Phase

A Mobile-First Layer-1 Blockchain

with Reputation-Based Consensus and a Mobile Proxy Network for AI Agents

Phase Core Team research@phase.xyz

phase.xyz

Version 6.1 — October 2026

Important Legal Notice. This document is provided for informational purposes only. It does not constitute an offer or solicitation to sell or purchase any security, financial instrument, or investment product in any jurisdiction. The PHASE token is a utility token intended for use within the Phase network as described herein. This whitepaper does not constitute investment advice or a prospectus of any kind. Recipients are solely responsible for their own assessment of the information herein. Participation in the Phase network involves significant risk, including the risk of total loss. Please read Section 20 carefully.

This document describes technology and mechanisms that are under active development. Architectural decisions described herein, including consensus parameters, economic parameters, and protocol mechanics, may change materially before mainnet launch. Statements about future capabilities or performance are forward-looking and subject to material uncertainty.

The PHASE token, if and when issued, will be issued only pursuant to separate offering documentation and applicable regulatory requirements in each relevant jurisdiction. Nothing in this whitepaper creates any obligation on the part of Phase or any affiliated entity to issue tokens or complete any described transaction.

Copyright © Phase Core Team 2026. All rights reserved.

Contents

1 Non-Technical Summary **5**

2 Introduction **6**

3 System and Threat Model **8**

3.1 Protocol Overview 8

3.2 Observable Data 10

3.3 Adversary Model 10

4 Reputation Scoring **12**

4.1 Beta-EMA Reliability Filter 12

4.2 Stress Weighting 14

4.3 LCB Scoring and Composite Score 15

4.4 Tracking Claim 16

5 Committee Selection **18**

5.1 Two-Tranche Selection via Thompson Sampling 18

5.2 Probation, Queue, and Churn Limit 20

6 Reward Mechanism **22**

6.1 QC-Based Reward Split 22

6.2 Finality-Gated Release 22

6.3 Tentative-Block Rewards: Fully Withheld 24

6.4 Provable Fault: Penalties 24

6.5 Adaptive Leader Bonus 25

7 Deterrence Without Stake Bonds **26**

7.1 The Earning Position as Collateral 26

7.2 Penalty Decomposition 26

7.3 One-Shot Sufficiency Condition 27

8 Coalition Considerations **28**

8.1 Correlation-Aware Scoring and Fault Attribution 29

8.2 Occupancy and Committee-Share Limits 30

8.3 Coalition Deterrence: Honest Assessment 30

9 Registry-Size Control **31**
9.1 Quorum Probability: Poisson-Binomial . . . 31
9.2 Soft-Cap Design Objective . . . 32

10 Tentative Blocks **32**
10.1 Protocol . . . 32
10.2 State Execution on TC Blocks . . . 32
10.3 Probabilistic Safety: Assumptions and Limitations . . . 35

11 Networking **38**
11.1 Transport: iroh over QUIC . . . 38
11.2 One connection per peer, demultiplexed by stream tag . . . 39
11.3 Beacons: signed-message relays for committee traffic . . . 39
11.4 Bandwidth engineering . . . 41

12 State Bootstrap: Snapshot Synchronization **44**
12.1 Export . . . 44
12.2 Manifest selection by quorum vote . . . 44
12.3 Verified parallel download with resume . . . 45
12.4 Checkpointed import and tail catch-up . . . 45
12.5 Degradation and trust model . . . 45

13 Execution Layer **46**
13.1 A Reth-based execution client . . . 46
13.2 Consensus–execution separation in a single process . . . 47
13.3 Validator registration as a system transaction . . . 48
13.4 Protocol system contracts . . . 49

14 The Proxy Layer **49**

15 Validator Pools **52**
15.1 What a pool is and is not . . . 52
15.2 Mechanics . . . 52
15.3 Income smoothing . . . 53
15.4 Incentive notes . . . 53

16 Tokenomics **54**
16.1 The PHASE Token . . . 54
16.2 Supply and Emission . . . 54
16.3 Fees and Burns . . . 55
16.4 Validator Economics . . . 56
16.5 Allocation and Vesting . . . 57

17 Governance **57**
17.1 Governance Architecture . . . 58
17.2 Governance Capture Analysis . . . 59
17.3 Governance Roadmap . . . 59

18 Security Analysis Summary **60**
18.1 Security Games Covered by the Model . . . 62

19 Calibration and Attack-Surface Testing **63**
19.1 Methodology . . . 63
19.2 Consensus Attack-Surface Tests . . . 63

20 Risks **64**
20.1 Protocol Risks . . . 64
20.2 Economic Risks . . . 65
20.3 Legal and Regulatory Risks . . . 66
20.4 Operational Risks . . . 66
20.5 Environmental Considerations . . . 67

21 Day-0 Bootstrapping **67**

22 Open Problems **68**
22.1 Base-Chain and Scoring Open Problems . . . 68
22.1.1 Non-provable censorship. . . . 68
22.1.2 Coordinated soft failure below detection threshold. . . . 68
22.1.3 Endogenous censoring in the tracking claim. . . . 68
22.1.4 Leader grief attack. . . . 68
22.2 Design-Level Open Problems . . . 68
22.2.1 Formal safety of the tentative-certificate mode. . . . 68

22.2.2 Coalition deterrence without capital bonds. 69
22.2.3 Committee sampling from a larger registry. 69
22.2.4 Selection randomness. 69

23 Conclusion **70**

Appendices **72**

A Notation Reference **72**

B Proof Sketch: Claim 4.1 (Beta-EMA Tracking) **74**

C Heuristic Bound: Claim 10.1 (Tentative Fork Risk) **76**

D Proof: Lemma 5.1 (Sybil Occupancy Bound) **76**

E Sketch: Claim 18.1 (Heuristic Manipulation Bound) **77**

F Leader Grief Analysis **77**

1. Non-Technical Summary

What is Phase? Phase is a blockchain designed to run on smartphones. Most blockchains require expensive server hardware or large capital deposits to secure the network. Phase requires neither. Participation is open to anyone with a smartphone, and trust is earned through demonstrated reliability rather than upfront capital.

The problem

More than six billion smartphones are active globally. In many regions, the smartphone is the only computing device most people own and the primary gateway to financial services. Yet block production remains concentrated in data centers, gated by hardware cost and capital requirements. The infrastructure does not match the distribution of the people it is meant to serve.

How Phase works

Phase tracks whether each validator shows up, votes on time, and behaves honestly. That track record feeds into a score, and the score determines who gets selected for committees and earns rewards. A validator who misbehaves is jailed and their reputation is reset, wiping out the economic value accumulated through prior honest participation — the penalty the design turns on, though in the current release the cryptographic cases of misbehavior are only logged, on whichever node saw both messages, and no penalty follows (§6.4).

This makes the cost of misbehavior scale with tenure. A validator's score is an earning position: it determines selection into reward-bearing committees, it cannot be bought or transferred, and the design intends it to take months of honest participation to build and to be slow to rebuild — a calibration target rather than the shipped behavior: under the shipped scoring parameters, which are uncalibrated placeholders, a score holds only about ten views' worth of discounted evidence (Appendix B). Jailing plus a reputation reset is designed to destroy that position outright, and because rewards are released only after the blocks they relate to are final — in the implementation, once a later proposer's committed watermark reaches them (§6.2) — a misbehaving validator is also designed to forfeit income not yet paid out. The longer a validator has participated honestly, the more they stand to lose.

What Phase is not

Phase is not a Proof-of-Stake network. Validators do not lock up large token deposits. This is deliberate: requiring large deposits would exclude smartphone users and reproduce the capital-concentration problem Phase is designed to avoid. Phase is also not Proof-of-Work; replacing stake with specialized hardware and energy spend would be just as inconsistent with a mobile-first network. Registration is open and free, and the selection mechanism filters for reliability using observed on-chain behavior. A validator with no history is not shut out. It registers, waits its turn in an activation queue, and enters on probation, where two observer seats per committee are reserved for probationers — they sign, and are scored

and paid for it, but their signatures count toward no quorum — so that a newcomer can accumulate the observations its score is built from; the route is deliberately narrow — only those two seats are reserved, and a probationer that has not graduated within a bounded number of periods returns to the back of the queue — but once it has enough observations, at a high enough measured reliability, it graduates to full membership and becomes eligible for the graduate voting seats — at most $\lfloor (a-1)/2 \rfloor$ beside the a genesis anchors, which are seated in every committee — and for leading a view. Committee selection additionally specifies an exploration path, but that path is for validators that have already graduated and whose records are still thin, not for newcomers, and in the software as shipped it is switched off and fills no seats today (§5).

The system beyond consensus

Consensus and its incentive layer are the analytical core of this paper, but they are not the whole system. The whitepaper also specifies the engineering substrate Phase runs on: a mobile-native networking stack built entirely on iroh, a snapshot-based bootstrap that lets a phone join the network without replaying chain history, an EVM execution layer built on Reth, an agent-facing mobile proxy layer metered on-chain whose revenue is designed to flow to participating validators, and native validator pools with a protocol-level referral primitive. These components are specified in §§11–15.

Honest about what is proved

This whitepaper contains formal mathematical analysis of Phase's security mechanisms. Some results are proved under stated assumptions, such as the Sybil occupancy bound; others — including single-validator deterrence — are proof sketches or design targets, and each claim is labeled with its status throughout (see the summary table in §18). Mechanisms that are specified but not yet active at launch are described as design; most of them are activated progressively as the network accumulates live data, but not all — the attestation bucket cap waits on a Sybil-resistant identity primitive, stress weighting on a consensus-measurable participation signal for failed views, and the correlated-fault jail escalation on misbehaviour proofs — and the text says which, where each mechanism is described.

2. Introduction

Proof-of-Stake blockchains tie validator eligibility to bonded capital [2]. This design is effective in server-grade environments, but creates a structural barrier: participating in block production requires capital that many users, particularly in regions where smartphones are the primary computing platform, cannot commit. Proof-of-Work avoids token bonds, but replaces them with hardware, energy cost, and operational concentration; that tradeoff is also misaligned with Phase's mobile-first objective.

Phase takes a different approach. **Validator participation is gated by demonstrated reliability, not locked capital.** This is an architectural choice, not a technical limitation.

Stake bonds are compatible with smartphones; Phase chooses not to require them because doing so would reproduce the capital-concentration problem the protocol is designed to avoid.

The central claim of this paper is that reputation-based deterrence can secure a BFT consensus protocol without capital bonds, provided three conditions hold:

1. Identity is Sybil-resistant at registration, via app-store device attestation;
2. The reputation filter is statistically calibrated to mobile-grade, non-stationary reliability data;
3. The economic value at risk upon a provable fault — comprising the discounted future income stream of an established validator, rewards withheld until the finality gate of §6.2 releases them, and the reputation position destroyed by jailing and reset — is large enough to dominate any single-shot bribe.

Condition (1) is tractable with current platform tooling. Condition (3) is the focus of the reward and deterrence design in §§6–7. Condition (2) is the hardest: non-stationary Thompson-sampling theory with Beta priors has limited published results, and the proofs that touch this area should be read as well-motivated claims rather than settled theorems.

Contributions

- A Beta-EMA scoring system for tracking time-varying validator reliability from consensus-observable data only, with a composite gating factor $\Pi(i,t)$ integrating anti-Sybil bucket controls (§4).
- Anchor-bounded committee selection: the genesis anchors seated whole, at most $\lfloor (a-1)/2 \rfloor$ graduate seats filled by a score-weighted exploit tranche and a Thompson-sampling explore tranche, and two observer seats that give probationers the committee exposures they need to graduate without a vote that counts (§5).
- A deterrence analysis without stake bonds, stated as a sufficient condition with an explicit parametric bound on jail duration: a provable fault is designed to trigger jailing, a reputation reset, and forfeiture of rewards not yet released at finality — in the implementation, not yet reached by a later proposer's committed watermark (§6.2) — so the earning position itself is the validator's skin in the game (§7; Remark 6.2 states how much of that stack the implementation has).
- A correlation-aware fault-attribution mechanism that escalates time-denominated penalties for clustered provable faults, together with an honest assessment of what income-stream deterrence can and cannot provide against coalitions (§8).
- A proof sketch for the EMA-discounted LCB, with boundary conditions stated explicitly (§4).
- A probabilistic safety analysis for tentative blocks under mobile churn (§10).

- A mobile-native networking stack built entirely on iroh, engineered for smartphone connectivity conditions (§11).
- A snapshot-first state bootstrap that lets a phone join the network in minutes without replaying chain history (§12).
- An on-chain-metered mobile-IP proxy layer serving agent traffic, with revenue designed to flow to participating validators (§14).
- A protocol-native pool and referral primitive for validator aggregation (§15).
- Full tokenomics, governance, and risk sections (§§16–20).

3. System and Threat Model

3.1 Protocol Overview

Phase runs chained HotStuff [1] with a rotating validator committee in discrete *views* $v \in \{0, 1, 2, \ldots\}$. Each view has one leader who proposes a block. If at least q_{full} of the n committee members sign on time, the leader aggregates their BLS signatures into a Quorum Certificate (QC). Block k is *finalized* when the QC for block $k{+}2$ forms, via the chained 3-chain rule; finality therefore costs two further certificates, not a fixed interval of wall-clock time. No minimum inter-block interval is enforced: a leader requests a block as soon as it enters a view and a view closes as soon as its certificate forms (the aggregator may hold a formed full quorum open for at most 150 ms so that a distant member's late vote still counts), so the realized cadence is bounded by certificate latency and by load, not by a target. The 1 s block time in Table 1 is a nominal accounting unit — the rate at which the emission schedule of §16.2 converts blocks into calendar time — and not a cadence the protocol paces to: a twelve-node development fleet measured 946 blocks in 120 s (7.9 per second) on a sustained window and 7.2–10.3 per second over shorter ones, so finality there is reached well inside one second. Only the upper bound is enforced: a view that fails to certify times out after 5 s, and that timeout doubles per consecutive failed view to a ceiling of 40 s.

The voting committee is $n = a + g$, where a is the number of genesis anchors and $g \le \lfloor (a-1)/2 \rfloor$ the seated graduates (§5), under a seat budget of 16 (32 is the wire-format ceiling); at the seven mainnet anchors $n \in [7, 10]$, $f \in [2, 3]$ and $q_{\text{full}} \in [5, 7]$. The quantitative sections below illustrate their analysis at the wire-format ceiling, $n = 32$, $f = 10$, $q_{\text{full}} = 22$: an illustrative large committee, not a configuration the network is moving toward. Under the anchor bound $n \le a + \lfloor (a-1)/2 \rfloor$, so the seven mainnet anchors seat at most ten voters whatever the seat budget, and seating 32 would take both a 32-seat budget (a client release) and at least 22 anchors, which only a hard fork adds (§21, §22). The quorum is derived as $\lfloor (n+f)/2 \rfloor + 1$ with $f = \lfloor (n-1)/3 \rfloor$, which is the threshold that keeps any two quorums intersecting in at least $f{+}1$ members; note this is *not* $2f{+}1$ except at sizes satisfying $n = 3f{+}1$.

Two limits of the current implementation are worth stating here rather than leaving to

be inferred. First, 32 is a ceiling of the wire format, not a tuning parameter: the block format bounds its validator array at 32 entries, and the signer bitfield carried by every certificate is 32 bits wide, so seating more than 32 validators is a block-format change. The seat count itself is a separate constant of the node software, set to 16 in the current configuration; it may be set below the ceiling without a format change — a 16-member committee simply fills 16 of the 32 slots — but cannot grow past it, and changing it on a chain that already carries history additionally requires scheduling the change on an epoch boundary, since two nodes disagreeing about the seat count verify identical vote preimages under different memberships. Second, the seat count bounds the voting population, and with it where deterministic safety holds. Deterministic safety holds while the anchor set is seated whole, whatever the registry size: any two quorums share an anchor, and finality survives up to $2(q_{\text{full}} - g) - a - 1$ compromised anchor keys. The certificate-verification condition $2q_{\text{full}} > |F_t| + f_{\text{reg}}$ over the voting population — the full rows competing for voting seats; probation rows never enter it — is needed only where the anchors are not seated whole (more anchors than seats, or no anchor left); there the certificate paths log the violation and certify anyway, deliberately, because refusing turned a committee going dark into a permanent halt — the boundary pass that would reseat it runs during block execution, which a chain that has stopped committing never reaches. Admission is clamped at a hard ceiling of 4096 eligible validators ($M_{\max}$, Remark 5.2), orders of magnitude above the seats, so that graduates can queue and be drawn; under the anchor bound that bench sits inside the model. Outside the model two conflicting blocks can both be certified and a block reported final can be reverted, after which the node converges on the strongest branch (§10); the coalition arguments in §8 that rest on a large voting population describe a regime the anchor bound forbids by design, and §22 records what remains open.

Consensus orders transactions for an EVM execution layer built on Reth; the execution architecture is specified in §13.

Table 1: Core Protocol Parameters (current configuration)

Parameter	Value	Description
n	$a + g \le 16$; 7–10 at seven anchors	Voting seats per view: a anchors plus $g \le \lfloor (a-1)/2 \rfloor$ graduates; 16 is the seat budget, 32 the wire-format ceiling
f	2–3 at seven anchors	Byzantine faults tolerated, $\lfloor (n-1)/3 \rfloor$
q_{full}	5–7 at seven anchors	Full quorum, $\lfloor (n+f)/2 \rfloor + 1$
q_{tent}	$f + 1$ (3–4 at seven anchors)	Static tentative floor over the voting seats; the activation band is searched (§10)
Leader rule	Round-robin	Full members only (§5)
Total supply	100,000,000,000	Fixed, PHASE (§16)
Block reward	158.55 PHASE	Linear, 10 years nominal (§16)
Block time	1 s nominal	Accounting unit only; no enforced floor
Finality	3-chain	Block k final when QC for k+2 forms

3.2 Observable Data

We model randomness on $(\Omega, \mathcal{F}, \mathbb{P})$ with history σ-field $\mathcal{F}_v := \sigma\big((C_u, L_u, S_u)_{u \le v,\, L_u = 1}\big)$, where C_u is the committee, $L_u \in \{0,1\}$ the QC-success indicator, and $S_u \subseteq C_u$ the QC signer set when $L_u = 1$.

Definition 3.1 (Consensus-measurable)**.** *A quantity is* consensus-measurable *if it is $\mathcal{F}_v$-measurable for some v: every full node computes it identically from shared chain history.*

A key constraint throughout is that all scoring affecting committee eligibility must be consensus-measurable. Network-local signals such as heartbeat timing or RTT may be useful for peer routing, but cannot enter the scoring system because different nodes would observe different values for the same validator.

- **Consensus-observable:** QC signer bitmaps, view numbers, leader identity.
- **Network-local (non-shared):** heartbeat timing, RTT, ϕ-accrual scores. Excluded from scoring.

Remark 3.1 (Scoring inputs are bound to the justification)**.** *The block proposer writes the scoring inputs — the signer set, the non-signers, and the leaders whose views timed out — into the block body, and every node reads them verbatim; the header commits to them (a scoring hash, checked wherever a body is admitted), so every node holds the same inputs. The signer section is no longer a proposer observation: it must equal the signer partition of the justifying certificate the proposal carries, decoded against the committee that certificate verified against, and every voting replica recomputes it from that certificate and refuses a proposal whose section differs. A tentative, genesis or boot-anchor justification carries no partition, and the section must then be empty. The charge list may name at most as many leaders as views the justification skipped, and only validators the registry marks non-probation.*

This binds S_v to a certificate rather than to a proposer's choice, with stated residuals. (a) The aggregator of a certificate chooses which genuine quorum of the votes it holds to justify with, so a narrower quorum may charge real seated signers a miss; it cannot credit a non-signer or pay an address outside the certificate's signers. (b) A tentative justification over the parent suppresses that height's signer section. (c) Within the views the justification skipped, any full validator may be charged. The check runs where a vote is decided, so a section that fails it is certified only by an adversary holding at least the tentative threshold of seats, outside Assumption 3.1. $X_{i,v}$ and the Case B leader charge of §4 inherit these residuals; leader-failure observations remain weaker still, since they originate in a view timer, a network-local signal of exactly the kind this section excludes from scoring.

3.3 Adversary Model

Assumption 3.1 (Byzantine fraction)**.** *At most f of the n voting seats are Byzantine per view: $f \in [2,3]$ of $n \in [7,10]$ at the seven mainnet anchors, which under the anchor bound means at most $f - g$ compromised anchor keys beside hostile graduates; the quantitative analysis below is*

illustrated at the wire-format ceiling, $f = 10$ of $n = 32$, a committee the shipped configuration cannot seat (§3).

Assumption 3.2 (Bribe model)**.** *An adversary may offer validator i a bribe B_i to commit a* provable deviation *(double-signing, equivocation). Provable deviations are detectable by any full node from cryptographic evidence.* As implemented, detectability is partial and node-local: a node detects only within the messages delivered to it — on the point-to-point vote path an equivocator that splits its two conflicting votes between two leaders is caught by neither — and although each report carries both signed statements as self-verifying evidence, that evidence stays on the detecting node, kept in its data directory and served to its operator over IPC only, and is neither sent to other nodes nor written to chain state (Remark 6.2). *This assumption does* not *cover censorship, selective delay, or other non-provable strategic behaviors; those are addressed via mechanism design rather than formal deterrence proofs.*

Assumption 3.3 (Sybil resistance)**.** Planned work: the current release performs no attestation of any kind, so this assumption describes the intended registration gate and every result that invokes it is conditional on work not yet shipped (Remark 3.2). *Each physical device maps to at most one registered identity per bucket, enforced by App Store / Google Play attestation at registration. Concretely, Apple App Attest and Google Play Integrity attest that a registration request originates from an unmodified app running on genuine hardware; the protocol uses the resulting device-bound key as the bucket identity. Each bucket is capped at* $\mathrm{Cap}_{\mathrm{bucket}}$ *eligible registrations per period. This is an* operational assumption*, not a cryptographically proved result. Its strength depends on platform attestation integrity, which is outside the protocol's control. Genuine device farms, emulator bypasses, and account-farm attacks remain possible, so attestation is intended to be paired with bucket caps, probation, churn limits, and minimum observations. Probation, the churn limit and the minimum-observation bar are active today; the bucket cap is not, and switching it on waits first on selecting and building a Sybil-resistant identity primitive, with calibration against live centralisation data only after that (Remark 3.2).*

Remark 3.2 (What the implementation enforces today)**.** *No device-attestation primitive — App Attest, Play Integrity, DeviceCheck or any other — exists in the implementation, and the registration transaction carries no attestation field. A registration is admitted on four checks, none of them an attestation: at most one registration per address per block, a lease that has expired or is inside its renewal window, a well-formed endpoint identifier, and a BLS proof-of-possession binding the consensus key and the endpoint identifier to the registering address. What* is *built is the enforcement half of the bucket mechanism: committee selection groups the epoch's frozen snapshot rows by a bucket identity and drops the surplus identities of any over-cap bucket, across both the exploit and explore tranches. Two gaps separate that from Assumption 3.3. First, the implemented* $\mathrm{Cap}_{\mathrm{bucket}}$ *caps* committee seats per epoch*, not eligible registrations per period — the quantity Lemma 5.1 bounds; no registration-side per-bucket cap exists. Second, nothing assigns shared bucket identities, so the snapshot writer defaults each row's bucket identity to the validator's own address: every validator is its own singleton bucket, and the cap would exclude nobody even if it were switched on — and its shipped value is zero, which disables the check outright. Activation therefore waits first on selecting and building a Sybil-resistant identity primitive; calibration against live centralisation*

data is the step after that, not the one that blocks.

