

# Towards Decentralized Proof-of-Location

Making use of mesh network technologies and permissionless consensus mechanisms, we present a novel **decentralized Proof-of-Location protocol** (1 - 3) and the implementation of a **proof-of-concept** (4 - 7), showing the achievement of **space and time synchronization** and the generation of **complete, verifiable, and spatio-temporally sound** location proofs.

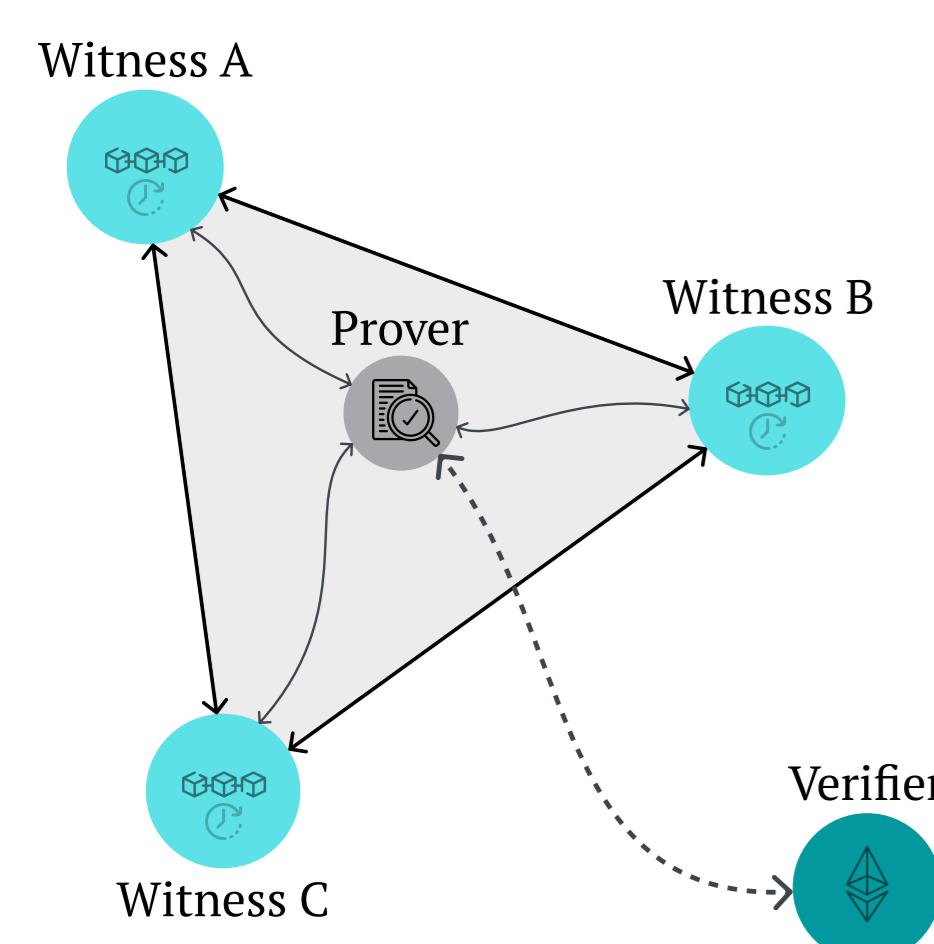
With the present surge of highly realistic generative AI tools, how would Joe Biden **prove the authenticity** of his review of that pizza place? How would a reporter **certify the integrity** of his pictures and videos, or a service provider prove the delivery or supply of goods, at a **specific location**?

