Witness Byzantine Fault Tolerance with Signature Tree and Proof-of-Navigation for Wide Area Visual Navigation

Nasim Paykari[1][a], Taylor Clark[2], Ademi Zain[1], Damian Lyons[1][b] and Mohamed Rahouti[2][c]
[1]*Computer and Information Science Department, Fordham University, Bronx, NY 10458, U.S.A.*
[2]*Computer and Information Science Department, Fordham University, New York, NY 10023, U.S.A.*

Keywords: Blockchain, Byzantine Fault Tolerance, Multi-Signature, Merkle Tree, Wide Area Visual Navigation, Cooperative Robotics.

Abstract: This paper presents Witness Byzantine Fault Tolerance (WBFT), a novel consensus protocol designed for Wide Area Visual Navigation (WAVN) systems, where cooperative robots share visual imagery in GPS-denied environments and reach agreement on navigation data via blockchain. WBFT addresses the limitations of traditional Byzantine Fault Tolerance (BFT) methods, particularly the high communication overhead of protocols like PBFT, by introducing a lightweight, secure, and signature-based consensus mechanism optimized for resource-constrained robotic networks. The protocol integrates a Proof-of-Stake-based leader election system, named Proof-of-Navigation (PoN), with a signature aggregation approach using Ed25519 cryptography and a Merkle tree structure, reducing verification complexity to $O(\log t)$ and achieving consensus with only $O(n)$ message complexity. WBFT tolerates up to $f \leq \lfloor (n-1)/3 \rfloor$ Byzantine faults and demonstrates superior resilience, scalability, and communication efficiency compared to existing BFT variants. Experimental results validate WBFT's performance across multiple metrics and network sizes, confirming its suitability for high-frequency, decentralized robotic coordination.

1 INTRODUCTION

Wide Area Visual Navigation (WAVN) enables a group of robots to collaboratively navigate by sharing visual imagery, assisting a robot in locating a designated home through a sequence of common landmarks (Lyons and Rahouti, 2023). WAVN integrates a blockchain where robots earn tokens for sharing imagery among other activities, and the robot with the most tokens wins the right to generate the next block (Paykari et al., 2024). However, faulty or malicious robots can disrupt consensus, risking the integrity of the blockchain and navigation process. In contrast, traditional consensus protocols like Practical Byzantine Fault Tolerance (PBFT) suffer from high message complexity and limited trust in an open network, impractical for open or resource-constrained networks (Castro et al., 1999).

To address this, we introduce the Witness Byzantine Fault Tolerance (WBFT) protocol, a consensus mechanism tailored to combine with the main consensus of WAVN's Blockchain. The WBFT protocol is designed to meet the needs of the WAVN system, where robots must reach consensus on navigation data while tolerating Byzantine faults. The protocol leverages a selected leader through the PoN phase and threshold signatures to optimize performance. WBFT leverages a multi-signatures with aggregation to minimize communication overhead, achieving consensus in two rounds (propose and witness/committed) with $O(n)$ message complexity and tolerating up to ($f \leq \lfloor (n-1)/3 \rfloor$) Byzantine faults. Unlike traditional PBFT (Castro et al., 1999), WBFT uses a predefined leader and aggregated signatures to reduce network traffic, making it suitable for resource-constrained robots.

The WBFT Byzantine fault tolerance method integrates a digital signature-based strategy to achieve consensus in a distributed network, improving the traditional voting mechanisms. In each consensus round, the designated leader, determined by a precomputed schedule in the Robostake component (Paykari et al., 2024), broadcasts a message to all nodes in the network. This message is accompanied by the leader's

[a] https://orcid.org/0009-0005-6793-7417
[b] https://orcid.org/0000-0003-1460-9741
[c] https://orcid.org/0000-0001-9701-5505

Paykari, N., Clark, T., Zain, A., Lyons, D. and Rahouti, M.
Witness Byzantine Fault Tolerance with Signature Tree and Proof-of-Navigation for Wide Area Visual Navigation.
DOI: 10.5220/0013961300003982

In *Proceedings of the 22nd International Conference on Informatics in Control, Automation and Robotics (ICINCO 2025) - Volume 2*, pages 633-644
ISBN: 978-989-758-770-2; ISSN: 2184-2809

digital signature, which allows each node to independently verify both the message's validity and the leader's identity using a predefined cryptographic scheme.

Upon receiving the message, a node validates the leader's signature and the message content. If valid, the node appends its own digital signature to the message, acting as a witness to both the message's correctness and the leader's signature. This doubly-signed message is then rebroadcast to the network. Upon receiving a message with two signatures, subsequent nodes verify both signatures. If the signatures are valid, the node replaces the second signature with its own and rebroadcasts the message. This process creates a chain of signature updates, where each node contributes to the message's validation.

To ensure scalability and efficiency, nodes organize signatures into a Merkle tree structure, with individual signatures forming the leaves. As signatures accumulate, the Merkle tree is updated, allowing nodes to track the number of valid signatures efficiently. Once the number of signatures surpasses a predefined threshold, the message is considered committed and eligible for inclusion in the blockchain. The committed message and its Merkle tree are then broadcast to the network for final inclusion in the blockchain.

This method leverages the Robostake component to precompute the leader for each block, ensuring deterministic leader selection and reducing the risk of leader-based attacks. The approach mitigates various Byzantine faults, including malicious leader behavior, by replacing voting with a signature-based validation and propagation mechanism. It enhances resilience against other attack vectors, such as Sybil attacks and message tampering.

The key contributions of this paper are summarized as follows.

- Propose *WBFT*, a novel consensus protocol tailored for WAVN in GPS-denied, resource-constrained robotic environments.
- Introduce a *Proof-of-Navigation (PoN)* mechanism, a PoS-inspired leader selection algorithm based on robots' navigation reliability and environmental adaptability, ensuring transparent and tamper-proof leader election.
- Leverage a *Merkle tree–based multi-signature aggregation* method that reduces verification complexity from $O(n)$ to $O(\log t)$, enabling efficient consensus validation among robotic agents.
- Achieve consensus in only two rounds (propose and witness) with $O(n)$ message complexity, significantly reducing communication overhead compared to classical PBFT and its variants.
- Demonstrate that WBFT tolerates up to $f \leq \lfloor (n-1)/3 \rfloor$ Byzantine nodes and ensures correctness and liveness in adversarial network settings.
- Provide comprehensive experimental benchmarks comparing WBFT to BLS, aggregate, and threshold signature schemes, and evaluate scalability across varying network sizes.
- Validate WBFT's suitability for high-frequency, decentralized robotic coordination tasks in visual navigation, particularly under bandwidth and resource constraints.

