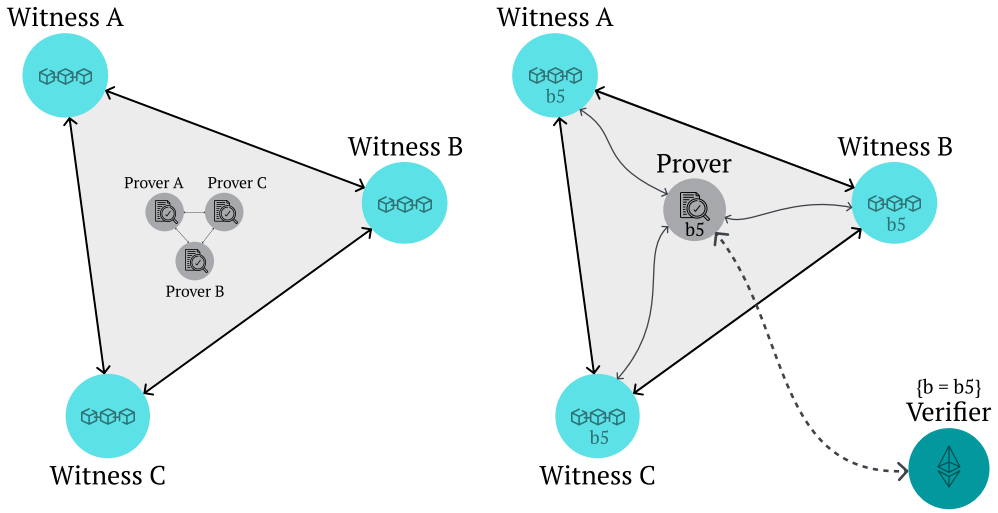


Figure 15. Some more sophisticated use cases of a Turing Complete Proof-of-Location protocol, where the execution of zone-relative smart contracts is enabled. On the left, a multi-prover configuration, where the witnesses are required to attest to the simultaneous existence of a group of provers, and on the right, the case where a verifier enforces the attestation of the prover's location at a specific block, or time.

With all the above considerations in mind, this thesis is set to demonstrate the simplest case of the application of a consensus mechanism to achieve time synchronization. The goal is to experimentally prove that the block interval of a consensus mechanism can be used to synchronize the clocks of a network of nodes, and thus, achieve zone-relative time consciousness. All the other considerations, like the extensibility of the proposed

approach, the possibility of achieving a more accurate location attestation, or making use of the Turing Completeness of the system for more complex logic, are left for future work. The next section will cover the proof generation process to finally produce a Proof-of-Location claim.

4.4 Relative Proof-of-Location

With the witnesses agreeing on a location, short-range communication, and internal clock synchronization, the infrastructure is ready to generate a verifiable Proof-of-Location certificate. This section will provide a description of the certificate generation process, its requirements, and a possible verification procedure that is derived from the protocol's design.

Building upon the steps achieved in the previous sections, it is assumed that a zone has been established, and that the zone members are able to achieve time-conscious consensus. Concretely, we assume that both the prover and the witnesses restrict their communications to the zone's physical boundaries, dictated by the coverage of their short-ranged communication means, and we also assume that the zone members are able to achieve zone-relative time synchronization, and so, pace the events of the protocol's execution at the same rate. The next step is to establish the proof generation process, which is based on the zone's relative space and time. The main goal is to assert that the prover is at a specific location, at a specific time, relative to the witnesses. This means, in practical terms, that the prover is able to communicate, via the same short-ranged communication means, with all the witnesses, and is able to demonstrate that it is, as well, synchronized with their internal clocks. However, before asserting such propositions, we also assume that every entity i has a unique key pair, composed of a public key K_i^{pu} and a private key K_i^{pr}. The public key represents the identity of an entity, and is used to verify the signature of the messages produced. The private key, on the other hand, is used to sign the messages. The public key of a zone member is known to all the other members, meanwhile, the private keys shall be kept secret.

Given that the witnesses $w \in W$ are pacing the zone events, i.e., generating new blocks, in average, at every T units of time, and the last block was generated at the zone-relative time t_x, with a block hash h_x, the proof generation process is as follows:

1. The prover ρ synchronizes itself at the zone-relative time t_x and learns the hash h_x of the last block B_x.
2. The prover ρ assembles a transaction Tx_ρ containing the input h_x.
3. The prover ρ signs the transaction Tx_ρ with its private key K_ρ^{pr}.
4. The prover ρ broadcasts the transaction Tx_ρ to the witnesses $w \in W$.

5. The witnesses $w \in W$ verify the transaction Tx_ρ.
6. The witnesses $w \in W$ assemble a block B_{x+1} with hash h_{x+1}, at time t_{x+1}, containing the transaction Tx_ρ.
7. The witnesses $w \in W$ sign the new block B_{x+1} with their private key K_w^{pr}.
8. The witnesses $w \in W$ broadcast the block B_{x+1} to the entire network.
9. The prover ρ verifies the hash h_{x+1}, the parent hash h_x, and the signatures of the block B_{x+1}, and the inclusion of its transaction Tx_ρ, with the matching input h_x.
10. The prover ρ, finally, assembles the Proof-of-Location certificate, containing the signed block B_{x+1}, by the witnesses $w \in W$, and the signed transaction Tx_ρ, by the prover ρ itself.

Having obtained a valid Proof-of-Location certificate, as the prover ρ was spatially synchronized with the witnesses $w \in W$ around the zone-relative time t_{x+1}, the next step would be to submit the certificate to a verification process. Without any verifier-specific information, the verification process would be identical to step 9 of the above procedure. Assuming that the verifier knows the identities of all the entities involved in the proof generation process, the verification process would go around verifying the signatures of both the block B_{x+1} and the transaction Tx_ρ, by the witnesses and the prover, respectively, and verifying, as well, that the transaction input matches the parent hash h_x of the block. This last verification step ensures that the prover was spatially synchronized with the witnesses around the zone-relative time t_{x+1}. A more well-founded analysis of the robustness, security, privacy, and, eventually, the correctness of this Proof-of-Location protocol, inspired by the works presented in Chapter 3, is left for future work. This would, as well, include a more detailed analysis of the security against the most common attacks and collusion scenarios. Bridging with the examples presented in Figure 2, in the first use case, the prover is represented by the reporter, and the witnesses are the bystanders that are passing by the accident site. The goal of the reporter is to prove that he was physically present at the location of the accident, factually covering the event. The proof, possibly containing a piece of media content, is attested by the onlookers and verified by a journalist when producing a news piece. In the second case, the ISP assumes the prover role and tries to gather a proof of the service provision within the customers' neighbourhood. Ultimately, the proof can be verified by regulators that are responsible for the enforcement of service level agreements.