Assumption 3.4 (Drifting reliability)**.** *Each validator i has latent true reliability $\theta_{i,t} \in [0,1]$ satisfying $|\theta_{i,t+1} - \theta_{i,t}| \leq \delta$ per period.*

Assumption 3.5 (Single-deviation, certain detection)**.** *The deterrence results in §7 assume a single validator considers a single provable deviation, detected with probability 1. Results weaken materially if detection is probabilistic, if multiple validators coordinate, or if the deviation yields recurring off-chain rents.*

4. Reputation Scoring

4.1 Beta-EMA Reliability Filter

The scoring system models each validator's reliability as a latent probability $\theta_{i,t}$, observes whether they vote on time each round, and maintains an on-chain estimate that updates with each observation. The Beta distribution is natural here: it is defined on $[0,1]$, updating after a success or failure is trivial, and its shape reflects how confident the protocol is in the estimate. A wide distribution means little evidence; a narrow distribution means a well-characterized validator.

The EMA discount λ reflects the reality that validator connectivity changes over time. Without discounting, a validator's score would be locked by early history. With it, recent performance matters more, and recovery from a bad patch is possible.

For each validator i and scoring period t, we maintain $(\alpha_{i,t}, \beta_{i,t})$ initialized at prior (α_0, β_0), e.g. $(1,1)$ (uniform). Let $\lambda \in (0,1)$ be the EMA discount factor and $w_v \in (0,1]$ the stress weight.

Case A ($L_v = 1$, QC formed): $\forall i \in C_v$:

$$\begin{aligned} \alpha_i &\leftarrow \lambda\alpha_i + w_v \cdot X_{i,v}, \\ \beta_i &\leftarrow \lambda\beta_i + w_v \cdot (1 - X_{i,v}), \end{aligned} \tag{1}$$

where $X_{i,v} := \mathbf{1}[i \in S_v]$. The set S_v is the one recorded in the block body, which every voting replica checks against the justifying certificate's signer partition; see Remark 3.1.

Case B ($L_v = 0$, no full QC, and the view timed out): only leader $j = l(v)$ is penalized:

$$\alpha_j \leftarrow \lambda\alpha_j, \qquad \beta_j \leftarrow \lambda\beta_j + w_v. \tag{2}$$

Non-leaders are left untouched, without even the discount λ; their vote outcome is *statistically censored* because no QC bitmap exists for failed views. The two cases do not cover every view: a view that closes on a tentative certificate (§10) satisfies neither and produces no update at all. That holds even when the view's timer fires before the certificate is committed: the replica whose timer fired records the view as a timeout, but an honest proposer charges only the leaders of views its justification skipped (Remark 3.1), and a block that

builds on that certificate is justified at the certificate's own view, so Case B reaches the view's leader only through a block justified by an older certificate.

Remark 4.1 (Per-view discounting under a per-period fold)**.** *The discount λ above is applied once per* view*, and the implementation honours that even though its scoring fold runs once per period. Within a period the evidence is accumulated by the same recurrence, $S \leftarrow \lambda S + w_v$, and the fold then carries the posterior forward by λ^k, where k is the number of views in which the validator was observed:*

$$\alpha_{i,t+1} = \lambda^k \alpha_{i,t} + \sum_{j:\, X_{i,j}=1} \lambda^{k-j} w_j, \qquad \beta_{i,t+1} = \lambda^k \beta_{i,t} + \sum_{j:\, X_{i,j}=0} \lambda^{k-j} w_j.$$

This is the closed form of (1)–(2) *over the period, not an approximation of it, so the batched update described in Appendix B and the per-view equations here describe the same filter. The distinction matters: applying λ once per period to an undiscounted sum of the period's evidence would leave the steady-state pseudo-mass at $\nu/(1-\lambda)$ rather than $1/(1-\lambda)$, with ν the observations per period — tightening the posterior by a factor that depends on how often a validator happens to be seated, and correspondingly weakening the conservatism the LCB of* (5) *is chosen for. A leader failure under Case B is an observed view like any other and is discounted on the same schedule.*

Remark 4.2 (Case B scoring limitation)**.** *Penalizing only the leader in failed views is a deliberate simplification, not a claim that non-leader non-participation is strategically irrelevant. In the full HotStuff protocol, a view can fail because the leader is faulty, because non-leaders are offline, or because of network partition. Phase does not implement the full HotStuff NewView* certificate *path and cannot distinguish these cases from consensus-measurable data alone. Penalizing non-leaders without reliable attribution evidence would introduce false negatives. The NVC partially addresses this when available; otherwise, Case B scoring should be understood as a lower bound on information extraction from failed views, not a complete treatment.*

Recurring soft reset. The two cases above are not the whole of the shipped filter. At each period boundary, after the λ^k carry and *before* the LCB is taken, both Beta parameters are shrunk a fixed fraction toward the prior:

$$\alpha_i \leftarrow (1-\rho_{\text{reset}})\,\alpha_i + \rho_{\text{reset}}\,\alpha_0, \qquad \beta_i \leftarrow (1-\rho_{\text{reset}})\,\beta_i + \rho_{\text{reset}}\,\beta_0. \tag{3}$$

Unlike stress weighting and the bucket factor below, (3) is *active* in the shipped configuration, at $\rho_{\text{reset}} = 0.02$ per period with $(\alpha_0, \beta_0) = (1, 1)$; setting $\rho_{\text{reset}} = 0$ recovers the unshrunk filter exactly. It does work that neither λ nor the concave transform g introduced below does: the factor λ scales α_i and β_i together, leaving the posterior *mean* untouched and widening only its spread, and g compresses score gaps while leaving the evidence behind them intact, whereas (3) moves the posterior itself back toward the newcomer prior mean $\alpha_0/(\alpha_0+\beta_0)$ every period in which the validator is observed, however long and clean the record. It is therefore the term that limits incumbency in the retained state rather than only in the transformed score. Because the shrink is applied to the persisted $(\alpha_{i,t}, \beta_{i,t})$, the LCB of (5), the Thompson draw of (8) and both composite scores all read the shrunk

posterior, and the steady-state pseudo-mass sits below the $1/(1-\lambda)$ quoted above — Appendix B gives the shrunk fixed point.

NewView Certificate (NVC). *Not implemented in the current release; specified here as planned work.* When an NVC — an aggregate of timeout votes with a signer bitmap, attached to the next leader's proposal — is available for view v, a separate *online-reliability filter* updates identically to (1)–(2) using the NVC signer set N_v. This separates vote responsiveness from online reliability and partially de-censors non-leader outcomes in failed views. NVC data is not guaranteed to exist for every failed view: it is produced only when the next leader successfully aggregates timeout votes into their proposal. Views that time out without a valid subsequent proposal produce no NVC bitmap, and non-leaders in those views receive no update at all, with no partial de-censoring.

4.2 Stress Weighting

During network-wide congestion, individual failure evidence carries less signal: many validators going offline simultaneously reflects a shared condition rather than individual unreliability. Define per-view participation fraction:

$$s_v := \begin{cases} |S_v|/n & \text{if } L_v = 1, \\ |N_v|/n & \text{if } L_v = 0 \text{ and NVC exists (planned)}, \\ 0 & \text{otherwise.} \end{cases}$$

The stress weight:

$$w_v := \min\left(1,\ \frac{n \cdot s_v}{q_{\text{full}}}\right). \tag{4}$$

When network participation falls below $q_{\text{full}}/n \approx 69\%$, the weight falls below 1, reducing the magnitude of reputation updates for that view. Stress weighting is *specified* but not active: the implementation holds $w_v = 1$ for all views.

Remark 4.3 (w_v is degenerate under the current s_v)**.** *As defined above, w_v cannot take an intermediate value, so the graceful degradation just described is not reachable. When $L_v = 1$ a full QC exists, hence $|S_v| \geq q_{\text{full}}$ and $w_v = \min(1, |S_v|/q_{\text{full}}) = 1$; when $L_v = 0$ the NVC branch is planned rather than available, so $s_v = 0$ and $w_v = 0$. The weight is therefore identically 1 on successful views and 0 on failed ones.*

The step between them is where a tentative certificate would sit, and the case analysis above does not reach it: L_v indicates a full *QC, and a tentative certificate carries no reputation update of its own — neither Case A credit for its signers nor a Case B charge against the leader. A view that reached only $q_{\text{tent}} \leq |S_v| < q_{\text{full}}$ is therefore normally not scored at all, even when it also times out. The implementation holds such a certificate for a fixed one-second window, continuing to collect votes toward a full QC; if that window closes first the certificate is committed and the view is scored not at all, and if instead the view timer fires first the pending certificate is broadcast anyway and the view is recorded locally as a timeout — but a block justified by that certificate may charge only the views its justification skipped, which do not include the certificate's own, so that view's*

leader is charged under Case B only if the next block is justified by an older certificate. The gap is worth stating explicitly, because 37–69% participation is exactly the regime a reader would expect an intermediate w_v to describe.

This matters for activation. A failed view is exactly where Case B charges the leader, and that charge is weighted by w_v, so activating (4) as written would multiply every leader penalty by zero and silently remove the only sanction the scoring layer applies to a failing leader. The constant $w_v = 1$ the implementation currently holds is what keeps Case B effective. Making the weight behave as this section describes requires an s_v that reflects observed participation in a failed view — either the NVC of §4 or another consensus-measurable signal — not merely switching on the formula.

4.3 LCB Scoring and Composite Score

We score conservatively. Rather than the posterior mean $\alpha/(\alpha+\beta)$, we use the α_{conf}-lower quantile:

$$\text{LCB}(i,t) := Q_{\alpha_{\text{conf}}}\big(\text{Beta}(\alpha_{i,t}, \beta_{i,t})\big), \quad \alpha_{\text{conf}} = 0.05. \tag{5}$$

This is a Lower Confidence Bound in the bandit sense [3]: we are 95% confident the true reliability exceeds this value. A newcomer with 3 perfect votes scores far lower than a veteran with 100 votes at 90%, because the LCB demands evidence, not just good outcomes.

To prevent incumbents from building insurmountable leads, we apply a concave transform $g(x) = x^\gamma$, $\gamma \in (0,1)$. This compresses score advantages at the top of the distribution. In plain terms, higher reliability still helps, but early leaders do not become unreachable just because they accumulated history first. The exponent γ and the confidence level α_{conf} are governance parameters in the design of §17 — as shipped, both are hard-coded defaults of the node's scoring configuration, changed by a client release (Remark 17.1) — calibrated against two objectives: γ controls how much of the score gap between a 90%-reliable and a 99%-reliable validator is preserved after transformation, with smaller γ flattening the top of the distribution more aggressively; $\alpha_{\text{conf}} = 0.05$ means the LCB is the 5th-percentile quantile of the Beta posterior, which biases selection toward well-evidenced validators over recent high-performers with few observations. Both parameters are subject to testnet calibration; sensible initialization ranges are $\gamma \in [0.3, 0.7]$ and $\alpha_{\text{conf}} \in [0.05, 0.15]$. Whether that compression reaches committee selection depends on a second exponent applied downstream; at the shipped parameters it does not (Remark 4.4).

The composite gating factor $\Pi(i,t)$ is the anti-Sybil bucket factor:

$$\Pi(i,t) := \Pi_{\text{bucket}}(i,t) \in \{0,1\}, \tag{6}$$

which equals 1 if validator i is within its bucket cap and 0 otherwise. Bucket-cap *enforcement* is implemented — selection drops the surplus identities of an over-cap bucket — but the device-attestation primitive that would assign shared bucket identities is not, so $\Pi_{\text{bucket}} \equiv 1$ today: every validator is its own singleton bucket, and the shipped cap is zero, which disables the check outright. Activation awaits that primitive first, and calibration

against live centralisation data only after it (Remark 3.2).

The composite score for the **exploit tranche** is:

$$\text{Score}^{\text{exploit}}(i,t) := \Pi_{\text{bucket}}(i,t) \cdot g\big(\text{LCB}(i,t)\big). \tag{7}$$

For the **explore tranche**, the deterministic LCB is replaced by a Thompson draw from the same posterior:

$$\tilde{s}_i := \Pi_{\text{bucket}}(i,t) \cdot g(\tilde{p}_i), \qquad \tilde{p}_i \sim \text{Beta}(\alpha_{i,t}, \beta_{i,t}). \tag{8}$$

This two-equation design is what would give full members with little history a path into the committee through the explore tranche, without grace periods or special-case rules; in the shipped configuration $\varepsilon_{\text{explore}} = 0$, so the explore tranche is implemented but fills no seats, and every graduate seat comes from the exploit draw (§5). It gives no such path to a brand-new validator in any configuration: newcomers enter through the activation queue and then probation, and probationers are in neither tranche until they graduate. It is also intended to limit low-history Sybil attacks: attestation and bucket caps would limit entry once built (Remark 3.2), while probation and the minimum-observation graduation bar — both active today — limit how quickly new identities can reach high-value roles.

Remark 4.4 (The selection exponent cancels g at the shipped γ)**.** *Neither score above is the quantity committee selection consumes. The implementation freezes, once per epoch, a* selection weight *obtained by raising the exploit score to a fixed exponent:*

$$w_{\text{sel}}(i,t) := \big[\text{Score}^{\text{exploit}}(i,t)\big]^{\eta_{\text{sel}}} = \Pi_{\text{bucket}}(i,t) \cdot \text{LCB}(i,t)^{\gamma\eta_{\text{sel}}}, \qquad \eta_{\text{sel}} = 2,$$

the second equality because $\Pi_{\text{bucket}} \in \{0,1\}$ *is unchanged by the exponent. Selection therefore responds to the product* $\gamma\eta_{\text{sel}}$*, not to* γ*. At the shipped* $\gamma = 0.5$ *that product is exactly* 1*, so the weight is proportional to the raw LCB and the concave transform has no effect whatsoever on committee composition; compression survives only for* $\gamma < 1/\eta_{\text{sel}} = 0.5$*, and for* $\gamma > 0.5$ *— the upper half of the range suggested above — the composition is convex and amplifies the very leads* g *is introduced to compress. This is the whole of* g*'s influence, because the exploit weight is the only place* $\text{Score}^{\text{exploit}}$ *enters consensus: probation graduation and the* M_t *liveness model both read* LCB *directly, and the explore tranche ranks Thompson draws, an ordering that any increasing* g *leaves unchanged. Two further properties of the shipped weight are worth stating. It is computed on the integer-scaled score and floored at one, so a validator whose score rounds to zero retains a minimal non-zero chance of selection. And* η_{sel} *is a compile-time constant of the node software, not a field of the scoring configuration at all, so the effective exponent can be moved only through* γ *— itself a chain-spec parameter that no runtime path sets. At the current parameters the mechanisms that limit entrenchment are the two observer seats for probationers (§5) and the periodic soft reset* ρ_{reset}*, not* g*.*

4.4 Tracking Claim

Note: The following is a *claim* (proof sketch). The derivation is directionally sound in a simplified setting but is not fully rigorous once endogenous censoring (Case B), variable stress weights w_v, and boundary behavior near $\theta_{i,t} \in \{0,1\}$ are accounted for. The quantity $N_{\text{eff}} = 1/(1-\lambda)$ used below is discounted pseudo-mass, not the variance-equivalent effective sample size for exponentially weighted data; the constants in the concentration argument are therefore not fully pinned down. Once discounting, censoring, and data-dependent stress weights are combined, the Beta lower quantile is best understood as a conservative scoring heuristic, not a calibrated 95% confidence statement about a drifting latent reliability process. Non-stationary Thompson-sampling theory with Beta priors has limited published results; we flag this as an open problem (§22).

Claim 4.1 (Beta-EMA Tracking, Proof Sketch)**.** *Under Assumption 3.4, fix $\eta \in (0, 1/2)$ and suppose $\theta_{i,t} \in [\eta, 1-\eta]$ for all t. In the simplified setting without censoring, with constant $w_v = 1$, and with the soft reset of (3) disabled ($\rho_{\text{reset}} = 0$):*

$$\mathbb{E}\big[|\text{LCB}(i,t) - \theta_{i,t}|\big] \lesssim \underbrace{\frac{c(\eta, \alpha_{\text{conf}})}{\sqrt{N_{\text{eff}}}}}_{\textit{statistical spread}} + \underbrace{\frac{\delta}{1-\lambda}}_{\textit{tracking bias}}, \tag{9}$$

where $N_{\text{eff}} \approx 1/(1-\lambda)$ and $c(\eta, \alpha_{\text{conf}}) > 0$ diverges as $\eta \to 0$. The heuristic optimal discount factor:

$$\lambda^* = 1 - \left(\frac{2\delta}{c(\eta, \alpha_{\text{conf}})}\right)^{2/3}. \tag{10}$$

This bound does not extend to censored or variable-weight regimes.

Remark 4.5. *The two-way tradeoff in (9) is the key practical guide: smaller λ reduces tracking bias but widens statistical spread. The optimal λ^* balances these at a rate that scales with the drift speed δ. A faster-changing validator environment calls for smaller λ.*

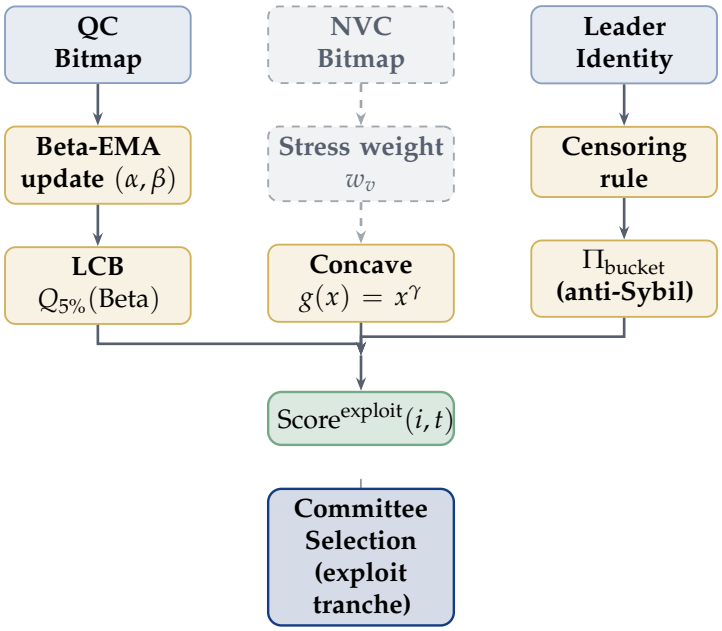

Figure 1: End-to-end scoring pipeline for the exploit tranche. The explore tranche replaces the deterministic LCB with Thompson draws from the same posterior, so full members with few observations can still earn committee exposures; a validator still on probation is in neither tranche and is seated only as a non-counting observer (§5). The dashed grey stages are drawn as specified rather than shipped: no NVC exists at all, and the stress weight is applied on every observation but pinned at $w_v = 1$ (§4). The columns group pipeline stages rather than trace operand flow: w_v multiplies the evidence entering the Beta-EMA update (1)–(2), and g is applied to LCB as in (7). Two of the solid stages leave committee composition untouched at the shipped parameters: the concave transform is cancelled downstream by the selection exponent at $\gamma = 0.5$ (Remark 4.4), and $\Pi_{\text{bucket}} \equiv 1$ — its enforcement branch is never entered, because the shipped bucket cap is zero and every validator is its own bucket (Remark 3.2).

5. Committee Selection

5.1 Two-Tranche Selection via Thompson Sampling

The selection problem is a classic explore/exploit dilemma. Always picking the highest-scoring validators selects proven performers but prevents under-observed full members from accumulating evidence. Pure random selection avoids ossification but risks missing quorum. Thompson sampling [4] balances both by sampling from each validator's uncertainty range. Veterans have narrow distributions and draw near their true reliability; a candidate with few observations has a wide distribution and occasionally draws high, winning a committee slot. As evidence accumulates the distribution narrows, and selection converges to merit.

Phase fills the voting seats in two tranches and seats observers beside them. The seat budget is 16 in the shipped configuration (32 the wire-format ceiling), while the registry

ceiling $\min(M_t, M_{\max})$ (§9, Remark 5.2) bounds how many validators may be eligible at all and does not size the committee. Voting seats are filled in two tranches: the anchors — the genesis validators — taken whole, then up to $\lfloor (a-1)/2 \rfloor$ graduated validators drawn without replacement with probability proportional to w_{sel} (uniform on stall-recovery rounds). Up to two probationers are seated as observers: they sign so their reliability can be scored and are paid as signers, but their signatures count toward no quorum and they never lead. Any graduate seat the *Full* pool cannot fill goes unused rather than to further probationers, and a two-anchor chain seats no graduate at all. Within the graduate seats the design specifies a further split:

- **Exploit tranche**: fill the graduate seats not carved out for the explore tranche by a deterministically seeded draw without replacement, in which each candidate still in the pool is selected with probability proportional to its selection weight $w_{\text{sel}}(i,t)$, the fixed power η_{sel} of $\text{Score}^{\text{exploit}}(i,t)$ from (7) pinned in Remark 4.4. This is score-*weighted* sampling rather than a top-k cut, and Remark 5.1 states what that does and does not guarantee. There is no eligibility requirement beyond graduated *Full* status — anchors are seated before the draw, and probationers only into the observer seats above — and the bucket factor Π_{bucket} already embedded in the score.
- **Explore tranche**: for each graduated candidate i not seated by the exploit tranche, draw $\tilde{p}_i \sim \text{Beta}(\alpha_{i,t}, \beta_{i,t})$, compute $\tilde{s}_i = \Pi_{\text{bucket}}(i,t) \cdot g(\tilde{p}_i)$ as in (8), and fill the carved-out seats — an $\varepsilon_{\text{explore}}$ fraction of the graduate seats, rounded down, and zero on the stall-recovery rotations of Remark 5.1 — ranked by $\tilde{s}_i$.

Seats filled by the explore tranche vote but are never leader-eligible, and the observer seats neither vote nor lead: the leader-eligible set is the anchors and the exploit draw, fixed before either of the other two is added. The split eliminates the ossification that pure score ranking would produce: a full member with a wide posterior occasionally draws high in the explore tranche, earns committee exposures, accumulates observations, and raises its selection weight toward regular exploit-tranche selection. That path opens only after graduation. A validator that registers after genesis waits in the activation queue, enters the registry on probation, and takes its first committee exposures from the observer seats above — a special case in selection, and the only route a newcomer has; only the genesis set skips both the queue and probation, entering as anchors.

The explore tranche is implemented but activated progressively as the network accumulates live data; in the current configuration $\varepsilon_{\text{explore}} = 0$, so every graduate seat is filled by the exploit draw, and the observer seats are the only route by which a newcomer accumulates the observations a scoring history needs. That route is narrow by construction: at most two probationers are seated per committee, and their draw, like the graduate draw, is weighted by w_{sel} (uniform on stall-recovery rounds) and keeps no turn order between epochs, so a probationer that has already signed well is favoured over one still on the newcomer prior, and a probation cohort larger than the reserved allocation graduates only over many epochs.

Remark 5.1 (The exploit tranche samples, it does not rank)**.** *The exploit tranche is a seeded random draw, not a deterministic top-k cut. For each seat the implementation walks the cumulative weight distribution over the candidates still in the pool and removes the one it lands on, so a candidate's chance of a seat is proportional to* $w_{\text{sel}}(i,t)$ *and a lower-scoring candidate can displace a higher-scoring one; the score order governs the committee's expected composition, not its membership. The draw is seeded deterministically from the epoch and the committee round, so it is reproducible and every node derives the same committee. Two consequences are worth stating plainly. First, the anchors are seated whole before the draw, and when the graduates are no more than* $\lfloor (a-1)/2 \rfloor$ *the draw is degenerate and seats every one of them too, which is the regime a young network is in. Second, the weights apply only to the canonical per-epoch committee: the rotations used to recover from a stall flatten every weight to one and sample uniformly, because scores cannot move while the chain is inserting no blocks and a weighted redraw would keep re-seating the same unresponsive cohort. Committee membership during stall recovery is therefore independent of* $\text{Score}^{\text{exploit}}$.