**How would you prove you are here, right now, reading this poster?**

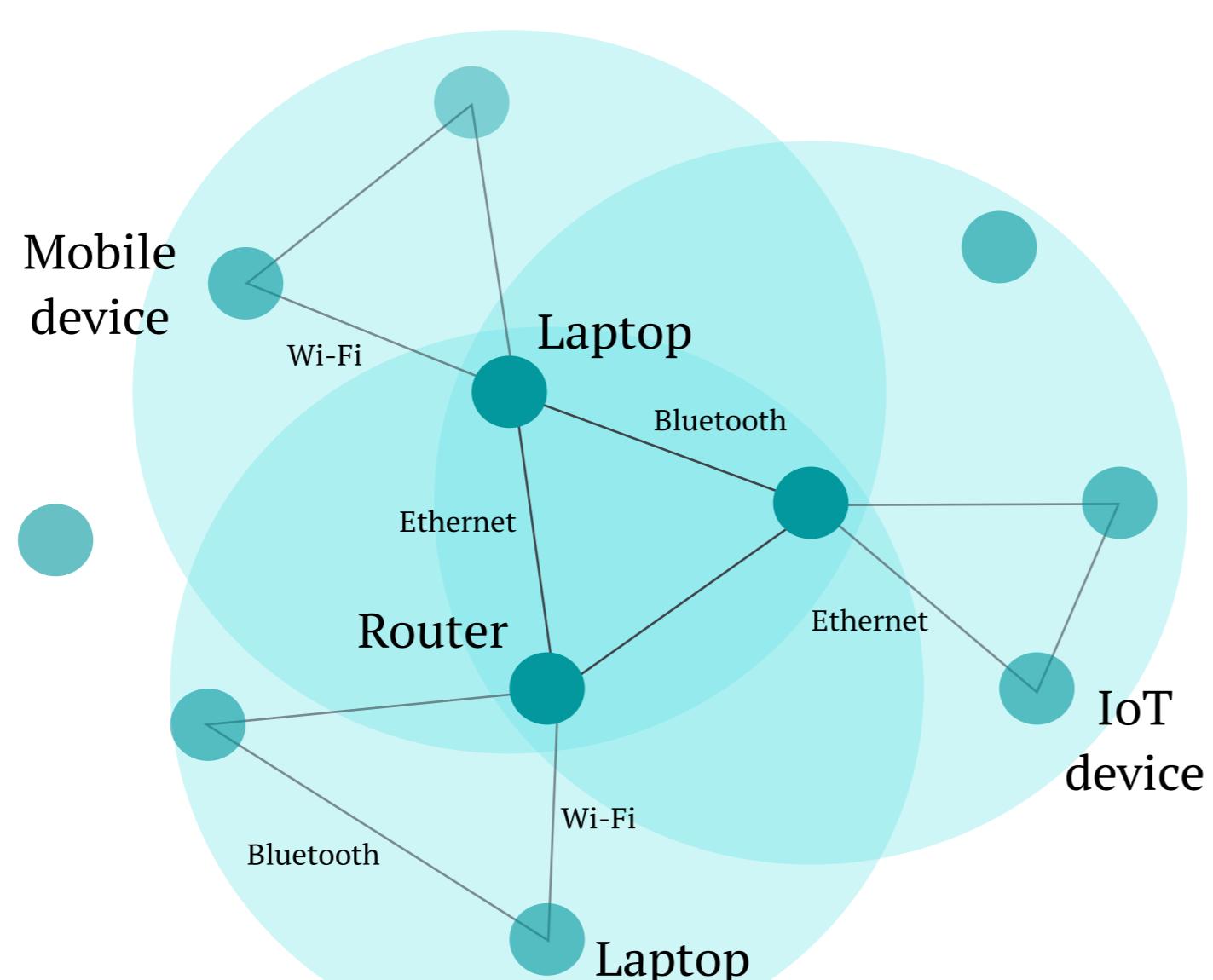


**A digital Proof-of-Location** is an electronic certificate that attests one's position in both space and time.

A **prover** engages in a short-range 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.