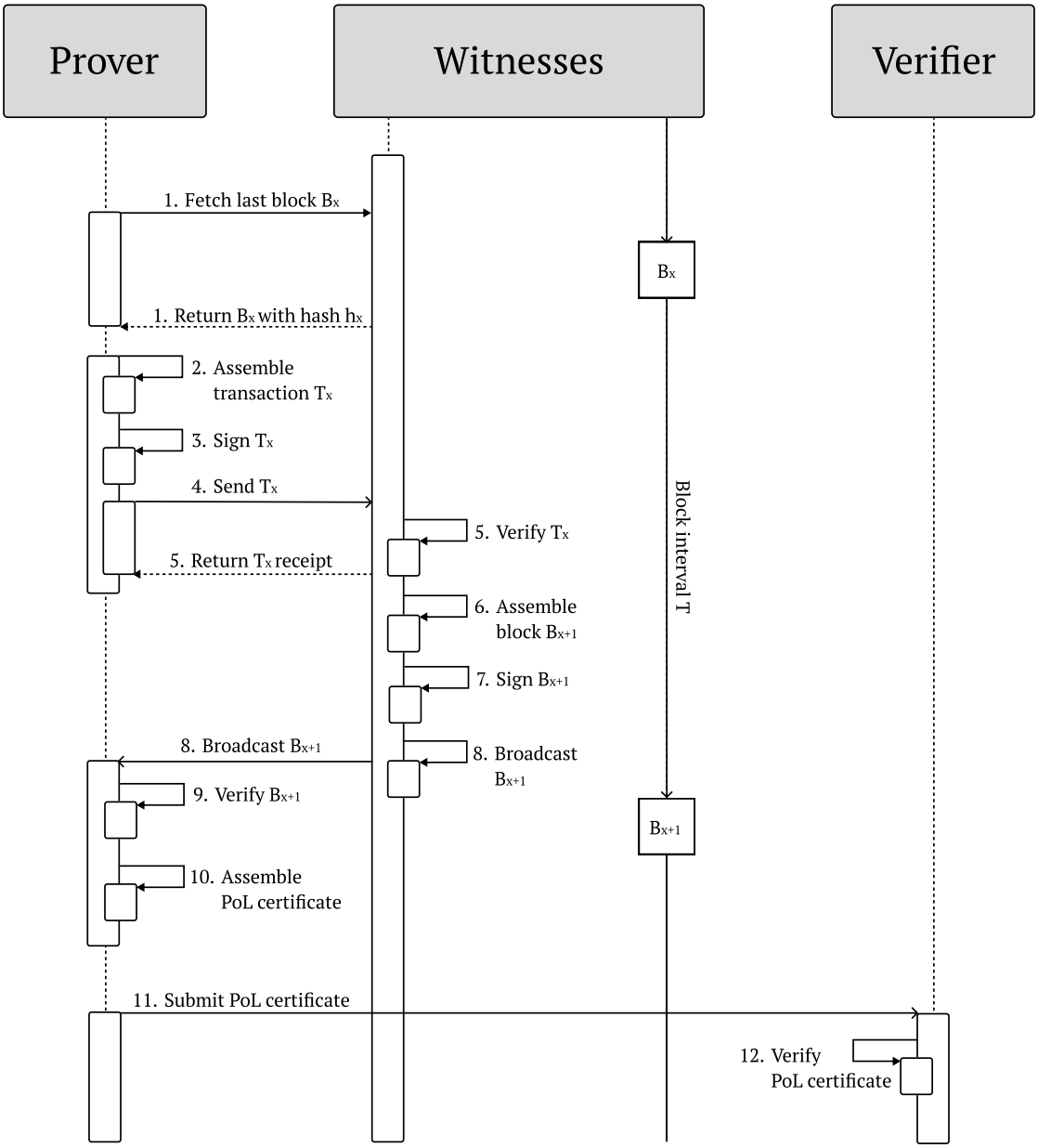


Figure 16. Sequence diagram overview of the proof generation process.

The next section will provide a rough outline of how this protocol can be extended to achieve Absolute Proof-of-Location, involving the introduction of verified space and time references.

4.5 Absolute Proof-of-Location

Having established a means for Relative Proof-of-Location, the next step would be to extend the protocol to finally achieve Absolute Proof-of-Location. A possible path consists in combining the procedure with a Global Time Synchronization Protocol and a Global Positioning System, as illustrated in Figure 17.

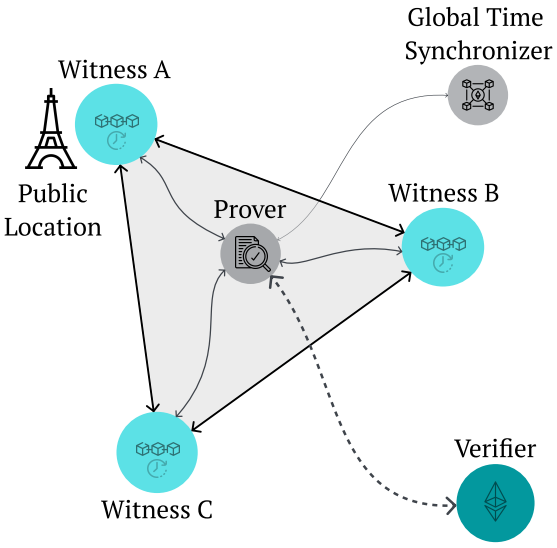


Figure 17. Achieving Absolute Proof-of-Location by combining Relative Proof-of-Location with a Global Time Synchronization Protocol and a Global Positioning System.

The goal is to produce a Proof-of-Location certificate that is spatio-temporally sound, relative to the zone, but which can, as well, be spatially and temporally acknowledged by any other node outside the zone. This effort may require, for instance, a global time-stamping server, or protocol that can assert a certain level of globally secure timestamp accuracy in a tamper-proof manner. In a decentralized fashion, one example of such a system is a public Blockchain network, such as Bitcoin [18], or Ethereum [19]. The same applies to the Global Positioning System (GPS). A standard system of geographic coordinates could be validated and embedded into the Proof-of-Location certificate, to globally assert the location of the whole zone [2, 38]. It is worth noting that the Relative Proof-of-Location protocol is flexible enough to accommodate any kind of higher level protocol, including more complex certificate formats, tighter trust levels, and composite information to be signed. This and further extensions are left for future work.

5 Proof-of-Concept

This chapter details the steps taken to develop and evaluate the proof-of-concept implementation. Section 5.1 provides an overview of the infrastructure, while Section 5.2 describes the implementation of the Proof-of-Location protocol, after the establishment of the baseline network architecture. Finally, Section 5.3 presents some measurements of the proof-of-concept evaluation. All the software developed, along with complementing documentation, is available at the repository referenced in Appendix I.

5.1 Infrastructure

Aiming at meeting the requirements established in Chapter 4, the system design process and infrastructure choices for accomplishing the establishment of a mesh network were crucial in ensuring its operability. The following sections describe the design of the system and the testbed setup, as well as the challenges faced during the establishment of the network infrastructure.

5.1.1 System Design

One of the critical decisions was the selection of an appropriate operating system (OS) for configuring and enabling an ad hoc and dynamic mesh network. Multiple Linux distributions were initially explored, including OpenWrt, Raspbian, and Ubuntu. After careful consideration and a handful of failed attempts, OpenWrt led the track to host the protocol implementation. Its automated build tools and configurable packages, including the straightforward B.A.T.M.A.N. kernel module integration, load, and setup, made it a suitable choice for the flexible customization of the mesh network. The OS's active development community and extensive documentation further supported its use. As mentioned in Section 2.2, OpenWrt is a lightweight solution for embedded devices, supporting a considerable set of platforms and toolchains for targeted integration of embedded software. Its choice turned out to be strategic regarding as well the role played in the spawning of decentralized and community-driven computer networks, that actively contribute to and are powered by such open source projects, all around the world[9].

