MeshGuardian Protocol Specifications

Packet Structure

MeshGuardian Packet Structure

The MeshGuardian protocol uses a modular, secure, and extensible packet structure designed for Delay-Tolerant Networks (DTNs) and challenging connectivity environments.

Packet Segments

Segment	Size	Description
Header	128 bytes	Routing, synchronization, control, encryption flags, and plugin metadata
Payload	Variable	Compressed and encrypted application data
Trailer	96 bytes	Signature + Audit Trail Hash for integrity and traceability

> Example total packet size: **491 bytes**
> Payload: 267B (based on compression)

Header (128 bytes)

Includes:
- `Version` (1B), `Packet Type` (1B), `Priority` (1B)
- `Capability Flags` (4B), `Protocol ID` (1B)
- `Source Node ID`, `Destination Node ID`, `Relay ID` (3 16B)
- `Sequence Number` (4B), `Timestamp` (8B)
- `Hop Count`, `TTL`, `Compression Type`, `Encryption Algorithm ID`
- `Plugin Data` (4B), `Biometric Hash` (32B)
- `Energy Mode`, `Congestion Flags`, `FEC Flags` + `Checksum` (4B)
- `Reserved` (2B padding for alignment)

Payload (Variable Size)

The payload includes:
- Encrypted JSON or binary message
- Optional Forward Error Correction (FEC) block

- Optional redacted fields (PrivacySafePod) for field-level encryption

Payload is compressed (e.g., zlib, Brotli) **before** encryption.

Trailer (96 bytes)

- **Signature (64B):** Ensures authenticity (e.g., Schnorr)
- **Audit Trail Hash (32B):** Used for local + blockchain logging (e.g., Solana Tier 1)

Security & Trust

Each packet is validated using:
- Sequence Number + Timestamp for deduplication
- Signature for zero-trust validation
- Optional blockchain sync via Tier 1 audit events

See Also

- [consensus_engine.md](./consensus_engine.md)
- [synchronization.md](./synchronization.md)

Consensus Engine

MeshGuardian Consensus Engine

The MeshGuardian Consensus Engine ensures decentralized agreement on packet validation, blockchain-backed audit logging, and trust propagationtailored for Delay-Tolerant Networks (DTNs), disaster zones, and interplanetary networks.

Purpose

- Establish global trust in decentralized environments
- Validate packets under zero-trust conditions
- Trigger audit trail writes on consensus events

- Adapt consensus rigor to context (e.g., energy or mission-criticality)

Supported Consensus Mechanisms

Proof-of-Stake (PoS)

- Used for low-priority packets (default)
- Energy-efficient, ideal for IoT and remote nodes
- Leader election based on stake and signal strength

Practical Byzantine Fault Tolerance (PBFT)

- Used for high-priority packets (`Bit 3 = 1`)
- Ensures strong agreement across a quorum of nodes
- Used in critical infrastructure and safety-of-life deployments

Hybrid Mode

- Toggles between PoS and PBFT based on:
  - `consensus_mode_flag`
  - Network density or energy flags
  - Packet priority bits

Validation Flow

1. **Packet arrives** Header is parsed.
2. **Signature checked** If valid, continue.
3. **Consensus mode triggered**
   - If PoS validate via local stake/role
   - If PBFT initiate quorum round
4. **Audit Trail Logging**
   - Tier 1 events blockchain (e.g., Solana)
   - Tier 2 events local + P2P

Packet Fields Involved

- `consensus_mode_flag`: indicates PoS, PBFT, or hybrid
- `pbft_leader_id`: used during quorum rounds
- `priority`: triggers escalation to PBFT
- `capability_flags`: determines consensus plugins enabled

Conflict Resolution

- Uses sequence numbers and logical timestamps (Lamport)
- Deterministic fork resolution with lowest hash + quorum sync

---

Security Features

- Zero-trust validation model
- Quantum-ready signature handling
- Optional post-quantum consensus plugin (future)

Related

- [packet_structure.md](./packet_structure.md)
- [synchronization.md](./synchronization.md)

Synchronization Mechanism

MeshGuardian Synchronization Mechanism

MeshGuardian ensures data consistency across Delay-Tolerant Networks (DTNs) through a robust, energy-aware synchronization engine. This system adapts to scenarios where nodes may be offline for extended periods or connected only intermittently.

Purpose

- Prevent data loss and duplication in disconnected networks
- Enable causal and temporal ordering across async nodes

MeshGuardian Protocol Specifications

- Maintain audit integrity via hash-consistent event chains
- Operate in both terrestrial and interplanetary DTNs

Key Features

- **Nano-Sync Packets**
  Minimal overhead sync messages (~10 bytes) for extreme low-power operation.

- **Event Hashing + Audit Chain**
  All events hashed and stored for traceability. Critical entries optionally anchored to Solana or other chains.

- **Logical Clocks & Lamport Timestamps**
  Ideal for async systems where UNIX time fails (e.g., Mars base vs. Earth).

- **Conflict Resolution**
  Merges conflicting updates via vector clock detection + plugin arbitration.

- **Offline Buffering**
  Nodes store unsent packets with TTL until reconnected.

Use Cases

- First responders syncing bodycam footage in disaster zones
- Mars rover syncing telemetry to Earth during blackout windows
- Remote wildlife sensors buffering environmental data for weeks

---

Relevant Packet Fields

Field	Size	Description
`sequence_number`	4B	For ordering + deduplication
`timestamp`	8B	Logical or real-time clock
`hop_count`	2B	Tracks number of relays
`ttl`	2B	Time-to-live; auto-decrements
`nano_sync_flag`	1B	Activates lightweight sync mode
`audit_trail_hash`	32B	Hash of previous event in chain

MeshGuardian Protocol Specifications

Audit + Integrity

- Tier 1: Critical events Blockchain
- Tier 2: Local log + P2P sync
- Sync checkpoints signed with Schnorr or Dilithium signatures

---

Plugin-Enhanced Sync

- AI-powered merge arbitration (planned)
- Redundancy-aware multi-path re-sync
- Burst-throttle control for congestion collapse

Related Specs

- [packet_structure.md](./packet_structure.md)
- [consensus_engine.md](./consensus_engine.md)