The rest of this paper is structured as follows. Section 2 provides background on the WAVN system, blockchain integration, and the hybrid PoS–BFT consensus mechanism. Section 3 reviews relevant work in Byzantine fault tolerance and consensus protocols. Section 4 details the design of the WBFT protocol, including its cryptographic components and signature aggregation scheme. Section 5 presents experimental results evaluating performance and scalability. Section 6 discusses key insights and limitations, and Section 7 concludes the paper with directions for future work.

2 BACKGROUND AND SYSTEM CONTEXT

To better understand the design rationale behind the proposed protocol, this section presents the key components and operational context of the WAVN system. It outlines the role of collaborative visual localization, blockchain-based incentives, and the hybrid consensus mechanism that combines PoS leader selection with Byzantine fault tolerance. These foundational concepts establish the system requirements and assumptions guiding our methodological design.

2.1 Wide Area Visual Navigation (WAVN)

In WAVN, a group of robots collaborates to guide a robot in a dynamic and GPS-denied environment to a designated location using shared visual imagery. Each robot captures and shares images of its surroundings. Any robot in the team that needs to navigate to a location, identified by a goal image, out of its immediate field of view, leverages the imagery from the team to find a sequence of landmarks seen in common between team members that can be followed to the target location.

2.2 Blockchain

Robots generate transactions that include imagery of the main robot and common landmarks between it and another robot. In the next level, robots compete to generate the next block, which leads to the reward of a constant amount of tokens. The system incentivizes participation through a blockchain where:

- Robots earn tokens for activities such as contributing valid imagery and common landmarks, participating in the network, or generating a block.
- A competition process, as described in (Paykari et al., 2025a), determines the next block generator based on token accumulation.
- The blockchain records imagery and common landmarks contributions, ensuring transparency, immutability, and accessibility.
- In case of navigation needs, a robot uses the ledger and retrieves transactions to generate a sequence of common landmarks from the current position to the home (Paykari et al., 2025b).

2.3 Hybrid Consensus

Whether silent (failing to respond) or Byzantine (sending malicious data), Faulty robots can disrupt navigation or manipulate records. Consensus addresses this by securing the blockchain, ensuring only valid imagery contributions are recorded.

2.3.1 PoS Based Consensus

The PoN system, as outlined in (Paykari et al., 2024; Paykari et al., 2025a), employs a blockchain-based Proof-of-Stake (PoS) mechanism to select a leader for cooperative navigation among robotic teams. Each robot is assigned a stake through a stake weight function, which evaluates its navigation reliability based on factors like historical performance. A PoS consensus score is then calculated, reflecting each robot's influence, with higher scores indicating greater trustworthiness. The robot with the highest consensus score is dynamically chosen as the leader for a given block, coordinating tasks like path planning using verified data stored on the blockchain. This ensures transparent, tamper-proof selection. A navigability function further tailors the stake to the specific environment, providing the leader is best suited for tasks like navigating agricultural fields or dense forestry. The system adapts dynamically, reassigning leadership as conditions or robot performance change, though challenges like faulty leaders or Byzantine behavior remain unsolved.

The leader selection process for the robotic team is managed through the PoS-based PoN mechanism, which is tailored for cooperative navigation Alg. 1. It works in this way:

1. Stake Weight Function:
   - Each robot in the team is assigned a stake based on its navigation reliability and activities. This reliability is determined by factors such as historical performance or contribution to the team's navigation goals.
   - The stake weight function quantifies the portion of the stake by the robot compared to the whole stake, giving a higher chance to robots with more reliable navigational sharing to be selected as the next leader.
2. PoS Consensus Score:
   - The PoN system calculates a consensus score for each robot based on its stake. This score reflects the robot's influence in the decision-making process.
   - Robots with higher consensus scores (i.e., those with higher stakes due to reliable navigation data) are prioritized for leadership roles.
3. Leader Selection:
   - The robot with the highest consensus score at a given time is selected as the leader for the next block generation. The leader is responsible for collecting transactions and broadcasting them as a new block that, along with the collective data verified through the blockchain, will be used for path planning.
   - The blockchain ensures that the selection process is transparent and tamper-proof, as all robots validate the leader's score using the decentralized ledger and the last block.
4. Dynamic Updates:
   - The leader role is not fixed; it can change dynamically as the robots' stakes and consensus scores are updated based on real-time performance and environmental conditions.
   - For example, if a robot's sensors degrade or its navigation data becomes less reliable, its stake and consensus score decrease, reducing its likelihood of being selected as the leader. Even if a robot becomes isolated from others, it has no chance of becoming the next leader.

2.3.2 Byzantine Fault Tolerance

To ensure the integrity and reliability of the WAVN blockchain in the presence of faulty or malicious

robots, the WBFT protocol is integrated with the PoN's Proof-of-Stake (PoS) mechanism to achieve robust consensus. WBFT addresses Byzantine faults, such as robots sending malicious imagery or manipulating navigation data, by leveraging a signature-based consensus mechanism that eliminates the high message complexity of traditional PBFT. Unlike PBFT's, WBFT achieves agreement through a two-phase process (propose and witness) using aggregated Ed25519 signatures and a Merkle tree structure. In the propose phase, the leader, selected via the PoN process based on its consensus score, broadcasts a block containing imagery transactions with its digital signature. Nodes validate this signature and contribute their own signatures to a Merkle tree, where each signature is hashed with the node's identifier to form a leaf, enabling efficient verification with $O(\log t)$ complexity for a threshold $t = \lfloor 2n/3 \rfloor + 1$. This approach tolerates up to $f \leq \lfloor (n-1)/3 \rfloor$ Byzantine faults.

In the witness phase, once the threshold number of valid signatures is collected, the Merkle tree's root is broadcast, allowing all nodes to verify the block's authenticity and the threshold condition independently. The use of Ed25519 signatures ensures fast signing and verification, with compact 32-byte public keys and 64-byte signatures, minimizing network overhead, critical for robots sharing high-frequency visual imagery. The protocol's integration with PoN ensures that only reliable robots with high consensus scores lead the consensus, reducing the risk of malicious leader behavior.