The OpenWrt image building process took advantage of the official Image Builder pre-compiled environment in order to create a custom image, skipping the need for source compilation[10]. The process was automated with the help of a *Dockerfile*, not only to allow for an easy integration and further testing of the mesh network pre-compiled packages, but also to enable the experimentation with different network settings and the quick embedding and deployment of the Proof-of-Location protocol software. The Image Builder setup is located under the "proof-of-concept/openwrt-builder" directory of

[9] `https://en.wikipedia.org/w/index.php?title=Freifunk`

[10] `https://openwrt.org/docs/guide-user/additional-software/imagebuilder`

the thesis repository. The packages were chosen to provide deployment and debugging support for mesh networking using *batman-adv*, wireless security using WPA/WPA2, working with the *ext4* file system, and interacting with an Ethereum network via the official *geth* client. The images were experimentally compiled for *x86-64*, *bcm2708*, and *bcm2709* targets, being the last two ideal for the Raspberry Pi Zero and Raspberry Pi 2/3/4 models, respectively[11].

5.1.2 Testbed Setup

The next step was to set up a testing environment for the deployment and running of the Proof-of-Location protocol. Up to the delivery of this thesis, efforts are being made to port the generated OpenWrt image and compiled protocol modules to physical Raspberry Pis. However, the deployment on physical devices and the configuration of wireless interfaces is proving to be a challenge on multiple levels. The current drawbacks we are facing relate to the automated enabling of the batman-adv network interface, in order to allow for the autonomous start of a dynamic and ad hoc mesh network. Nevertheless, even when manually configuring the interfaces, the protocol process of dynamic neighbourhood discovery is also failing, as the wireless interfaces are not able to easily detect each other, or establish a stable connection. We have been further investigating these issues to identify the root cause of these problems, and to finally uncover the solution for the deployment on physical devices. These attempts and the physical demonstration of the Proof-of-Location protocol are set for future work.

Meanwhile, to ease such development and testing hustle and to allow for a more flexible and faster deployment of the protocol, we have set up a virtualized environment, using QEMU (see Section 2.2). The environment was configured to emulate a set of x86-64 target devices, and to run the OpenWrt image with the embedded and target-compiled Proof-of-Location software. The testbed setup followed the guidelines of the official batman-adv documentation[12], and the source code is available under the "proof-of-concept/qemu" directory. To reduce manual input, an automated script was programmed, expecting the previously built OpenWrt image, and defining variables for the instance type, either “witness” or “prover”, the instance number, and a GDB port, for kernel debugging. The script creates a shared bridge for the cluster and a unique tap interface for each virtual machine, assigning to them an individual MAC address. Finally, the script launches QEMU by creating a new virtual disk image, as a copy-on-write snapshot of OpenWrt, and by setting up each virtual machine with 1 GB of memory, 2 virtual CPUs, and a virtual SCSI disk. It assigns, as well, internal network interfaces to the instances. One uses the previously created tap interface to flush the mesh network traffic, and a second one is a virtual NIC to allow for the usual internet

[11] `https://firmware-selector.openwrt.org/`

[12] `https://www.open-mesh.org/doc/devtools/Emulation_Environment.html`

connection[13]. KVM is also enabled, in order to speed up the emulation, as well as a virtual RNG (Random Number Generator) and a virtual serial port. Up and running, with three witnesses and a prover instance, the testbed looks as depicted in Figure 18.

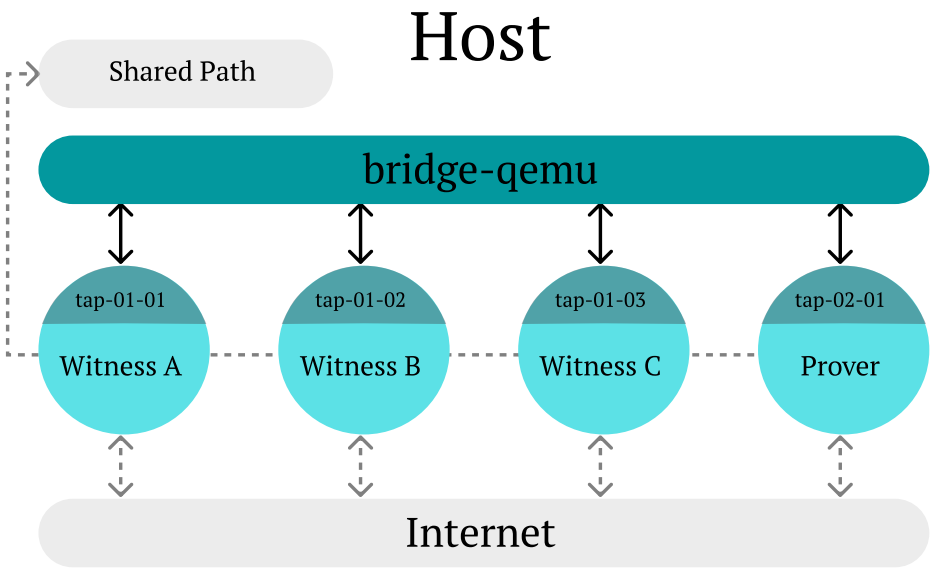


Figure 18. Testbed setup for the Proof-of-Location protocol.

5.1.3 Network Architecture

After the establishment of the testbed, the physical networking layer is emulated by the bridge interface, which is set to pool all the mesh traffic that flows into and from the tap interfaces. The step that follows is the configuration of the batman-adv kernel module, responsible for the dynamic and ad hoc mesh network creation. This step was automated at the OpenWrt image building process, with the inclusion of custom instructions in the image startup scripts. These instructions set the routing algorithm and enable the virtual Ethernet interface eth0, linking the tap interface created before to the batman-adv interface bat0. The interval time between the broadcasting of neighbourhood discovery messages is set to 5 seconds, which determines how often a node should broadcast its presence in the network. The bridge loop avoidance mechanism is activated, penalizing the routing of traffic through routes with more hops. This last step is important to avoid the creation of loops in the network and to ensure the traffic is always routed through the shortest path, forcing the nodes to communicate directly with each other, within the

[13] https://www.open-mesh.org/doc/devtools/OpenWrt_in_QEMU.html

testbed. Finally, the script disables the firewall to avoid any interference with batman-adv, and flushes the IP addresses of both the eth0 and bat0 interfaces.

Subnetting is then done via the assignment of an IP address to the bat0 interface. The IP address of each instance is generated based on the MAC address of eth0, establishing a non-conflicting address in the 192.168.0.x/24 subnet. After this step, the network is ready to use the TCP/IP stack, and the Proof-of-Location protocol can be deployed, configured, and run. The final network topology is depicted in Figure 19.