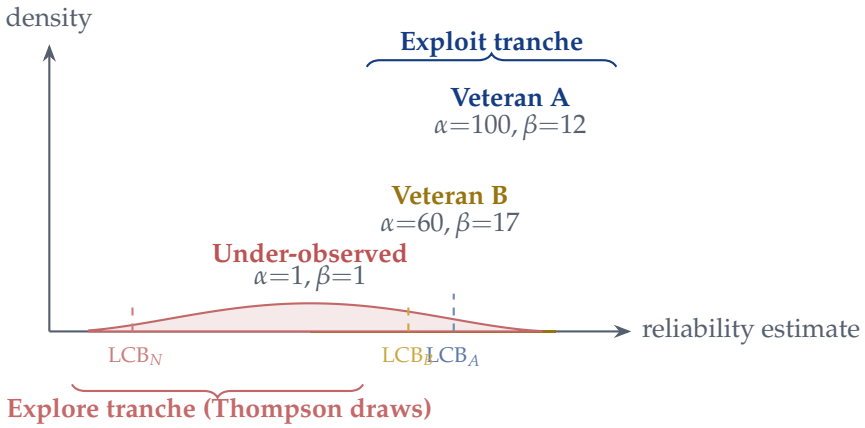

Figure 2: Two-tranche Thompson sampling over the graduated full members, for the at most $\lfloor (a-1)/2 \rfloor$ graduate seats; the anchors are seated whole before either tranche. Veterans (narrow distributions) carry the largest exploit-tranche selection weights and so are drawn most often. Full members with few observations (wide distributions) enter via explore-tranche Thompson draws; as evidence accumulates, their posteriors narrow and selection converges to merit. Probationers are in neither tranche: they are seated only as non-counting observers.

5.2 Probation, Queue, and Churn Limit

Registration is permissionless; admission into the eligible registry is rate-limited. Validator states: $s_{i,t} \in \{\texttt{queued}, \texttt{probation}, \texttt{full}, \texttt{jailed}\}$, with $E_t = P_t \cup F_t$.

At each period boundary: (1) release jailed validators whose sentence ends; (2) promote probationers satisfying $\text{obs}(i,t) \geq N_{\min}$ and $\text{LCB}(i,t) \geq \tau_{\text{grad}}$ (Remark 5.3); (3) return to `queued` any probationer that has not graduated within the probation timeout — 50 periods in the current configuration — placing it, like a released jailed validator, at the back

of the activation queue with its scoring state left untouched by the transition and no further penalty (a queued validator is dropped from every selection snapshot written from that boundary on and thereafter accumulates no observations, and a scoring row left unchanged for a full active duration of 320 blocks is dropped by the epoch-boundary staleness eviction, so a long enough wait in the queue means re-entry on the newcomer prior); (4) admit new entrants in queue order, up to the churn limit χ (Remark 5.2) and no further than the room left under the registry ceiling $\min(M_t, M_{\max})$.

Remark 5.2 (The shipped churn limit is a flat constant)**.** *As implemented, χ is a compile-time constant of* 2 *admissions per period boundary, the same at every registry size: it is the $\chi_{\min} = \chi_{\max}$ instance of the proportional rule $\chi_t = \min(\chi_{\max}, \max(\chi_{\min}, \lfloor \rho_{\text{churn}} M_t \rfloor))$, whose ρ_{churn} term the node software does not implement. Nor is χ reachable from chain state or from a chain specification: changing it is a client release, which is what Stage 1 of the governance roadmap (§17) describes. The second bound is structural — admission also stops once the eligible set reaches $\min(M_t, M_{\max})$, where $M_{\max}$ is a hard maximum on the eligible registry (*4096 *in the shipped build). The flat limit is the binding one in steady state: the eligible registry grows by at most two validators per period whatever the queue depth, so filling it from a standing start to $M_{\max}$ takes on the order of two thousand periods. What that clamp enforces is the registry ceiling, not a safety bound: under the anchor bound the graduates beyond $\lfloor (a-1)/2 \rfloor$ are a bench inside the model (§3), so a registry of any size up to $M_{\max}$ leaves deterministic safety where it was, and the quorum-intersection rule over the voting population is consulted only where the anchors are not seated whole.*

Remark 5.3 (The implemented graduation predicate)**.** *Step (2) is the whole promotion rule: the boundary pass reads each probationer's freshly folded scoring state and promotes it to* `full` *when its cumulative raw observations — QC signatures plus misses, against $N_{\min} = 40$ — and its LCB — against $\tau_{\text{grad}} = 0.5$ — both clear their thresholds. No fault history enters the test, and no per-validator fault record exists to consult: equivocation detection is wired into both the proposal and the vote path, but it only logs and counts, writes no chain state, and nothing reacts to a report. A further clause requiring no cryptographic fault within a fixed window of periods would therefore be vacuous, and step (2) deliberately carries none. The guarantee such a clause would express is structural rather than a term in the predicate:* `jailed` *is disjoint from* `probation`*, a jailed validator is excluded from the eligible registry for its whole sentence, and release returns it to the* back *of the activation queue, so it must be re-admitted and serve a fresh probation before it can be promoted again — the window is J, and no separate parameter is needed. That route is itself dormant: jailing exists as a callable state transition with no production caller, so nothing currently moves a validator into* `jailed` *(§6.4).*

Lemma 5.1 (Sybil occupancy bound under attestation)**.** *Under Assumption 3.3, an adversary controlling k attested devices occupies at most $k \cdot \text{Cap}_{\text{bucket}}$ slots in E_t.*

Proof. See Appendix D. □

Remark 5.4 (Committee-share bound, conditional)**.** *An expected committee-fraction bound of $k \cdot \text{Cap}_{\text{bucket}} / M_t$ follows only under a further assumption: that adversary-controlled validators*

draw weights in committee selection no larger than those of an average incumbent. Against a patient adversary who farms scores before attacking, that assumption is not guaranteed, and score-weighted sampling can concentrate committee mass on adversarial validators. We do not claim this as a theorem.

Remark 5.5. *This lemma is only as strong as Assumption 3.3. Collusion between platform attestation services or successful account-farm attacks on app stores would undermine it.*

6. Reward Mechanism

6.1 QC-Based Reward Split

For each finalized view v with reward budget R_v:

$$\text{Pay}_{\text{leader}}(v) := \beta_t \cdot R_v \quad \text{to leader } l(v), \tag{11}$$

$$\text{Pay}_i(v) := (1-\beta_t) \cdot R_v \cdot \frac{W_{i,v}}{\sum_{j \in S_v} W_{j,v}} \quad \forall i \in S_v, \tag{12}$$

where $W_{i,v}$ is the signer weight (uniform in the baseline design; score-proportional weighting is a governance option under the design of §17, not a switch the implementation exposes). Non-signers ($i \in C_v \setminus S_v$) receive zero. The disposition of the non-signers' share — burnt, or redistributed among signers — is likewise a governance-design choice with no parameter behind it: the shipped split divides the whole signer pot equally among the recorded signers, giving the indivisible dust to the leader, so no such share arises (Remark 17.1). In the current configuration R_v is protocol emission alone — the fixed per-height amount of the emission schedule — with no transaction-fee component: fees are burnt in full, so they are neither split by β_t nor released at finality. Folding fee revenue into the validator payout is the fee-offset design of §16, which is not yet active.

6.2 Finality-Gated Release

The release policy is that rewards are paid *only* for blocks finalized under a full quorum certificate. Reward settlement is performed by the execution layer with a lag, crediting heights at or below a *finality watermark*: a header scalar each proposer stamps as the greater of its own observed full-QC 3-chain committed height and the value its parent carried. When every proposer in the recent chain is honest, this credits exactly the heights covered by a full QC and finalized under the 3-chain rule; the watermark itself is not backed by a certificate, and Remark 6.1 states what that leaves open. The *entitlement* is reorg-stable, but not because the paying block is final: the per-height amount and the leader and signer identities are keyed on the paid height rather than on the block that pays it, and the payout path pays only heights at or below the watermark the paying block commits. That watermark is proposer-asserted — importers check only that it does not regress and does not exceed the parent's height less two (three heights below the paying block), and every node's execution layer applies that same depth clamp again on its own — so it is a structural cap rather than a finality proof, and a proposer that over-reports inside that band can

pay a height no full-QC 3-chain has committed. The *credit* is reorgable outright: it is ordinary state written by whichever canonical descendant pays it, and that descendant is never itself final at the moment it writes, so a reorg reverts the balance increments together with the paid-up-to watermark that records them; a payout becomes irreversible only once the block that wrote it is itself finalized. What holds in the meantime is reorg-invariance rather than reorg-proofness: the surviving descendant pays the same heights again and re-emits the credit, so no payout is lost to the reorg, none is paid twice, and none survives on a chain segment that subsequently fails to finalize. The re-derived payout is the *same* payout only when the fork point is above the paid height: the beneficiary and signer records for a height are written by the block at that height itself, so a reorg deep enough to replace that block — which, absent an outright safety violation, requires the over-reported watermark above — can have that height pay a different leader, and a different signer set: the beneficiary is the replacing proposer's own, and the signer record is the signer section of the full QC that block was justified by, if any. Branch-dependent is the distribution rather than the per-height total: pool membership is read at the paying block and restricted to members who had joined by the paid height (§15), so a reorg that changes who has since left redistributes an unchanged total across a different set of members. Two further inputs are likewise resolved against the paying block's state — the beneficiary and signer records, retained in an 8192-height ring beyond which a height pays nothing or degrades to leader-only, and the reward schedule, which sits behind an upgradeable proxy.

This gating has a second function beyond accounting hygiene: at any moment, every active validator carries a balance of *withheld, not-yet-final* rewards — income already earned by observable work but not yet released because finality has not yet caught up. We denote this balance $\mathrm{Wh}_i \geq 0$. It consists of the finality-lag tail (a few views of income in flight under the 3-chain rule) plus any tentative-block rewards awaiting a full QC (§6.3). On a provable fault, Wh_i is forfeited under the penalty design of §6.4; no implemented path forfeits it (Remark 6.2). Wh_i is structurally small in normal operation; it is a hygiene mechanism, not the primary deterrent.

Remark 6.1 (The finality gate is proposer-committed)**.** *Validating nodes check the committed watermark for range only: it must be at least the parent's value (itself capped at the upper bound) and at most the parent's height less two, the 3-chain commit depth. The execution layer applies that depth clamp again on its own — a block may credit no height above its parent's height less two — but nothing compares the scalar against a certificate, or against the validating node's own committed height. A Byzantine leader can therefore stamp the largest value the range admits, and heights that carry only a tentative certificate are credited; and because the range floor is inherited from the parent, an honest successor cannot lower it and carries the inflated value forward until its own finality overtakes it. The cheapest setting for this is exactly the stall that tentative certificates exist for.*

The durability property survives, for a different reason than the gate: a credit is written into the state of the block that pays it, so it becomes irreversible exactly when that block does — and when a paying block finalizes, so do the strictly lower heights it credited. The exposure is early credit into

tentative state, not permanent payment for a height that never finalizes. Binding the watermark to the quorum certificate that justifies it — carrying that certificate, or a commitment to it, in the block and verifying it on import — is a block-format change, and it is not implemented.

Table 2: Reward Release Policy. The *committed finality watermark* is the proposer-stamped header scalar of §6.2: range-checked on import and depth-clamped in execution, but never compared against a certificate (Remark 6.1).

Reward type	**Payout**	**Condition**
Consensus block reward (leader share)	$\beta_t \cdot R_v$	Once a later watermark reaches the height
Consensus block reward (signer share)	per (12)	Once a later watermark reaches the height
Tentative-block reward	0 (withheld)	Released when a later watermark reaches it

6.3 Tentative-Block Rewards: Fully Withheld

Blocks certified only by a tentative certificate (§10) advance the chain but not finality. Their rewards are 100% withheld until a later block's committed finality watermark reaches them — on a chain of honest proposers, exactly when a subsequent full QC finalizes the covering chain segment (Remark 6.1); only then are they released under the same rule as ordinary rewards. On a provable fault, any pending tentative-block rewards are forfeited as part of Wh_i under that same unimplemented design (Remark 6.2). This removes any incentive to farm income through low-quorum segments that never achieve finality.

6.4 Provable Fault: Penalties

On cryptographic proof of a provable fault (double-signing, equivocation), the design applies three penalties — none of which the implementation currently triggers (Remark 6.2):

1. **Forfeiture of withheld rewards:** the entire not-yet-final balance Wh_i — tentative-block rewards pending confirmation and any finality-lag income in flight — is forfeited (burned or redirected — the disposition is a governance-design choice and no parameter for it exists; the forfeiture path is in any case dormant with jailing, Remark 5.3).
2. **Jail:** exclusion from the eligible registry $E_{t'}$ for $t' \in [t, t+J)$.
3. **Reputation reset:** the Beta-EMA state is reinitialized, $(\alpha_{i,t+J}, \beta_{i,t+J}) \leftarrow (\alpha_{\mathrm{pen}}, \beta_{\mathrm{pen}})$ with $\alpha_{\mathrm{pen}} \ll \beta_{\mathrm{pen}}$: the validator restarts at (or below) the newcomer prior and re-enters through probation like a new participant — minimum observations, graduation threshold τ_{grad}, and no leader eligibility until graduation (§5).

No deposit is seized, because none exists: registration is free and there is nothing to slash. The penalty is denominated in *time and forgone income* — the jail term, the forfeited in-flight rewards, and the accumulated evidence that made the validator's score valuable. Section 7 quantifies why, for an established validator, this is a material loss.

Remark 6.2 (Nothing triggers the penalty stack)**.** *The three penalties above are design. Detection exists for the cryptographic cases, but as node-local observability only: the equivocation detector is fed from both the proposal path and the vote path — the proposal path by broadcast, the vote path only at whichever leader a vote was sent to, so an equivocator that splits its two conflicting votes between two leaders is caught by neither — and on the first conflicting pair from one validator in one view and one message kind it increments an in-process counter, logs at error level (rate-limited per validator) and records the evidence; the detector itself is a bounded LRU keyed by validator, view and kind. The record carries both signed statements and can be checked offline against the accused key, but it stays node-local — held in memory, where the operator reads it over IPC, and appended to a bounded file in the node's data directory — and is neither gossiped nor written to chain state; the consensus crate's own documentation records that nothing in the protocol acts on a report. There is no per-validator* provable-fault *record anywhere in the database for a later stage to consult — the only per-validator failure counters are the scoring system's missed-signature and failed-leader tallies, which are soft-failure data. Penalties 2 and 3 exist together as a single state transition rather than as a policy: it sets the status to* `jailed` *and resets the Beta state to the configured success prior against a fixed failure mass of* 50*, writing an LCB of exactly* 0 *— under the shipped, explicitly uncalibrated* $\mathrm{Beta}(1,1)$ *prior that is a posterior mean of roughly* 0.02*, and the reset is at or below the newcomer prior for any prior whose own failure mass is below* 50*, which the constant does not itself enforce. A jailed validator is then skipped by the selection-snapshot writer, but each snapshot is written a full epoch in advance, so the exclusion begins two epoch boundaries after the jailing; the same epoch-boundary pass returns the validator to the back of the activation queue once ten boundaries have passed (*$J = 10$ *in the shipped build), from where it must be re-admitted and serve a fresh probation. Two qualifications on that transition: it has no production caller (one definition, two re-exports, and three calls, all inside two unit tests), and its own notes name a prerequisite for wiring one — the reputation write is not reproducible by the clear-and-replay scoring unwind, so a node that unwound into the jailing epoch would keep the* `jailed` *status and lose the reset. It also applies the reset at the moment of jailing rather than at* $t + J$*. Penalty 1 is not merely untriggered but absent: the reward path credits heights at or below the committed finality watermark (§6.2) and reads no fault record, and no code forfeits, burns, or redirects a withheld balance. Remark 5.3 reaches the same conclusion from the promotion side; the analysis of §7 prices a stack the implementation does not yet apply.*

6.5 Adaptive Leader Bonus

The leader share β_t is designed as a slowly adaptive governance-controlled parameter:

$$\beta_t = \mathrm{clip}_{[\beta_{\min},\beta_{\max}]}\left(\beta_{\text{base}} + k_P \log \frac{\mathrm{TWAP}_{\text{ref}}}{\mathrm{TWAP}_t} + k_L(L_{\text{target}} - \hat{L}_{t-1})\right), \tag{13}$$

where TWAP_t is a long time-weighted average of the PHASE token price, $\mathrm{TWAP}_{\text{ref}}$ its reference level, and $\hat{L}_{t-1}$ is the observed liveness signal — specified as the QC-success rate, while the value the node software carries to the reward split today is a finality-gap proxy recomputed from the block's own committed watermark, a windowed QC-success rate being a stated prerequisite for activation rather than something measured. The base

share β_{base} is unrelated to the Beta-EMA prior parameter β_0 of §4. Updates are rate-limited ($|\beta_t - \beta_{t-1}| \le \Delta_\beta$ per week); no such limiter is implemented. The adaptive rule is part of the mechanism design and is activated progressively as the network accumulates live data; until activation, $\beta_t = \beta_{\text{base}}$ is a fixed constant ($\beta_{\text{base}} = 0.6$ in the current configuration, with the clip band pinned to it so the share cannot move), and the price term is design only — no price feed or TWAP exists in the node software.

Remark 6.3 (Procyclicality of deterrence)**.** *The deterrent described in §7 is denominated in future income. Its market value therefore scales with the token price and with fee flow: in a downturn, the same jail term and reputation reset represent a smaller dollar-denominated loss, and deterrence against dollar-denominated bribes weakens accordingly. We state this plainly rather than obscure it; the risk discussion in §20 returns to it.*

7. Deterrence Without Stake Bonds

7.1 The Earning Position as Collateral

Phase has no bonded stake, so the deterrence question must be answered honestly from the only asset a validator actually holds at protocol level: an *earning position* — a reputation state that generates a stream of expected future income. Let $\pi_i = \pi_i(s_{i,t}, M_t) > 0$ denote validator i's expected per-period net income given status $s_{i,t}$ (probation vs. full, exploit-tranche eligibility) and registry size M_t, and let $\rho \in (0,1)$ be the per-period discount factor. The validator's *continuation value* is the expected net present value of that stream:

$$V_i(t) := \mathbb{E}\left[\sum_{k\ge 0} \rho^k \, \pi_i(s_{i,t+k}, M_{t+k}) \;\middle|\; \mathcal{F}_t\right]. \tag{14}$$

$V_i(t)$ is increasing in the validator's reputation state: a well-evidenced score raises selection probability, unlocks the exploit tranche, and (for graduated validators) leader income. The earning position itself is the validator's skin in the game.

7.2 Penalty Decomposition

The penalties of §6.4 — design, none of which the implementation triggers (Remark 6.2) — price a provable deviation in three non-overlapping components:

$$\text{Pen}_i(J) := \underbrace{\text{Wh}_i}_{\substack{\text{withheld rewards}\\ \text{forfeited at detection}}} + \underbrace{\sum_{k=0}^{J-1} \rho^k \, \pi_i}_{\substack{\text{income foregone}\\ \text{during jail}}} + \underbrace{\rho^J \Big(V_i^{\text{normal}}(t+J) - V_i^{\text{reset}}(t+J) \Big)}_{\substack{\text{reputation-rebuild cost}\\ \text{after jail}}}, \tag{15}$$

where $V_i^{\text{normal}}(t+J)$ is the continuation value the validator would have held at $t+J$ absent the fault, and $V_i^{\text{reset}}(t+J)$ is the continuation value starting from the reset state $(\alpha_{\text{pen}}, \beta_{\text{pen}})$ — which includes the probation re-entry path: minimum observations, the graduation threshold, and depressed selection probability while the score is rebuilt from the prior.

Since honest continuation satisfies $V_i(t) \approx \sum_{k<J} \rho^k \pi_i + \rho^J V_i^{\text{normal}}(t+J)$, the decomposition admits the compact reading

$$\text{Pen}_i(J) \;\approx\; \text{Wh}_i + V_i(t) - \rho^J V_i^{\text{reset}}(t+J)\,: \tag{16}$$

a provable fault costs the in-flight rewards plus the difference between the validator's current earning position and the discounted value of the position they must rebuild from scratch.

7.3 One-Shot Sufficiency Condition

Scope. This is a *sufficient condition* for one-shot, single-validator, provable-fault deterrence under certain detection, stated at proof-sketch rigor. It is not a coalition result, a repeated-game result, or a bound on non-provable strategic behavior. Outside Assumptions 3.1–3.5 — in particular under probabilistic detection, off-chain hedges, or recurring censorship rents — the condition may fail to be sufficient. Coalition considerations are in §8; residual open problems in §22.

Proposition 7.1 (Deterrence: sufficient condition, proof sketch)**.** *Under Assumptions 3.1–3.5, a provable single-validator deviation by i is unprofitable if $B_{\max} \le \text{Pen}_i(J)$. Dropping the reputation-rebuild term in (15) (conservative, since it is ≥ 0) yields the closed-form parametric condition*

$$J \ge J^*(B_{\max}, \text{Wh}_i, \pi_i, \rho) := \frac{\log\left(1 - \dfrac{(B_{\max} - \text{Wh}_i)(1-\rho)}{\pi_i}\right)}{\log \rho}, \tag{17}$$

well-defined when $\text{Wh}_i \le B_{\max} < \text{Wh}_i + \pi_i/(1-\rho)$. When $B_{\max} \le \text{Wh}_i$, any jail duration suffices on the monetary margin.

Proof sketch. Sufficiency. If $B_{\max} \le \text{Pen}_i(J)$, then $B_i - \text{Pen}_i(J) \le 0$ for every feasible bribe $B_i \le B_{\max}$; the deviation is weakly dominated, strictly when the inequality is strict. *Parametric form.* Dropping the third term of (15), sufficiency requires $B_{\max} \le \text{Wh}_i + \pi_i(1-\rho^J)/(1-\rho)$; solving $(1-\rho^J) \ge (B_{\max} - \text{Wh}_i)(1-\rho)/\pi_i$ for J gives (17). This is a sketch, not a full proof: π_i is treated as a constant expectation rather than a stochastic stream, and the dropped rebuild term is bounded below by zero without characterizing it. It is not an "if and only if": under probabilistic detection or recurring off-chain rents, sufficiency can fail. □

Remark 7.1 (No seizable stock)**.** *Because Wh_i is structurally a few views of income in flight, condition (17) is carried almost entirely by the flow terms. In particular, if $B_{\max} \ge \text{Wh}_i + \pi_i/(1-\rho)$ — a bribe exceeding the full NPV of the validator's income stream — no finite jail term satisfies the conservative bound, and deterrence must rest on the dropped rebuild term and, more importantly, on the non-economic layers: BFT safety itself (a single validator cannot violate safety; Assumption 3.1), coalition-scale defenses (§8), and the bootstrap design (§21). We regard economic deterrence as a cost overlay on cryptographic safety, not a substitute for it.*

Three honest observations follow directly from (16).

(a) Deterrence grows with tenure and expected income. $V_i(t)$ compounds with honest participation: graduation from probation, exploit-tranche entry, and a well-evidenced LCB all raise π_i and selection probability. The penalty a veteran faces is therefore a multiple of what a newcomer faces — the system naturally concentrates consensus power in exactly the validators for whom deviation is most expensive.

(b) At genesis, deterrence is near zero. At launch, $V_i \approx 0$ for every participant: there is no income history, little fee flow, and $\mathrm{Wh}_i \approx 0$. Equation (15) then provides essentially no protection. We do not paper over this: genesis-period security rests on the permanent anchor set and its published trust assumptions: the genesis validators are seated whole in every committee and graduates hold at most $\lfloor (a-1)/2 \rfloor$ voting seats beside them, and no decay of that share is scheduled. This is treated in §21.

(c) The penalty cannot be repurchased. A bonded validator who loses a deposit can restore their position instantly with fresh capital; the deterrent is fungible and replenishable. Here the lost asset is *evidence accumulated over time*: after a reset, the validator must serve the jail term, re-enter probation, and re-earn observations at the protocol's own pace — no payment accelerates the rebuild. A validator (or the same operator across attempts) attacking K times pays K full rebuild periods in calendar time, serially within an identity; and spawning parallel identities to amortize resets is limited by churn limits today, and by attestation bucket caps once those are built (Lemma 5.1, Remark 3.2). Repeated attacks are therefore expensive in a way that capital-denominated penalties are not: capital can be re-supplied at market speed, whereas reputation is re-earned only at chain speed.

