RFC-011: Byzantine Consensus

Status: Proposed **Date:** January 2026 **Author:** Derrell Piper ddp@eludom.net

---

Abstract

This RFC specifies Byzantine fault-tolerant consensus for Cyberspace federation, enabling agreement among distributed vaults even when some participants are faulty or malicious. N nodes tolerate up to f failures where $N \geq 3f + 1$.

Motivation

Federation (RFC-010) assumes honest peers. Reality differs:

- **Crash failures:** Nodes go offline
- **Byzantine failures:** Nodes lie, equivocate, or attack
- **Network partitions:** Messages delayed or lost
- **Sybil attacks:** Fake identities flood the network

Byzantine consensus provides:

1. **Safety:** Honest nodes agree on same value
2. **Liveness:** System makes progress despite failures
3. **Fault tolerance:** Survives f failures with 3f+1 nodes

From Lamport, Shostak, and Pease (1982):

> *The Byzantine Generals Problem: reaching agreement in the presence of traitors.*

Specification

System Model

```
Nodes:        N = 3f + 1 (tolerates f Byzantine faults)
Network:      Asynchronous with eventual delivery
Cryptography: Ed25519 signatures (authenticated channels)
```

Consensus Properties

Agreement: If honest node i decides v, honest node j decides v.

Validity: If all honest nodes propose v, decision is v.

Termination: All honest nodes eventually decide.

Protocol: Practical Byzantine Fault Tolerance (PBFT)

```
Phase 1: PRE-PREPARE
  Primary broadcasts  PRE-PREPARE, v, n, sig

Phase 2: PREPARE
  On valid PRE-PREPARE, broadcast  PREPARE, v, n, sig
  Collect 2f PREPARE messages

Phase 3: COMMIT
  On 2f+1 PREPARE, broadcast  COMMIT, v, n, sig
  Collect 2f+1 COMMIT messages

Decision:
  On 2f+1 COMMIT, decide v
```

Message Formats

```
(consensus-message
  (type pre-prepare)
  (view 0)
  (sequence 42)
  (value-hash "sha512:...")
  (from #${primary-pubkey})
  (signature #${ed25519-sig}))

(consensus-message
  (type prepare)
  (view 0)
  (sequence 42)
  (value-hash "sha512:...")
  (from #${replica-pubkey})
  (signature #${ed25519-sig}))

(consensus-message
  (type commit)
  (view 0)
  (sequence 42)
  (from #${replica-pubkey})
  (signature #${ed25519-sig}))
```

View Change

When primary fails or is Byzantine:

```
1. Replica timeout on PRE-PREPARE
2. Broadcast  VIEW-CHANGE, v+1, prepared-proofs
3. New primary collects 2f+1 VIEW-CHANGE
4. New primary broadcasts  NEW-VIEW, v+1, proofs
5. Resume protocol in new view
```

Application to Cyberspace

Federation Ordering

```
(consensus-propose
  (action release-publish)
  (version "2.0.0")
  (proposer #${alice-key}))

;; After consensus:
(consensus-decided
  (sequence 42)
  (action release-publish)
  (version "2.0.0")
  (decided-by (quorum ...)))
```

Threshold Governance Integration

Combine with RFC-007: - Consensus on *what* to do - Threshold signatures on *authorization*

```
(governance-decision
  (consensus-sequence 42)
  (action deploy-production)
  (threshold-met 3-of-5)
  (signers (alice carol dave)))
```

Optimizations

Speculation

Execute optimistically before commit:

```
Tentative execution after 2f+1 PREPARE
Rollback if COMMIT fails
```

Batching

Amortize consensus over multiple operations:

```
(consensus-batch
  (sequence 42)
  (operations
    (release-publish "2.0.0")
    (release-publish "2.0.1")
    (config-update ...)))
```

Fast Path

When all replicas agree initially:

```
Skip PREPARE phase
Direct to COMMIT with 3f+1 matching responses
```

Security Considerations

Threat Model

Tolerates: – f Byzantine nodes (arbitrary behavior) – Network delays and reordering – Message loss (with retransmission)

Requires: – $N \geq 3f + 1$ total nodes – Authenticated channels (signatures) – Eventual message delivery

Attack Resistance

Attack	Mitigation
Equivocation	Signatures prove inconsistency
Replay	Sequence numbers, view numbers
Denial of service	View change, rate limiting
Sybil	SPKI admission control

Complexity

Metric	Value
Message complexity	O(N^2) per decision
Communication rounds	3 (normal case)
Cryptographic operations	O(N) signatures/verifies

Implementation Notes

State Machine

```
(define-record-type <pbft-state>
  (make-pbft-state view sequence log prepared committed)
  pbft-state?
  (view pbft-view)
  (sequence pbft-sequence)
  (log pbft-log)                ; sequence → messages
  (prepared pbft-prepared)      ; sequence → value
  (committed pbft-committed)) ; sequence → value
```

Dependencies

- `crypto-ffi` – Ed25519 signatures
- `audit` – Decision logging
- Network transport (TCP, QUIC)

References

1. Lamport, L., Shostak, R., & Pease, M. (1982). The Byzantine Generals Problem.
2. Castro, M., & Liskov, B. (1999). Practical Byzantine Fault Tolerance.
3. Yin, M., et al. (2019). HotStuff: BFT Consensus with Linearity and Responsiveness.
4. RFC-007: Threshold Signature Governance
5. RFC-010: Federation Protocol

Changelog

- **2026-01-06** – Initial specification

Implementation Status: Proposed **Fault Tolerance:** f failures with 3f+1 nodes **Protocol Basis:** PBFT