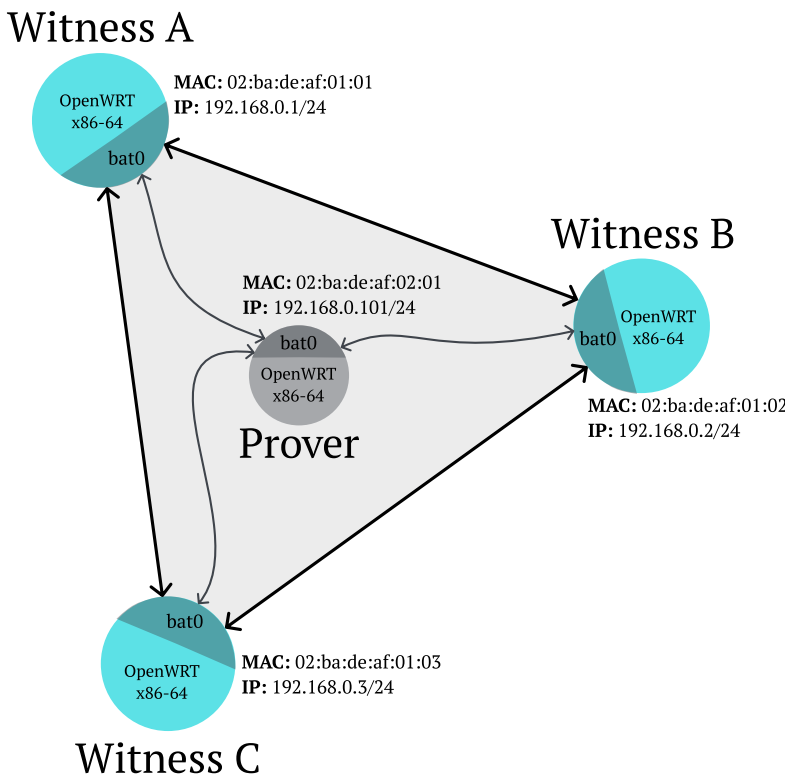


Figure 19. Network topology of the system under test, after configuring the batman-adv interface and assigning IP addresses to the instances.

5.2 Protocol Implementation

The steps above ensured the establishment of the physical, data link, network, and transport layers, drawing up the foundation for the application layer, featuring the actual implementation of the Proof-of-Location protocol. In this section, we provide a description of the protocol setup, including the reasoning and choice for the Blockchain framework to use. The processes of proof generation and verification are also detailed.

5.2.1 Practical Permissionless Consensus

The outline provided in Section 4.3 identified the need for a clock synchronization mechanism that would finally enable spatio-temporal soundness. Space synchrony is achieved with the assumptions around the short-ranged communication means. Time synchrony, on the other hand, is achieved with a clock synchronization mechanism. The hypothesis involved the employment of a permissionless consensus protocol to establish zone-relative time consciousness, but with the added benefit of providing strongly consistent serialization of transactions and total order of multidimensional events, instead of simply counting time in a unidimensional manner, via a plain clock synchronization protocol. The practicalities of employing a permissionless consensus mechanism involved the experimentation with a blockchain framework, the deployment of an interacting client, and the setup of the network.

The beginning of the exploration process included the prototyping of an ad hoc Proof-of-Work based consensus protocol, to assess the feasibility of the hypothesis. This work was part of the 2022 Fall Semester's edition of the Distributed Systems seminar, where an analytical approach was taken to survey multiple permissionless consensus mechanisms, pointing out the challenges of porting such protocols to resource constrained environments[14]. The results of the experiment were sufficiently encouraging to warrant the development or choice of a more robust and scalable implementation. Multiple open source projects were considered, and the ultimate decision relied on Ethereum and its flexible tooling for creating private networks[15]. Initially, the successful attempts at interacting with Ethereum concealed a handful of problems that would be later uncovered. The smart contracts' functionality, for instance, was presenting issues at the execution and output conversion. The problems were ultimately traced back to the use of a different version of the contract's compiler, which was not compatible with the version of the client used. Some of the issues persisted, which led us to ask for help from the Ethereum community, in GitHub[16], StackExchange, and other forums. We have eventually solved the problem, with a custom compilation of the software, and have successfully blended its official client implementation, *geth*, into the OpenWrt image.

The prerequisites for the establishment of an ad hoc Ethereum network expect first the configuration of a *Genesis* file and a local data directory, to set the initial state of the network and progressively save the history, as the blocks get mined. Configuring the genesis block resorts, as well, to the choice of a consensus protocol. The geth client supports two main consensus protocols, a Proof-of-Work (PoW) and a Proof-of-Authority (PoA) based mechanism. Both protocols were tried out, but PoA was ultimately chosen for carrying out the remaining tests, since it offered a more controlled environment for the flexible adjustment of multiple parameters, like the block time. A more thorough analysis

[14] `https://github.com/edurbrito/dist-sys-seminar`

[15] `https://geth.ethereum.org/docs/fundamentals/private-network`

[16] `https://github.com/ethereum/go-ethereum/issues/27009`

of the PoW and PoA protocols, as well as trade-offs and performance comparisons, are left for future work. The PoA protocol relies on a list of authorized signers, which are allowed to mine blocks [45]. The signers' list is defined in the genesis block, and the private key of each signer is used to sign the blocks. We have automated the key pair generation at startup and developed multiple utility programs to facilitate the deployment of the nodes and the execution of the protocol. These custom tools were written in Golang and compiled for the target platform, to be used as part of the OpenWrt image, serving also as demonstration of the feasibility of the process of embedding custom software into the system. These command-line programs can be found under the "proof-of-concept/src/geth-utils" directory. Among other tasks, they automate the initialization of the blockchain nodes, via the genesis file, and the establishment of the network, via the discovery of bootnodes and connection of new peers. Each node exposes the necessary API endpoints to allow for the interaction with the rest of the protocol participants. Network rules were also configured to restrict the communication to the zone subnet, as well as a caching policy to avoid unnecessary resource consumption. Everything was accomplished with the help of the Ethereum geth client, and the final blockchain network arrangement is illustrated in Figure 20.

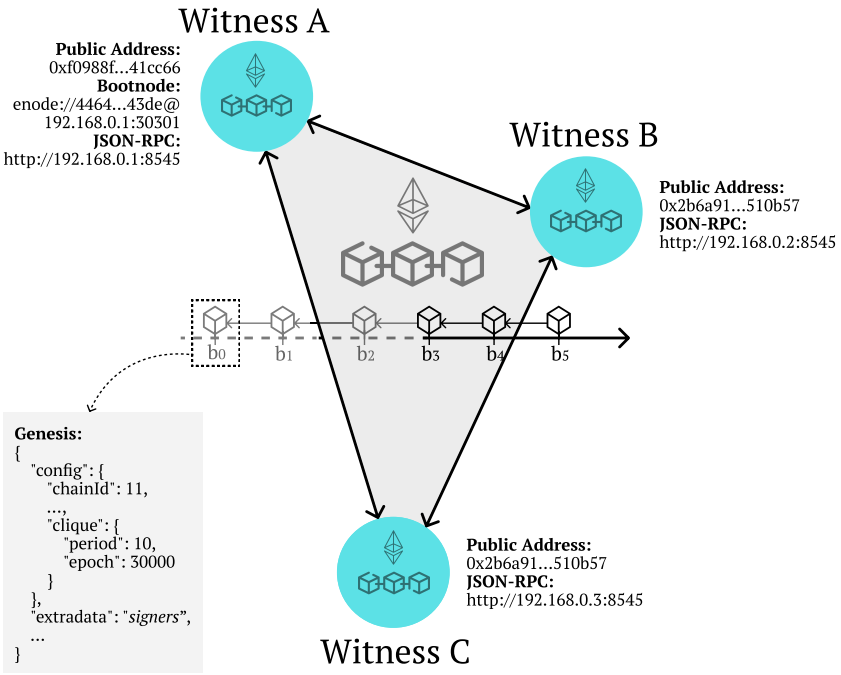


Figure 20. The Ethereum blockchain network setup, featuring a Proof-of-Authority consensus protocol with a block time of 10 seconds.

5.2.2 Proof Generation and Verification

Guaranteed the establishment of a permissionless consensus mechanism, the witnesses are finally synchronized in time. The prover can now take advantage of this synchrony to request the generation of a Proof-of-Location certificate. To accomplish such task, the prover instance, which is also part of the mesh network communication channel and has been assigned an IP address in the same subnet, needs to interact with the running blockchain and follow the protocol specified in Section 4.4.