3 STATE-OF-THE-ART

Blockchain technology has emerged as a powerful tool for enabling decentralized trust in robotics and navigation systems (Aditya et al., 2021), particularly in applications like WAVN (Paykari et al., 2024). The PBFT protocol, a cornerstone of BFT, achieves consensus in partially synchronous networks but incurs high communication overhead with $O(n^2)$ messages due to its all-to-all communication in the prepare and commit phases (Castro et al., 1999). Recent BFT variants, such as HotStuff and Tendermint, improve scalability by reducing message complexity and optimizing leader rotation. Yet, they still struggle in resource-constrained environments due to computational demands (Yang and Bajwa, 2019). Threshold signature schemes, like BLS, provide compact signatures but require complex setup phases, which can be impractical for dynamic robotic networks (Boneh et al., 2018). In contrast, our WBFT protocol leverages a lightweight, signature-based consensus mechanism with aggregated Ed25519 signatures and a Merkle tree structure, achieving $O(n)$ message complexity and eliminating the need for complex setup, making it ideal for bandwidth-constrained robotic systems.

BFT remains critical for ensuring reliable consensus in distributed systems despite malicious or faulty nodes, with recent advancements enhancing scalability and adaptability for modern applications like blockchain and cyber-physical systems (CPS). Liu and Junwu (Liu and Zhu, 2024) propose AP-PBFT. This enhanced PBFT framework incorporates a Verifiable Random Function (VRF) to select consensus and primary nodes, ensuring fair proposal aggregation for multi-value consensus in decentralized autonomous organizations (DAOs). An incentive mechanism promotes honest participation, reducing collusion risks and improving efficiency in complex decision-making scenarios. Unlike AP-PBFT's focus on multi-value consensus and incentive-driven participation, WBFT prioritizes a streamlined, signature-based approach using a Merkle tree to achieve efficient verification with $O(\log t)$ complexity, tailored for resource-constrained robotic networks in WAVN.

Wang et al. (Huang et al., 2022) introduce WRBFT, a PBFT variant that uses workload-based node selection and VRF for dynamic primary node assignment, addressing fixed node roles and high communication overhead. By enabling flexible node entry and exit, WRBFT enhances scalability and resilience, making it suitable for large-scale blockchain networks. WBFT diverges by integrating a precomputed leader schedule via the PoS-inspired PoN mechanism and employing aggregated Ed25519 signatures, reducing communication overhead and optimizing performance for robotic systems with limited bandwidth.

Wu et al. (Wu et al., 2023) develop RPBFT, tailored for CPS, with improved master node election, robust malicious node detection, and optimized view changes to reduce communication delays and increase throughput. This method ensures reliable consensus in resource-constrained IoT-enabled blockchain systems while maintaining security against Byzantine faults. In contrast, WBFT uses a Merkle tree-based signature aggregation to achieve efficient verification and incorporates a leader rotation mechanism, making it more suitable for high-frequency, bandwidth-constrained robotic navigation tasks in WAVN.

Byzantine-resilient Distributed Coordinate Descent (ByRDiE) (Yang and Bajwa, 2019) tackles Byzantine faults in high-dimensional distributed learning, enabling robust optimization in decentralized machine learning tasks across convex and non-convex settings. Its focus on data-driven applica-

tions expands BFT's scope beyond blockchain, offering resilience in adversarial environments. WBFT, however, is explicitly designed for blockchain-based robotic navigation, using a signature tree method to ensure efficient consensus with $O(\log t)$ verification complexity, distinct from ByRDiE's optimization-centric approach.

Asynchronous Algorand (Abraham et al., 2025) advances BFT for blockchain systems by achieving near-linear communication complexity and constant expected time, even in dynamic networks with unknown participation. Its asynchronous design overcomes limitations of synchronous BFT protocols in handling network delays and partitions, enhancing scalability. Unlike Asynchronous Algorand's focus on asynchronous consensus, WBFT employs a synchronous, signature-based mechanism optimized for deterministic leader selection and compact signature aggregation, prioritizing efficiency in resource-constrained robotic environments.

The reviewed methods, AP-PBFT, WRBFT, RPBFT, ByRDiE, and Asynchronous Algorand, demonstrate significant strides in addressing scalability, efficiency, and adaptability in BFT protocols across diverse applications. WBFT distinguishes itself by combining a signature-based consensus with a Merkle tree structure and PoS-based PoN leader selection, achieving low communication overhead, tailored for WAVN's robotic blockchain system.

Algorithm 1: PoN leader selection.

Data: Set of robots $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ with navigation histories and performance metrics

Result: Selected leader r^* for the next block generation

foreach $r_i \in \mathcal{R}$ **do**

  Retrieve navigation reliability score s_i based on last committed block (B_{n-1});

  Compute stake weight $w_i = \frac{s_i}{\sum_{j=1}^{n} s_j}$;

  Compute PoS consensus score $c_i = f(w_i, \text{navigability context})$;

Select robot r^* with the highest consensus score c_i;

if *multiple robots share max* c_i **then**

  Break tie using secondary metrics (e.g., recency of successful navigation or token age);

return r^* *as the PoN leader for the current round*;

4 METHODOLOGY

This section details the methodology and structure of the WBFT protocol, providing a comprehensive explanation of its components and their roles in achieving consensus in a distributed system. Each component of WBFT is justified with respect to its contribution to fault tolerance, security, and efficiency. Additionally, we highlight that multiple solutions exist for each component of the protocol, and by substituting or refining these modular elements, the method's performance, scalability, or resilience may be further improved.

4.1 Signatures and Verifications

To ensure secure and efficient message validation in the WBFT protocol, we evaluate three cryptographic methods, including ECDSA-secp256r1 (Kramer et al., 2018), Ed25519 (Bisheh-Niasar et al., 2021), and RSA-2048-bit (Sadikin and Wardhani, 2016) for generating public/private key pairs, creating digital signatures, and verifying them across a distributed network. These methods are compared based on signing time, verification time, and the sizes of public keys, private keys, and signatures, as these metrics directly impact the protocol's performance and network overhead. Keys and signatures are serialized using the DER format for ECDSA and RSA, and raw encoding for Ed25519, ensuring compatibility with network transmission requirements. The evaluation reveals Ed25519 offers the fastest signing and verification with compact key and signature sizes, ideal for bandwidth-constrained networks; ECDSA provides a balanced approach with moderate sizes and performance; and RSA, while secure, incurs higher computational and size overheads. By analyzing these metrics, we identify Ed25519 as a promising candidate for WBFT, with the flexibility to adopt alternative methods based on specific security or performance needs. The results are visualized in Figure 1, highlighting the suitability of each technique for scalable Byzantine fault tolerance.