## 1. Dynamic Mesh Networks



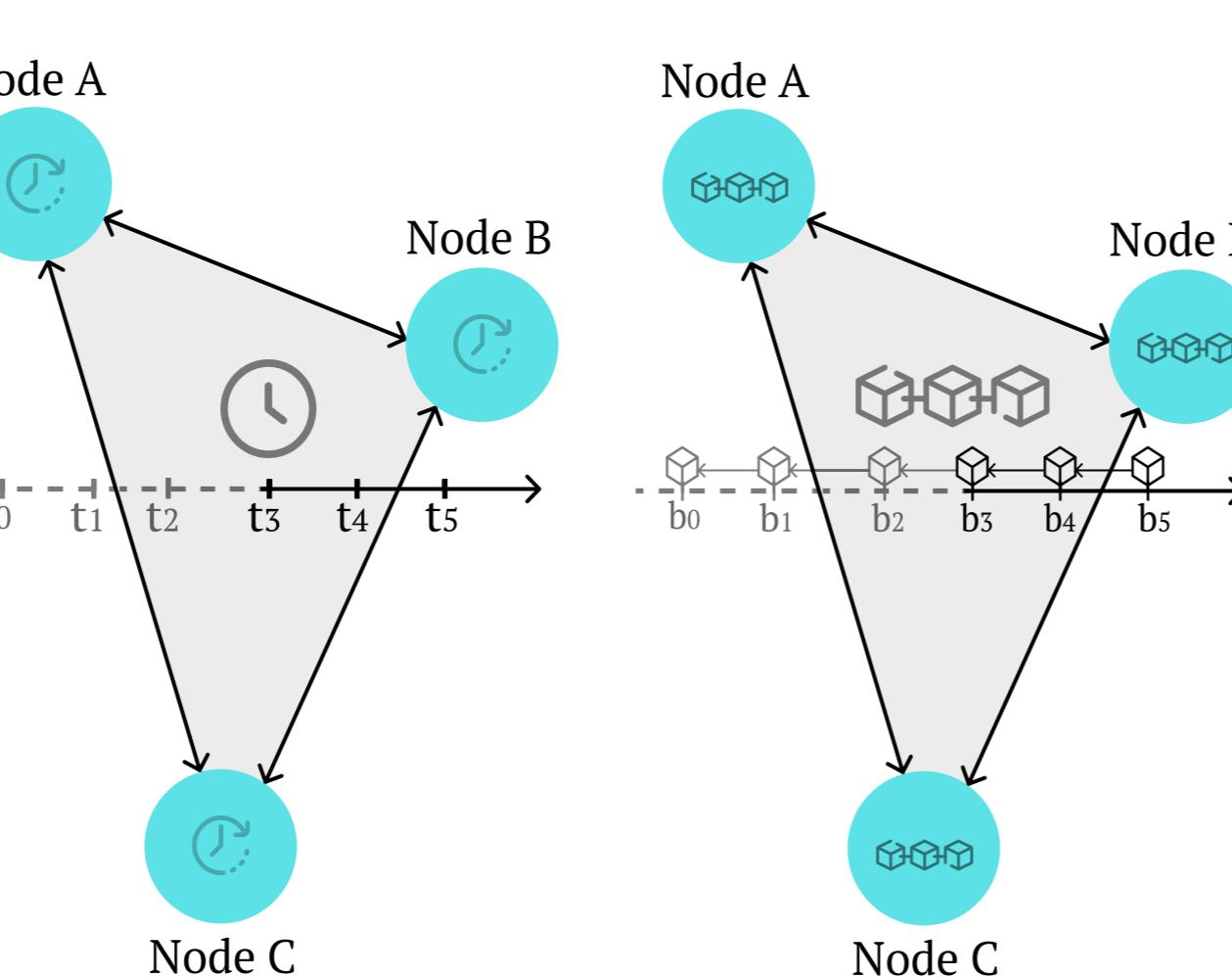
In mesh topologies, nodes are directly and dynamically connected, in a **non-hierarchical** way. This trait allows for peer-to-peer communication between devices, to efficiently route the data. The mesh nodes are expected to **dynamically self-organize** and self-configure.

