Aerium Consensus Protocol

Formal Mathematical Specification

Aerium Team

https://aerium.network

Protocol Implementation:
https://github.com/aerium-network/aerium/tree/main/consensus

October 13, 2025

Abstract

This document presents the formal mathematical specification of the Aerium consensus protocol, extracted from the TLA+ specification. The protocol is designed to achieve Byzantine fault-tolerant consensus in distributed systems with up to F faulty nodes among N total nodes.

Contents

1 System Parameters

1.1 Constants

$$N : \text{Total number of nodes in the network} \tag{1}$$
$$F : \text{Maximum number of faulty nodes} \tag{2}$$
$$\text{FaultyNodes} \subseteq \{0, 1, \ldots, N-1\} \tag{3}$$

1.2 Threshold Values

Definition 1.1 (Quorum Thresholds)**.** *The protocol uses the following threshold values:*

$$3F + 1 : \textit{Absolute majority (super-quorum)} \tag{4}$$
$$2F + 1 : \textit{Quorum (standard majority)} \tag{5}$$
$$F + 1 : \textit{Minority threshold} \tag{6}$$

1.3 System Assumptions

Invariant 1.1 (Network Size)**.** *The number of nodes must be sufficient to tolerate F faults:*

$$N \geq 3F + 1 \tag{7}$$

Invariant 1.2 (Faulty Node Constraint)**.** *The cardinality of faulty nodes cannot exceed the maximum allowed:*

$$|FaultyNodes| \leq F \tag{8}$$

2 Proposer Selection

Definition 2.1 (Proposer Function)**.** *The proposer for a given round is determined by:*

$$IsProposer(i) \iff round_i \bmod N = i \tag{9}$$

where i is the node index and $round_i$ is the current round of node i.

3 Message Predicates

3.1 PRECOMMIT Vote Predicates

Definition 3.1 (Absolute PRECOMMIT Majority)**.** *A node has received absolute majority of PRECOMMIT votes:*

$$HasPreCommitAbsolute(i) \iff \Big|\Big\{msg \in logs_i : msg.type = \text{“PRECOMMIT”} \wedge msg.round = round_i\Big\}\Big| \geq 3F + 1 \tag{10}$$

Definition 3.2 (PRECOMMIT Quorum)**.** *A node has received quorum of PRECOMMIT votes:*

$$HasPreCommitQuorum(i) \iff \Big|\Big\{msg \in logs_i : msg.type = \text{“PRECOMMIT”} \wedge msg.round = round_i\Big\}\Big| \geq 2F + 1 \tag{11}$$

3.2 Change-Proposer (CP) Pre-Vote Predicates

Definition 3.3 (CP Pre-Vote Quorum). *A node has received quorum of CP:PRE-VOTE messages:*

$$\begin{aligned} \mathit{CPHasPreVotesQuorum}(i) \iff \Big|\Big\{ & \mathit{msg} \in \mathit{logs}_i : \mathit{msg.type} = \text{“CP:PRE-VOTE”} \\ & \wedge \mathit{msg.round} = \mathit{round}_i \\ & \wedge \mathit{msg.cp\_round} = \mathit{cp\_round}_i \Big\}\Big| \geq 2F + 1 \end{aligned} \tag{12}$$

Definition 3.4 (CP Pre-Vote Quorum for Yes). *A node has received quorum of CP:PRE-VOTE messages with value 1 (yes):*

$$\begin{aligned} \mathit{CPHasPreVotesQuorumForYes}(i) \iff \Big|\Big\{ & \mathit{msg} \in \mathit{logs}_i : \mathit{msg.type} = \text{“CP:PRE-VOTE”} \\ & \wedge \mathit{msg.round} = \mathit{round}_i \\ & \wedge \mathit{msg.cp\_round} = \mathit{cp\_round}_i \\ & \wedge \mathit{msg.cp\_val} = 1 \Big\}\Big| \geq 2F + 1 \end{aligned} \tag{13}$$

Definition 3.5 (CP Pre-Vote Quorum for No). *A node has received quorum of CP:PRE-VOTE messages with value 0 (no):*

$$\begin{aligned} \mathit{CPHasPreVotesQuorumForNo}(i) \iff \Big|\Big\{ & \mathit{msg} \in \mathit{logs}_i : \mathit{msg.type} = \text{“CP:PRE-VOTE”} \\ & \wedge \mathit{msg.round} = \mathit{round}_i \\ & \wedge \mathit{msg.cp\_round} = \mathit{cp\_round}_i \\ & \wedge \mathit{msg.cp\_val} = 0 \Big\}\Big| \geq 2F + 1 \end{aligned} \tag{14}$$