Remark 7.2 (Non-provable deviations)**.** *Censorship, selective vote withholding, and coordinated soft failure are not provable on-chain and are not covered by Proposition 7.1. They are addressed through mechanism design — scoring pressure on leaders (Appendix F), committee rotation on stall, and governance escalation, for which no mechanism is deployed yet (Remark 17.1) — rather than formal deterrence bounds. Remaining open problems are collected in §22.*

8. Coalition Considerations

Under full-QC finality, HotStuff's BFT safety proof [1] guarantees no fork for $|\mathcal{K}| \leq f$ Byzantine validators regardless of off-chain coordination — and, under the anchor bound, while the anchor set is seated whole: any two quorums then share an anchor, so finality survives up to $2(q_{\text{full}} - g) - a - 1$ compromised anchor keys whatever the registry size (§3). Only where the anchors are not seated whole — more anchors than seats, or no anchor left — do the certificate paths log the broken quorum-intersection bound and certify anyway, so the guarantee this section layers cost on top of is deterministic in the configuration the chain launches with. Economic penalties add cost on top of that conditional guarantee;

they are not the primary safety mechanism at the BFT level. Proposition 7.1 covers single-validator deviations only; this section examines what the design does — and honestly, does not — provide against coalitions. Quantitative statements in this section are illustrated at the wire-format ceiling ($n = 32$, $f = 10$, $q_{\text{full}} = 22$), not at the shipped $n \in [7, 10]$ (§3).

8.1 Correlation-Aware Scoring and Fault Attribution

Correlation enters the scoring system in two directions, and the asymmetry between them is deliberate.

Protecting honest validators from correlated outages. Mass simultaneous failures (carrier outage, regional connectivity loss) carry little signal about individual reliability. The stress-weighting correction in the scoring layer (§4) is intended to reduce the magnitude of reputation updates in low-participation views, so that once active a genuine correlated outage does not destroy individual reputations. This correction is specified but not yet active — the implementation holds $w_v = 1$ — so the protection described here is a property of the design, not of current behaviour.

Escalating penalties for correlated provable faults. Conversely, *provable* faults that cluster in time are more dangerous than isolated ones, because clustered equivocation is the signature of a coordinated safety attack. Let $\mathcal{K}_v$ denote the set of validators with a provable fault recorded within a rolling window of W_{corr} views ending at view v. Since the penalty currency in Phase is time rather than seized capital, the progressive mechanism escalates in time:

$$J(k) := \min\Big(J_{\max},\ J_{\text{base}} \cdot \Big(1 + \kappa_{\text{corr}} \cdot \tfrac{\min(k,f)-1}{f-1}\Big)\Big), \qquad k := |\mathcal{K}_v|, \tag{18}$$

with governance-set $\kappa_{\text{corr}} \geq 1$: an isolated fault draws the base jail term, while a fault that is part of an f-sized cluster draws a term up to $(1+\kappa_{\text{corr}})$ times longer, together with the same reputation reset. The mechanism is a design element and nothing behind it is built: no per-validator *provable-fault* record exists in chain state for $\mathcal{K}_v$ in (18) to be computed from, equivocation detection only logs, counts and keeps the signed evidence node-local (in memory and in a file in the node's data directory, served to the operator over IPC), nothing in production calls the jail transition — so the base term the rule escalates has no trigger either — and the shipped jail term is a flat constant of ten periods, the same for every fault (§6.4; Remark 6.2). Activation waits on misbehaviour proofs and a jail caller, not on accumulated live data.

Remark 8.1 (Scope: provable faults only)**.** *The escalation rule* (18) *applies only to provable faults (equivocation, double-signing), where cryptographic evidence is unambiguous. It deliberately does not extend to soft failures (going offline, missing views): genuine correlated outages would be indistinguishable from coordinated soft attacks and would trigger unacceptable false-positive penalties against honest mobile validators. Soft-failure correlation is handled by the protective direction above, not the punitive one.*

8.2 Occupancy and Committee-Share Limits

Before a coalition can attack, it must hold seats, and the selection pipeline meters seat acquisition in time:

- **Bucket occupancy.** Under device attestation, an adversary controlling k attested devices occupies at most $k \cdot \mathrm{Cap}_{\mathrm{bucket}}$ slots in the eligible registry (Lemma 5.1). This bound inherits the strength and the limitations of platform attestation (Assumption 3.3), and it is prospective rather than live: no attestation is performed today, bucket identity defaults to the validator's own address, and the implemented cap governs committee seats per epoch rather than registry slots (Remark 3.2).
- **Churn limit.** New admissions per period are capped at χ, so even a large identity supply converts into registry share only at a bounded rate; probation and minimum-observation requirements add further calendar time before any new identity carries selection weight.
- **Bounded graduate seats.** Under the anchor bound the voting population is not sampled: the anchors are seated whole and at most $\lfloor (a-1)/2 \rfloor$ graduates sit beside them — three at seven anchors — so a coalition of graduates holds at most that many voting seats whatever its registry share, and quorum intersection holds whatever the registry size (§3). Which graduates fill those seats is decided by score-based selection over the eligible registry, capped at $\min(M_t, M_{\max})$ (the score-weighted exploit tranche plus, once $\varepsilon_{\mathrm{explore}} > 0$, the Thompson-sampled explore tranche; at the shipped $\varepsilon_{\mathrm{explore}} = 0$ every graduate seat comes from the exploit draw, §5), from deterministic per-epoch snapshots. A coalition holding registry share σ can expect a share of the graduate seats that tracks its weighted registry share, and a patient adversary who farms high scores before attacking can hold above-average selection weight. The selection seed mixes in no chain state, deliberately, so that the next committee can be prewarmed; whether an anchored seed or a VRF is worth its cost to the prewarm is open (§22). Sampling a larger voting population from the registry — the regime in which this share argument would bear on safety rather than on seats and income — is one the anchor bound forbids by design (§22).

8.3 Coalition Deterrence: Honest Assessment

With the escalation rule, the per-member penalty for a coalition of size k is $\mathrm{Pen}_i(J(k))$ from (15) evaluated at the escalated jail term, and a natural design condition against a total bribe budget B_{total} split across members is

$$\mathrm{Pen}_i\big(J(|\mathcal{K}|)\big) \geq \frac{B_{\mathrm{total}}}{|\mathcal{K}|} \qquad \text{for each } i \in \mathcal{K}. \tag{19}$$

Design target (not a proved result), and an honest comparison. Condition (19) is a design target: a full proof would require formalizing the bribe-splitting model, coalition

symmetry, and detection timing, and none of that is done here. More fundamentally, coalition deterrence *without capital bonds is weaker than in bonded systems*: a coalition of k members jointly forfeits k earning positions and in-flight rewards, and the escalation in (18) lengthens the rebuild, but there is no superlinear capital seizure of the kind a correlation-scaled deposit penalty provides. Phase does not claim otherwise. The design addresses the gap through three layers:

1. *Committee-scale randomization* — occupancy and share limits above make assembling f correlated seats slow, observable, and probabilistic rather than purchasable;
2. *The permanent anchor set* (§21) — while earning positions are too young to deter, and thereafter, safety rests on the genesis anchors, seated whole in every committee with graduates bounded to $\lfloor (a-1)/2 \rfloor$ voting seats; the anchor share is not retired as income streams accumulate, since no decay is implemented or planned;
3. *Detection-and-jail with correlated-fault attribution* — clustered provable faults are designed to draw escalated, non-repurchasable time penalties per (18). Unlike the two layers above, this one has no implementation behind either half: detection logs and counts without writing chain state, and no code jails (Remark 6.2).

Each layer is stated as design intent with its assumptions visible; none is presented as a theorem. Formal coalition analysis remains open (§22).

9. Registry-Size Control

9.1 Quorum Probability: Poisson-Binomial

If registration is free and unlimited, a free-entry equilibrium drives expected validator profits toward zero, at which point validators stop caring about uptime. The soft cap prevents this by limiting the eligible registry without restricting registration.

For committee C_v in period t, the timely-vote count $K_v^* := \sum_{i \in C_v} Y_{i,v}$, where $Y_{i,v} \sim \text{Bernoulli}(\theta_{i,t})$ are conditionally independent given $\mathcal{F}_{v-1}$, follows a Poisson-Binomial distribution [5]. Using conservative probabilities $\underline{p}_{i,t} := \text{LCB}(i,t)$:

$$\mathbb{P}(K_v^* \geq q_{\text{full}} \mid \mathcal{F}_{v-1}) \geq \text{TailPB}\big(q_{\text{full}};\, \{\underline{p}_{i,t}\}_{i \in C_v}\big). \tag{20}$$

For correlated mobile failures, apply conservative shrinkage $\underline{p}'_{i,t} := (1-c)\,\underline{p}_{i,t}$ for calibration constant $c > 0$. The primary source for calibrating c is testnet connectivity data; network-local signals such as RTT latency and ϕ-accrual failure rates — which cannot enter scoring but are observable off-chain — provide a cross-check. If pre-launch testnet evidence is weak, a conservative default of $c = 0.1$ can be used as a placeholder, with the understanding that this may over-shrink liveness estimates and produce a tighter M_t than necessary.

9.2 Soft-Cap Design Objective

Scope. A *design objective*, not a proved theorem. The liveness constraint holds by definition of M_t. What is not proved is that the Poisson-Binomial bound is tight enough to be useful, that c can be reliably calibrated, or that $\mathrm{Rev}_q(M)$ is a sufficient proxy for validator economic viability across all market conditions.

$$M_t := \max\{M : \mathrm{TailPB}(q_{\mathrm{full}}; \{p'_{i,t}\}) \geq L_{\mathrm{target}} \ \wedge \ \mathrm{Rev}_q(M) \geq \mathrm{Rev}_{\mathrm{target}}\}, \tag{21}$$

where $L_{\mathrm{target}} = 0.95$ and $q = 0.25$ (25th-percentile validator profit floor). M_t is designed to move at most daily, with $|M_t - M_{t-1}| \leq \Delta M_{\max}$; the implemented controller instead recomputes it at every epoch boundary (every 20 blocks), with a step of at most 4 per epoch. Dynamic registry sizing — which bounds the eligible registry only, while the committee's seat budget stays a fixed constant of the node software (§5) — is implemented as a design mechanism and activated progressively as the network accumulates live data; until activation the controller is inert: both targets ship at zero — the sentinel that disables the feasibility search — so the computation returns the structural maximum and the hard registry ceiling $M_{\max}$ is the only bound in force, itself a node constant changed by a client release (Remark 17.1).

10. Tentative Blocks

10.1 Protocol

Mobile churn may cause the full quorum q_{full} to be temporarily missed, stalling the chain. Tentative blocks address this: the chain continues making progress under a weaker certificate, and the full QC retroactively finalizes those blocks once quorum is restored.

Definition 10.1 (Tentative Certificate (TC))**.** *A TC for view v is an aggregate BLS signature plus signer bitmap proving $\geq q_{\mathrm{tent}} < q_{\mathrm{full}}$ committee members signed the proposed block.*

Current configuration. In the current configuration the tentative quorum is set to $q_{\mathrm{tent}} = f + 1$ over the voting seats — 3 at seven anchors with no graduate seated ($n = 7$, $f = 2$, $q_{\mathrm{full}} = 5$), 4 with the three graduate seats filled ($n = 10$, $f = 3$, $q_{\mathrm{full}} = 7$). This choice has a simple structural property: because at most f committee members are Byzantine (Assumption 3.1), every TC necessarily carries at least one honest signature. A dynamic tentative quorum — which adjusts q_{tent} within bounds fixed in the node software ($f + 1 \leq q_{\mathrm{tent}} \leq q_{\mathrm{full}}$) based on the measured reliabilities of the validators in the voting seats and a split-vote rate that is a node-software constant — is implemented in the node software and is activated progressively as the network accumulates live data.

10.2 State Execution on TC Blocks

Phase adopts a differentiated execution policy for TC blocks. The probabilistic safety bound (Claim 10.1) gives the per-view fork probability as a function of q_{tent}: at the thresh-

old band the feasibility search targets it is $P(\text{Bad}_v) \leq 10^{-10}$ per view, and for most transactions executing normally on TC blocks is then safe in expectation. That bound does not describe the configuration in force today: at the static $q_{\text{tent}} = f + 1$ we have $q_{\text{tent}} - f = 1$, so (22) requires only one honest vote on each side of a fork, and what protects tentative execution is the structural property stated above — every TC carries at least one honest signature — rather than a numerical rarity. For operations where even a negligible fork probability creates disproportionate harm, execution waits for full QC.

Table 3: Execution Policy by Transaction Type on TC Blocks

Transaction type	**TC block**	**Rationale**
Ordinary transfers	Execute, `tentative` status	Reversible; unwound with the branch
Contract execution	Execute, `tentative` status	Deterministic replay; unwound with the branch
Validator reward release	Executes; pays up to the watermark	Own reward waits; proposer-committed (§6)
Validator-set and parameter changes	Execute, `tentative` status	Unwound with the branch on a TC fork; see below

In the design, client software and off-chain systems observe the `tentative` status on TC blocks, and applications requiring settlement certainty wait for the subsequent full-QC upgrade to `final`. That status is not implemented: nothing on the block, in storage, or in any block-returning RPC response marks a block as tentative or upgrades it when a certificate forms, and `safe` and `final` move only at finalization (Remark 10.2, which treats what a client can and cannot see in full). On a TC fork, state is unwound with the branch down to the common ancestor — never below the node's finalized block (the paragraph on what a TC fork costs, below) — not to the last full-QC block. Validator reward release is gated on the height being paid, not on the block that pays it: a TC block still runs the payout path and credits ancestors at or below the committed finality watermark it carries — on a chain of honest proposers, the heights a full QC has finalized under the 3-chain rule (Remark 6.1) — into the ordinary, not-yet-final state of that block, from which the credit reverts with the branch and is re-emitted on the surviving one (§6.2). The hold is therefore real but narrow — it withholds the reward *for* a height the watermark has not yet reached, not reward settlement *on* a tentative block. Validator-set and parameter changes carry no hold of either kind: the epoch-boundary transition — jail release, probation graduation and timeout, churn-limited admission, and the write of the next epoch's committee snapshot — executes with the block on the sole condition that the height is an epoch boundary, and is reverted by the journalled unwind if the tentative branch is abandoned. What bounds the exposure is a lookahead rather than a hold: the snapshot written at block $20E$ governs epoch $E + 1$, which begins at block $20(E + 1)$, so in normal operation the full QC arrives well inside that twenty-block lead.

Remark 10.1 (The lookahead is not a guarantee)**.** *The margin above is a timing property, not an invariant, and it is thinnest in exactly the regime tentative certificates exist for. A stall lasting longer than one epoch pushes the full QC past the twenty-block lead, so admissions, graduations and the next committee are computed from — and take effect on — state that has never obtained a full QC. Safety then rests on the unwind rather than on the ordering: if that branch is abandoned the writes are reverted, but a committee seated from unfinalized state may have acted in the meantime. Making this a genuine hold would require deferring the boundary transition until a full QC covers the boundary height — a stronger condition than the reward path's, which defers on a proposer-committed watermark (Remark 6.1) rather than on a certificate.*

Remark 10.2 (The head advances before any vote is counted)**.** *Execution and head adoption are not gated on q_{tent}. A node that accepts a valid proposal executes it and adopts it as its local fork-choice head immediately: a follower the moment the proposal's justification has verified against a committee it holds, with no vote counted at all; the leader as it proposes, holding only the vote it has just signed itself. From that moment the block is served as canonical chain state to* `latest`*-tagged queries. The head moves whether or not the node can then vote — if the parent state is unavailable the node executes and adopts the block but does not vote on it, because that is the path by which it recovers the state it is missing — but a proposal whose justification it cannot verify never reaches the execution layer.*

No block-returning interface distinguishes such a block from one that has since obtained a certificate: the block, the state it is served from, and every RPC response that returns it are identical either way. The `tentative` *status of the table above, and of Figure 3, is a design property and not yet an implemented one: no marking is written to the block, to storage, or to any block-returning RPC response when a certificate forms, and* `safe` *and* `final` *move only at finalization. The distinction is reported to clients in exactly one place, the consensus telemetry endpoint* `phase_getViewRecords`*, which returns one record per certificate the node formed or observed, each carrying the certifying view, the block hash, and a flag saying whether that certificate was tentative or full; a view that times out instead yields a record naming no block. It is telemetry rather than a status: it is keyed by view and served best-effort from a bounded in-memory ring buffer rather than from state, a certificate whose signing committee the node can no longer reconstruct leaves no record at all, and a view that closes tentatively and is later upgraded emits a second record, which replaces the tentative one when it is still the most recent entry and sits beside it otherwise — so a view may carry two records, and the flag on each is the certificate that record saw, not the one under which the block was executed. The class of blocks bearing executed, client-visible, revertible state is therefore strictly wider than the TC class of Figure 3 — it also contains every proposal that never reaches q_{tent}, by Table 4 some* 1.5–3% *of views at the candidate thresholds. Nor is such a proposal discarded when its view times out: the timeout advances the view and clears the vote pool, and leaves the block tree, the fork-choice head and the execution layer untouched. The uncertified block stays the node's head, and keeps being served, until a certificate for a competing block arrives and reorgs it away.*

Two things bound the exposure. Chain progression is certificate-gated: a leader extends only the block named by the highest certificate it holds — which may itself be a tentative one — so an uncertified proposal is never built upon, and the reward and view-advance rules are untouched.

And executing before the vote is structural rather than a choice, since the vote signs a computed state root. What does not carry over is the bound: ε in Claim 10.1 is the probability of a fork between two competing tentative certificates, and it says nothing about a sub-q_{tent} proposal a client may already have acted on. An application that must not act on revertible state waits for `final`*, because below that no block-returning interface tells the two regimes apart: the only discriminator the node exposes is negative and out-of-band — the absence, among the telemetry records above, of any certificate naming that block — and it is best-effort and bounded by the ring buffer's retention rather than by the chain.*

What a TC fork costs. A tentative-certificate fork can unwind only down to the node's finalized block, never below it: a block at or below the finalized height is not admitted to the block tree, and the forkchoice update names the finalized block, which the execution layer refuses to drop. Within that bound the node does not revert to the last full-QC block either: it switches its head to the competing tip, and the execution layer reorgs to it, unwinding as far as the common ancestor of the two branches — a block that nothing requires to carry a full QC, because certificates are ranked by view first and a full certificate is preferred over a tentative one only as a tie-break between equal views. A higher-view tentative certificate displaces a full one only over a held block that strictly extends the block of the highest full QC the node has verified, so it never takes the head from that block: the common ancestor lies at or above it. What a tentative certificate cannot do is move finality in either direction: finalization requires three consecutive full quorum certificates in consecutive views (the 3-chain rule), so no tentative certificate moves the finalized height, and none demotes a finalized block. Outside the fault model, a newer conflicting full quorum certificate demotes this node's finality: it reorganises onto the strongest branch, re-finalizes there and reports the reversion loudly; an older conflicting three-chain is refused and the node keeps its finality.

10.3 Probabilistic Safety: Assumptions and Limitations

Scope. A proof sketch. The split-probability model assigns each honest online validator an independent probability ρ_s of voting for a conflicting proposal. This does not fully specify fork-choice behavior, abstention, or routing under network partition. The bound is a heuristic risk estimate, not a derived protocol theorem.

Definition 10.2 (Bad event)**.** Bad_v*: two conflicting tentative blocks at the same height both obtain a TC before any full QC resolves the fork.*

Claim 10.1 (Probabilistic Safety Bound, Proof Sketch)**.** *Under Assumption 3.1, the Poisson-Binomial vote model, and the modeling assumption that each honest online validator votes for a conflicting block independently with probability $\rho_s \in [0, 1/2]$:*

$$\mathbb{P}(\text{Bad}_v) \leq \sum_{k \geq 2(q_{\text{tent}} - f)} \mathbb{P}(K_v^* = k) \cdot \mathbb{P}\big(A \geq q_{\text{tent}} - f \wedge B \geq q_{\text{tent}} - f \mid k, \rho_s\big). \tag{22}$$

For any $\varepsilon > 0$ and $1 - \delta_{\text{live}}$, and for sufficiently small ρ_s, a feasible q_{tent} satisfying both constraints

exists and is found by binary search over $\{1, \ldots, q_{\text{full}} - 1\}$.

This claim does not constitute a full BFT safety proof under tentative quorum. *Full HotStuff safety applies exclusively at* q_{full}*, and there only while quorum intersection holds over the eligible registry (§3).*

A proof sketch appears in Appendix C. The numerical analysis below is illustrated at the wire-format ceiling ($n = 32$, $f = 10$, $q_{\text{full}} = 22$), an illustrative large committee rather than the shipped one (§3); the same feasibility search applies at the shipped $n \in [7, 10]$, whose candidate thresholds are not tabulated here.

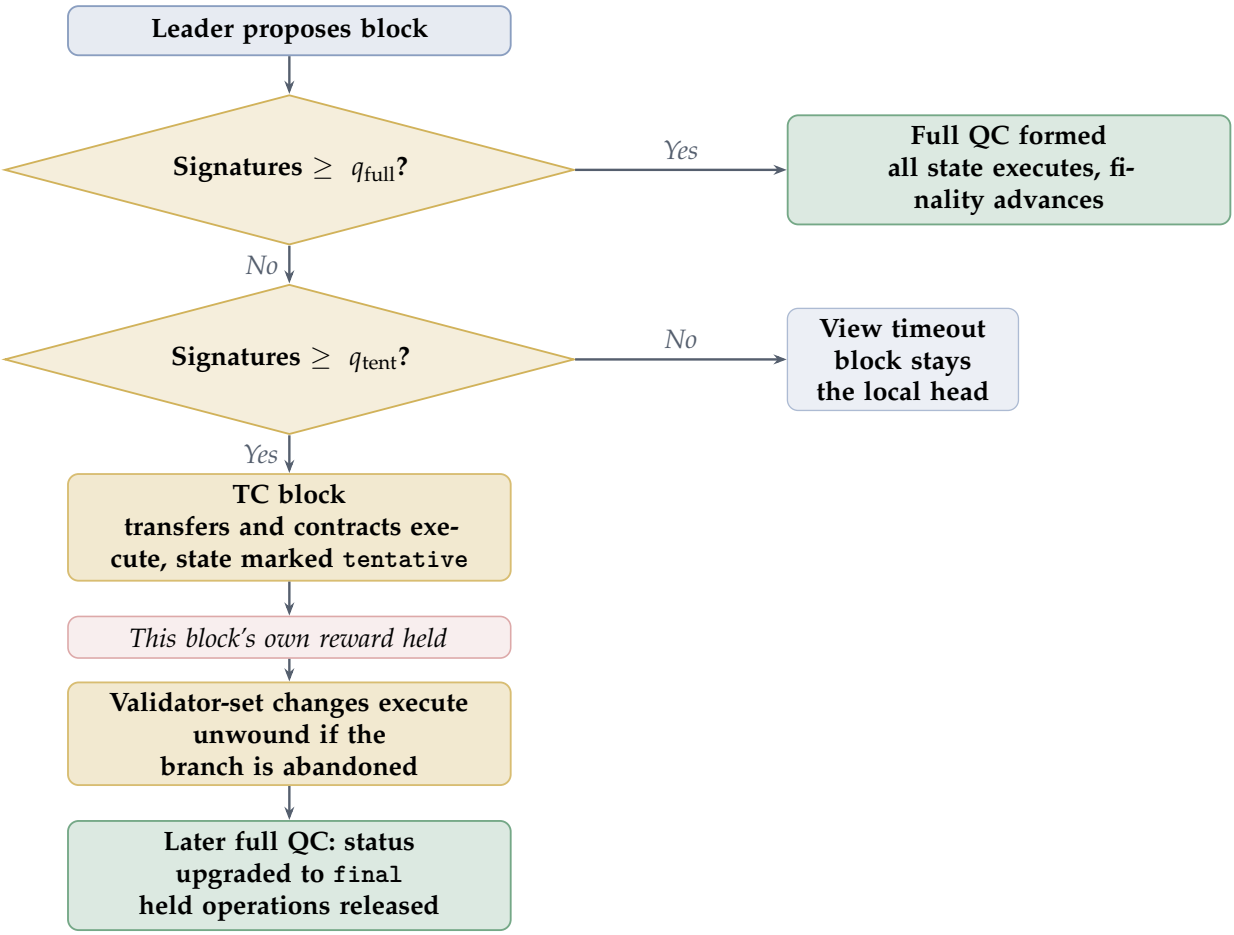

Figure 3: TC protocol (differentiated execution). Ordinary transfers and contract execution proceed on TC blocks with tentative status. A TC block's own reward waits for the committed finality watermark to reach its height (Remark 6.1), though the block still runs the payout path for the heights at or below the watermark it itself commits; validator-set changes execute with the block and are unwound if the tentative branch is abandoned. The tentative path never lowers the full-QC threshold itself; what bounds full-QC safety is the registry-size condition of §3, which nothing in this figure affects. The two decision nodes govern view advance, finality and reward release, not whether the block executes: a valid proposal is executed and adopted as the accepting node's local head before any certificate exists, and the timeout branch leaves that state in place rather than discarding it (Remark 10.2).