The WBFT protocol employs the DER format for serializing and deserializing public and private keys for ECDSA and RSA and raw encoding for Ed25519 due to their compact binary representations. For instance, Ed25519 public keys are 32 bytes, ECDSA public keys are approximately 70 bytes in DER format, and signatures are 64 bytes and approximately 70 bytes, respectively. These compact sizes minimize network overhead, ensuring efficient transmission and processing in WBFT's broadcast-heavy consensus mechanism.

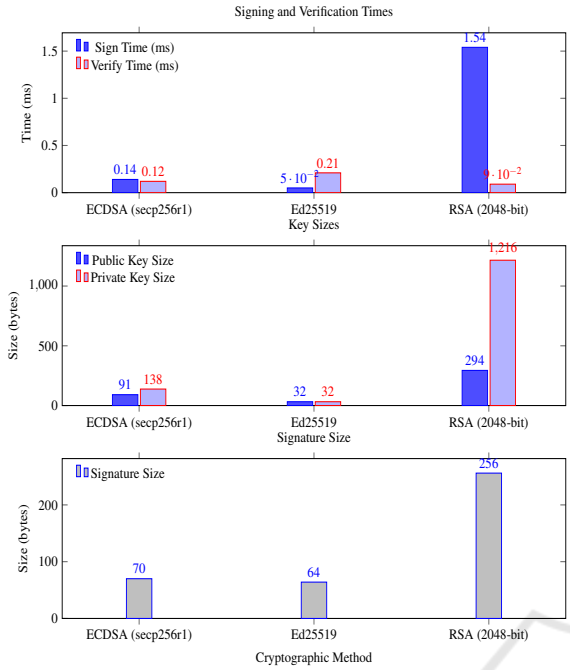


Figure 1: Comparison of ECDSA (secp256r1), Ed25519, and RSA (2048-bit) in terms of signing/verification times and key/signature sizes, demonstrating Ed25519's efficiency.

The Ed25519 signature scheme operates over the Edwards curve defined by $x^2 + y^2 = 1 + dx^2y^2$ over a finite field $\mathbb{F}_p$, where $p = 2^{255} - 19$ and d is a curve-specific constant. For a private key $k \in \mathbb{Z}$, the public key is computed as $A = [k]B$, where B is the curve's base point. To sign a message M, the signer computes a scalar $r = H(k,M)$ (where H is a cryptographic hash function, typically SHA-512), a point $R = [r]B$, and a scalar $s = r + H(R,A,M) \cdot k \mod \ell$, where ℓ is the order of the curve's base point. The signature is the pair (R,s). Verification involves checking if $[s]B = R + [H(R,A,M)]A$, ensuring the signature's validity without revealing k. This scheme's compact 32-byte keys and 64-byte signatures, combined with its fast scalar multiplication, make it a good candidate for WBFT's high-frequency signature broadcasts.

4.2 Signature Tree

A digital signature authenticates the origin and integrity of a message, while a multisignature scheme enables multiple parties to collaboratively sign a message, enhancing security in distributed systems like WBFT. Several multisignature approaches exist, each with unique trade-offs. The BLS (Boneh-Lynn-Shacham) scheme, leveraging bilinear pairings on the BLS12-381 curve, aggregates individual signatures into a compact 48-byte signature, verified via a single pairing operation, though verification time scales with the number of signers (Bacho and Loss, 2022). Aggregate signatures concatenate individual signatures into a single byte string, simplifying verification but resulting in a signature size that grows linearly with the number of signers, making it less efficient for large groups (Bellare and Neven, 2006). Threshold signature schemes require a minimum number of signatures (t-of-n) to validate a message, offering flexibility in signer participation but necessitating individual verification for each signature, which can be computationally intensive (Chu et al., 2023). To address these limitations, we propose a signature tree method that integrates threshold signatures with a Merkle tree structure, enabling efficient verification of multiple signatures in WBFT.

Our signature tree method enhances threshold signatures for the WBFT protocol by integrating them with a Merkle tree structure, offering efficient verification of multiple signatures in distributed systems. In this approach, a leader node broadcasts a message signed by participating nodes to produce individual signatures. Each signature, concatenated with the node's unique identifier, is hashed to form a leaf in the Merkle tree. The tree is constructed by pairwise hashing leaves up to a single root, representing the collective signatures. To validate the message, a node verifies that the Merkle root corresponds to a threshold number of valid signatures, using the associated public keys and Merkle paths. This method reduces verification complexity from $O(n)$ to $O(\log n)$ for n signatures, as only the root and relevant paths need verification, making it highly scalable for large networks. However, constructing the tree introduces computational overhead, particularly for dynamic node sets, which must be balanced against the efficiency gains in verification.

The method leverages the cryptographic properties of Merkle trees to ensure integrity and authenticity in WBFT's consensus mechanism. Requiring a designated leader's signature in the tree ensures a consistent message origin, while witness signatures provide fault-tolerant agreement. The use of a threshold ensures robustness and efficiency, as each robot that meets the threshold faster has the proof for committing and adding the block to the chain. The compact Merkle root, typically 32 bytes (using SHA-256), minimizes network overhead during broadcast, making this approach ideal for WBFT's high-frequency, bandwidth-constrained environment. Compared to BLS and aggregate signatures, the signature tree method offers a balance of scalability and flexibility, particularly suited for scenarios where partial verification of large signer sets is critical. It also

eliminates needing a trusted channel or third party to communicate required keys.

Our multi-signature method is chosen for its compatibility with Ed25519, which offers fast signing and verification, ideal for resource-constrained robots. Unlike BLS signatures, which require a complex setup phase, our method is straightforward, concatenating individual signatures while maintaining verifiability. Compared to ring signatures, it provides stronger accountability, as each signer's contribution is explicit. The method aligns with WBFT's fast path, reducing bandwidth usage in WAVN's visual data exchanges.