Definition 3.6 (CP Pre-Vote Minority for Yes). *A node has received minority threshold of CP:PRE-VOTE messages with value 1:*

$$\begin{aligned} \mathit{CPHasPreVotesMinorityForYes}(i) \iff \Big|\Big\{ & \mathit{msg} \in \mathit{logs}_i : \mathit{msg.type} = \text{“CP:PRE-VOTE”} \\ & \wedge \mathit{msg.round} = \mathit{round}_i \\ & \wedge \mathit{msg.cp\_round} = \mathit{cp\_round}_i \\ & \wedge \mathit{msg.cp\_val} = 1 \Big\}\Big| \geq F + 1 \end{aligned} \tag{15}$$

Definition 3.7 (CP Pre-Vote Split Decision). *A node has received both yes and no CP:PRE-VOTE messages:*

$$\begin{aligned} \mathit{CPHasPreVotesForYesAndNo}(i) \iff & \Big|\Big\{ \mathit{msg} \in \mathit{logs}_i : \mathit{msg.type} = \text{“CP:PRE-VOTE”} \\ & \wedge \mathit{msg.round} = \mathit{round}_i \wedge \mathit{msg.cp\_round} = \mathit{cp\_round}_i \\ & \wedge \mathit{msg.cp\_val} = 0 \Big\}\Big| \geq 1 \\ \wedge & \Big|\Big\{ \mathit{msg} \in \mathit{logs}_i : \mathit{msg.type} = \text{“CP:PRE-VOTE”} \\ & \wedge \mathit{msg.round} = \mathit{round}_i \wedge \mathit{msg.cp\_round} = \mathit{cp\_round}_i \\ & \wedge \mathit{msg.cp\_val} = 1 \Big\}\Big| \geq 1 \end{aligned} \tag{16}$$

3.3 Change-Proposer (CP) Main-Vote Predicates

Definition 3.8 (CP Main-Vote in Previous Round (No)). *A node has received at least one CP:MAIN-VOTE with value 0 in previous CP round:*

$$\begin{aligned} \mathit{CPHasOneMainVotesNoInPrvRound}(i) \iff \Big|\Big\{ & \mathit{msg} \in \mathit{logs}_i : \mathit{msg.type} = \text{“}\mathit{CP{:}MAIN\text{-}VOTE}\text{”} \\ & \wedge \mathit{msg.round} = \mathit{round}_i \\ & \wedge \mathit{msg.cp\_round} = \mathit{cp\_round}_i - 1 \\ & \wedge \mathit{msg.cp\_val} = 0\Big\}\Big| > 0 \end{aligned} \tag{17}$$

Definition 3.9 (CP Main-Vote in Previous Round (Yes)). *A node has received at least one CP:MAIN-VOTE with value 1 in previous CP round:*

$$\begin{aligned} \mathit{CPHasOneMainVotesYesInPrvRound}(i) \iff \Big|\Big\{ & \mathit{msg} \in \mathit{logs}_i : \mathit{msg.type} = \text{“}\mathit{CP{:}MAIN\text{-}VOTE}\text{”} \\ & \wedge \mathit{msg.round} = \mathit{round}_i \\ & \wedge \mathit{msg.cp\_round} = \mathit{cp\_round}_i - 1 \\ & \wedge \mathit{msg.cp\_val} = 1\Big\}\Big| > 0 \end{aligned} \tag{18}$$

Definition 3.10 (CP All Main-Votes Abstain in Previous Round). *A node has received quorum of CP:MAIN-VOTE messages with value 2 (abstain) in previous CP round:*

$$\begin{aligned} \mathit{CPAllMainVotesAbstainInPrvRound}(i) \iff \Big|\Big\{ & \mathit{msg} \in \mathit{logs}_i : \mathit{msg.type} = \text{“}\mathit{CP{:}MAIN\text{-}VOTE}\text{”} \\ & \wedge \mathit{msg.round} = \mathit{round}_i \\ & \wedge \mathit{msg.cp\_round} = \mathit{cp\_round}_i - 1 \\ & \wedge \mathit{msg.cp\_val} = 2\Big\}\Big| \geq 2F + 1 \end{aligned} \tag{19}$$