Table 4: Tentative Threshold Candidates (illustrated at the wire-format ceiling $n = 32$, $q_{\text{full}} = 22$, not the shipped committee)

q_{tent}	$\mathbb{P}(K_v^* \geq q_{\text{tent}})$	$\mathbb{P}(\text{Bad}_v)$	Assessment
22 (full)	78%	≈ 0	Liveness too low
18	92%	$\lesssim 10^{-15}$	Liveness marginal
16	97%	$\lesssim 10^{-12}$	Acceptable
15	98.5%	$\lesssim 10^{-10}$	Acceptable
14	99.2%	$\lesssim 10^{-8}$	Safety near margin
12	99.8%	$\lesssim 10^{-5}$	Safety violated

The safety budget — the per-view probability that two conflicting blocks both obtain a tentative certificate, which the protocol is willing to tolerate in tentative mode — is a node-software constant, not a governed parameter (§17), and is distinct from the explore fraction $\varepsilon_{\text{explore}}$ of §5. The value $\varepsilon \leq 10^{-10}$ is the target at activation, not a bound in force today: the shipped safety and liveness budgets are both zero, the sentinel that disables the feasibility search and holds q_{tent} at the static floor $f + 1$. It bounds forks within the TC class only, and says nothing about a proposal that never reaches q_{tent} yet whose executed state was served (Remark 10.2). The candidate table shows that at that illustrative committee the budget admits a comfortable band of thresholds ($q_{\text{tent}} \in [15, 16]$) under the stylized split model; the operative caveat is that ρ_s, the real-world split-vote rate of mobile validators, must be measured on the live network before the dynamic tentative quorum is activated.

11. Networking

Every mainstream L1 networking stack was designed under the assumption that a validator is a server: a public IP address, open inbound ports, and a stable link. A phone satisfies none of these. It sits behind carrier-grade NAT with no public address, its IP changes when it moves between Wi-Fi and LTE, its operating system aggressively suspends background connections, and its data plan makes every redundant byte a real cost. For a network whose validators *are* phones (§3), the transport layer is not an optimization target; it is an existence condition.

11.1 Transport: iroh over QUIC

Phase replaces the standard Ethereum networking stack (devp2p) entirely with iroh,[1] a production peer-to-peer transport built on QUIC. Three of its properties are load-bearing for mobile validation:

- **Identity-based addressing.** Nodes are addressed by an ed25519 public key, not an IP address ("dial keys, not IPs"). A validator's network identity is stable across address

[1] https://www.iroh.computer

changes and is the same identity registered on-chain (§13); the master key of the device deterministically derives the wallet, BLS, and network keys.

- **NAT traversal with relay fallback.** Connections are established by QUIC hole punching where possible; where a direct path cannot be punched, traffic transparently falls back to relay infrastructure. A phone behind cellular NAT is reachable in either case. Relays and identity discovery (pkarr publishing and DNS lookup) are Phase-operated services at launch; bootnodes and anchors are reachable from a compiled-in static table, every other node only through discovery.
- **Network-switch survival.** QUIC path migration lets an established connection survive the device moving between networks. A validator switching from Wi-Fi to LTE does not drop out of its committee; the transport re-validates the new path underneath the running protocols.

To our knowledge, Phase is the first L1 whose complete networking stack—consensus, block and transaction gossip, state synchronization, and bootstrap—runs over iroh. We state this as an observation about deployments known to us, not as a verified census; iroh itself is a mature transport with substantial non-blockchain production use, so the novelty claimed is architectural, not technological.

11.2 One connection per peer, demultiplexed by stream tag

All node-to-node protocols share a *single* multiplexed QUIC connection per peer. Each QUIC stream begins with a one-byte protocol tag that routes it to the appropriate handler: a lightweight heartbeat carrying liveness and the peer's current chain height; a request/response protocol for headers, bodies, transactions, and snapshot chunks (§12); gossip for blocks and transactions; the HotStuff message family (proposals, votes, NewView, Advance and Decide, and the block request/response pair); and blob-sidecar distribution between well-provisioned nodes, a path that carries nothing while blobs are switched off on that chain (Remark 13.1). The connection handshake verifies the genesis hash, so peers of an incompatible network are rejected before any protocol traffic flows. The proxy service (§14) runs on a second iroh endpoint of the same node, under its own identity and its own ALPN, with its own UDP sockets, relay link and discovery record; it shares the host, the process runtime and the uplink with protocol traffic, but none of the node endpoint's peer slots, windows or read budgets.

The practical consequence for a phone is that it maintains *one* NAT binding, one TLS handshake, and one keepalive per peer, rather than one per subprotocol. This reduces battery drain, shrinks the state the carrier's NAT tables must hold, and makes recovery after a network switch a single reconnection rather than a cascade.

11.3 Beacons: signed-message relays for committee traffic

Two phones frequently cannot reach each other directly even with hole punching. Phase therefore designates a set of well-provisioned nodes—*beacons*—that relay consensus mes-

sages between committee members lacking a direct path: a validator sends a framed relay request to the two nearest connected beacons by measured round-trip time—or, when the beacon that last delivered a message the target signed did so within the last minute and is still connected, to that beacon and the nearest other—and the beacon forwards the enclosed message to the target; a block response instead goes only through the beacon that last delivered a block request from the requester, when that beacon meets the same two conditions. Beacons also serve state snapshots (§12) and re-broadcast the proposals, NewView, Advance and Decide messages a peer sends them directly (one forwarded to them goes no further) to the peers connected to them, the committee's voting seats first—and, once a beacon knows the current committee, only proposals go beyond the other beacons and the voting and observer seats of the previous, current and upcoming committees, since the next proposal carries its leader's highest certificate as its justification; block gossip is not relayed by any node today—the receiving swarm exposes an engine stream for it that nothing consumes—and transactions travel a bounded path: a phone sends only its own transactions, to at most four beacons, and relays nothing it received; a transaction crosses a second hop only through a beacon's pool, which re-propagates it to the committee's voting seats and the other beacons.

Beacons are accelerators, not trusted parties. Every consensus message must authenticate to its recipient before it can move that recipient's consensus state, and the seven message kinds — *Prepare*, *Vote*, *NewView*, *Advance*, *Decide*, *BlockRequest* and *BlockResponse* — do not all authenticate the same way: proposals, votes, NewView messages and block requests carry a BLS signature by their originator, while *Advance*, which broadcasts a newly formed certificate, *Decide*, which announces the full QC that finalizes a block, and block responses carry no originator signature at all; the first two are authenticated instead by the aggregate BLS signature inside the certificate they carry, which the recipient verifies against the committee that formed it (Remark 11.1), and a block response by the hash the requester asked for and the certificate it carries. A certificate whose committee the recipient cannot yet reconstruct—the normal condition of a node that is catching up—is not checked by that recipient, and may drive catch-up only: it moves no lock, no view, and no finalized frontier. Reconstructing a committee the node does not already hold is itself budgeted—a fixed number of reconstructions per window, so that a peer cannot make the node rebuild a committee for every message it sends—and a certificate that would need a fresh reconstruction after that budget is spent is not checked at all—one naming the round the node is already on is served from its cache and still verifies, which is why the bound is invisible to a correct peer: it is refused rather than admitted unverified, and on the proposal and *Advance* paths the message carrying it is dropped outright. Transactions carry their own signatures. A beacon can delay or drop traffic—a liveness concern bounded by view changes and committee rotation (§10)—but it cannot forge a vote, alter a proposal, or impersonate a validator. The beacon infrastructure is operated by the Foundation at launch, as are the relays and the identity-discovery services above; all of them are a Foundation-run availability dependency on the decentralization roadmap, and broadening the beacon set is part of it (§17).

Remark 11.1 (What the certificate does not cover)**.** Advance *and* Decide *each carry a view number in the frame alongside the certificate, and that field sits outside the aggregate signature, so a relay can rewrite it. No handler reads it: the* Advance *handler takes the view from the certificate, the* Decide *handler uses none, and neither handler is passed a sender identity—the index the dispatcher resolves for the delivering peer feeds a per-peer rejection counter and the logs, nothing else. No consensus decision on these two messages therefore rests on anything but what the certificate itself proves. Binding the frame with an originator signature, as the four signed message kinds are bound, would close the gap by construction; that is not implemented.*

Committee handovers receive special treatment: whenever a node installs a committee it also attempts to *prewarm* connections to the members of the next epoch's committee (§5), and while the chain is stalled it prewarms the committee of the next stall-recovery round once that round is within thirty seconds of activating — half of the sixty-second period that promotes those rounds — so that the first views of a new committee do not begin with a burst of simultaneous hole-punching attempts. The next epoch's committee is prewarmed once its selection snapshot is written, about 18 blocks before the boundary; only the voting seats are dialled, and observers dial in themselves. The prewarm is best-effort and idempotent, dials only a committee that seats the node, as a voting member or an observer, and dials none while the node is catching up; what it has to warm is small in any case: the anchors are seated in every committee, so consecutive committees differ only in the graduate and observer seats and most of the connections the incoming committee needs are already open. Committee rotation during a stall is driven by wall-clock time elapsed since an on-chain anchor, read from each node's local clock, which SNTP must vouch for before the node derives a rotation round from it; a node whose clock cannot be vouched for keeps voting but holds its round, so a fleet-wide time-source outage suspends stall recovery, never consensus.

11.4 Bandwidth engineering

The stack assumes bandwidth is expensive and treats economy as a design requirement rather than a tuning pass:

- **Frame deduplication on the beacon paths.** On the blob-sidecar and consensus-re-broadcast paths, a beacon content-hashes the raw frame and tests the digest against a bounded LRU of recently seen digests, so a replay costs it a hash rather than the work behind the frame—the KZG check of a sidecar, or the fan-out of a re-broadcast. The two caches on the sidecars a beacon accepts—a replay cache per peer, and a cache, shared by all peers, of the frames it has already verified—and the cache on the consensus frames it re-broadcasts are beacon-side; the cache on the sidecars sent to the beacons runs on every node, holding a beacon's relays and its own sidecars and a phone's own, so a node sends the same bytes to the beacons once. On a received frame, each cache is consulted after the frame has been read in full and its message decoded, because the frame reader hands back the decoded message and the raw bytes from a single call, so nothing upstream holds the bytes any earlier. The signed-message relay

above is deliberately exempt: a validator sends the same signed vote to the proposer and to the next leader, and BLS signatures are deterministic, so identical bytes are legitimate there and a replay is charged against the relay budget rather than dropped. Block and transaction gossip carries no frame-level cache: duplicate transactions are dropped by the pool as hashes it already holds, and the re-propagation engine that would deduplicate multi-hop gossip is written but not yet wired into the node.

- **Per-peer seen-tracking.** For each peer, the node maintains a bounded LRU of the transaction hashes it takes that peer to hold: hashes the peer announced, transactions it broadcast to the node, transactions served to it on request, and transactions the node sent towards it—the last entered when the send is issued, not on delivery. The per-peer record of what a peer holds suppresses re-sends from a beacon, which skips a recipient it records as holding the transaction; a phone, which sends only its own transactions, consults it only to top up a fetch request already directed at a peer with other buffered hashes that peer announced. Block and proposed-header gossip consults no per-peer cache at all.
- **Forwarding without re-encoding.** A phone publishes its own transactions to a small, fixed number of beacons—at most four, and to no client peers—and no node forwards a block or transaction gossip frame it received; a transaction travels a second hop only as a fresh message from a beacon's pool, to the voting seats and the other beacons, as described above. Where a beacon does forward a frame—a client's blob sidecar on to the other beacons, a consensus frame on re-broadcast—it sends the bytes it received rather than decoding and re-encoding the message, and the re-broadcast fan-out is capped, with the committee's voting seats served first. Skipping the re-encode saves CPU and therefore battery.
- **Bandwidth-aware peer selection.** Download throughput per peer is tracked with an exponentially weighted moving average; the block downloader's header and body requests are routed ε-greedily—the large majority to the currently fastest peers, a small exploration fraction to others so the estimates stay current—while each snapshot chunk goes to the least busy serving peer with room for it.
- **Hard frame caps.** Each protocol class bounds frame size, checked against the frame's declared length before any payload byte is read: 64 KiB on the control plane (handshake and heartbeat), 1 MiB on inbound requests, 8 MiB on blob sidecars, 10 MiB on consensus frames and on the length any gossip frame may declare, and 20 MiB on responses to requests the node itself issued. Within those, a gossip frame is held, from its kind byte and before its body, to just over 2 MiB for transactions and 172 KiB for a hash announcement, the retired block kinds being refused outright, and a response to its request kind's ceiling—4 MiB for headers, pooled transactions or blob sidecars, about 16 MiB for block bodies, and the full 20 MiB only for a snapshot manifest or chunk. Consensus sits at the *loose* end of that range rather than the tight one—only responses the node asked for are looser—and deliberately so: a proposal carries a whole block, so a cap below what an honest proposer can emit would drop proposals,

a liveness failure worse than the flooding a tighter cap would prevent. Only a proposal or a block response may approach the 10 MiB cap; every other consensus kind is refused above 256 KiB before its body is read. A phone is protected less by the size of its cap than by how the body is read: an over-cap declaration is refused before the body, an under-cap one buys nothing because bodies accumulate in doubling slices that start at 4 KiB (128 bytes for a frame declared longer than 16 KiB that is not an inbound request), and in-flight reads draw on three separate budget lanes—control plane, consensus, and everything else—so bulk traffic cannot starve a consensus or heartbeat read.

- **Peer reputation and banning.** Each peer carries a local score; timeouts, malformed frames, and invalid snapshot chunks are penalized, and peers falling below a threshold are banned—except a configured beacon, which is logged and has its score reset instead, on the grounds that isolating oneself from a relay is worse than tolerating a noisy one; a genesis anchor that is not a beacon is, once a committee is installed, spared a score-driven ban the same way, since the anchors are seated in every committee, though a chain mismatch still bans it. The penalty is applied only where the observing code holds a handle to the peer table; two receive loops do not, so on those paths a malformed frame costs the sender nothing (Remark 11.2).

This transport-level reputation is deliberately distinct from the consensus reputation of §4: it is network-local, differs between observers, and therefore never enters committee selection, consistent with the consensus-measurability constraint of §3.

Remark 11.2 (Two receive loops are unscored)**.** *Penalties are deducted only where the observing code holds a handle to the peer table: the handshake, the heartbeat, the stream router, the requesting side of request/response (bad responses, malformed manifests, invalid snapshot chunks), consensus receive on every node—a client's consensus receiver meters, caps and charges through the same ingress gate as a beacon's—and, on a beacon, blob-sidecar receive. Two receive loops deduct nothing: block/transaction gossip and the serving side of request/response. On those, a malformed frame or a read timeout is logged—rate-limited on the gossip loop (both the decode and the timeout branch), since the input is remote-controlled, but not on the serving side's non-overload failures, which emit a line per occurrence—and, on the gossip loop, counted and answered with an exponential, capped backoff once consecutive failures pass a threshold; but the peer's score is untouched and it is never disconnected for the fault. For the gossip loop this is an implementation gap rather than a design choice: its own documentation names the penalty call as the right response and records that the loop, as constructed, cannot reach it. The serving side of request/response gives no reason either way; it bounds a hostile peer by a read deadline and serving-capacity limits rather than by score. One adjacent decision* is *deliberate: the gossip loop never disconnects a peer for a frame it cannot parse. A gossip frame is a tag byte followed by an RLP body and carries no protocol version, so a peer mid-upgrade and a peer sending garbage are indistinguishable, and cutting such peers off would partition the fleet during a rolling upgrade. The consensus path answers the same question without exempting a malformed frame's sender from scoring: its frames carry a version byte, so a*

skewed frame is refused and counted, with nothing reported against its sender for the skew, while a malformed frame is still charged, on a beacon and on a phone alike.

12. State Bootstrap: Snapshot Synchronization

A classical node bootstraps by downloading the full block history and re-executing every transaction from genesis. On a phone this is not slow; it is impossible in any practically relevant sense, and a node that takes days to appear can never sustain meaningful participation. Phase instead bootstraps from *verified state snapshots*: a new node downloads the network's state at a recent finalized block, re-executes none of the history up to it, and executes only the blocks between it and the network's tip.

12.1 Export

Nodes configured as snapshot exporters—primarily beacons (§11)—export the full state at the network-configured cadence, a fixed block count of 86,400 blocks on mainnet and 3,600 on every other network (daily and hourly at the nominal 1 s accounting unit of §3): account balances and nonces, contract storage and bytecode, the validator registry, and the consensus tables that committee selection and scoring are derived from. Derived structures—the hashed state tables and the Merkle trie—are deliberately *not* exported: the importer rebuilds them from the plain state it received, which is what lets the recomputed state root act as a non-circular proof of everything imported (§12.4). Snapshots are taken *only* at finalized blocks, so no reorganizable state can enter a snapshot. An export consists of a manifest—block number and hash, state root, the chain's genesis hash, the encoded block header together with a window of ancestor headers (serving the `BLOCKHASH` history below the snapshot), the list of all chunks with their hashes and sizes, and an optional publisher signature (§12.5)—plus the chunks themselves: serialized state segments, compressed with zstd, at 2 MB of uncompressed payload per chunk. Chunks are content-addressed: a chunk's identifier is the Keccak-256 hash of its compressed bytes, so the name is the checksum.

12.2 Manifest selection by quorum vote

A bootstrapping node does not trust the first manifest it receives. It queries its serving peers—the beacon set (§11)—clamps candidate heights to at most 400 blocks above the network tip, the highest height any connected beacon reports (unbounded when none reports one), and accepts a manifest only when a quorum of peers, at least two and a strict majority of all responders, agrees twice over: on the manifest's identity (its block hash, state root, and genesis hash) and on a digest of its full content, chunk inventory included. If no height reaches quorum, the node retries at earlier export epochs and ultimately falls back (below) rather than resolving a conflict heuristically. A single compromised or corrupted serving node therefore cannot steer a new node onto a fabricated state, and a quorum member cannot substitute chunks while echoing an honest identity.

12.3 Verified parallel download with resume

Chunks are fetched in parallel from multiple serving nodes with many concurrent streams, load-balanced toward the least-busy and fastest peers (§11). Every chunk is verified on arrival against the size and hash recorded in the manifest; a failed check triggers a bounded number of retries from *different* peers, and the offending peer is penalized through the transport reputation mechanism; since every source is a configured beacon, the penalty lowers its score but never bans it. Completed, verified chunks are never re-downloaded: an interrupted connection, a killed app, or a reboot resumes the download at the granularity of a single chunk.

12.4 Checkpointed import and tail catch-up

Import is not a special mode; it is a stage of the node's staged execution pipeline (§13). The stage proceeds through explicit phases—bytecode, accounts, contract storage, the validator registry, and the consensus tables—and persists both the current phase and the index of the last applied chunk as a stage checkpoint. After the last chunk, the hashed tables and the Merkle trie are rebuilt locally and the recomputed state root is proved against the snapshot header before the import activates. The import consequently survives process death at any point: the user can background the app, lose signal, or reboot the phone, and the import continues from its checkpoint rather than restarting. This property is not cosmetic; on mobile, being killed mid-way is the common case.

After import, the node synchronizes the remaining *tail*—the blocks between the snapshot height and the network's tip—through the ordinary sync path and joins consensus. Because snapshots are exported on a fixed network-wide cadence, the tail is normally at most one export interval, plus the blocks produced while the next export is still being written and while the node downloads and imports the snapshot. Each scheduled export that the beacons skip because the previous one is still running, and each fallback to an older export (§12.5), adds one more interval; ordinary sync closes the tail in every case. The snapshot path engages only when a node is behind by more than a configured distance; a briefly offline node simply resumes normal sync.

12.5 Degradation and trust model

If the newest snapshot cannot be downloaded, the node tries it again—three attempts in all, keeping the chunks it has already verified—then falls back to the previous export epoch, and ultimately, after three failed snapshots or a 30-minute deadline on the whole download, to full synchronization. A node that selects no manifest within 180 s, cannot store the snapshot, or whose imported state fails verification (§12.4) goes straight to full synchronization. The fast path is an optimization, not a dependency.

Two properties are worth separating. *Integrity* is cryptographic: every byte imported is checked against the manifest's content hashes, so a serving node cannot substitute data within an accepted manifest. *Manifest choice* rests on the two-vote quorum above: the bootstrap assumes a majority of the responding serving peers are honest at the moment of

selection. Operators may instead configure *trusted publishers*: a manifest carrying a valid signature from a configured publisher key—a dedicated key, neither a validator key nor a network identity—is accepted in place of the quorum at its own height, but only while fresh relative to the network tip, so a historically signed snapshot cannot be replayed onto a new node. Selection still runs highest height first, so a quorum at a higher height is adopted before a signed manifest is examined: a publisher lets a node bootstrap from a single source, but does not protect it from a colluding majority of its serving peers. With no publishers configured, no single source is ever accepted. Both are explicit, weaker-than-consensus assumptions; an operator who rejects them can run full sync, which requires no such trust. In the current deployment the serving set *is* the Foundation-operated beacon set, and the relays and discovery services that every node outside the static table depends on to be reached are Foundation-operated too (§11); widening both is part of the decentralization roadmap.

A consequence worth stating plainly: sync time is independent of chain age. The chain can run for years, and a new phone still downloads only the current state—a volume comparable to a large mobile-app update—rather than a history that grows without bound.

13. Execution Layer

13.1 A Reth-based execution client

Phase's execution layer is built on Reth, a production-grade Ethereum execution client written in Rust, and runs the Ethereum EVM with the fork set through Cancun—Shanghai and Cancun are active from genesis, and no later Ethereum fork is scheduled. Contract semantics and Ethereum's standard transaction types behave as on Ethereum: existing contracts deploy and execute unchanged, and the submission path—signing a transaction and sending it—is unmodified. The one EVM-level addition is a protocol precompile at `0x...0080` that answers validator-registry queries, described below. Two of Ethereum's transaction types are exceptions: EIP-7702 set-code transactions (type `0x04`) are refused on every Phase chain, because Prague—the fork that introduced them—is not in the fork set, and blob-carrying EIP-4844 transactions (type `0x03`) are refused on mainnet, for the reason Remark 13.1 gives. The block, header, and receipt surfaces are *not* Ethereum's, and that remark sets out where they differ; tooling that decodes blocks or receipts rather than only submitting transactions needs a chain-specific extension. Building on Reth rather than a bespoke VM is a deliberate risk decision—EVM correctness is inherited from a widely reviewed implementation instead of re-earned—and it provides the staged synchronization pipeline into which snapshot import integrates as a native stage (§12).