The automation of the prover process was also achieved with the help of a Golang utility program, found under the "proof-of-concept/src/geth-utils" directory. A random witness is first chosen and inquired about the most recent block. After that, a transaction is assembled, signed, and submitted. All the requests are accomplished via the JSON-RPC API exposed by the geth client running in the witnesses' machines. The transaction is then broadcasted to the network and the prover waits for it to be included in a block. Once the transaction is confirmed, the prover can verify its validity. If valid, it can finally request the block signatures from the witnesses. The transaction validation process may be automated via the deployment of a smart contract in the network, decentralizing and automating the procedure. An example of a smart contract, located in the "proof-of-concept/src/block-verifier" directory, was also developed and successfully deployed. The contract was written in Solidity[17], compiled to the Ethereum Virtual Machine (EVM) bytecode, and included in the genesis file. It compares the input hash, submitted by the prover, with the hash of the last block in the network. The contract is invoked via a typical blockchain transaction, and returns a boolean value, indicating whether the transaction is valid or not, automating the validation process that would otherwise be performed manually, by either the prover or the verifier. This approach serves as demonstration of the feasibility of the deployment of smart contracts in the zone's blockchain. More sophisticated contracts can be developed and deployed according to application needs, expanding the possibilities of the protocol and its use cases for providing decentralized and zone-relative location services (see Section 4.3). Furthermore, the success of the proof generation process is dependent on the adjustment of the block time, which is a parameter that can be configured in the chosen PoA protocol. We have set the block time T to 10 seconds, in our current implementation. Similar to the time-limited approach presented by Nosouhi et al. [38], the block interval serves the purpose of preventing proxy or wormhole attacks, as depicted in Figure 21, shortening the chances for an adversary that is synchronized with the witnesses to interact with a remote prover and still be able to generate a spatio-temporally sound Proof-of-Location certificate. This time interval should be planned and adjusted according to the application needs and the desired levels of security.

[17] `https://docs.soliditylang.org/en/v0.8.19/`

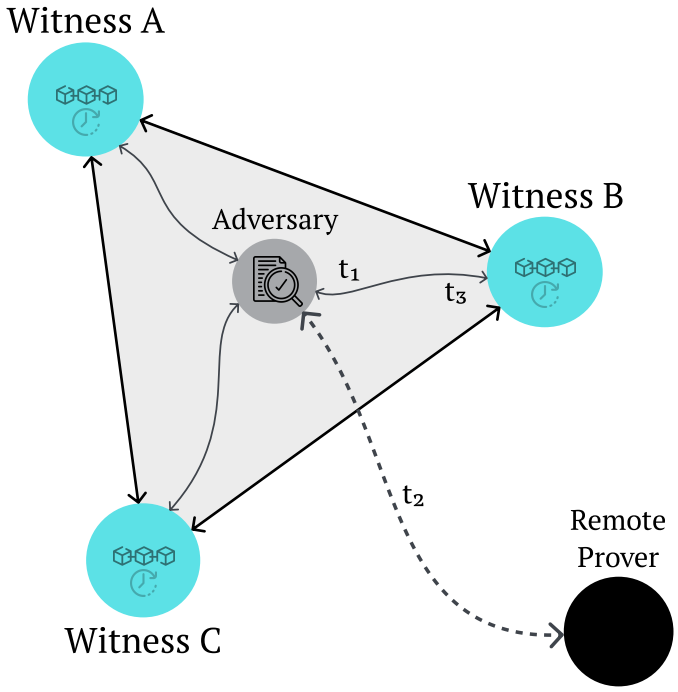


Figure 21. A possible attack scenario, where a malicious prover is able to generate a Proof-of-Location certificate without being nearby the witnesses. If $t_1 + t_2 + t_3 \leq T$, the adversary is able to act on behalf of the prover, synchronizing with the zone, asking the remote prover for a transaction signature, and generating a valid certificate [38].

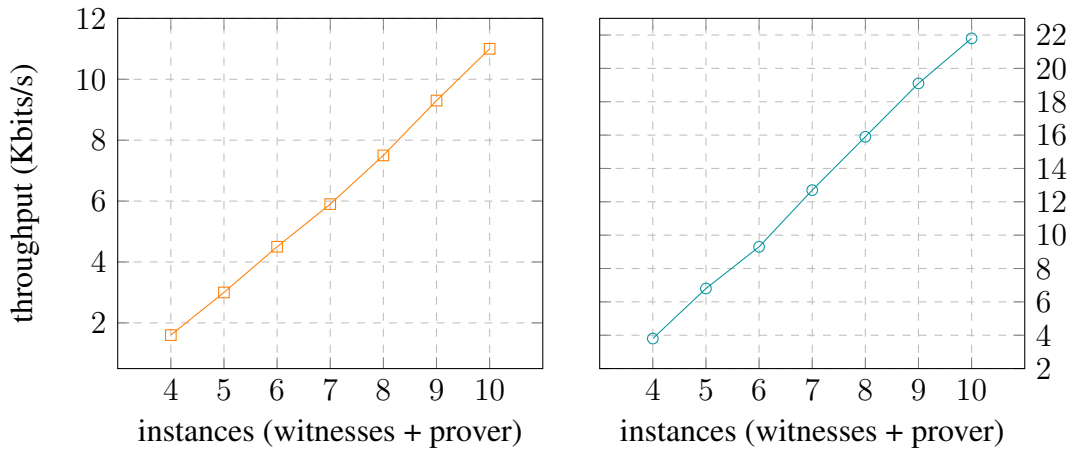
The process of verifying the certificate validity was demonstrated along with the generation process, for testing purposes. However, the verification process can be independently automated as well, for instance, by making use of smart contracts. We have also developed an example of a smart contract, under the "proof-of-concept/src/pol-verifier" directory, that verifies the witnesses and prover signatures. Such contract is ready to be deployed in any private or public blockchain network, acting as a decentralized record keeper of the Proof-of-Location certificates. Nevertheless, it is important to note that the verification procedure is not limited to the use of smart contracts. The process can be performed by any verifier that has access to the entities' public keys and the Proof-of-Location certificate, just like any typical digital signature verification, integrated with digital applications of all kinds.

5.3 Measurements

Along with the implementation of the proof-of-concept, we have conducted multiple experiments to evaluate the networking and computational performance of the proposed

approach. The testbed was empirically hosted on a laptop with an AMD Ryzen 7 4700U CPU and 16 GB of RAM, running an x86-64 Linux 6.1.24-1-lts system. Every instance got assigned 1 GB of RAM and 2 virtual CPUs.

The bridge interface, pooling all the mesh traffic exchanged between the tap interfaces, was the starting point for the networking measurements. All these interfaces had a maximum virtual bandwidth of 10 Mbit per second, assigned by the hosting system. The traffic was monitored using Wireshark[18], and the metrics were collected throughout the various stages of the experiment, by listening to packets of different kinds, flowing through the bridge interface. Establishing the mesh network connectivity, the Ethernet frames belonging to the batman-adv protocol sized an average of 74 bytes, and the IPv4 related packets averaged at 278 bytes, presenting both protocols a seemingly linear throughput increase with the increase in the number of instances, as shown in Figure 22.



(a) Batman-adv traffic throughput.

(b) IPv4 traffic throughput.