**Mesh networks** enable decentralized and short-range exchange of messages, leading to space synchronization.

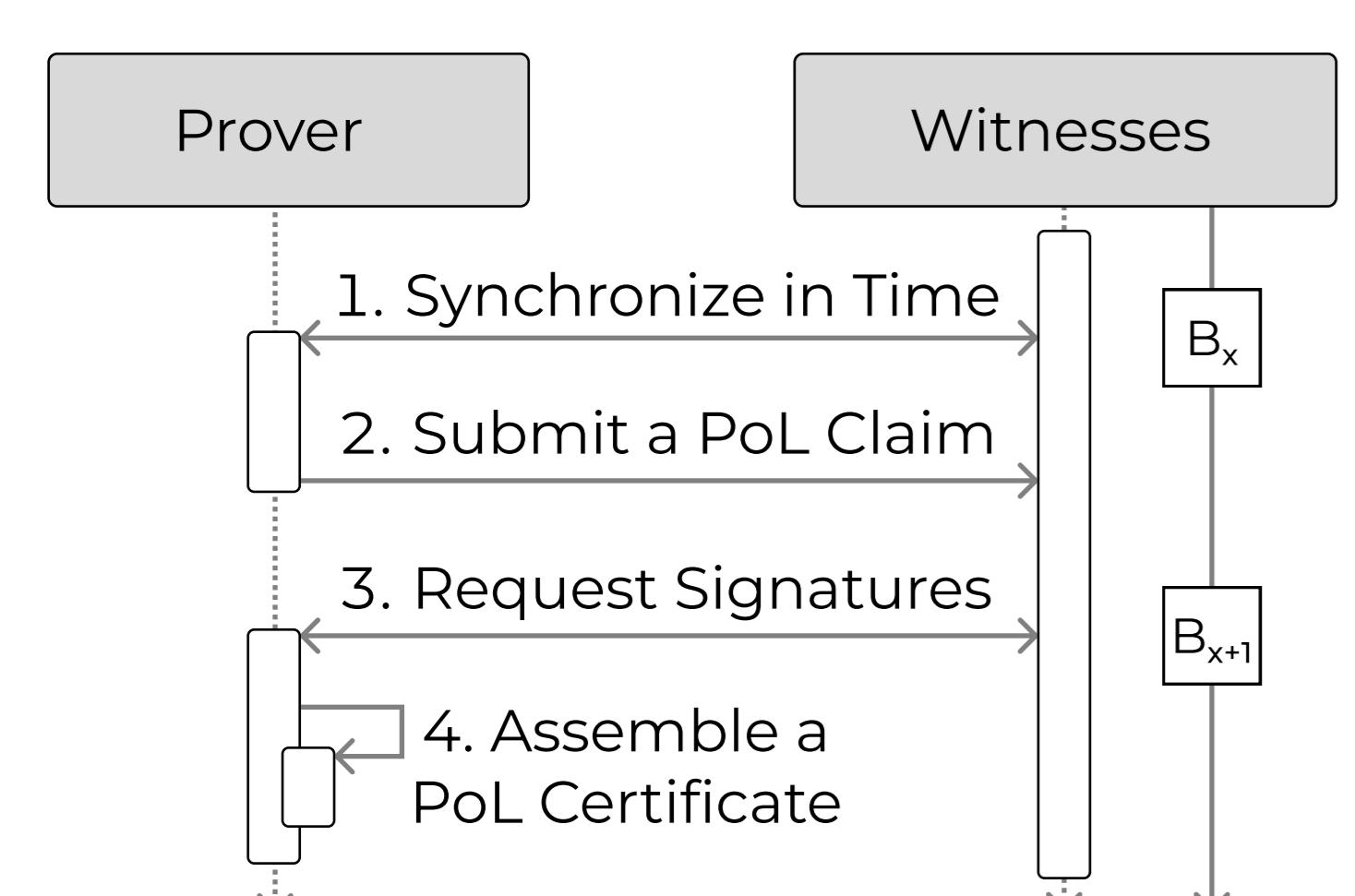
## 2. Permissionless Consensus

In **decentralized** and **trustless** environments, achieving time synchronization

is the problem of achieving permissionless consensus, and the need for **ordering** and **synchronizing** events at the same pace, when participants are **not necessarily trusted**. Permissionless consensus with a Turing-Complete system enables the decentralized and **time-conscious** agreement on the execution of arbitrary logic.