Remark 13.1 (Where the execution layer departs from Ethereum)**.** *Five differences are visible to anything that reads blocks and receipts rather than only submitting transactions.*

A reduced header. *The consensus header is Phase-specific: it adds* `validatorsHash` *and* `scoringHash`*, the commitments to the body's validator array and scoring payload, and it has no* `withdrawalsRoot`*,* `blobGasUsed`*,* `excessBlobGas` *or* `parentBeaconBlockRoot` *field, nor*

`mixHash`, `nonce` or `sha3Uncles`. The block object returned by `eth_getBlockByNumber` is that header's fields directly, with constant `sha3Uncles`, `mixHash` and `nonce` values restored for compatibility: it carries two fields Ethereum blocks do not and omits the first four. With no `excessBlobGas` to price blob gas from, `eth_blobBaseFee` returns an error rather than a value.

No withdrawals and no beacon root. *EIP-4895 withdrawals and the EIP-4788 beacon-roots system call are both couplings to a beacon chain. Phase has none, and neither mechanism is implemented on any path: no pre-execution call touches the beacon-roots contract, and the block body has no withdrawals list.*

No blob fee market, and blobs off on mainnet. *EIP-4844 transactions are part of the envelope, but with no `excessBlobGas` in the header there is nothing for a fee market to feed back into; on a chain where blobs are switched on, blob gas is priced at a flat 1 wei per unit—131,072 wei for a whole blob—at any level of demand. Blobs are moreover off unless a chain's specification names an activation timestamp, and mainnet's does not: its per-block blob count is zero, so the transaction pool and the block builder reject every blob-carrying transaction. Switching blobs on is a chain-specification entry—it carries no fork identifier and adds no hardfork variant—but it is still a consensus-rule change that every node must adopt in lockstep: a node still running the blobs-off schedule rejects any block carrying a blob transaction.*

A floored base fee, and no priority fee to the proposer. *On a chain whose genesis configuration sets a minimum base fee—mainnet and fabric, both at 1 gwei—the EIP-1559 base fee never falls below that value, however empty the blocks: it still rises on blocks above the gas target and falls back after them, but stops at the floor instead of decaying towards zero. A transaction priced under the floor is never included. The priority fee, which Ethereum pays to the block proposer, is burnt with the base fee, so the fee a transaction pays is destroyed in full (§16). Validator registration is not affected: it runs at a zero gas price that no base fee is compared against.*

A protocol transaction type. *Validator registration, described below, is an EIP-2718 typed transaction at type identifier `0x7d`, a value within the EIP-2718 type space that Ethereum has not assigned. It does not arise in ordinary application traffic: it is built, signed and submitted by the node's own registration-renewal service—the pool accepts the type from any submitter, but no wallet or contract produces one. It is an ordinary member of the transaction list and surfaces over JSON-RPC as a receipt whose type field reads `0x7d`, a value no Ethereum receipt takes. The removal of pool members whose validator registration has lapsed (§15) is not a transaction: the block executor performs it before a block's first transaction, from state, and no block may carry the unsigned type `0x7e` that once did it. The block body additionally carries the proposal's validator set and its scoring payload.*

13.2 Consensus–execution separation in a single process

Consensus and execution are separated by an internal interface deliberately shaped like Ethereum's engine interface: consensus drives execution through fork-choice updates, payload construction, and payload validation, and execution reports back validity and state roots. Unlike Ethereum's two-process split, both sides run in a single process and the interface is a function boundary, not an RPC boundary. The separation preserves the ar-

chitectural discipline—consensus code cannot reach into execution state except through the defined interface, and either side can be tested against the other in isolation—while avoiding the memory footprint, IPC overhead, and operational complexity of running two clients on a phone.

13.3 Validator registration as a system transaction

Validator registration is a protocol-level system transaction with zero gas cost—there is no fee and no deposit (§3). A registration binds together the validator's Ethereum-style address, its BLS public key for consensus voting, and its network identity key (§11), and carries a proof-of-possession over the BLS key bound to the sender address and the chain identifier—the standard defence against rogue-key attacks on BLS aggregation, with the binding preventing replay of a registration across accounts or networks. All three keys are derived deterministically from the device's single master key, so a phone manages one secret.

A registration submitted as a system transaction is valid for 320 blocks and is renewed automatically while the node runs. The short TTL makes the registry self-cleaning for such registrations: an abandoned or dead device falls out of the active validator set within minutes, without governance action or explicit deregistration. Genesis validators are exempt: a validator declared in the chain spec's genesis validator set holds no lease, is never renewed automatically, is permanently active, and is skipped by the expiry sweep, so the lease never removes it from the eligible set, whether or not its device is still running; eligibility still runs through the status filter, which the genesis row satisfies as `full` from block 0 and which would drop a jailed validator regardless of its lease — a status nothing sets today (Remark 5.3). The exemption is a bootstrap carve-out that cannot be acquired—a registration submitted after genesis carries the number of the block it landed in, so no later transaction can produce such a row. It is given up only voluntarily, by the operator submitting a fresh registration from its own address carrying its existing BLS key, which converts the row into an ordinary leased one, and only from block 160 onwards — before that the renewal window has not opened and the fresh registration is refused; the automatic renewal daemon never performs that conversion. Converting the last anchor would end the anchor model permanently (§21); the operating rule forbids it and no code refuses it. The registry is maintained natively by the node, and the EVM sees it only through a mirror: the precompile at `0x...0080` never reads the node's registry tables, which each node persists at its own lag behind the executed head, but answers from the leases the block executor writes into EVM state from each block's own registration transactions, and from the chain specification for genesis validators, so its answer is a function of the block being executed and the same on every node. The validator-pool contract (§15) gates pool creation and joining on it for live active-validator status.

13.4 Protocol system contracts

A small set of protocol system contracts is deployed at genesis: the rewards contract encoding the block-reward emission schedule (§6, §16), the validator-pool contract (§15), two per-block record contracts holding, for a rolling window of the most recent 8192 heights, each height's beneficiary and the signer set of the full quorum certificate that justified that height, if one did (Remark 3.1), and, for the proxy layer, the settlement contract together with the billing token its balances are denominated in—itself a genesis-deployed ERC-20, on mainnet `USDT.e`, backed 1:1 by USDT and minted by the inbound bridge (§14). They are ordinary EVM contracts deployed behind an upgradeable delegate-call proxy pattern whose administration is held by a governance-controlled multisig (§17). The deployed multisig implements no timelock: an upgrade requires a threshold number of owner confirmations — a threshold the multisig can itself change through its own m-of-n process — and once that threshold is reached it can be executed immediately, with no enforced delay between confirmation and effect. This allows parameter and logic changes without hard forks, at the price of an explicit upgrade-authority trust surface; that surface is bounded by the governance framework and is scheduled to narrow along the decentralization roadmap. Two of them are part of the state transition rather than incidental record-keeping. In every block's pre-execution phase the executor writes that block's beneficiary and its asserted signer set into the two record contracts by system call, so both writes are committed into the state root; it reads them back when a finalized height's deferred reward is released, to resolve that height's payee and to split what remains of the height reward, after the pool-owner bonus (§15), between the leader and the signer set recorded at that height. That recorded set is the signer set of the certificate that justified the height, not the S_v of (12). Eviction from the 8192-height window is neither recoverable nor neutral: a height whose beneficiary record has been overwritten pays nothing at all, and one whose signer record has been overwritten still carves out the pool-owner bonus and then pays the leader its β_t share of what remains and nothing to the signers, with the signers' remainder never minted. An upgrade of either contract can therefore change which addresses a finalized height whose reward has not yet been released will pay; heights already paid are fixed by a paid-height watermark the contract enforces monotone — though that enforcement is itself implementation code behind the same upgradeable proxy, so an upgrade can lift it.

14. The Proxy Layer

Each Phase validator holds an asset that cannot be provisioned in a datacenter: a genuine mobile-carrier IP address, the class of traffic origin least likely to be blocked or challenged on the public internet. Simultaneously, AI agents—among the fastest-growing consumers of web traffic—are systematically refused when originating from datacenter ranges. The Phase proxy layer connects the two: a validator phone can opt in as a *metered egress node*, and agents route traffic through it, exiting to the internet from the phone's mobile IP. Usage is metered per session and settled on-chain against a prepaid balance. The wire-level

specification is published separately as the *Phase Proxy Protocol* technical specification; this section summarizes the design commitments relevant to the protocol as a whole.

Same transport, second service. The proxy is not a parallel product with separate infrastructure; it is a second service of the same node over the same iroh transport (§11), on a second endpoint under its own identity and ALPN. A client runs a local SOCKS5 listener; tunneled traffic traverses QUIC—with the same NAT traversal and relay fallback as consensus traffic—to the exit and leaves from the exit's own connection. Enabling the service is a node configuration flag.

Three design commitments. First, *billing is mandatory*: an exit refuses to serve without a configured billing service, and no connection is relayed unmetered. Second, *discovery is registry-only*: clients obtain exits from a registry of live, heartbeating participants (currently a 15 s heartbeat with 45 s expiry) rather than dialing arbitrary endpoints, so every reachable exit is a metered one. Third, *the exit never terminates TLS*: application-level encryption runs end-to-end between the agent's software and the target; the exit observes target addresses and byte counts—which it must, to meter and to enforce egress policy—but relays opaque bytes and cannot read or modify payload content.

Prepaid, watermark-idempotent settlement. Accounts pre-fund an on-chain balance in the protocol's billing token: a six-decimal, dollar-denominated ERC-20 fixed at a genesis address, hard-coded by both the settlement contract and the billing service, and the unit in which every price, deposit, charge and withdrawal is quoted. On mainnet the token is `Bridged PHASE USDT` (symbol `USDT.e`) and is backed 1:1 by USDT: units minted by the inbound bridge are backed by the USDT the bridge received at its deposit address on the source chains, and the 10,000 USDT.e genesis supply, allocated to the treasury, is backed by a USDT reserve held by Phase Labs. The bridge contract is the token's only minter and mints only against a confirmed source-chain deposit, within a per-deposit cap of 1,000 USDT and a 24-hour cap of 10,000 USDT; the upgrade multisig (`0x...93`, §13) administers the token and can reassign the minter. The bridge is one-way for now: USDT moves to Phase, but USDT.e cannot yet be withdrawn back to a source chain. Exits report cumulative per-session usage; billing folds reports into charges; and a settlement process periodically submits, per account, the *cumulative* charge total to the on-chain settlement contract. The contract debits only the difference between the submitted target and a monotone per-account watermark. Resubmitting a batch, or two billing services settling overlapping charges, debits zero the second time: double-charging is structurally impossible rather than operationally avoided. The unspent remainder of a balance is withdrawable on two paths: the settler approves pending withdrawals in the normal course, and failing that the account holder may claim the remainder directly once a fixed dispute window has elapsed since the request. The second path is what makes the first non-custodial — settlement may charge the account right up until the claim, and the payout is clamped to whatever deposit survives, so the escape releases only what settlement has already left behind. Until a network's billing contract is upgraded to include it, releasing a remainder depends on the settler acting.

Egress protection. The exit enforces a normative egress policy on every connection: refusals cover loopback, RFC-1918 private ranges, link-local and cloud-metadata addresses, carrier-grade NAT space, multicast, and their IPv6 counterparts—including IPv4 addresses embedded in IPv6 forms (mapped, NAT64, 6to4), which are unwrapped and re-checked against the IPv4 rules. Name resolution happens exactly once, and the connection is made to the already-verified address, eliminating DNS-rebinding time-of-check/time-of-use attacks. The validator's phone can therefore not be used to reach its own node, its local network, or its cloud environment. Connection caps and rate limits separately bound how much proxy work the node admits: a global cap on concurrent proxy connections (default 50), a cap of eight connections per peer identity, and a per-account token bucket on new tunnel requests (default 100 per second). These bound sessions and tunnel-open rate, not bytes. No bandwidth or throughput budget for relayed traffic is implemented—the only volume bound is economic, a session being cancelled once its prepaid balance is exhausted—and the exit's own endpoint separates the paid service from protocol traffic only below QUIC: the two endpoints share the host, the process runtime and the uplink, with nothing ordering one behind the other. These caps therefore do not by themselves keep proxy load from competing with the node's consensus duties for CPU and bandwidth, and no other mechanism in the implementation does; what the separate endpoint does buy is that proxy connections take none of the node's peer slots, windows or read budgets.

Attribution and economics. The exit runs on a second iroh endpoint with its own identity, derived from the same master key as the validator's but distinct from it and not bound on chain; every session is attributed on the billing ledger to that exit identity, so each metered byte is attributable to a specific device, though not through the validator registry of §13. Proxy fees are designed to flow to the validators whose devices carry the traffic alongside block rewards (§6). Usage is priced, charged and collected in the billing token, so the *volume* of proxy revenue is set by demand for mobile egress rather than by token-market conditions. What reaches a validator is not insulated in the same way: the implemented payout accounting does not forward the billing token—it uses each exit's share of collected revenue as the weight for dividing a separately purchased pot of PHASE, transferred outside the protocol—and the liability that accounting tracks is denominated in PHASE. At launch the Foundation's stated operating policy is to commit the whole of collected proxy revenue to open-market PHASE purchases and to distribute the purchased PHASE daily in proportion to each exit's revenue share—the weighting the implemented accounting already computes. No contract enforces that policy; moving its execution on-chain is a roadmap item, and the payout denomination is therefore a policy commitment rather than a settled property of the protocol.

Coordination trust. The billing layer—the registry, the session ledger, and settlement submission—is operated by the Foundation at launch and is architected for progressive decentralization: its authority to CHARGE is bounded by the on-chain contract's watermark accounting, its authority to WITHHOLD a withdrawal is bounded by the claim window above, every economically meaningful transition (deposit, settlement, withdrawal) is on-

chain and idempotent, and the metering pipeline's failure modes are deliberately biased in the paying account's favor. We state this trust assumption explicitly rather than describe the current coordinator as trustless.

15. Validator Pools

Phase includes a protocol-native pool primitive: a mechanism by which validators form on-chain groups that share rewards, and by which the organizer of a group earns a protocol bonus that grows with the group's active size. Functionally it is a referral system enforced by block execution—the relationship "organizer recruited member" is an on-chain join operation, and every downstream payment consequence follows from state, with no off-chain tracking, no manual payouts, and nothing to dispute.

15.1 What a pool is and is not

A pool aggregates *payments*, not *power*. Unlike proof-of-work pools, no work is combined: every member remains an independent validator with its own device, its own reputation score (§4), and its own committee participation (§5). Pool membership has no effect on committee selection, scoring, or voting; it changes only how finalized reward payouts (§6) are distributed.

15.2 Mechanics

A registered validator creates a pool and becomes its organizer; other validators join by referencing the organizer's address. Both operations are *registry-gated*: creation and joining are restricted to registered, active validators, verified at call time through the precompile at `0x...0080`, which answers from the registration leases the block executor records in EVM state and, for genesis validators, from the chain specification (§13). A pool cannot be populated with inert accounts—recruiting means growing the actual validating fleet. Each address belongs to at most one pool, and an organizer cannot join another pool.

Reward sharing is applied at the execution level, inside the finality-gated payout path: when a payout is due to a validator who belongs to a pool of N active members, the amount is divided equally among all N. The organizer additionally receives a bonus on payouts earned by pool members as block leaders; the bonus scales linearly with the number of active members, at 0.1 PHASE per member, and saturates at 10 PHASE per height at 100 members; it is in any case capped at a tenth of the height's reward. The bonus is carved out of the height's block reward *before* the leader/signer split, so per-block issuance never exceeds the schedule value: it is funded from within the fixed validator reward budget (§16), not minted on top — by the leader and the signers of that height together, in proportion to their split. On mainnet, membership is capped at 100: the pool contract's `joinPool` refuses a full pool, and the bonus saturates at the same count.

Three guards keep the mechanism clean. Short entry and exit locks—20 blocks each in the current configuration, i.e. 20 s at the nominal 1 s accounting unit of §3 and about

2.5 s at the 7.9 blocks per second measured there—prevent hopping between pools to intercept payout timing; they are designed as anti-gaming friction, not loyalty enforcement. Both lengths are compile-time constants of the pool contract rather than governed parameters, so lengthening either is an implementation upgrade by the multisig that holds the upgrade-admin role of every system proxy (§17), not a parameter write (Remark 17.1). Membership is read at the paying block and then filtered by the height being paid, so a member earns only for heights at or after the block in which it joined. And a per-block system call automatically removes members whose validator registration has lapsed (§13), so pool membership tracks the live fleet without manual curation. (An organizer whose own registration lapses stops earning validator payouts of its own; the pool itself persists, and by design the organizer keeps receiving its equal share of the members' payouts and the leader bonus — members who disagree can leave once their entry lock expires.)

15.3 Income smoothing

For an ordinary member, the dominant effect of pooling is variance reduction. A phone validator earns leader-scale rewards only in the infrequent views in which it leads; between them, income arrives as smaller signer shares. In a pool of N members with per-period reward flows of mean m and variance s^2, approximately independent across members, the equal split preserves each member's expected income at m while reducing its variance by roughly a factor of N: fewer jackpots, a steadier stream. For a retail participant on a phone, predictability of income is itself a product property—it is the difference between an earning device and a lottery ticket.

15.4 Incentive notes

The organizer bonus pays for a real externality: each recruited member is a registered, active validator that contributes observations, committee capacity, and (if opted in) proxy egress. Because entry is registry-gated and leased registrations are self-cleaning, the bonus cannot be farmed with idle accounts—the genesis carve-out of §13 is fixed at launch and cannot be entered later, so it opens no farming path; sustaining it requires keeping recruited devices actually validating. The pool split and bonus are additive to the emission design and interact with the reward calibration of §6; the split rule, bonus scaling, membership cap, and lock durations are all governed parameters in the design of §17; today the bonus scaling is a compile-time constant of the node and the split rule is node code, while the lock durations and, on mainnet, the membership cap sit in contract bytecode — none of them reachable by any deployed governance process (Remark 17.1). We note one honest limitation: equal splitting mildly weakens the coupling between an individual member's marginal effort and their marginal income (a $1/N$ share of their own payouts plus a share of others'); the per-validator reputation system (§4) is unaffected by pooling and continues to bear on selection, so the performance incentive is preserved where it matters for security.

16. Tokenomics

Status. This section is the canonical description of PHASE token economics. Total supply, the allocation split, the emission schedule, and vesting terms are fixed as stated below. A small number of secondary operating parameters (such as the mobile-cost-floor reference constant) are designed to be set through governance; no governance process is deployed, and the shipped node carries no value for the mobile cost floor (Remark 17.1). This section is not a token sale document, an offer to sell tokens, or a valuation forecast.

16.1 The PHASE Token

PHASE is the native utility token of the Phase network, serving three functions:

1. **Fee denomination.** Transaction fees are denominated and paid in PHASE through the standard execution-layer fee flow (§13), and are the intended input to the fee-offset emission mechanism of §16.2, which is not yet active.
2. **Validator rewards.** Block rewards are paid in PHASE, released only under the finality gate of §6.2. In the current configuration they are a validator's only protocol income: fees are burnt rather than shared (§16).
3. **Governance.** PHASE holders are intended to vote on protocol parameters via timelocked proposals; no voting or timelock contract is deployed, and parameters are changed today by a client release for the consensus constants and by the upgrade multisig for those held in the system contracts (§17).

Remark 16.1 (Zero-cost registration and the non-PoS claim)**.** *Validator registration requires no minimum PHASE holdings. This is the design intent: a meaningful token-ownership requirement would reintroduce a capital gate and contradict the non-PoS objective. Free registration is intended not to create a free-entry security problem, because access to the eligible registry is to be controlled by device attestation (Assumption 3.3), probation with minimum observation count $N_{\min}$, reputation scoring, and the M_t soft cap. Of those four gates, attestation is planned work: today entry is gated only by proof-of-possession and probation, so nothing yet binds a registered identity to a distinct physical device (Remark 3.2); what keeps free identities from a voting quorum is the anchor bound of §5, not a price or an attestation. The economic exposure that deters misbehavior is the validator's accumulated earning position — the discounted future income stream — not a registration deposit.*

16.2 Supply and Emission

Total supply is fixed at **100,000,000,000 PHASE**; no issuance exists outside the allocation of Table 5. Half of the supply — **50,000,000,000 PHASE (50%)** — is committed to validator rewards and is emitted on a **linear per-block schedule over ten years**: at the nominal 1 s accounting unit of §3 this is

$$R_{\text{block}} = \frac{5 \times 10^{10}}{10 \cdot 365 \cdot 86{,}400} \approx 158.55 \text{ PHASE per block} \quad (\approx 5 \text{ billion PHASE per year}).$$

The schedule is laid out for ten years on the assumption of one block per second; the realized block time may differ from that assumption. The per-block reward and the total budget are the fixed quantities, so a faster or slower chain reaches the end of the budget earlier or later in calendar time while the per-block amount stays the same. Fixing the reward budget as a supply share bounds dilution ex ante and makes the reward pool a governance-visible constant.

A linear schedule was selected over frontloaded and adoption-coupled alternatives for three reasons: it is cohort-fair per unit of time (a validator joining in year seven faces the same per-block reward as one joining at genesis); it is maximally legible; and it still concentrates per-validator income early in the network's life without any explicit frontloading, because the same budget is divided among the small early pool M_t. Emission ends at a fixed height rather than on a running account of what has been minted: the shipped schedule pays 158.55 PHASE at heights 1 through 315,357,931 — the largest number of whole payments the fifty-billion budget admits, 2,069 short of the $10 \cdot 365 \cdot 86{,}400$ blocks the per-block rate was derived from, because the shipped constant is exactly 158.55 rather than the 158.5489 . . . the division gives — and nothing from height 315,357,932 on. That cutoff falls in year ten at the nominal 1 s accounting unit of §3 and sooner in calendar time at a faster realized cadence; from it on, validator income consists of transaction fees — once the fee offset below routes them to validators; the current configuration burns them in full — and the proxy economy (§14).

Fee-offset emission. Emission is designed to be *net of fees*. In each interval, collected transaction fees fund the scheduled validator payout first; only the shortfall between the fee flow and the scheduled rate is minted from the reward budget, and fee revenue *above* the scheduled rate is burnt. Three consequences follow. First, the scheduled rate is a ceiling on gross minting: as usage grows, net emission falls, and at sufficiently high usage the system turns deflationary. Second, validator income is smoothed: validators receive the scheduled rate whether it is financed by fees or by minting. Third, budget tokens left unminted by fee offset are not destroyed but are meant to remain available and extend the effective reward tail past the schedule's last paying height; the shipped schedule does not implement that carry-forward — `rewardAt` is a pure function of height, paying a flat 158.55 PHASE below the end height and zero from it on — so the tail extension arrives only with the fee offset itself. The offset mechanism's accounting boundaries belong to the governance design of §17; the mechanism is not implemented, so no such boundary exists today, in the node or in the genesis rewards contract (Remark 17.1).

16.3 Fees and Burns

Transaction fees are denominated in PHASE. In the current configuration every fee component is burnt in full — the base fee by the fee-market rule and the priority fee by the execution layer — so validator income is the block reward alone. The base fee follows the EIP-1559 rule with a floor: it never falls below a minimum set in the genesis configuration (1 gwei on mainnet and fabric), however empty the blocks, so every transaction except

a validator registration, which runs at a zero gas price (§13), pays at least that price per unit of gas; above the floor it still rises on blocks over the gas target and falls back after them. A consequence worth stating plainly: in this interim state a proposer's income does not depend on what its block contains, so the protocol offers no marginal incentive to include or order transactions — inclusion rests on validators following the reference block builder, until the fee offset below is activated and fees start financing validator income. The fee-offset mechanism of §16.2 is the design target, activated progressively: once active, fees finance validator payouts up to the scheduled per-second rate (displacing minting one-for-one) and only the surplus is burnt. There is no separate fee-split parameter — the offset rule *is* the fee allocation. An additional discretionary burn (a fraction of circulating supply per year) belongs to the same governance design; it is zero today, and neither a burn-fraction parameter nor the timelocked process that would introduce one exists in the implementation (Remark 17.1).