Definition 3.11 (CP Main-Vote Quorum). *A node has received quorum of CP:MAIN-VOTE messages:*

$$\begin{aligned} \mathit{CPHasMainVotesQuorum}(i) \iff \Big|\Big\{ & \mathit{msg} \in \mathit{logs}_i : \mathit{msg.type} = \text{“}\mathit{CP{:}MAIN\text{-}VOTE}\text{”} \\ & \wedge \mathit{msg.round} = \mathit{round}_i \\ & \wedge \mathit{msg.cp\_round} = \mathit{cp\_round}_i\Big\}\Big| \geq 2F + 1 \end{aligned} \tag{20}$$

Definition 3.12 (CP Main-Vote Quorum for Yes). *A node has received quorum of CP:MAIN-VOTE messages with value 1:*

$$\begin{aligned} \mathit{CPHasMainVotesQuorumForYes}(i) \iff \Big|\Big\{ & \mathit{msg} \in \mathit{logs}_i : \mathit{msg.type} = \text{“}\mathit{CP{:}MAIN\text{-}VOTE}\text{”} \\ & \wedge \mathit{msg.round} = \mathit{round}_i \\ & \wedge \mathit{msg.cp\_round} = \mathit{cp\_round}_i \\ & \wedge \mathit{msg.cp\_val} = 1\Big\}\Big| \geq 2F + 1 \end{aligned} \tag{21}$$

Definition 3.13 (CP Main-Vote Quorum for Abstain). *A node has received quorum of CP:MAIN-VOTE messages with value 2 (abstain):*

$$\begin{aligned} \mathit{CPHasMainVotesQuorumForAbstain}(i) \iff \Big|\Big\{ & msg \in logs_i : msg.type = \text{“CP:MAIN-VOTE”} \\ & \land msg.round = round_i \\ & \land msg.cp\_round = cp\_round_i \\ & \land msg.cp\_val = 2 \Big\}\Big| \geq 2F + 1 \end{aligned} \tag{22}$$

3.4 CP Decide Predicates

Definition 3.14 (CP Decide Vote for Yes). *A node has received at least one CP:DECIDED message with value 1:*

$$\begin{aligned} \mathit{CPHasDecideVotesForYes}(i) \iff \Big|\Big\{ & msg \in logs_i : msg.type = \text{“CP:DECIDED”} \\ & \land msg.round = round_i \\ & \land msg.cp\_val = 1 \Big\}\Big| > 0 \end{aligned} \tag{23}$$

4 State Transitions

4.1 Propose Phase

Definition 4.1 (Propose Transition). *A non-faulty node in the propose state transitions to precommit state:*

$$\mathit{Propose}(i) : state_i = \text{“propose”} \Rightarrow state_i' = \text{“precommit”} \tag{24}$$

If the node is the proposer, it broadcasts a PROPOSAL message.

4.2 Precommit Phase

Definition 4.2 (Precommit Transition). *A non-faulty node in precommit state that has received a proposal sends PRECOMMIT vote:*

$$\mathit{PreCommit}(i) : state_i = \text{“precommit”} \land \mathit{HasProposal}(i) \Rightarrow \mathit{send\ PRECOMMIT} \tag{25}$$

4.3 Commit Conditions

Definition 4.3 (Absolute Commit). *A node commits immediately upon receiving* $3F + 1$ *PRECOMMIT votes:*

$$\mathit{AbsoluteCommit}(i) : \mathit{HasPreCommitAbsolute}(i) \Rightarrow state_i' = \text{“commit”} \tag{26}$$

Definition 4.4 (Quorum Commit). *After the change-proposer phase decides, a node commits upon receiving* $2F + 1$ *PRECOMMIT votes:*

$$\begin{aligned} \mathit{QuorumCommit}(i) : state_i = \text{“precommit”} \land decided_i = \mathit{TRUE} \\ \land \mathit{HasPreCommitQuorum}(i) \Rightarrow state_i' = \text{“commit”} \end{aligned} \tag{27}$$

Definition 4.5 (Commit and Announce). *A node in commit state broadcasts ANNOUNCEMENT:*

$$\mathit{Commit}(i) : state_i = \text{“commit”} \Rightarrow \mathit{send\ ANNOUNCEMENT} \tag{28}$$

4.4 Timeout Transition

Definition 4.6 (Timeout)**.** *A node transitions to change-proposer phase on timeout:*

$$\mathit{Timeout}(i) : \mathit{state}_i = \text{“}\mathit{precommit}\text{”} \wedge \mathit{decided}_i = \mathit{FALSE} \Rightarrow \mathit{state}'_i = \text{“}\mathit{cp{:}pre\text{-}vote}\text{”} \tag{29}$$