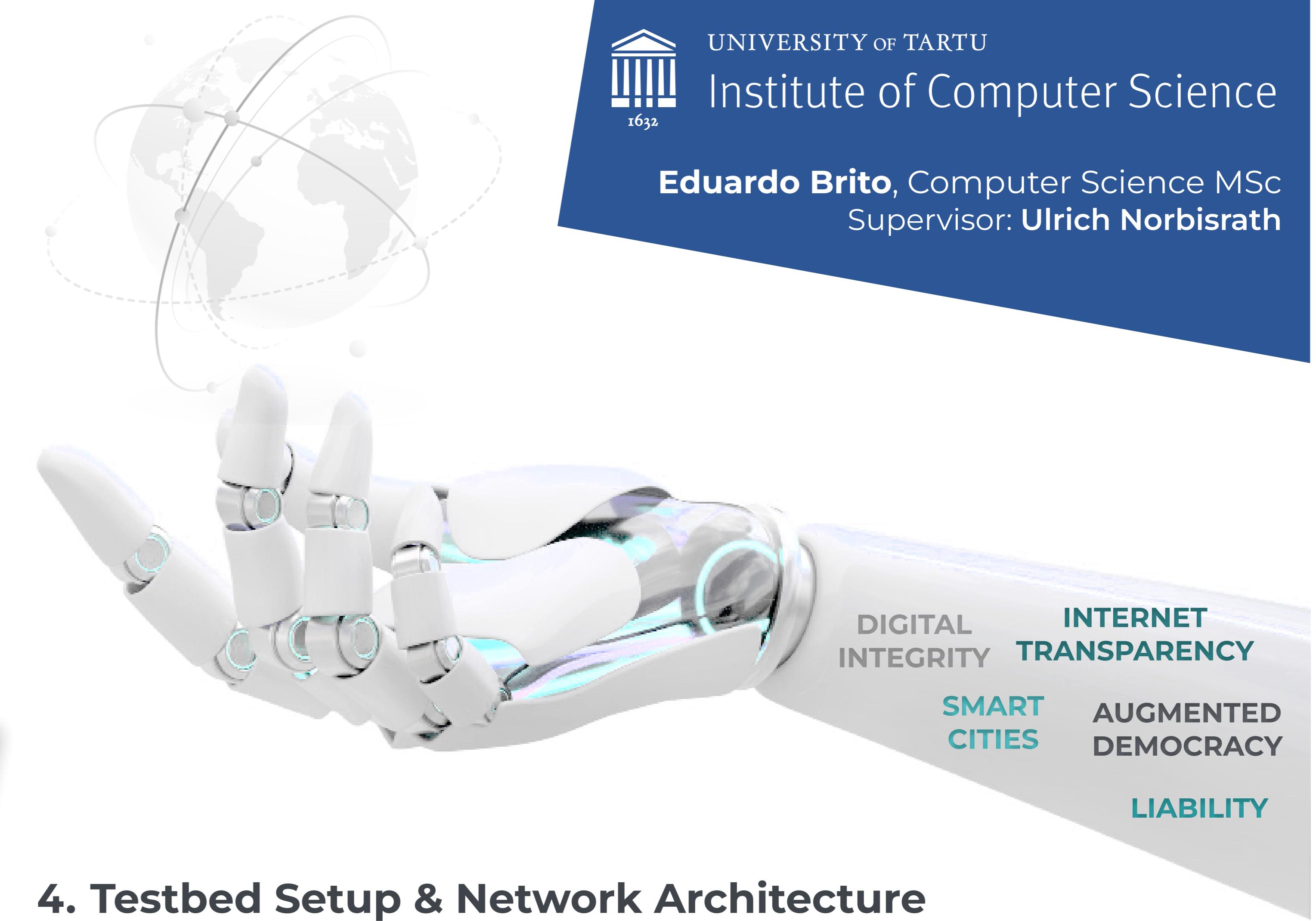
## 3. Verifiable Proof-of-Location



With the establishment of space and time synchronization, one can now generate

**complete, verifiable and spatio-temporally sound** location claims, achieving decentralized Proof-of-Location.

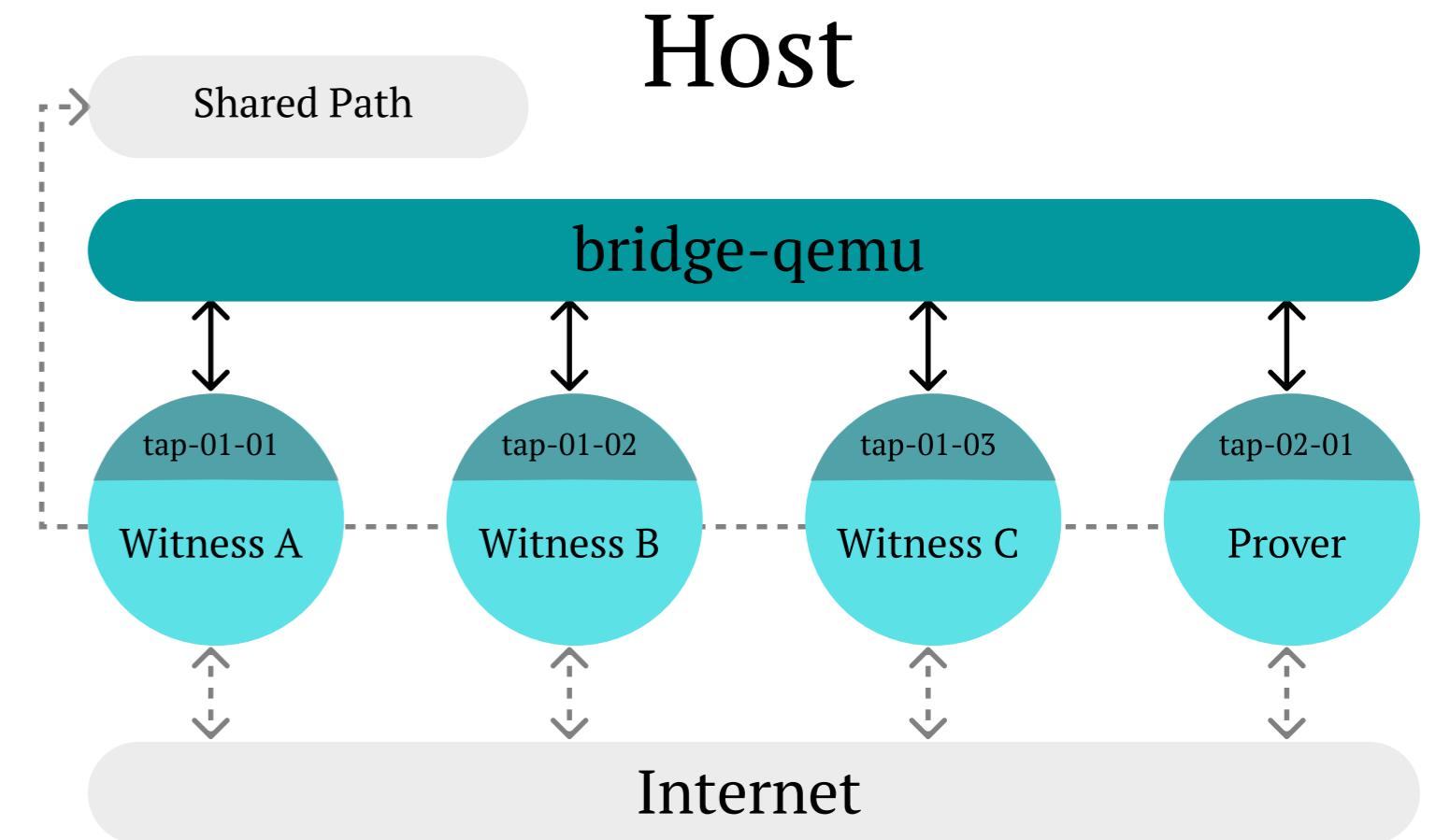
The verification process requires the nodes' public keys and the Proof-of-Location certificate, just like any **digital signature verification**, integrated with applications of all kinds.