16.4 Validator Economics

A validator's income has three designed components: block rewards from the emission schedule (§16.2), a share of transaction fees, and income from the proxy economy (§14), which is designed to flow to validators as the proxy marketplace matures. For a fully-active validator in a registry of size M_t, expected per-period income from the protocol components is approximately

$$\pi_i \approx \frac{\theta_{i,t}}{\bar{\theta}_t} \cdot \frac{r_{\text{period}}}{M_t}, \tag{23}$$

where r_{period} is the total reward budget over a scoring period and $\bar{\theta}_t$ the registry-average score weight: the budget divides across the pool, tilted toward higher-scoring validators. Concretely, the daily budget of ≈13.7 million PHASE at the nominal 1 s accounting unit implies, at an illustrative genesis-scale pool of $M_t = 2{,}000$, an average of ≈6,850 PHASE per validator-day before fees. That is a registry-wide average, not a typical member's income: a height's reward goes only to its leader and signers (§6) and to their validator pools (§15), and the anchors are seated, and leader-eligible, in every committee (§5), so the anchor seats draw the larger part of the emission — at seven anchors, with the three graduate and two observer seats filled and every member signing, $0.6 \cdot \frac{7}{10} + 0.4 \cdot \frac{7}{12} \approx 0.65$ of what remains for the leader/signer split after any pool-owner bonus, at $\beta_{\text{base}} = 0.6$ — and the validators drawn into the remaining seats share the rest. Income per non-anchor validator then declines as the registry grows, which is exactly the pressure the soft-cap controller manages. Rewards are released only once a later block's committed finality watermark reaches the height — on a chain of honest proposers, only after full-QC finality (§6.2); rewards accrued on tentative blocks are withheld until then (§10).

The calibration anchor for validator economics is the *mobile cost floor*: the marginal cost of running a Phase validator on a phone — battery, bandwidth, storage, and device wear — which is small but not zero. The soft-cap controller (§9) must keep expected per-validator income above this floor; if the registry grows faster than the reward and fee flow, income falls below cost, honest uptime incentives weaken, and the income-stream deterrence ar-

gument degrades. The mobile cost floor is therefore an input to registry-size governance, not merely a marketing observation. Its numerical value is to be estimated from device measurements and published as a governance reference constant; the shipped node carries none — its soft-cap revenue floor is zero, the sentinel that disables that constraint (§9) — so no cost floor enters the soft cap the node applies today.

16.5 Allocation and Vesting

Table 5: Token Allocation and Release Schedule

Category	Share	Tokens	TGE unlock	Cliff	Release
Validator rewards	50%	50.0B	—	—	Linear per-block emission, ten years at the nominal 1 s unit
Core team	25%	25.0B	0%	6 months	Nine equal quarterly tranches, months 6–30
Foundation & treasury	25%	25.0B	2%	none	Linear, 36 months
Total	**100%**	**100.0B**			

Only the Foundation's operating float is liquid at TGE: 2% of its bucket (500,000,000 PHASE, 0.5% of total supply), which pays the network's first governance and settlement transactions. Everything else vests. The core team allocation is locked for six months and then released in nine equal quarterly tranches — the first six months after TGE, the last thirty months after it — so insiders are fully aligned with the network's first two and a half years. The remaining 98% of the Foundation bucket vests linearly over 36 months from TGE with no cliff, funding ecosystem programs, the bootstrap infrastructure of §11, and grants. Circulating supply therefore starts at 0.5% of total and grows through per-block emission and vesting.

Two principles are fixed permanently:

- **No validator staking requirement — ever.** Registration requires no minimum PHASE holdings, and no future allocation or governance decision may introduce a bond, lockup, or holding requirement for validation. This is a standing design commitment, not a launch parameter.
- **Community majority.** Half of the total supply is committed to validator rewards alone; together with Foundation-funded ecosystem programs, the majority of supply flows to network participants rather than to the team.

17. Governance

17.1 Governance Architecture

Phase's target governance model is timelocked on-chain control of protocol parameter changes, first reached at Stage 3 of the roadmap below. None of it is deployed today (Remark 17.1); under that design, governance controls:

- Scoring parameters: λ, γ, ρ_{reset}, $\varepsilon_{\text{explore}}$;
- Deterrence parameters: jail duration J and the reputation-reset rule — both compile-time constants today, changeable only by a client release;
- Committee and leader parameters: leader-bonus bounds $\beta_{\min}/\beta_{\max}$, churn bound χ;
- Soft-cap targets for the eligible registry M_t (§9), including the validator-viability floor;
- Emission mechanics: the 50% validator budget and its flat per-block schedule — ten years at the nominal 1 s accounting unit — are fixed constants outside governance; governed parameters are the fee-offset accounting bounds and the discretionary burn fraction (§16).

Governance changes are bounded by parameter class. Routine tuning can move faster; security-critical changes require longer timelocks and capped per-proposal deltas.

Remark 17.1 (No governance mechanism is deployed)**.** *The parameter list above, the bound table below, and the 7- and 30-day timelock minima of the capture analysis below are the design target; none of them describes a mechanism now in force. No proposal, voting or timelock contract exists among the deployed system contracts (§13), so those delays currently bound nothing. Every parameter named above that the implementation has at all is a node-software value fixed at build time, with one exception and one gap: the emission schedule is a constant of the genesis rewards contract, which the upgrade multisig can replace (§13), and neither the fee-offset accounting bounds nor the discretionary burn fraction exists in the node or in any contract (§16). The scoring parameters are hard-coded defaults of the chain specification's scoring configuration — $\lambda = 0.9$, $\gamma = 0.5$, $\alpha_{\text{conf}} = 0.05$, $\rho_{\text{reset}} = 0.02$, on the $(\alpha_0, \beta_0) = (1, 1)$ prior — and that configuration is neither serialised into genesis nor parseable from a chain-specification file: every chain specification in the tree takes the struct's default, so no runtime path sets it. The remainder are compile-time constants of the node: $\varepsilon_{\text{explore}} = 0$, the jail duration $J = 10$ periods and the pessimistic Beta failure mass the reputation reset would write — both inert, since no production path jails a validator (Remark 5.3) — the churn bound $\chi = 2$ (Remark 5.2), the leader-bonus bounds pinned at $\beta_{\min} = \beta_{\max}$ with the adaptation gain at zero, and both soft-cap targets — the liveness target and the validator-viability revenue floor — shipped at 0, the sentinel that disables the M_t controller and leaves the hard structural ceiling on the eligible registry as the only limit. Changing any of them is a client release — the Stage 1 posture of the roadmap below, though the transparency log Stage 1 describes does not exist either. The one upgrade authority that does exist is the multisig administering the genesis system contracts, and it enforces no delay at all: once a threshold of owner confirmations is reached a proposal is immediately executable, and the multisig can move that threshold itself (§13). It reaches contract logic and the parameters held in contract storage or bytecode, not the consensus constants listed above.*

Parameter class	Examples	Bound style
Scoring	λ, γ, α_{conf}	Hard range plus max per-proposal change
Deterrence	J, reset rule	Security-critical timelock and emergency floor
Committee / leader	$\beta_{\min}/\beta_{\max}$, churn bound χ	Hard range plus max per-proposal change
Registry size	M_t targets, viability floor	Daily rate limit and validator-viability floor
Emission	Reward schedule, fee split, burn fraction	Security-critical timelock; capped relative change per epoch
Admin	Pause, upgrade, guardian rotation	Multisig/timelock and public notice window

Table 6: Governance parameter bounds. Exact ranges are governance parameters, but the bound structure is part of the security design.

17.2 Governance Capture Analysis

Capture risk. A single entity accumulating sufficient voting power can alter protocol parameters unilaterally. Relevant attack surfaces:

- **Deterrence manipulation:** Reducing J (jail duration) or weakening the reputation-reset rule to lower the cost of provable faults.
- **Soft-cap manipulation:** Moving registry-size targets to suppress competitors' selection probability or to dilute per-validator income below the viability floor.
- **Emission manipulation:** Reshaping the reward schedule or fee split to favor a particular validator cohort.

Designed mitigations, none of them yet deployed (Remark 17.1): timelocks (minimum 7 days for routine parameters, 30 days for security-critical parameters); maximum single-proposal change bounds; a protocol guardian multisig with emergency veto power during the decentralization transition. Until they are, the capture surface is not smaller but different in kind: the deterrence and registry-size parameters above are compile-time node constants moved by a client release under Stage 1, the reward schedule is a constant of the genesis rewards contract, and the deployed upgrade multisig can replace that contract's logic — or any other system contract's — with no delay between confirmation and effect.

17.3 Governance Roadmap

Stage 1: team-controlled parameter updates with a public transparency log.
Stage 2: off-chain signaling with team execution.
Stage 3: on-chain governance with timelocks; guardian multisig retains emergency veto.
Stage 4: guardian multisig deprecated; full on-chain governance.

18. Security Analysis Summary

This section consolidates the formal status of the paper's claims. The framing is deliberately conservative: a result is listed as *proved* only where a self-contained argument exists under stated assumptions; *proof sketch* marks arguments with known gaps; *design target* marks properties the design aims at but does not establish. None of the statuses below are aspirational upgrades.

Table 7: Formal Status of Claims

Result	Status	Caveat
Sybil occupancy (Lemma 5.1, App. D)	Proved	Eligible-slot occupancy only; depends on attestation (Assumption 3.3); fails against a patient score-farming adversary at the committee-share level
Beta-EMA tracking (Claim 4.1, App. B)	Proof sketch	Not rigorous with censoring, variable w_v, boundary conditions; N_{eff} is discounted pseudo-mass
Deterrence sufficiency, income-stream form (§7)	Proof sketch	One-shot, single-validator, provable-fault only; the value of the forfeited income stream depends on the calibrated emission and fee path; and the penalty stack it prices has no trigger in the implementation (Remark 6.2)
Tentative fork risk (Claim 10.1, App. C)	Proof sketch	Stylized split model; feasible q_{tent} for sufficiently small ρ_s only; and, like the full-QC path, it assumes at most f of the voting seats are Byzantine, which under the anchor bound means at most $f - g$ compromised anchor keys beside hostile graduates (§22)
Full-QC finality (§3)	Holds under the anchor bound	Deterministic while the anchor set is seated whole, up to $2(q_{\text{full}} - g) - a - 1$ compromised anchor keys; outside that (more compromised anchor keys, more anchors than seats, or no anchor left) the certificate paths log the broken quorum-intersection bound and certify anyway, so two conflicting blocks can both be certified, a finalized block can be reverted and the node converges on the strongest branch (§10, §22)
Explore manipulation (Claim 18.1, App. E)	Proof sketch	One attack template; "material widening" undefined
Leader grief (App. F)	Proof sketch	Detection latency at large M_t unresolved
Coalition deterrence (§8)	Design target	Simulation-supported; no formal coalition-proof result without capital bonds; and the escalated jail term it is evaluated at is unimplemented, with no fault record for it to read (Remark 6.2)
Liveness / soft cap (§9)	Design target	Calibration requires live network data
Progressive mechanisms (dynamic q_{tent}, dynamic registry size, explore tranche)	Design target	Implemented in node software and activated progressively as the network accumulates live data; in the current configuration all three are off (q_{tent} fixed at $f + 1$, both soft-cap targets at zero, $\varepsilon_{\text{explore}} = 0$)
Attestation buckets (Assumption 3.3)	Design target	Cap enforcement implemented and disabled; no attestation primitive selected, so bucket identity defaults to the validator address and the cap binds nobody (Remark 3.2)
Stress correction	Design target	Specified only; $w_v = 1$ in the implementation

18.1 Security Games Covered by the Model

The deterrence model applies only when the relevant action is attributable, observable, and penalizable under protocol rules. The games below are the ones the model covers; behaviors outside this set are listed honestly in §22.

Table 8: Security games and status

Game	Adversary action	Deterrence / detection path	Status
Provable validator fault	Double-signing, equivocation, cryptographic fault evidence	Jail for J periods, reputation reset, forfeiture of withheld pre-finality rewards, loss of the future income stream	Proof-sketch deterrence (income-stream form, §7); no trigger in the implementation — detection logs only (Remark 6.2)
Sybil entry	Mass registration of identities	Device-attestation buckets; occupancy bound (Lemma 5.1)	Proved, conditional on attestation, which is not yet implemented (Remark 3.2)
Explore-tranche manipulation	Deliberate underperformance to widen the posterior and re-enter via the explore tranche	Fixed tranche budget; opportunity-cost bound (Claim 18.1)	Proof sketch
Leader grief	Withholding proposals to deny committee income	Case B penalties lower the leader's LCB, reducing its committee-selection weight; no demotion is implemented	Proof sketch; detection latency open
Coordinated soft failure	Coalition degrades performance below quorum without provable faults	Stress weighting is intended to partially absorb correlated outages (specified, not active)	Open (§22)
Censorship / inclusion	Excluding or delaying valid transactions without cryptographic fault	Round-robin rotation bounds per-view exclusion; pattern observability via event logging	Open (§22)

Claim 18.1 (Heuristic sufficient condition against explore-tranche manipulation)**.** *Deliberately underperforming to widen the Beta distribution and exploit the explore tranche is heuristically unprofitable when* $\varepsilon_{\text{explore}} \leq (1 - \rho_{\text{reset}}) \cdot p_{\text{sel}}(s_{\text{normal}}, M_t)$.

Note. This is a heuristic sufficient condition, not a theorem. The phrase "materially widen the Beta" used in the argument below is not formally defined, and the role of p_{sel} in the bound is not derived from first principles; the argument captures the right order of magnitude for one specific attack template (deliberate underperformance to

induce re-selection via the explore tranche) but does not rule out related strategies such as score-farming followed by attack, or coordinated underperformance across multiple identities.

Sketch. Gain from the single-identity underperformance attack is bounded by $G \leq \varepsilon_{\text{explore}} \cdot n \cdot r_{\text{avg}}$ (fixed tranche budget). Cost grows linearly as $\Delta \approx T_{\text{bad}} \cdot \pi_i(s_{\text{normal}})$. For T_{bad} large enough to shift the Beta posterior far enough that the explore tranche selection probability rises appreciably — a threshold we do not formalize here — $\Delta \geq G$ under the stated bound. A full proof would require a formal definition of "material posterior widening" and a derived relation between T_{bad} and the resulting change in p_{sel}. □

19. Calibration and Attack-Surface Testing

19.1 Methodology

Calibration is performed by agent-based simulation over multi-year horizons. Simulated validator populations carry heterogeneous latent reliabilities, realistic mobile churn processes, and scripted adversarial strategies; the scoring, selection, reward, and deterrence mechanisms are modelled from the design rather than run from the node software — including the explore tranche and M_t sizing, both switched off in the shipped configuration (§5, §9), and the penalties that nothing in the implementation triggers (Remark 6.2). Scenarios are expressed in a simple language of adoption paths: a baseline variant with faster network growth and a conservative variant with slower growth. The shape of the scoring engine is stable across scenarios; what changes materially is the sustainable registry size M_t and the reward calibration. All outputs are engineering diagnostics used to set operating ranges and to identify workstreams — they are not theorems, and they are not price forecasts. Registry-size operating ranges derived from these runs are published as governance reference values at launch.

19.2 Consensus Attack-Surface Tests

The attack-surface suite exercises the consensus and scoring mechanisms against the adversarial templates from §18.1. Readings: "Comforting" means substantial margin in the sweep; "Watch" means nominally safe but dependent on assumptions that live-network data must confirm; "Open" means a design or governance workstream is still required: both "Open" rows were planned to close before mainnet, which launched with them open.

Tested surface	Read	Implication for Phase
Score farming	Comforting	Honest participation dominates deliberate score manipulation across the sweep. Continue monitoring abnormal score trajectories.
Sybil re-entry cost	Comforting	Re-entering after a reputation reset requires a full probation cycle from the prior; the time cost is high relative to plausible bribes, conditional on device-attestation integrity.
Leader grief	Watch	LCB degradation detects a griefing leader but ejects no one: it lowers only the griefer's selection weight, and slowly at large M_t (App. F); leader-specific attribution should be tested on the live network.
Tentative fork risk	Watch	Safe at low split-vote rates; the real mobile split-vote frequency ρ_s must be measured before the dynamic tentative quorum is activated.
Soft-cap race-to-bottom	Open	Registry-size control must enforce the validator economic viability floor (§16.4), not just a quorum-probability objective.
Genesis deterrence	Open	No-stake security starts with a short income history, so the deterrence stack is weakest at Day 0. Carried by the permanent anchor set (§21), which is shipped, and by emission-schedule selection (§16.2).

Table 9: Attack-surface findings used as design diagnostics.

Two workstreams follow directly from the "Open" rows: the anchor set must be made fully explicit — the mainnet genesis writes seven anchors and carries no voucher, which leaves only the operating rule on converting an anchor's row open — and the soft-cap controller's economic viability floor must be activated. The controller shipped to mainnet with both of its targets at 0, the sentinel that disables it (Remark 17.1), so the floor can now be activated only by a later client release. The "Watch" rows define the first two measurement campaigns of the live network: leader-grief attribution latency and the empirical split-vote rate.

20. Risks

Risk Disclosure. The following is a non-exhaustive summary of material risks associated with the Phase network and the PHASE token. Participation involves significant risk, including the risk of total loss. Prospective participants should conduct their own independent assessment.

20.1 Protocol Risks

- **Deterrence may be insufficient at low tenure or low token price.** The income-stream penalty is proportional to the validator's expected future earnings. At early stages, when tenure is short and token price is low, the deterrence condition may fail. The

protocol relies on the permanent anchor set (§21) during exactly this window — and, since the anchor bound never lifts, after it; deterrence strengthens over time as income histories accumulate.

- **Coalition deterrence is a design target, not a proved result.** The correlation-escalated penalty (§8) has neither been formally analyzed nor implemented — no per-validator provable-fault record exists for it to read, and the base jail term it would scale has no trigger (Remark 6.2) —, and a coalition operating below the detection threshold for soft failures remains an open problem.
- **Sybil attack via attestation compromise.** If App Store / Google Play attestation is compromised or circumvented at scale, the anti-Sybil layer it is designed to provide fails. That layer is planned work: the current release performs no device attestation (Remark 3.2), and what keeps free identities from a voting quorum is the anchor bound of §5. There is no purely cryptographic Sybil resistance.
- **Mobile churn causing persistent quorum failure.** Correlated connectivity failures (tower outages, OS updates, regulatory restrictions on background processes) could cause sustained quorum misses. Tentative certificates mitigate the liveness impact, but their safety rests on a heuristic bound (Claim 10.1), not a full BFT proof.
- **Non-provable strategic behavior.** Censorship and coordinated soft failure are not formally deterred. The reputation system provides long-run disincentives but not cryptographic guarantees.
- **Proof incompleteness.** Claims 4.1 and 10.1 are proof sketches. If the missing rigor reveals errors, the protocol's security properties may differ materially from those described.

20.2 Economic Risks

- **Procyclicality of monetary deterrence.** The jail-period income loss $\sum_{k=0}^{J-1} \rho^k \pi_i$ scales with token price. The less price-sensitive component is the time cost of rebuilding reputation from the reset prior: the required periods of probation and re-scoring do not shrink when price falls, although the value of regained access still depends on future protocol utility and fee flow.
- **Emission miscalibration.** If the reward schedule underpays relative to the mobile cost floor, honest participation weakens; if it overpays, it invites over-entry pressure on the soft cap. The schedule and budget are fixed (§16.2); the fee-offset mechanism and registry-size governance are the designed levers for correcting miscalibration within the fixed budget, neither of them active yet (Remark 17.1).
- **Validator race-to-the-bottom.** If the soft cap fails to prevent over-entry, expected validator profits approach zero, degrading liveness.

- **Proxy-economy demand risk.** Proxy income is designed to flow to validators but depends on external demand for proxy services (§14); that demand may not materialize at the scale the calibration assumes.
- **Billing-asset risk.** Proxy prices, prepaid balances and collected revenue are denominated in the genesis billing token (§14; on mainnet `USDT.e`, backed 1:1 by USDT held at the bridge deposit address and, for the genesis supply, in a Phase Labs reserve). The backing is held in custody by Phase Labs rather than enforced on chain, the bridge is one-way for now, so USDT.e cannot yet be withdrawn to a source chain, and the token's implementation and minter are replaceable by the upgrade authority (§17); client balances therefore carry custody and redemption risk. The implemented payout path divides a pot of PHASE bought with the collected revenue, so what reaches a validator is exposed to token-market conditions either way.
- **Governance capture.** A single entity could reduce J or reshape emission and registry-size parameters to weaken deterrence or favor a cohort. See §17.
- **Tokenomics execution risk.** Supply, allocation, emission, and vesting are fixed (§16); the residual risks are execution — implementing the schedule, vesting, and fee-offset mechanics correctly in the protocol contracts — and market risk on the value of the reward stream, which no schedule controls. §16 is not a statement of offering terms.

20.3 Legal and Regulatory Risks

- **Token classification.** The PHASE token may be classified as a security, financial instrument, or other regulated product in one or more jurisdictions.
- **AML/KYC.** Permissionless access may conflict with anti-money-laundering and know-your-customer requirements in certain jurisdictions.
- **MiCA and equivalent frameworks.** Under the EU Markets in Crypto-Assets Regulation (MiCA) and equivalent frameworks, issuance of PHASE tokens would require compliance with whitepaper, disclosure, and authorization requirements. Phase intends to seek appropriate legal advice prior to any public token distribution.

20.4 Operational Risks

- **Software bugs.** Vulnerabilities in the consensus implementation, mobile client, or proxy infrastructure may result in loss.
- **Key management.** The upgrade multisig that administers the system contracts (§17) relies on team key management practices during the decentralization transition, and so does the permanent anchor set (§21): finality survives at most $2(q_{\text{full}} - g) - a - 1$ compromised anchor keys beside g hostile graduates. At the seven mainnet anchors that bound is 2 with $g = 0$ or 1 graduates seated and 0 with $g = 2$ or 3: once two or more graduates are seated, a single compromised anchor key together with hostile graduates can finalize conflicting blocks. That is $f+1$ Byzantine voting seats, the

generic BFT limit, but it is reached with one anchor key. Key compromise could result in loss.

- **Third-party dependencies.** App Store / Google Play attestation and mobile OS background-execution policies are outside Phase's control; changes to either can affect validator availability and the anti-Sybil layer.

20.5 Environmental Considerations

Phase uses BFT consensus without computational puzzles. Energy consumption scales with the number of active validators and communication overhead, comparable to running a background mobile application. Phase does not use proof-of-work at any layer. A formal carbon footprint assessment will be commissioned prior to mainnet launch.

21. Day-0 Bootstrapping

At genesis, the reputation system has no signal. Phase bootstraps via:

Permanent anchor set. The genesis validator set is a permanent anchor set: anchors are seated whole in every committee, graduated validators hold at most $\lfloor (a-1)/2 \rfloor$ voting seats beside them, and probationers sign only as non-counting observers (§5). The anchors carry honest consensus while permissionless validators accumulate evidence, and they keep carrying it afterwards: there is no decay schedule; an anchor leaves only by converting its row to a lease (§13), and the last anchor never does. The permissioned share $\rho_{\text{perm}}(t)$ is therefore 1 by construction; a decaying share $\rho_{\text{perm}}(t) = \max(0, 1 - t/T_{\text{boot}})$ is not implemented and is not planned. The chain spec gives the anchors their standing through two rows: the registration exemption of §13—a validator in the genesis validator set is permanently active and is never swept, so the lease never removes it from the eligible set—and its status row, written directly as `full` at epoch 0, which skips the activation queue, the churn limit, and the probation gate of §5 that every validator first registering after genesis passes through. The trust assumption is the anchor operators', and it is permanent rather than temporary; adding an anchor after launch is a hard fork, since no transaction creates one.

Non-transferable genesis voucher (optional). A non-transferable airdrop voucher, forfeitable on provable misbehavior, would provide early skin-in-the-game without a permanent capital requirement. This is a genesis-only mechanism: it creates no ongoing holding requirement, and beyond genesis, deterrence rests on the income-stream stack of §7 under the anchor set above. The mainnet genesis carries no voucher: it has no voucher contract and no voucher allocation.

22. Open Problems

22.1 Base-Chain and Scoring Open Problems

Non-provable censorship.. A leader may selectively exclude transactions. Round-robin rotation limits individual censorship to one view. Structured event logging makes repeated patterns observable. No cryptographic proof of selective exclusion is available without mempool commitment schemes or inclusion proofs. Formal treatment: open.