4.3 WBFT Protocol

The WBFT protocol is a consensus mechanism tailored for PoN blockchain systems, enabling a group of n nodes to agree on a shared block of blockchain ledger despite the presence of up to $f \leq \lfloor (n-1)/3 \rfloor$ faulty nodes, ensuring resilience in dynamic, resource-constrained environments. WBFT achieves consensus through three main phases: propose, witness, and commit, requiring a threshold of $t = \lfloor 2n/3 \rfloor + 1$ valid signatures to validate an entry, balancing security, scalability, and efficiency. In the propose phase, a designated leader node, selected by the PoN process (see Algorithm 1), initiates consensus by broadcasting a new entry (block) accompanied by the Ed25519 cryptographic signature. This signature ensures the entry's authenticity and integrity, allowing nodes to verify the leader's identity. During the witness phase, participating nodes validate the leader's signature against the leader's public key and, if valid, contribute their own signatures to a Merkle tree, where each signature is hashed to form a leaf. The Merkle tree structure organizes these signatures hierarchically, pairwise hashing leaves to compute parent nodes until a single root is derived, enabling efficient verification with $O(\log t)$ complexity compared to $O(n)$ for traditional threshold schemes.

Once the threshold number of signatures is collected in the commit phase, the fastest reached broadcasts the Merkle tree, allowing independent verification of the threshold condition for others. Each node recomputes the Merkle root using the provided tree and signatures, ensuring the leader's signature is included and the threshold is met, before appending the entry to the blockchain. This will eliminate the need for communication overhead in the commit phase compared to PBFT. A secure key exchange ensures that all nodes share public keys before consensus begins, maintaining network reliability.

Consider a set of n nodes in the WBFT protocol, with a designated leader node and a threshold $t \leq n$ required for block validation. Let B_m be the Block to be signed, and each node i (for $i = 1, \ldots, n$) has a private key $sk_i \in \mathbb{Z}$ and corresponding public key $pk_i = [sk_i]G$, where G is a generator of an elliptic curve group. The leader node signs B_m to produce a signature $\sigma_L = \text{Sign}(sk_L, B_m)$, where Sign is the signing function. Each witness node i (for $i \neq L$) signs (B_m, σ_L) to produce $\sigma_i = \text{Sign}(sk_i, (B_m, \sigma_L))$. These signatures will create leaves $l_i = H(ID||\sigma_j)$, and the leader's leave also includes the block $l_i = H(ID||B||\sigma_j)$.

Algorithm 2: Witness Byzantine Fault Tolerance (WBFT).

Data: Set $N = \{r_1, ..., r_n\}$ with PKs $\{pk_1, ..., pk_n\}$, leader r_L
Result: Committed block B_m
Propose:
r_L creates B_m, $\sigma_L \leftarrow \text{Sign}(sk_L, B_m)$
Broadcast (B_m, σ_L)
Witness:
foreach $r_i \in N \setminus \{r_L\}$ **do**
  Construct Merkle tree $\mathcal{T}$
  Let $\Sigma \leftarrow \{\sigma_L\} \cup \{\sigma_j\}$ (received signatures)
  if $\forall \sigma \in \Sigma$, $Verify(pk_\sigma, B_m, \sigma)$ **then**
    $l_\sigma \leftarrow H(ID_\sigma \parallel \sigma)$ for each $\sigma \in \Sigma$
    $l_i \leftarrow H(ID_i \parallel \sigma_i)$
    Broadcast $(\sigma_i \leftarrow \text{Sign}(sk_i, (B_m, \sigma_L)))$
Commit:
if $|\mathcal{L}| \geq \lfloor 2n/3 \rfloor + 1$ **then**
  $R \leftarrow \text{Root}(\mathcal{T})$
  Broadcast $(R, \mathcal{T})$
  foreach $r_j \in N$ **do**
    if $VerifyMerklePath(R, l_{\sigma_L}, \mathcal{T})$ **then**
      Append B_m to chain
return B_m

For pair signatures σ_i, σ_j, then $p_i = H(\sigma_i||\sigma_j)$ is computed, where H is a cryptographic hash function and $||$ denotes concatenation. The Merkle tree is constructed by organizing the t leaves $\{l_1, \ldots, l_t\}$ (including the leader's leaf l_L) into a binary tree. For a level with leaves l_i, l_{i+1}, as mentioned the parent node is $p_{i,i+1} = H(l_i||l_{i+1})$. This process continues, hashing pairs of nodes until the root $R = H(p_k||p_m)$ is computed, where p_k, p_m are the final nodes at the top level. Verification involves checking that t signatures, including σ_L, are valid using their public keys (i.e., $\text{Verify}(pk_i, B_m, \sigma_i) = \text{True}$) and reconstructing the Merkle root R' from the provided signatures and

their Merkle paths, as expressed in Algorithm 2. The message is valid if $R' = R$ and the leader's signature is included, ensuring authenticity and threshold compliance with $O(\log t)$ complexity for path verification.

5 EVALUATION

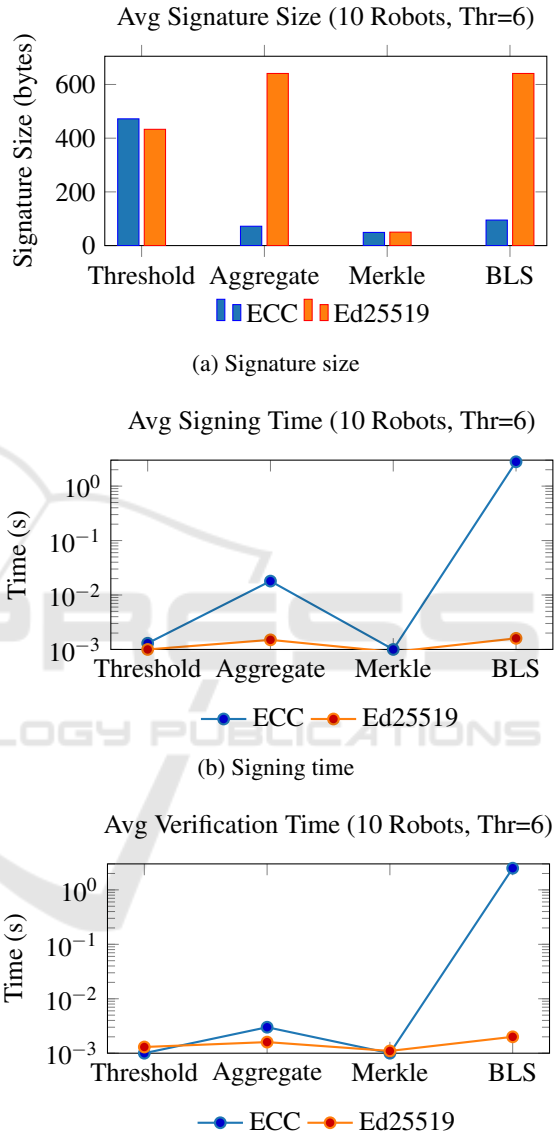
In this section, we evaluate the performance and scalability of the WBFT protocol's signature tree method. We conducted two experiments benchmarking its efficiency against other multisignature schemes and assessing its scalability across varying network sizes. These experiments focus on key metrics such as signing time, verification time, and signature size, which are critical for resource-constrained robotic networks in the WAVN system.