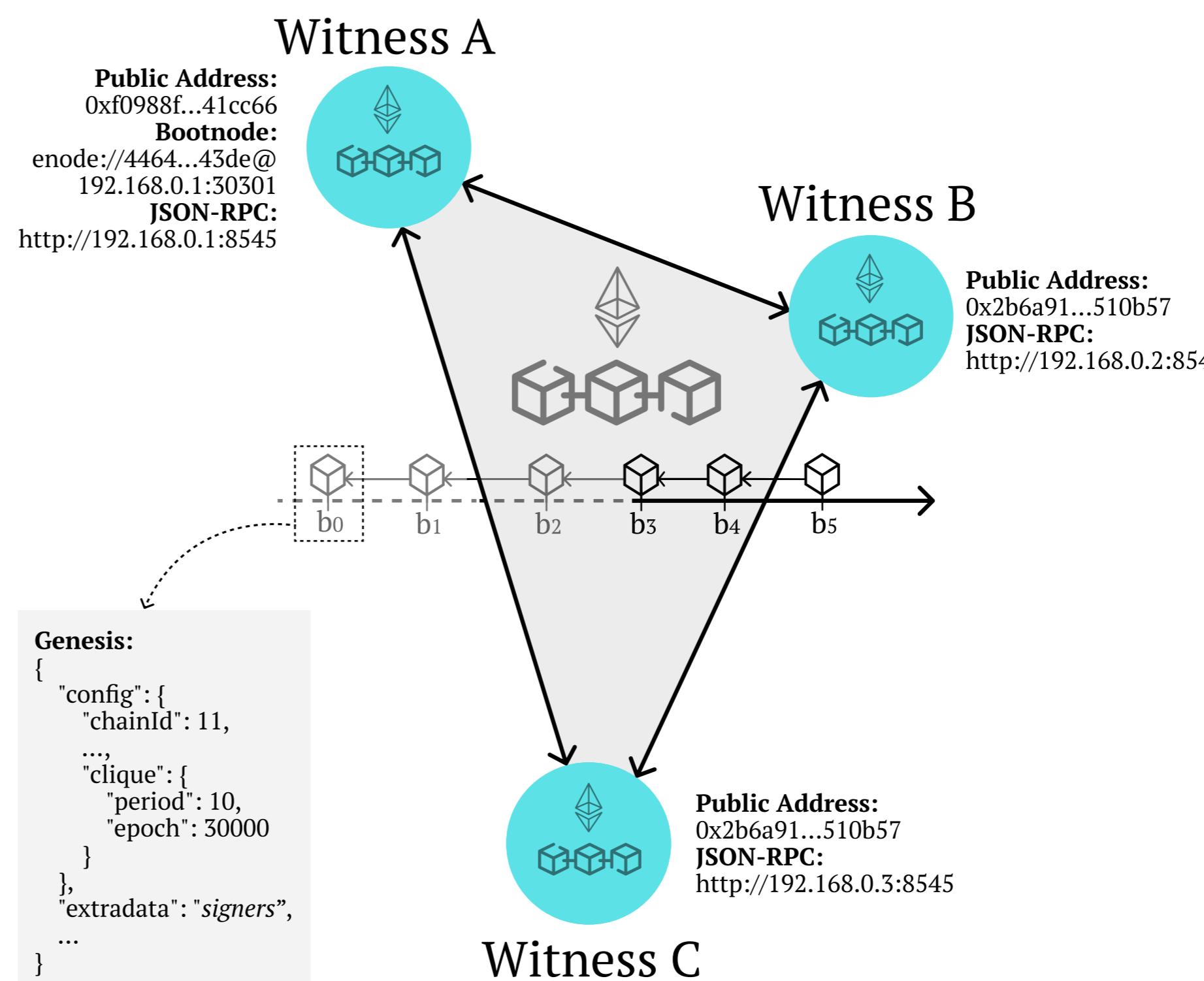
## 4. Testbed Setup & Network Architecture

The setup was emulated using **QEMU**. Each node ran **OpenWrt** as the OS, and **batman-adv** as the layer 2 routing protocol for mesh networking.