5 Change-Proposer Protocol

5.1 CP Pre-Vote Phase

Definition 5.1 (CP Pre-Vote Initial Round)**.** *For the initial CP round (cp_round* $=$ 0*), a node votes based on its PRECOMMIT status:*

$$\mathit{CPPreVote}(i, \mathit{cp\_round} = 0) : \begin{cases} \mathit{vote\ 1\ (yes)} & \mathit{if}\ \neg \mathit{HasPrecommited}(i) \\ \mathit{vote\ 0\ (no)} & \mathit{if\ HasPreCommitQuorum}(i) \\ \mathit{vote\ 1\ (yes)} & \mathit{otherwise} \end{cases} \tag{30}$$

The decision requires:

$$\begin{aligned} \big|\{&\mathit{msg} : \mathit{msg.type} = \text{“}\mathit{PRECOMMIT}\text{”} \\ &\vee (\mathit{msg.type} = \text{“}\mathit{CP{:}PRE\text{-}VOTE}\text{”} \wedge \mathit{msg.cp\_round} = 0)\}\big| \geq 2F + 1 \end{aligned} \tag{31}$$

Definition 5.2 (CP Pre-Vote Subsequent Rounds)**.** *For subsequent CP rounds (cp_round* > 0*), a node votes based on previous main-votes:*

$$\mathit{CPPreVote}(i, \mathit{cp\_round} > 0) : \begin{cases} \mathit{vote\ 0\ (no)} & \mathit{if\ CPHasOneMainVotesNoInPrvRound}(i) \\ \mathit{vote\ 1\ (yes)} & \mathit{if\ CPHasOneMainVotesYesInPrvRound}(i) \\ \mathit{vote\ 0\ (no)} & \mathit{if\ CPAllMainVotesAbstainInPrvRound}(i) \end{cases} \tag{32}$$

Note: The protocol is biased toward 0 when all previous votes abstained.

5.2 CP Main-Vote Phase

Definition 5.3 (CP Main-Vote Decision)**.** *A node transitions from cp:main-vote based on received pre-votes:*

$$\begin{aligned} &\mathit{CPMainVote}(i) : \mathit{CPHasPreVotesQuorum}(i) \Rightarrow \\ &\qquad \begin{cases} \mathit{decided}_i \leftarrow \mathit{TRUE}, \mathit{state}'_i \leftarrow \text{“}\mathit{precommit}\text{”} \\ \quad \mathit{if\ CPHasPreVotesQuorumForNo}(i) \\ \mathit{send\ MAIN\text{-}VOTE}(1), \mathit{state}'_i \leftarrow \text{“}\mathit{cp{:}decide}\text{”} \\ \quad \mathit{if\ CPHasPreVotesQuorumForYes}(i) \\ \mathit{send\ MAIN\text{-}VOTE}(2), \mathit{state}'_i \leftarrow \text{“}\mathit{cp{:}decide}\text{”} \\ \quad \mathit{if\ CPHasPreVotesForYesAndNo}(i) \end{cases} \end{aligned} \tag{33}$$

5.3 CP Decide Phase

Definition 5.4 (CP Decide Transition). *A node in cp:decide state transitions based on main-votes:*

$$CPDecide(i) : CPHasMainVotesQuorum(i) \Rightarrow \begin{cases} send\ DECIDED(1), round_i' \leftarrow round_i + 1, \\ \quad state_i' \leftarrow \text{``propose''} \\ \quad \text{if } CPHasMainVotesQuorumForYes(i) \\ cp\_round_i' \leftarrow cp\_round_i + 1, \\ \quad state_i' \leftarrow \text{``cp:pre-vote''} \\ \quad \text{if } CPHasMainVotesQuorumForAbstain(i) \end{cases} \tag{34}$$

5.4 CP Strong Termination

Definition 5.5 (Strong Termination Condition). *Nodes can terminate the CP phase early under specific conditions:*

$$CPStrongTerminate(i) : \begin{cases} state_i' \leftarrow \text{``precommit''}, decided_i \leftarrow TRUE \\ \quad \text{if } cp\_round_i = MaxCPRound \\ \quad \land HasPreCommitQuorum(i) \\ round_i' \leftarrow round_i + 1, cp\_round_i' \leftarrow 0, \\ \quad state_i' \leftarrow \text{``propose''} \\ \quad \text{if } CPHasDecideVotesForYes(i) \end{cases} \tag{35}$$

6 Safety and Liveness Properties

6.1 Committed State

Definition 6.1 (Committed Proposal). *A proposal is committed when a quorum of nodes announce the same proposal:*

$$\begin{aligned} IsCommitted \iff \exists S \subseteq \{msg \in network : msg.type = \text{``ANNOUNCEMENT''}\} :& \\ |S| \geq 2F + 1& \\ \land \forall msg_1, msg_2 \in S : msg_1.round = msg_2.round& \end{aligned} \tag{36}$$

6.2 Success Property

Theorem 6.1 (Eventual Success). *All non-faulty nodes eventually commit:*

$$\Diamond IsCommitted \tag{37}$$

where ◊ denotes the temporal operator "eventually".