5.1 Experimental Setup

All benchmarks in Sections 5.1–5.3 were conducted in a controlled simulation environment. The WBFT and PBFT protocols were implemented in Python 3.11 using the **TCP sockets** for reliable message passing and threading for concurrent message handling. Cryptographic operations (Ed25519 signing/verification) were performed using the `cryptography` library. Experiments were executed on a workstation equipped with an Intel Core i7-1260P (2.50 GHz), 16 GB RAM, running Windows. To emulate network latency and packet loss, we used the Linux `tc/netem` utility, introducing randomized delays in the range specified for each experiment (e.g., 0–1 s in Sec. 5.3). Each data point reflects the average over 20 independent runs. This setup allowed us to capture both protocol-level message complexity and cryptographic verification overhead in conditions approximating bandwidth-constrained robotic networks.

5.2 Multisignature Scheme Benchmarking

In the first experiment, we benchmarked four multisignature schemes: BLS, aggregate, threshold, and our novel signature tree method, using a network of 10 nodes with a threshold of 6 signatures, over 20 runs. The experiment measured average signing time, verification time, and signature size for each scheme, implemented with elliptic curve cryptography. The signature tree method organizes threshold signatures into a Merkle tree, enabling efficient verification with $O(\log t)$ complexity. Results, visualized in Figure 2, demonstrate that the signature tree method achieves a balance of fast verification and moderate signature size, outperforming BLS in verification speed and aggregate signatures in size efficiency, making it well-suited for WBFT's high-frequency consensus in robotic networks.



(a) Signature size

(b) Signing time

(c) Verification time

Figure 2: Performance comparison of BLS, aggregate, threshold, and signature tree methods in terms of average signature size, signing time, and verification time for 10 nodes with a threshold of 6, over 20 runs, highlighting the signature tree method's efficiency for WBFT.

5.3 Scalability Analysis

In the second experiment, we assessed the scalability of the same four multisignature schemes by varying the number of nodes from 5 to 20, with a threshold of $\lfloor n/2 \rfloor + 1$, over 20 runs. This experiment measured average signing time, verification time, and signature size for ECC and Ed25519 implementations. Results in Figure 3 visualize the signature tree method's superior verification efficiency and moderate signature size across increasing node counts, confirming its suitability for WBFT's distributed consensus in large-scale, fault-tolerant robotic systems.

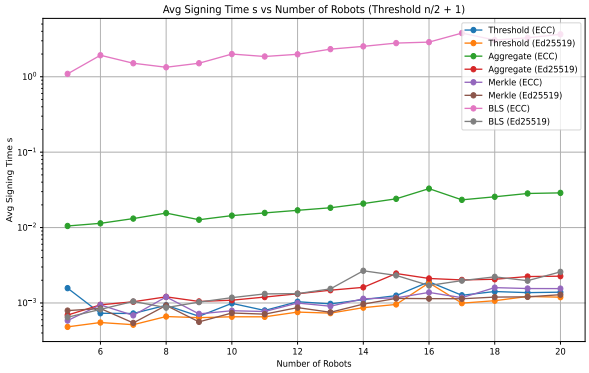
The comparative analysis of WBFT and PBFT based on logs from a four-node system (robot 0 as leader, robot 1, robot 2, robot 3 as followers) reveals distinct performance and resilience characteristics. PBFT demonstrates superior efficiency, achieving an average commit time of approximately 0.0067 seconds (ranging from 0.00277 to 0.01155 seconds across robot 2 and robot 3) compared to WBFT's 0.027 seconds (ranging from 0.01477 to 0.05403 seconds). This performance edge in PBFT stems from its streamlined vote-based mechanism, which minimizes message exchanges by enabling parallel vote processing, unlike WBFT's sequential propose-confirm-commit phases. PBFT's ability to commit 14 entries (0–13) versus WBFT's 12 (0–11) further underscores its robustness, particularly in handling Byzantine faults, such as those injected by the leader and robot 1. However, WBFT's structured approach offers unique strengths, particularly in environments prioritizing deliberate consensus validation.

5.4 WBFT

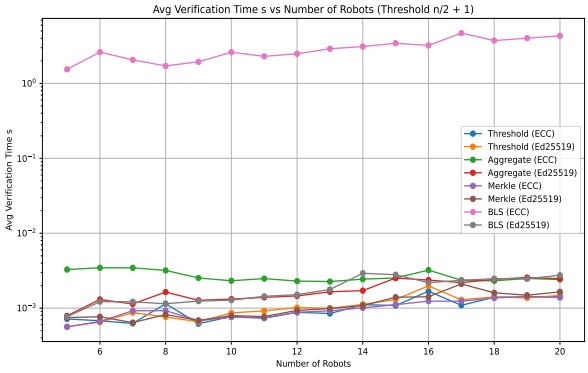
An experiment comparing PBFT and WBFT highlights distinct differences in their performance under identical conditions, focusing on commit times, communication efficiency, and fault-handling mechanisms. Both systems were tested with 7 robots with random delay between 0 and 1, using a threshold of 5 (calculated as $(\lfloor 2n/3 \rfloor + 1)$, where $n = 7$). The WBFT leader log shows the system successfully committing 5 entries with propose times ranging from 0.687 to 1.644 seconds (1.644 is for the Byzantine case) and a total commit time for entry 0 at 0.63519 seconds. Communication metrics indicate 36 proposed messages sent, 26 witnesses received, and 21 committee messages, with only 10 rewrite attempts, demonstrating efficient recovery and consensus finalization. In contrast, the PBFT leader log reports 6 committed entries, with propose times averaging 0.683–0.686 seconds and faster commit times (e.g., 0.08321 seconds for entry 0). However, PBFT required 60 rewrite attempts, suggesting higher overhead in handling message inconsistencies, despite sending 30 propose messages, 25 votes, and 60 commit messages. While PBFT shows slightly faster commit times in optimal scenarios, WBFT's lower rewrite attempts indicate better efficiency in managing communication overhead.