The instances are connected to a **bridge interface**, pooling all the raw mesh traffic, simulating the physical medium. Each interface gets a **MAC address**, and batman-adv is set up for the discovery of nodes, peer-to-peer connections, and the **formation of a witnessing zone**. Taking advantage of the TCP/IP suite of protocols and by **subnetting**, the typical Internet connections are enabled on top of the peer-to-peer mesh network topology.



## 5. Turing-Complete Clock Synchronization



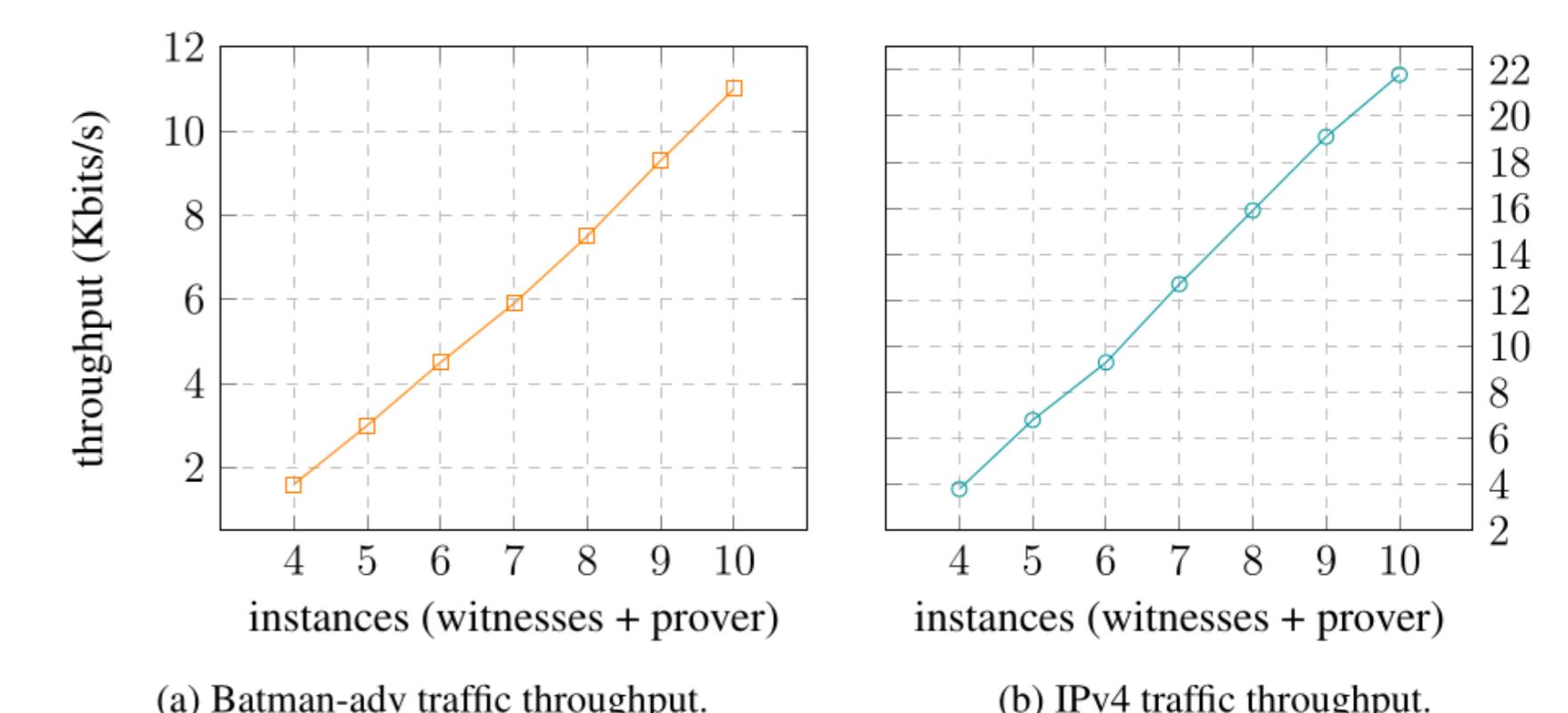
Ethereum was the chosen **blockchain** framework. The ad-hoc network was configured via the Genesis file, choosing a **consensus protocol** and some network rules, f.e., the **block time**. The nodes exposed API endpoints for the discovery of neighbors and peer-to-peer connections.