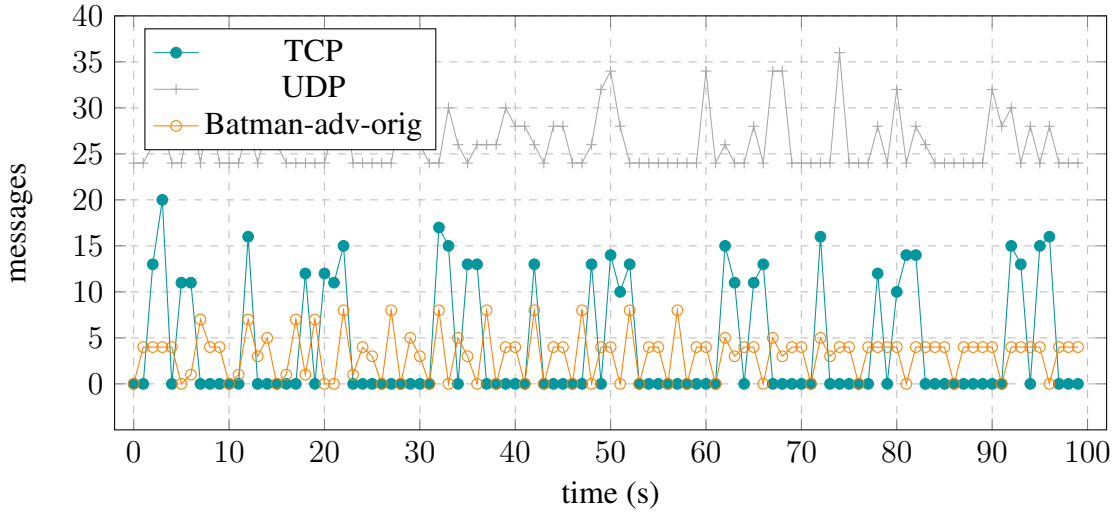
Figure 22. Average protocol throughput, measured in the bridge interface.

The Blockchain activity was also monitored, in order to observe the protocol behaviour, regarding the block generation and proposal phases. Figure 23 captures the number of messages exchanged between the instances, during a time frame of typical protocol activity. The interval time between blocks, set to 10 seconds, corresponds to the higher peaks of TCP traffic, while the UDP traffic is more evenly distributed. This behaviour gets more pronounced as the number of instances increases.

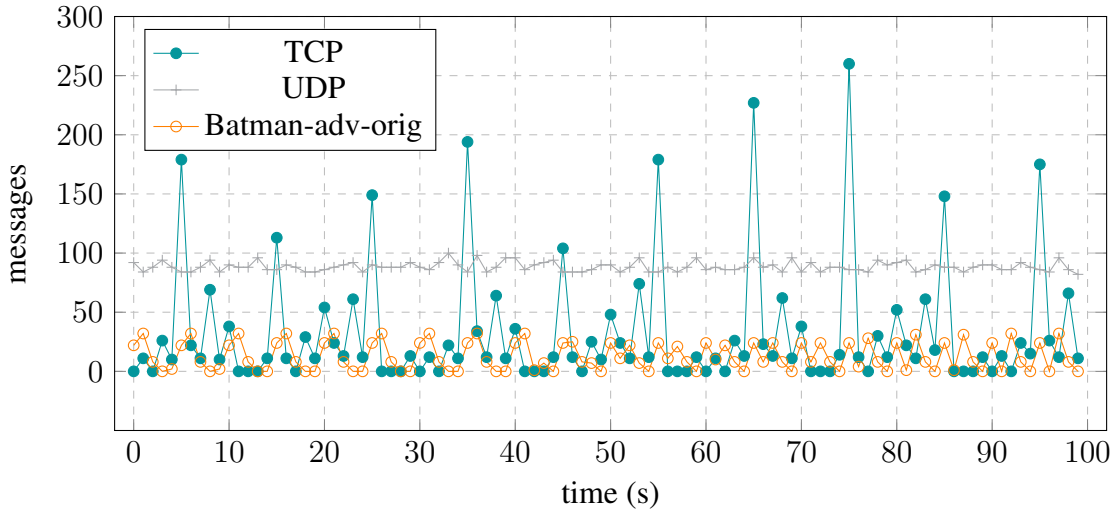
Both CPU and RAM usages were also continuously measured across the whole experiment. The two virtual cores of each instance saw the CPU usage averaging at 2%, peaking at 20% when the prover would interact with a witness, or vice versa, in order to produce a Proof-of-Location certificate. The overall RAM usage did not go

[18] `https://www.wireshark.org/`

beyond 200 MB. These numbers are in line with the expected behaviour of the protocol, running the PoA consensus algorithm, as a lightweight mechanism that may not require much computational power, suitable and adaptable to resource-constrained environments. The PoW consensus algorithm, on the other hand, would require a much higher and variable computational power, that would be difficult to predict and control, since it is not only manually configured, but dependent, as well, on the dynamic difficulty adjustment mechanism, in order to meet a fixed block time. Nevertheless, both protocols should present similar network traffic patterns. In terms of storage, the disk space used did not exceed 65 MB, for the entire file system.



(a) 4 instances.



(b) 8 instances.

Figure 23. The network activity, with a block time of 10 seconds.

The proof generation process did not possess enough relevancy for time measurement purposes. Its execution speed, during the transaction creation, signing, and broadcasting phases is highly dependent on the running environment, and not limited to the showcased implementation. It can be effectively optimized in multiple ways, using different libraries, programming languages, or parallelization techniques. Moreover, the total execution time is ultimately bounded to the block time, since the prover needs to wait a maximum of T units of time for the generation of the new block that may contain its transaction. The proof generation process is also not a bottleneck in the protocol, as it is not a part of the consensus mechanism, and may be executed in parallel with the other protocol activities. The certificate assembly and verification processes can, as well, be performed later and asynchronously. Nonetheless, the success rate of the generation of Proof-of-Location certificates may still be assessed. For such experiment, we adjusted the interval time between blocks and measured the number of valid Proof-of-Location certificates. A similar test was conducted by Nosouhi et al. [38], targeting the effectiveness of their protocol against prover-prover collusions, or proxy and wormhole attacks. The test is still dependent on the characteristics of the running environment, but the results sustain the conclusion that the block time is a crucial parameter in the protocol. Figure 24 shows a direct relation between the block time and the success rate in generating valid certificates. The higher the block time, the lower the failures. The consequences of such relation are twofold. A more permissive block time allows for a lower number of invalid certificates, but also for a higher probability of witnessing malicious activity, such as collusions between adversaries and remote provers, as explained in Figure 21. The block time is, therefore, a trade-off between the two, and should be carefully adjusted to the running environment. Further reasoning is still needed, to deeper assess the impact of this heuristic, or to propose a more robust solution.