WBFT introduces several strengths over PBFT, particularly in its use of cryptographic mechanisms to enhance security and accountability. By employing Ed25519 signatures at each stage, WBFT ensures that messages are tamper-proof, preventing unauthorized changes during transmission and thwarting attacks like man-in-the-middle. Witnesses are cryptographically linked to the leader's signed proposal, making any attempt at forgery or equivocation immediately detectable. WBFT can handle network issues, even if the leader fails after sending only one proposal, without the risk of messages being altered during the voting/witness phase. Additionally, WBFT's Merkle tree proof mechanism guarantees that all witnesses are accurately reflected in the final commit, allowing nodes to independently verify results without relying on intermediate messages. This contrasts with PBFT, which depends on simpler message counts and integrity checks, lacking the robust cryptographic verification of WBFT. In scenarios where a leader might act unreliably, WBFT's design ensures that even in optimistic cases, a threshold of $(n/2+2)$ nodes can prevent a malicious leader from disrupting consensus, offering a stronger defense against such threats.

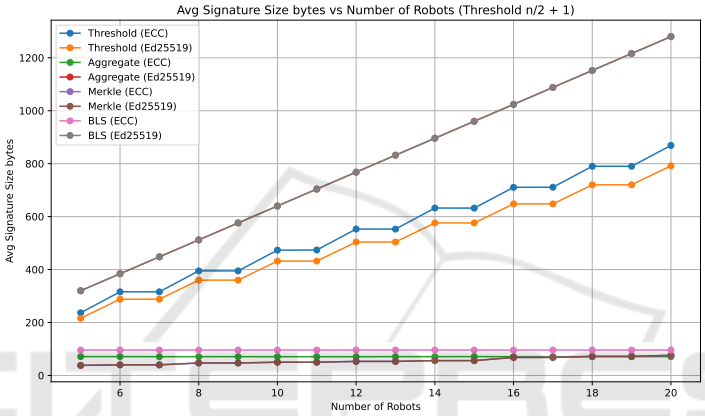
WBFT provides enhanced fault tolerance and recovery mechanisms compared to PBFT. Its dynamic leader transition ensures system liveness during timeouts, maintaining consensus even when delays or failures occur. Public key requests and proof-based commits enable nodes to recover from missing or delayed messages, improving resilience. WBFT also excels in accountability by generating cryptographic evidence to identify and exclude faulty nodes, mitigating risks like forking, where a leader sends inconsistent proposals to different node subsets. PBFT, while effective, is more vulnerable to such forking due to its weaker threshold and lack of signature-based detection, leading to inefficiencies like the observed high rewrite attempts. WBFT's ability to produce verifiable evidence and maintain consensus with fewer retries underscores its superiority in ensuring both security and operational efficiency in distributed systems.



(a) Average signing time



(b) Average verification time



(c) Average signature size

Figure 3: Scalability comparison of BLS, aggregate, threshold, and signature tree methods in terms of average signing time, verification time, and signature size for 5 to 20 nodes with a threshold of $\lfloor n/2 \rfloor + 1$, over 20 runs, demonstrating the signature tree method's scalability for WBFT.

6 DISCUSSION

WBFT protocol introduces a novel signature-based consensus mechanism that significantly enhances the trustworthiness and scalability of distributed consensus in the WAVN system. Our multi-signature method, leveraging a Merkle tree structure, reduces verification complexity from $O(n)$ to $O(\log n)$ for n signatures, offering a substantial improvement over traditional aggregate signatures, which scale linearly with the number of signers (Bellare and Neven, 2006). The use of Ed25519 signatures, with their compact 32-byte public keys and 64-byte signatures, ensures fast cryptographic operations, critical for real-time navigation tasks in resource-constrained robotic networks. This efficiency is particularly vital in WAVN, where robots frequently exchange high-volume visual imagery to construct navigation paths. Additionally, the integration of digital signatures with practical Byzantine tolerance enhances security. The deterministic leader selection via the PoN's PoS mechanism further strengthens the protocol by prioritizing reliable nodes, ensuring transparency and tamper-proof leadership assignment in dynamic environments like agricultural fields or dense forestry (Paykari et al., 2024).

Additionally, in mobile robotic settings, intermittent connectivity and delays are common. WBFT's threshold-based design means that as long as a connected component of size $t = \lfloor 2n/3 \rfloor + 1$ is maintained, consensus can still progress—albeit with higher latency as witness signatures are collected opportunistically. In cases where partitions leave all components smaller than t, WBFT preserves safety (no conflicting commits) but temporarily halts liveness until connectivity resumes. The use of compact Merkle proofs limits the bandwidth cost of retransmissions, while leader rotation provides recovery during extended delays. Thus, under adverse conditions WBFT is expected to degrade gracefully in la-

Table 1: Security comparison of WBFT and PBFT.

Dimension	PBFT (Message Counting)	WBFT (Cryptographic Proofs)
Commit Rule	Relies on counting prepare/commit messages; vulnerable to equivocation	Requires $t = \lfloor 2n/3 \rfloor + 1$ valid digital signatures, embedded in a Merkle root
Fault Detection	Limited; inconsistent views not easily attributable	Misbehavior produces verifiable cryptographic evidence (invalid or missing signatures)
Leader Accountability	Weak; malicious leaders may fork proposals without proof	Strong; equivocation leaves signed evidence traceable to leader
Recovery from Delays	Rewrites and retries; higher communication overhead	Compact witness proofs ($O(\log n)$) allow independent verification
Scalability under Byzantine Behavior	High retries, costly in bandwidth	Safety preserved with fewer retries; evidence enables exclusion of faulty nodes

tency while retaining safety, with future work aimed at adaptive timeouts and opportunistic forwarding to better match the dynamics of robotic mesh networks.