7 Type Invariants

Invariant 7.1 (State Type Correctness)**.** *For all nodes $i \in \{0, 1, \ldots, N-1\}$:*

$$\begin{aligned} state_i.name &\in \{\text{“propose”}, \text{“precommit”}, \text{“commit”}, \\ &\qquad \text{“cp:pre-vote”}, \text{“cp:main-vote”}, \text{“cp:decide”}\} && (38) \\ state_i.decided &\in \{TRUE, FALSE\} && (39) \\ 0 \le state_i.round &\le MaxRound && (40) \\ 0 \le state_i.cp\_round &\le MaxCPRound && (41) \end{aligned}$$

Invariant 7.2 (Message Type Correctness)**.** *For all messages $msg \in network$:*

$$\begin{aligned} msg.type &\in \{\text{“PROPOSAL”}, \text{“PRECOMMIT”}, \text{“CP:PRE-VOTE”}, \\ &\qquad \text{“CP:MAIN-VOTE”}, \text{“CP:DECIDED”}, \text{“ANNOUNCEMENT”}\} \\ & && (42) \\ msg.index &\in \{0, 1, \ldots, N-1\} && (43) \\ 0 \le msg.round &\le MaxRound && (44) \\ 0 \le msg.cp\_round &\le MaxCPRound && (45) \end{aligned}$$

Invariant 7.3 (Commit Correctness)**.** *If a node is in commit state, then:*

$$\begin{aligned} state_i.name = \text{“commit”} \Rightarrow & \left|\left\{msg : msg.type = \text{“PRECOMMIT”} \right.\right. \\ & \left.\left. \wedge\, msg.round = round_i\right\}\right| \ge 2F+1 \\ & \wedge \left|\left\{msg : msg.type = \text{“PROPOSAL”} \right.\right. \\ & \left.\left. \wedge\, msg.round = round_i\right\}\right| = 1 \\ & \wedge\, \forall msg_1, msg_2 \in \{msg : msg.type = \text{“ANNOUNCEMENT”}\} : \\ & \quad msg_1.round = msg_2.round \qquad (46) \end{aligned}$$

Invariant 7.4 (New Round Correctness)**.** *If a node enters a new round (beyond round 0), then the previous round must have received CP:DECIDED votes and no ANNOUNCEMENT:*

$$\begin{aligned} state_i.name = \text{“propose”} \wedge round_i > 0 \Rightarrow & \left|\left\{msg : msg.type = \text{“CP:DECIDED”} \right.\right. \\ & \wedge\, msg.round = round_i - 1 \\ & \left.\left. \wedge\, msg.cp\_val = 1\right\}\right| > 0 \\ & \wedge \left|\left\{msg : msg.type = \text{“ANNOUNCEMENT”} \right.\right. \\ & \left.\left. \wedge\, msg.round = round_i - 1\right\}\right| = 0 \\ & \qquad (47) \end{aligned}$$

8 Protocol Summary

The Aerium consensus protocol operates in phases:

1. **Propose Phase**: The designated proposer broadcasts a proposal.
2. **Precommit Phase**: Nodes that receive the proposal send PRECOMMIT votes.

3. **Commit Decision**:
   - If a node receives $3F + 1$ PRECOMMIT votes, it commits immediately (absolute commit).
   - If a node receives $2F + 1$ PRECOMMIT votes after CP decides, it commits (quorum commit).
4. **Timeout & Change-Proposer**: If a node times out without committing:
   (a) **CP Pre-Vote**: Nodes vote yes/no based on their local state.
   (b) **CP Main-Vote**: Based on pre-votes, nodes send main-votes (yes/no/abstain).
   (c) **CP Decide**: Based on main-votes:
      - If quorum votes yes: move to next round with new proposer.
      - If quorum abstains: repeat CP phase with next CP round.
      - Strong termination allows early exit under specific conditions.

9 Conclusion

The Aerium consensus protocol provides Byzantine fault-tolerant consensus with a change-proposer mechanism that ensures liveness even in the presence of faulty proposers. The protocol guarantees safety through quorum-based voting and achieves liveness through the change-proposer sub-protocol with strong termination conditions.