The Ethereum nodes started producing new blocks, achieving time synchronization.

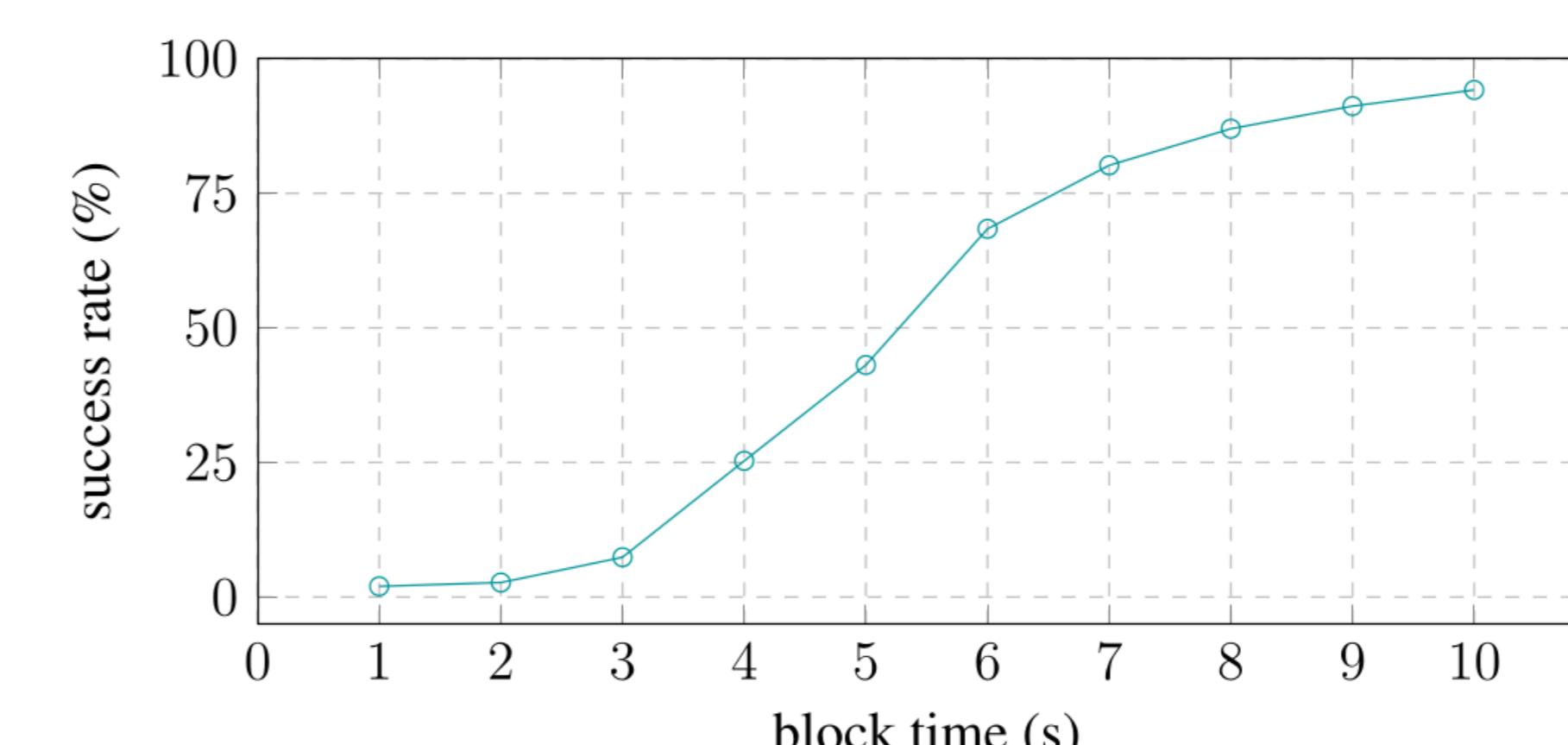
The proof generation and verification processes were automated via **smart contracts**, written in Solidity and running in the EVM.

## 6. Performance Measurements

Both the **batman-adv** and the **IPv4** related protocols showed a seemingly **linear increase of the average throughput**, with the increase in the number of witnesses. The blockchain traffic had low impact on the network overhead. CPU, RAM and Disk usages also showed **low resource consumption**, demonstrating the adaptability and suitability of the protocol to resource-constrained environments.



## 7. Attack Vectors



A more **permissive block time** allows for a **larger success rate**, but opens the possibility for the so-called **proxy attacks**: when an adversary has enough time to fetch the latest block, ask a remote prover for a signature, and submit a transaction. This allows to conclude that

the **block time** plays a crucial role in the soundness of the protocol.