To formalize the contrast between WBFT and PBFT, Table 1 highlights how WBFT's signature-based design strengthens security compared to PBFT's message-counting approach. Whereas PBFT relies primarily on message tallies that can be equivocated without trace, WBFT embeds every commit in cryptographic evidence, enabling both stronger safety and accountability. This design reduces the risk of forks, facilitates faster recovery, and ensures that malicious behavior can be attributed and penalized.

Despite its strengths, WBFT faces several limitations that warrant consideration, such as constructing the Merkle tree, introducing computational overhead, particularly in dynamic networks with frequent node joins or exits, which may impact performance in large-scale deployments or the behavior of the network in case of a byzantine leader or a non-responsive leader.

Future work aims to address these limitations and further enhance WBFT's performance and adaptability. Developing an adaptive fault tolerance mechanism that dynamically adjusts the threshold based on network conditions could improve resilience in asynchronous settings, potentially achieving optimal fault tolerance without sacrificing liveness. Introducing a second or third leader to the markle tree can be another alternative. Exploring sharding techniques to partition the network into smaller consensus groups could reduce communication overhead and enhance scalability, particularly for large-scale robotic swarms. Additionally, optimizing the Merkle tree construction process by leveraging incremental hashing or parallel processing could mitigate computational overhead in dynamic node sets. Integrating lightweight cryptographic alternatives, such as post-quantum signature schemes, could future-proof WBFT against emerging threats while maintaining efficiency. Finally, incorporating machine learning-based anomaly detection to identify and isolate Byzantine nodes in real-time could further strengthen the protocol's security, ensuring robust consensus in adversarial environments. These advancements will enhance WBFT's applicability to a broader range of distributed robotic systems, reinforcing its role as a scalable and fault-tolerant consensus mechanism for WAVN and similar applications.

7 CONCLUSION

This paper proposed the Witness Byzantine Fault Tolerance (WBFT) protocol, a scalable and secure consensus mechanism tailored for Wide Area Visual Navigation (WAVN) in robotic systems. By combining Ed25519 threshold signatures, a Merkle tree structure, and a novel Proof-of-Navigation (PoN) leader selection approach, WBFT achieves efficient consensus with reduced communication overhead and strong Byzantine fault resilience. Experimental results validate WBFT's performance advantages over traditional schemes like PBFT and BLS-based multisignatures, particularly in verification speed, scalability, and fault recovery. While Merkle tree construction introduces some overhead, WBFT remains well-suited for resource-constrained, adversarial environments. Future work will optimize tree construction, explore adaptive thresholds, and enhance resilience using machine learning and post-quantum cryptography, extending WBFT's applicability to broader decentralized robotic networks.

REFERENCES

Abraham, I., Chouatt, E., Gilad, Y., Stern, G., and Yakoubov, S. (2025). Asynchronous algorand: Reaching agreement with near linear communication and constant expected time. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 28–38.

Aditya, U. S., Singh, R., Singh, P. K., and Kalla, A. (2021). A survey on blockchain in robotics: Issues, opportunities, challenges and future directions. *Journal of Network and Computer Applications*, 196:103245.

Bacho, R. and Loss, J. (2022). On the adaptive security of the threshold bls signature scheme. In *proceedings of the 2022 ACM SIGSAC conference on computer and communications security*, pages 193–207.

Bellare, M. and Neven, G. (2006). Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 390–399.

Bisheh-Niasar, M., Azarderakhsh, R., and Mozaffari-Kermani, M. (2021). Cryptographic accelerators for digital signature based on ed25519. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(7):1297–1305.

Boneh, D., Drijvers, M., and Neven, G. (2018). Bls multi-signatures with public-key aggregation. *URL: https://crypto. stanford. edu/dabo/pubs/papers/BLSmultisig. html*.

Castro, M., Liskov, B., et al. (1999). Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186.

Chu, H., Gerhart, P., Ruffing, T., and Schröder, D. (2023). Practical schnorr threshold signatures without the algebraic group model. In *Annual International Cryptology Conference*, pages 743–773. Springer.

Huang, B., Peng, L., Zhao, W., and Chen, N. (2022). Workload-based randomization byzantine fault tolerance consensus protocol. *High-Confidence Computing*, 2(3):100070.

Kramer, M., Gerstmayer, F., and Hausladen, J. (2018). Evaluation of libraries and typical embedded systems for ecdsa signature verification for car2x communication. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, pages 1123–1126. IEEE.

Liu, X. and Zhu, J. (2024). An improved practical byzantine fault tolerance algorithm for aggregating node preferences. *Scientific Reports*, 14(1):31200.

Lyons, D. M. and Rahouti, M. (2023). Wavn: Wide area visual navigation for large-scale, gps-denied environments. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2039–2045. IEEE.

Paykari, N., Alfatemi, A., Lyons, D. M., and Rahouti, M. (2025a). Integrating robotic navigation with blockchain: A novel pos-based approach for heterogeneous robotic teams. *arXiv preprint arXiv:2505.15954*.

Paykari, N., Lyons, D., and Rahouti, M. (2024). Robostake: Pioneering cooperative navigation with a novel blockchain-powered proof-of-stake in robotic teams. In *2024 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–4. IEEE.

Paykari, N., Rahouti, M., and Lyons, D. M. (2025b). A novel blockchain-driven proof-of-stake model for cooperative navigation in visual homing robotic teams. *Distributed Ledger Technologies: Research and Practice*.

Sadikin, M. A. and Wardhani, R. W. (2016). Implementation of rsa 2048-bit and aes 256-bit with digital signature for secure electronic health record application. In *2016 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pages 387–392. IEEE.

Wu, Y., Wu, L., and Cai, H. (2023). Reinforced practical byzantine fault tolerance consensus protocol for cyber physical systems. *Computer Communications*, 203:238–247.

Yang, Z. and Bajwa, W. U. (2019). Byrdie: Byzantine-resilient distributed coordinate descent for decentralized learning. *IEEE Transactions on Signal and Information Processing over Networks*, 5(4):611–627.