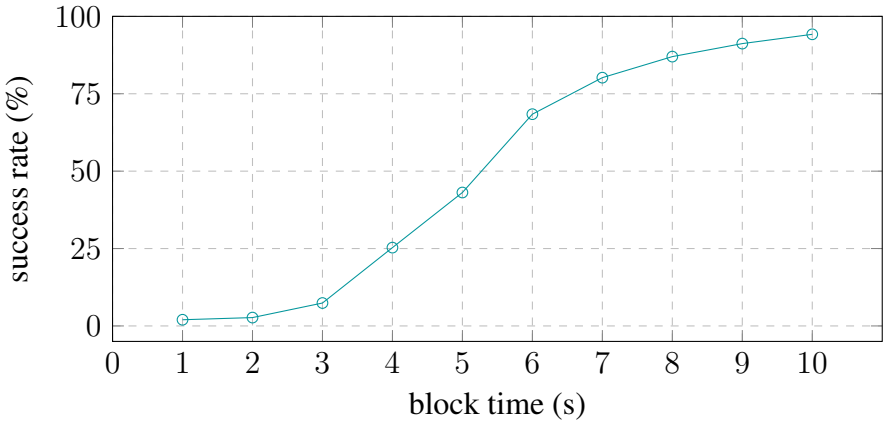


Figure 24. The success rate of the generation of Proof-of-Location certificates.

In retrospective, the main goal of achieving a fully functional proof-of-concept for the Proof-of-Location protocol was, indeed, reached. We have fulfilled the requirements proposed in Chapter 4 and made a demonstration possible. The protocol was implemented in a distributed setting, with a modular network architecture, and a clear separation of concerns between the different stages. We have also engineered the solution with enough flexibility, allowing for an easy integration of new components, or the replacement or tuning of the existing ones, in order to adapt to different environments, or to improve the overall performance. Further measurement considerations, along with the demonstration in a physical environment, are already in our horizon. The next chapter will address the overall conclusions and the future work that may be conducted, in order to improve and expand this Proof-of-Location protocol.

6 Conclusion

The Proof-of-Location problem was the main target of this thesis. Through theoretical and practical contributions, we attempted at dissecting its roots. The work started with the identification of the underlying concepts, hypotheses, and real-world applications, and was followed by a review of the state of the art, encompassing a wide range of trust levels and infrastructural scenarios.

Inspired by previous work, this thesis delivered a novel approach to the problem. The proposed solution involved the specification of a decentralized Proof-of-Location protocol and the implementation of a proof-of-concept. The final work mixes routing algorithms for mobile ad hoc mesh networks and permissionless consensus mechanisms, in order to finally achieve collective agreement on one's location at a certain point in time. We demonstrated the protocol using a network architecture that is both distributed and modular, where each stage has its own distinct set of responsibilities, covering the entire networking spectrum. It started in the physical and data link layers, taking advantage of mesh topologies. Followed was the establishment of spatially synchronized zones, exploiting the flexibility of a routing protocol to further enable network and transport layer capabilities. Reaching the application layer, a permissionless consensus mechanism was used to achieve time synchronization, finally allowing location attestations to be generated and verified. The enabling of Turing-complete smart contracts allowed us to showcase the protocol's extensibility and the possibility of implementing more complex logic, for the provision of decentralized location services. An attack vector was also identified, and a mitigation strategy was discussed. The chosen technologies for both the routing algorithm and the blockchain framework showed optimistic results that make the work suitable and adaptable to resource-constrained environments. We believe the proposed protocol is a promising step towards achieving Proof-of-Location, in a decentralized and trustless manner.

The groundwork for the implementation of a novel Proof-of-Location protocol has been established, but further investigation is still needed. For instance, we are looking forward to identifying the most effective identity management systems and crypto-economic incentives to motivate nodes to collaborate and establish and maintain coverage zones, with the research on the processes of zone discovery and zone affinity management revealing to be essential for the overall success of the protocol. Another aspect that deserves further investigation is the shift from deterministic-finality Byzantine fault-tolerant algorithms to probabilistic finality consensus mechanisms. Formal verifications, measurements, and comparisons between the two approaches are necessary to determine the best candidates for decentralized Proof-of-Location. Furthermore, we have demonstrated the simplest case of applying a consensus mechanism to achieve time synchronization. Extending the approach to reach more accurate location attestations, making full use of the system's Turing completeness for more complex logic, and ensuring the extensibility of the approach are all areas for future work. A thorough analysis of the robustness,

security, privacy, and correctness of the baseline solution is also to be presented. We point our interests as well towards the integration of privacy preserving mechanisms, such as zero-knowledge proofs. Lastly, the live deployment of the solution and the evaluation of the performance in real-world scenarios, contributing to the production of more secure and tamper-proof location data, is the ultimate step we are aiming for.

Acknowledgments

I would like to express gratitude towards my supervisor, Ulrich Norbisrath, for all the guidance, feedback, and support provided throughout the development of this thesis. Eero Vainikko and my classmates of the Distributed Systems seminar, my friends, family, and all the people that have contributed to the reasoning and validation of the work are also acknowledged.

Words of appreciation should go, as well, to the multiple tools that aided the writing and development of this thesis. ChatGPT, Grammarly, Code Spell Checker, LTeX, and GitHub Copilot provided invaluable support throughout the working process, without discrediting all the other tools, extensions, and development environments that, directly or indirectly, with or without AI features, contributed to the efficiency of the work. Appendix III provides a detailed description of the writing workflow and the tools used to produce this thesis.

Finally, I would like to thank the University of Tartu for providing multiple opportunities and access to resources and funding, supporting the development of this work. In particular, I would like to mention the IT Academy, the XRP Ledger Foundation, and Cybernetica AS, for the scholarship support provided.

References

[1] W. Luo and U. Hengartner, "Veriplace: a privacy-aware location proof architecture," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 23–32, 2010.

[2] M. Amoretti, G. Brambilla, F. Medioli, and F. Zanichelli, "Blockchain-based proof of location," in *2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 146–153, IEEE, 2018.

[3] B. Nasrulin, M. Muzammal, and Q. Qu, "A robust spatio-temporal verification protocol for blockchain," in *Web Information Systems Engineering–WISE 2018: 19th International Conference, Dubai, United Arab Emirates, November 12-15, 2018, Proceedings, Part I 19*, pp. 52–67, Springer International Publishing, 2018.

[4] W. Li, H. Guo, M. Nejad, and C.-C. Shen, "Privacy-preserving traffic management: A blockchain and zero-knowledge proof inspired approach," *IEEE access*, vol. 8, pp. 181733–181743, 2020.

[5] A. Dupin, J.-M. Robert, and C. Bidan, "Location-proof system based on secure multi-party computations," in *Provable Security: 12th International Conference, ProvSec 2018, Jeju, South Korea, October 25-28, 2018, Proceedings*, pp. 22–39, Springer, 2018.

[6] S. Saroiu and A. Wolman, "Enabling new mobile applications with location proofs," in *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, pp. 1–6, 2009.

[7] E. Pournaras, "Proof of witness presence: Blockchain consensus for augmented democracy in smart cities," *Journal of Parallel and Distributed Computing*, vol. 145, pp. 160–175, 2020.

[8] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer networks*, vol. 47, no. 4, pp. 445–487, 2005.

[9] A. Cilfone, L. Davoli, L. Belli, and G. Ferrari, "Wireless mesh networking: An iot-oriented perspective survey on relevant technologies," *Future Internet*, vol. 11, no. 4, p. 99, 2019.

[10] M. L. Sichitiu, "Wireless mesh networks: opportunities and challenges," in *Proceedings of World Wireless Congress*, vol. 2, p. 21, 2005.

[11] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke, "Ieee 802.11 s: the wlan mesh standard," *IEEE Wireless Communications*, vol. 17, no. 1, pp. 104–111, 2010.

[12] S. Bari, F. Anwar, and M. Masud, "Performance study of hybrid wireless mesh protocol (hwmp) for ieee 802.11 s wlan mesh networks," in *2012 international conference on computer and communication engineering (ICCCE)*, pp. 712–716, IEEE, 2012.