Coordinated soft failure below detection threshold.. A coalition may jointly degrade performance below quorum without any single member committing a provable fault and without meeting the threshold for the correlation penalty to activate. The stress-weighting mechanism is intended to partially absorb correlated outages, but is specified rather than active; the correlation penalty (§8) is explicitly restricted to provable faults. Formal characterization of minimum coalition size and coordination cost: open.

Endogenous censoring in the tracking claim.. Claim 4.1 is proved only in the uncensored setting ($L_v = 1$ always, $w_v = 1$). In the full protocol, failed views produce no QC bitmap, introducing endogenous censoring that correlates with latent reliabilities $\theta_{i,t}$. A rigorous tracking result under endogenous censoring is the primary open theoretical problem in the scoring system.

Leader grief attack.. A malicious leader can withhold block proposals, denying income to the entire committee at a cost-to-harm ratio of approximately $1{:}(n-1)$ — 1:6 to 1:9 at the shipped $n \in [7, 10]$, 1:31 at the wire-format ceiling $n = 32$. The leader's LCB degrades under repeated Case B penalties, which lowers its committee-selection weight. It does not lead to expulsion: no demotion or jailing of a *Full* validator is implemented, and the jail rule is deliberately restricted to provable faults, which withholding a proposal is not. The consequence is therefore weaker than a threshold crossing, and for an anchor it is nothing: an anchor is seated in every round whatever its score, so for the permanent anchor set a degraded LCB costs the griefer nothing at all; only the graduate seats respond to score. There the effect is a reduced sampling weight rather than a removal, and at large registry sizes ($M_t > 500$) leader selection is infrequent enough that even that signal accumulates slowly. A formal bound on minimum registry size for adequate grief detection, and the interaction between grief attacks and coalition formation, are open problems.

22.2 Design-Level Open Problems

Formal safety of the tentative-certificate mode.. Claim 10.1 is a heuristic bound over a stylized split-vote model. A full BFT-style treatment of the tentative mode — specifying fork choice, abstention, message delivery under partition, and the interaction between TC chains and the retroactive full-QC upgrade — does not exist. Until it does, the tentative mode should be understood as an engineering mitigation with a quantified but model-dependent risk

estimate, not a proved extension of HotStuff safety.

Coalition deterrence without capital bonds.. The deterrence stack is designed to replace bonded collateral with jail, reputation reset, finality-gated rewards, and forfeiture of the future income stream; of those four, only the finality gate is implemented, and nothing triggers the fault-side penalties (Remark 6.2). For a single validator this yields a proof-sketch bound (§7); for coalitions, the correlation-escalated jail term is a design target only. Whether an income-stream-only deterrence model admits a formal coalition-proof result — and under what assumptions about bribe financing and coordination cost — is open.

Committee sampling from a larger registry.. Under the anchor bound the committee never samples its voting population: the anchors are seated whole and the graduate seats are capped at $\lfloor (a-1)/2 \rfloor$, so quorum intersection holds by construction whatever the registry size (§3), and the graduates beyond the cap wait on a bench inside the model. What is open is therefore not safety but the shape it forces. The voting population is $a + \lfloor (a-1)/2 \rfloor$ — ten at seven anchors — and no registration, graduation or score can raise it; seating more graduates needs more anchors, which no transaction creates and only a hard fork can. Sampling graduates beyond the cap from a larger registry, which the coalition arguments of §8 presuppose, would reopen the population-wide rule $2q_{\text{full}} > |F_t| + f_{\text{reg}}$, which among proper samples admits only a population that is a multiple of three with a single seat withheld, and no choice of threshold recovers it — at a 16-of-22 sample it needs a quorum of 15 while liveness admits at most 9. Closing that needs a different construction — a seated committee large enough to be the population, which past 32 is a block-format change, or a sampling rule whose safety argument is not a bound over two unconstrained draws — and until one exists the coalition arguments of §8 live in a regime the anchor bound forbids by design, while the tentative-mode analysis of Appendix C assumes at most f of the voting seats are Byzantine (Assumption 3.1), which under the anchor bound means at most $f - g$ compromised anchor keys beside hostile graduates.

Selection randomness.. Committee selection is seeded from the epoch and committee round alone and mixes in no chain state, so every future committee is computable offline by anyone; this is deliberate, so that the next committee's connections can be prewarmed (§11). The voting power a chosen registration can buy is bounded by the anchor model, not by seed unpredictability: free identities reach only the graduate seats. An anchored seed, mixing in the hash of the block that wrote the epoch's snapshot, exists behind an activation epoch that is not scheduled, and a VRF is not planned; what remains open is whether either is worth its cost to the prewarm, given that the proposer of the anchor block could still grind its own hash and that the explore tranche's Thompson sampling would make selection a floating-point consensus computation.

23. Conclusion

Phase builds a formally grounded case that a mobile-first BFT blockchain can be secured without bonded stake. The deterrence stack — jail, reputation reset from the prior, rewards released only after finality, and forfeiture of the discounted future income stream — is designed to make the earning position itself the validator's skin in the game; of it, the watermark-gated release is live — in the implementation, gated on a proposer-committed watermark rather than on a verified certificate (§6.2, Remark 6.1) — and the fault-triggered penalties are specified but have no trigger in the implementation (Remark 6.2). The paper is deliberate about the status of each claim: Sybil occupancy is proved under the attestation assumption; Beta-EMA tracking, deterrence sufficiency, tentative fork risk, explore manipulation, and leader grief carry proof sketches with stated gaps; coalition deterrence and the liveness soft cap are design targets awaiting live-network calibration.

Beyond consensus, the system is an engineering substrate and an economy. The networking, synchronization, and deterministic execution layers are built for devices that sleep, roam, and churn; tentative certificates keep the chain live through quorum dips without relaxing full-QC safety; and mechanisms implemented ahead of their activation — dynamic registry sizing, dynamic tentative quorum, the explore tranche — are switched on progressively as the network accumulates live data. Two further mechanisms are not on that path. Stress correction is specified only: the implementation holds $w_v = 1$, and giving the weight its intended behaviour needs a consensus-measurable participation signal for failed views before the formula can be switched on at all (§4). Attestation buckets have only their enforcement half built, and switching that on waits first on selecting and building a Sybil-resistant identity primitive (Remark 3.2). On top of this substrate, the proxy economy gives validators an income source rooted in real service provision, with proxy revenue designed to flow to the same devices that secure consensus, tightening the link between honest uptime and earnings.

The remaining work is concrete. Coalition deterrence needs a proof, not a simulation; the tentative-certificate mode needs a formal safety treatment; genesis security leans on the permanent anchor set, whose operators' keys are the chain's trust root; and censorship and coordinated soft failure remain open. These are the areas where live network data and implementation choices must harden the protocol, now that mainnet has launched.

References

[1] M. Yin, D. Malkhi, M. K. Reiter, G. G. Gueta, and I. Abraham, "HotStuff: BFT consensus with linearity and responsiveness," in *Proc. ACM PODC*, 2019, pp. 347–356.

[2] V. Buterin et al., "Combining GHOST and Casper," *arXiv:2003.03052*, 2020.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, pp. 235–256, 2002.

[4] D. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, "A tutorial on Thompson sampling," *Found. Trends Mach. Learn.*, vol. 11, no. 1, pp. 1–96, 2018.

[5] Y. Hong, "On computing the distribution function for the Poisson binomial distribution," *Comput. Stat. Data Anal.*, vol. 59, pp. 41–51, 2013.

[6] M. J. Wainwright, *High-Dimensional Statistics*. Cambridge Univ. Press, 2019.

Appendices

A. Notation Reference

Table 10: Principal Notation

Symbol	Meaning
v	HotStuff view index
t	Scoring period index
n	Voting committee size, $a + g$ (7–10 at seven anchors under a 16-seat budget; 32, the wire-format ceiling, in the illustrative analysis)
f	BFT fault parameter of the voting seats (2–3 current; 10 at $n = 32$)
f_{reg}	BFT fault parameter of the voting population — the full rows competing for voting seats, probation rows excluded — $\lfloor (\lvert F_t \rvert - 1)/3 \rfloor$
q_{full}	Full quorum threshold (5–7 current; 22 at $n = 32$)
q_{tent}	Tentative quorum threshold ($f + 1$; 3–4 current)
C_v, S_v, N_v	Committee, QC signers, NVC signers at view v (N_v planned; see §4)
$L_v \in \{0, 1\}$	Full-QC success indicator
$l(v)$	Leader for view v (round-robin)
$\theta_{i,t}$	Latent true reliability (unobserved)
$\alpha_{i,t}, \beta_{i,t}$	Beta-EMA posterior parameters
α_0, β_0	Beta-EMA prior parameters and soft-reset target (3) ($(1, 1)$ as shipped)
λ	EMA discount factor
δ	Per-period drift bound (Assumption 3.4)
w_v	Stress weight
N_{eff}	Approximate effective sample size $\approx 1/(1 - \lambda)$
$\text{LCB}(i, t)$	α_{conf}-lower quantile of Beta posterior
γ	Concave-transform exponent
g	Concave transform $g(x) = x^\gamma$ applied to the reliability estimate
$\Pi_{\text{bucket}}(i, t)$	Anti-Sybil bucket factor $\in \{0, 1\}$
$\Pi(i, t)$	Composite gating factor

continued on next page

Symbol	Meaning
$\mathrm{Cap}_{\mathrm{bucket}}$	Per-bucket cap on eligible registrations per period (Assumption 3.3); the shipped cap is a per-epoch seat cap, at zero (Remark 3.2)
$\mathcal{A}$	Adversary-held subset of the eligible registry
$\mathrm{Score}^{\mathrm{exploit}}(i,t)$	Exploit-tranche composite score
η_{sel}	Selection exponent applied to $\mathrm{Score}^{\mathrm{exploit}}$ (fixed at 2; not governance-settable)
$w_{\mathrm{sel}}(i,t)$	Committee-selection weight $[\mathrm{Score}^{\mathrm{exploit}}(i,t)]^{\eta_{\mathrm{sel}}}$
$\varepsilon_{\mathrm{explore}}$	Explore-tranche fraction of the graduate committee seats (0 current)
M_t	Eligible registry soft cap (max eligible validators)
$M_{\max}$	Hard ceiling on the eligible registry; admission stops at $\min(M_t, M_{\max})$ (4096 as shipped)
$\Delta M_{\max}$	Per-update bound on movement of the soft cap M_t (§9); unrelated to $M_{\max}$
E_t	Eligible registry, $E_t = P_t \cup F_t$
P_t	Probationers (at most two observer seats; they sign but count toward no quorum, and never lead)
F_t	Full members: the a genesis anchors, seated whole, and the graduates the exploit and explore draws choose from
$\mathrm{obs}(i,t)$	Cumulative raw observations (QC signatures plus misses) credited to i
$N_{\min}, \tau_{\mathrm{grad}}$	Probation graduation bars: minimum observations (40) and minimum LCB (0.5) as shipped
χ	Churn limit: queued validators admitted per period (flat 2 as shipped)
R_v	Per-view reward budget
β_t	Leader bonus fraction
$\beta_{\mathrm{base}}, \beta_{\min}, \beta_{\max}$	Base leader share and clip band in (13) (0.6, with the band pinned to it)
k_L	Liveness gain in (13) (a constant, shipped at 0, which disables the adaptation)
$k_P, \mathrm{TWAP}_t, \mathrm{TWAP}_{\mathrm{ref}}$	Price gain, PHASE-price TWAP and its reference level in (13); design only — no price feed exists in the node software

continued on next page

Symbol	Meaning
$L_{\text{target}}, \hat{L}_{t-1}$	Target and observed quorum-success rate (0.95 specified); L_{target} in (13) and (21), $\hat{L}_{t-1}$ in (13) alone
Δ_β	Cap on the weekly change in β_t in (13) (design only; no limiter is implemented)
J	Jail duration (periods; 10 in the shipped build, though nothing enters `jailed` to serve it)
ρ	Income discount factor
$V_i(t)$	Reputation value (discounted future income)
π_i	Expected per-period *net* validator income (also written π_i^{net})
$\text{Pen}_i(J)$	Total penalty from a provable fault with jail length J, from (15)
Wh_i	Withheld pre-finality rewards; forfeited on provable fault by design, not in the implementation
ρ_{reset}	Soft-reset shrinkage of the Beta posterior toward the prior
p_{sel}	Per-period committee selection probability
δ_{live}	Liveness slack in the tentative-quorum feasibility condition
ρ_s	Split-vote rate: modelled probability an honest online validator votes for a conflicting block
ε	Tentative-mode safety budget: tolerated per-view fork probability (§10); cf. $\varepsilon_{\text{explore}}$ above
$B_{\max}$	Maximum adversary bribe
κ_{corr}	Correlation penalty multiplier for provable-fault coalitions
$\mathcal{K}_v$	Coalition fault set within correlation window
W_{corr}	Correlation window (views)

B. Proof Sketch: Claim 4.1 (Beta-EMA Tracking)

Status and limitations. Valid in a simplified setting: no censoring ($L_v = 1$ always), constant $w_v = 1$, no soft reset ($\rho_{\text{reset}} = 0$, against the shipped 0.02), and $\theta_{i,t} \in [\eta, 1-\eta]$ for some $\eta > 0$. Does not extend without modification to endogenous censoring (Case B), variable $w_v < 1$, the shipped $\rho_{\text{reset}} > 0$, or reliability near the boundaries. Treat as a qualitative guide in those regimes.

Proof sketch (simplified setting). **Step 1: Effective sample size (heuristic).** Under $L_v = 1$ and $w_v = 1$ for all v, after T views: $\alpha_{i,T} + \beta_{i,T} = \lambda^T(\alpha_0 + \beta_0) + \sum_{s=0}^{T-1} \lambda^{T-1-s}$. As $T \to \infty$, this converges to $1/(1-\lambda) =: N_{\text{eff}}$. Under variable $w_v \in (0, 1]$, steady-state mass is $\leq 1/(1-\lambda)$. The shipped filter additionally applies the soft reset (3) once per period, so this

is not the mass it actually carries. With k observed views per period and $w_v = 1$ the mass recursion is $m \leftarrow (1-\rho_{\text{reset}})\big(\lambda^k m + \frac{1-\lambda^k}{1-\lambda}\big) + \rho_{\text{reset}}(\alpha_0+\beta_0)$, whose fixed point is

$$N_{\text{eff}}^{\rho} = \frac{(1-\rho_{\text{reset}})\dfrac{1-\lambda^k}{1-\lambda} + \rho_{\text{reset}}(\alpha_0+\beta_0)}{1-(1-\rho_{\text{reset}})\lambda^k},$$

strictly below N_{eff} whenever $\alpha_0+\beta_0 < 1/(1-\lambda)$, as at the shipped $\lambda = 0.9$, $\rho_{\text{reset}} = 0.02$, $(\alpha_0,\beta_0) = (1,1)$: about 1.8% below it for a validator observed once in every block of a twenty-block period ($k = 20$; 1.6% in the large-k limit), rising to about 14% at $k = 1$. A positive ρ_{reset} therefore both widens the statistical-spread term of (9) and adds a standing shrinkage of the posterior mean toward $\alpha_0/(\alpha_0+\beta_0)$; (9) is stated for $\rho_{\text{reset}} = 0$ and accounts for neither. We emphasize that N_{eff} here is the steady-state *discounted pseudo-mass* of the recursion, not the variance-equivalent effective sample size for exponentially weighted Bernoulli data; the two agree up to constants in the stationary case but need not agree under non-stationarity.

Step 2: Beta-quantile concentration (two legs). The argument runs in two legs. *Leg A: LCB stays close to the posterior mean.* For $X \sim \text{Beta}(a,b)$ with $a+b = N$, $\mu = a/N$ bounded away from 0 and 1 (Wainwright [6], Ch. 2):

$$|Q_{\alpha_{\text{conf}}}(\text{Beta}(a,b)) - \mu| \le \frac{c\,\Phi^{-1}(1-\alpha_{\text{conf}})}{\sqrt{N}} + O(N^{-1}),$$

where $c > 0$ depends on μ and $c \to \infty$ as $\mu \to 0$ or $\mu \to 1$. Setting $N = N_{\text{eff}}$ gives the statistical spread term. *Leg B: the posterior mean tracks true reliability up to drift bias* (Step 3 below). Combining both legs via the triangle inequality then gives $|\text{LCB}(i,t) - \theta_{i,t}| \le |\text{LCB} - \mu_{i,t}| + |\mu_{i,t} - \theta_{i,t}|$, which is (9).

Step 3: Tracking bias (corrected). Under the simplified setting, the steady-state posterior mean is a geometric average of past Bernoulli outcomes,

$$\mu_{i,t} \approx (1-\lambda)\sum_{s=0}^{\infty}\lambda^s X_{i,t-s}, \qquad \mathbb{E}[X_{i,t-s} \mid \mathcal{F}_{t-s-1}] = \theta_{i,t-s}.$$

By Assumption 3.4, $|\theta_{i,t-s} - \theta_{i,t}| \le s\delta$, so the lag-$s$ observation differs from the present latent reliability by *up to* $s\delta$ — not δ. Taking the absolute deviation of the expected posterior mean:

$$|\mathbb{E}[\mu_{i,t}] - \theta_{i,t}| \le (1-\lambda)\sum_{s=0}^{\infty}\lambda^s \cdot s\delta = \delta \cdot \frac{\lambda}{1-\lambda}.$$

The final rate is therefore $O(\delta/(1-\lambda))$, the same order of magnitude as the flat-sum version but derived correctly. An earlier version of this appendix summed $\lambda^s\delta$ directly; that step was not rigorous.

Step 4: Triangle inequality. Combining Steps 2 and 3 yields (9).

Step 5: Optimal λ. Let $x := 1-\lambda$. The bound becomes $h(x) := cx^{1/2} + \delta x^{-1}(1-x)$. For

small x this reduces to $h(x) \approx cx^{1/2} + \delta x^{-1}$; setting $h'(x) = 0$ gives $x^* = (2\delta/c)^{2/3}$, hence $\lambda^* = 1 - (2\delta/c)^{2/3}$ at leading order. This is engineering guidance for testnet calibration, not a guarantee. □

C. Heuristic Bound: Claim 10.1 (Tentative Fork Risk)

Proof sketch. **Step 1.** Bad_v requires $\geq q_{\text{tent}} - f$ honest votes for each of two conflicting proposals. Hence Bad_v is impossible if $K_v^* < 2(q_{\text{tent}} - f)$.

Step 2. $\mathbb{P}(\text{Bad}_v) \leq \sum_{k \geq 2(q_{\text{tent}}-f)} \mathbb{P}(K_v^* = k) \cdot \mathbb{P}(A \geq m \wedge B \geq m \mid k)$, where $m = q_{\text{tent}} - f$ and K_v^* is Poisson-Binomial with $\{\underline{p}_{i,t}\}$.

Step 3. Given k honest online validators, each votes for A or B independently with probability ρ_s. The model does not specify fork-choice behavior, abstention, or asynchronous delivery; treat as a risk estimate.

Step 4. Feasibility: liveness and safety constraints are monotone in opposite directions in q_{tent}; a crossing point exists for small enough ρ_s, found by binary search over $\{1, \ldots, q_{\text{full}} - 1\}$. □

D. Proof: Lemma 5.1 (Sybil Occupancy Bound)

Proof. Under Assumption 3.3, registration is capped at $\text{Cap}_{\text{bucket}}$ identities per device per bucket. With k attested devices controlled by the adversary, the total number of adversary-held eligible slots in E_t is at most $k \cdot \text{Cap}_{\text{bucket}}$. This is the occupancy bound stated in Lemma 5.1. There is no cryptographic Sybil resistance independent of Assumption 3.3. □

On the committee-share bound. Write $\mathcal{A}$ for the adversary-held subset of the eligible registry E_t. Remark 5.4 discusses why we do not promote the uniform committee-share bound

$$\mathbb{E}\left[\frac{|C_v \cap \mathcal{A}|}{n}\right] \leq \frac{k \cdot \text{Cap}_{\text{bucket}}}{M_t}$$

to a proved result. The bound holds under uniform sampling, but Phase uses score-weighted sampling in the exploit tranche. A patient adversary who registers devices and participates honestly for a sufficient number of periods can accumulate reputation scores comparable to incumbents' before executing an attack. In that regime, the adversarial committee fraction is bounded above by the adversary's share of the eligible registry's total *selection weight* — the score raised to η_{sel} (Remark 4.4), which concentrates more sharply than a score-proportional draw — and this can exceed the uniform share. The mechanisms that push against this are the M_t soft cap (limiting adversarial registry share for any fixed k once activated; it ships inert, §9) and the probation gate (delaying score accrual). Neither yields a clean closed-form committee-share bound, and we treat this as conditional rather than proved.

E. Sketch: Claim 18.1 (Heuristic Manipulation Bound)

Sketch. Let T_{bad} be periods of deliberate underperformance. Cost: $\Delta \geq T_{\text{bad}} \cdot \pi_i^{\text{net}}(s_{\text{normal}}, M_t)$. Gain: $G \leq \varepsilon_{\text{explore}} \cdot n \cdot r_{\text{avg}}$ (fixed tranche budget). For T_{bad} large enough that the Beta posterior widens appreciably and the explore-tranche selection probability rises by a non-negligible amount, $\Delta \geq G$ under the stated bound. We do not formalize "widens appreciably" here; a rigorous version would require an explicit function relating T_{bad} to the change in p_{sel} induced by the posterior's second moment. The bound captures the right order of magnitude for the single-identity underperformance template only. □

F. Leader Grief Analysis

The quantitative analysis below is illustrated at the wire-format ceiling ($n = 32$, with registry size $M_t = 300$ in the simulation scenario), not at the shipped $n \in [7, 10]$; the mechanism is size-independent.

A malicious leader can withhold block proposals, denying income to all $n - 1$ non-leader committee members at a cost of one view of own income plus one Case B reputation penalty. The cost-to-harm ratio is approximately $1 : (n - 1) = 1 : 31$.

Griefer trajectory. Under the Beta-EMA, each withheld-as-leader event produces a Case B update ($\beta_{i,t}$ += 1) while the griefer's honest non-leader observations produce Case A updates. With committee size $n = 32$ and registry size M_t, the griefer is selected as leader with probability $1/n$ of their committee appearances, and selected to the committee with probability n/M_t. Expected grief events per period: $n/M_t \cdot 1/n \cdot \text{views} = \text{views}/M_t$.

At the simulation resolution of 50 views per period and $M_t = 300$: $50/300 \approx 0.17$ grief events per period. The griefer itself appears in $\sim 50 \cdot n/M_t \approx 5.3$ committees per period, so ~ 5.2 honest observations push α up by $\sim 0.80 \times 5.2 \approx 4.2$ while grief events push β up by ~ 0.17. The ratio of honest to grief evidence is $\sim 25 : 1$: a single period of griefing barely moves the posterior, and the effect accumulates only across many periods, as the detection-time estimate below shows.

Detection time. After 30 periods of always-griefing-as-leader, the griefer's LCB is approximately 0.564. For scale, $\tau_{\text{grad}} = 0.500$ — but that threshold is the *graduation* gate a probationer must clear to become *Full*, not a jail trigger, and nothing re-applies it to a validator that has already graduated. Crossing it downward therefore has no effect: the griefer is not approaching expulsion, because no expulsion path exists, and an anchor is seated in every round whatever its score, so only a graduate's seat responds to the degraded weight at all. At $M_t = 1000$, the detection time stretches proportionally because leader selection is rarer.

Implication. Leader grief is cheap per event (harm ratio 31:1) but self-correcting via LCB degradation. The correction speed depends on registry size. At large M_t, a leader-specific Beta-EMA tracking filter (lower λ, separate from the consensus LCB) would provide faster detection. This is a candidate mechanism for a later stage of the rollout, acti-

vated progressively as the network accumulates live data.

Copyright © Phase Core Team 2026. All rights reserved.