[13] D. Seither, A. König, and M. Hollick, "Routing performance of wireless mesh networks: A practical evaluation of batman advanced," in *2011 IEEE 36th Conference on Local Computer Networks*, pp. 897–904, IEEE, 2011.

[14] "Open-mesh. originator message version 2 (ogmv2)." `https://www.open-mesh.org/projects/batman-adv/wiki/OGMv2`. Accessed: 2023-02-16.

[15] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *Journal of the ACM (JACM)*, vol. 27, no. 2, pp. 228–234, 1980.

[16] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," in *Concurrency: the works of leslie lamport*, pp. 203–226, Springer, 2019.

[17] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OsDI*, pp. 173–186, 1999.

[18] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008.

[19] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, pp. 2–1, 2014.

[20] A. Miller, Y. Xia, K. Croman, E. Shi, and D. Song, "The honey badger of bft protocols," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 31–42, 2016.

[21] C. Decker, J. Seidel, and R. Wattenhofer, "Bitcoin meets strong consistency," in *Proceedings of the 17th International Conference on Distributed Computing and Networking*, pp. 1–10, 2016.

[22] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019.

[23] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[24] Y. Xiao, N. Zhang, J. Li, W. Lou, and Y. T. Hou, "Distributed consensus protocols and algorithms," *Blockchain for Distributed Systems Security*, vol. 25, p. 40, 2019.

[25] S. Bouraga, "A taxonomy of blockchain consensus protocols: A survey and classification framework," *Expert Systems with Applications*, vol. 168, p. 114384, 2021.

[26] B. Lashkari and P. Musilek, "A comprehensive review of blockchain consensus mechanisms," *IEEE Access*, vol. 9, pp. 43620–43652, 2021.

[27] C. Natoli, J. Yu, V. Gramoli, and P. Esteves-Verissimo, "Deconstructing blockchains: A comprehensive survey on consensus, membership and structure," *arXiv preprint arXiv:1908.08316*, 2019.

[28] B. Waters and E. Felten, "Secure, private proofs of location," *Department of Computer Science, Princeton University, Tech. Rep. TR-667-03*, 2003.

[29] M. Graham and D. Gray, "Protecting privacy and securing the gathering of location proofs-the secure location verification proof gathering protocol.," in *MobiSec*, pp. 160–171, Springer, 2009.

[30] C. Javali, G. Revadigar, K. B. Rasmussen, W. Hu, and S. Jha, "I am alice, i was in wonderland: secure location proof generation and verification protocol," in *2016 IEEE 41st conference on local computer networks (LCN)*, pp. 477–485, IEEE, 2016.

[31] M. R. Akand, R. Safavi-Naini, M. Kneppers, M. Giraud, and P. Lafourcade, "Privacy-preserving proof-of-location with security against geo-tampering," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[32] Z. Zhu and G. Cao, "Applaus: A privacy-preserving location proof updating system for location-based services," in *2011 Proceedings IEEE INFOCOM*, pp. 1889–1897, IEEE, 2011.

[33] X. Wang, A. Pande, J. Zhu, and P. Mohapatra, "Stamp: Enabling privacy-preserving location proofs for mobile users," *IEEE/ACM transactions on networking*, vol. 24, no. 6, pp. 3276–3289, 2016.

[34] S. Gambs, M.-O. Killijian, M. Roy, and M. Traoré, "Props: A privacy-preserving location proof system," in *2014 IEEE 33rd International Symposium on Reliable Distributed Systems*, pp. 1–10, IEEE, 2014.

[35] M. R. Nosouhi, S. Yu, M. Grobler, Y. Xiang, and Z. Zhu, "Sparse: privacy-aware and collusion resistant location proof generation and verification," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, IEEE, 2018.

[36] Z. Yang, C. Jin, J. Ning, Z. Li, A. Dinh, and J. Zhou, "Group time-based one-time passwords and its application to efficient privacy-preserving proof of location," in *Annual Computer Security Applications Conference*, pp. 497–512, 2021.

[37] W. Wu, E. Liu, X. Gong, and R. Wang, "Blockchain based zero-knowledge proof of location in iot," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, pp. 1–7, IEEE, 2020.

[38] M. R. Nosouhi, S. Yu, W. Zhou, M. Grobler, and H. Keshtiar, "Blockchain for secure location verification," *Journal of Parallel and Distributed Computing*, vol. 136, pp. 40–51, 2020.

[39] R. J. King, "Foam: The importance of time synchronization." https://blog.foam.space/foam-the-importance-of-time-synchronization-3934755ccc4e, Feb 2020. Accessed: 2023-03-10.

[40] F. Corp, "Foam whitepaper." `https://foam.space/publicAssets/FOAM_Whitepaper.pdf`. Accessed: 2023-03-10.

[41] L. L. Peterson and B. S. Davie, *Computer networks: a systems approach*. Elsevier, 2007.

[42] S. Misra, S. C. Misra, and I. Woungang, *Guide to wireless mesh networks*, vol. 9. Springer, 2009.

[43] "B.a.t.m.a.n. v." `https://www.open-mesh.org/projects/batman-adv/wiki/BATMAN_V`. Accessed: 2023-02-16.

[44] M. R. Malekpour, "A self-stabilizing hybrid fault-tolerant synchronization protocol," in *2015 IEEE Aerospace Conference*, pp. 1–11, IEEE, 2015.

[45] E. I. Proposals, "Eip-225: Clique proof-of-authority consensus protocol." `https://eips.ethereum.org/EIPS/eip-225`, 2017. Accessed: 2023-04-17.

Appendix

I. Repository

The source code, documentation, and other materials produced for this thesis are available in the following repository: `https://github.com/edurbrito/master-thesis-ut`.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Eduardo Ribas Brito**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Towards Decentralized Proof-of-Location,

   supervised by Ulrich Norbisrath.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Eduardo Ribas Brito
09/05/2023

III. Writing Workflow

The writing and development process of this work was supported and enhanced by multiple tools, extensions, and development environments. This section details the workflow and tools used to produce this thesis.

The writing environment was based on LaTeX, a document preparation system that allows writers to focus on the content, rather than the formatting. Visual Studio Code was the development environment chosen to write the LaTeX code and the thesis content. This lightweight and extensible code editor, combined with the LaTeX Workshop extension, provided a rich set of functionalities, such as syntax highlighting, code completion, source compilation, and live preview. Overleaf was the alternative considered, but it was discarded due to the less flexible and more limited development environment. The LaTeX source code was versioned using Git, and hosted on GitHub, a web-based version control hosting service. The repository also contains the source code of the proof-of-concept, as well as the supplementing documentation (see Appendix I). Within Visual Studio Code, the extensions Code Spell Checker and LTeX LanguageTool were also used to dynamically check the spelling and grammar of the text. Additionally, GitHub Copilot was enabled to provide code completion and debugging support, during the proof-of-concept programming. Several other extensions were also used to enhance the coding experience, with rich language-specific support.

Outside the development and writing environment, ChatGPT helped with summarizing, rephrasing, contextualizing, correcting, and paraphrasing indiscriminate sections of the writing content. This AI tool helped, as well, with the generation of code snippets and the explanation of multiple command-line interfaces. Grammarly was occasionally used to check the grammar and spelling of the text. Figma and Diagrams.net were used to create the diagrams and figures of this thesis, which contain several images and icons from Flaticon and Freepik catalogues.