Anemos — Soundness of Consensus, Oracle, Economics, and Mathematics

A native, protocol-level over-collateralized stablecoin on a fair-launch BFT chain

The Anemos Team

Version 0.13.0 · 2026-07-04 · Working Paper

Abstract

We present the design and a soundness analysis of **Anemos**, an L1 that integrates an over-collateralized, dual-tranche stablecoin directly into a Byzantine-fault-tolerant (BFT) proof-of-stake consensus. Four properties are analysed. (1) **Consensus**: instant-finality BFT with VRF sortition, where the stablecoin module is folded into the block state root and must be **bit-for-bit deterministic** to avoid forks. (2) **Economics**: a no-treasury, no-premine, no-supply-cap monetary policy that mints each block's reward from a geometric-decay schedule converging to a **supply-indexed** perpetual tail (so the validator security budget and reserve funding scale with an elastic supply rather than decaying to dust), split between the block proposer and the stablecoin reserve by a **health-dependent** rule under a hard validator-reward floor (PhiMin = 75%). (3) **Stablecoin**: a senior peg token plus a junior reserve coin, with a minting collateral floor, an O(1) rebase index for "golden-age" interest, a per-period circuit breaker, an intermediate **debt-restructuring** mode (recovery tokens) for a moderate shortfall, and an orderly pro-rata wind-down for the catastrophic tail. The senior tranche is the *only* tranche with an APR — a single, capped ($\leq 16\%$/yr), **smoothed** golden-age yield paid at a steady per-block rate; the junior reserve coin earns *no* cash yield and accrues value only as residual equity ($\max(0, \text{reserve value} - \text{liabilities})$). (4) **Oracle**: a consensus-embedded price feed in which a deterministic per-block committee subset signs prices that the proposer aggregates into a block-body oracle section attested by the existing certificate, eased into a slow **time-weighted average (price TWA, $\sim$ 3.5 h)** of the subset median, where *deviation* is slashed but *absence* is never slashed, fed by an off-chain multi-source price feeder; the price TWA's security compensator is a committee **term limit** ($\sim$ 8 h tenure rotation). We prove conservation of the native unit across every mint/burn path, give the fixed-point determinism rules, and state — and **do not contradict** — an impossibility result (Theorem 1): no over-collateralized design backed by its own volatile native coin can be made unconditionally solvent. We therefore engineer for *graceful* failure and quantify ruin probability by Monte-Carlo simulation.

Contents. 1 Introduction · 2 Consensus · 3 Monetary policy & emission · 4 The stablecoin · 5 The oracle · 6 Determinism & math · 7 Risk & ruin · 8 Comparison · 9 Conclusion · 10 References · Appendix A Parameters · Appendix B Notation.

Contents

1 Introduction 3
2 Consensus Layer 5
3 Monetary Policy and Emission 7
4 The Native Stablecoin 15
5 The Oracle 31
6 Determinism and Mathematical Foundations 53
7 Risk Analysis and Ruin Probability 54
8 Comparison to Prior Systems 56
9 Conclusion 59
10 References 60
Appendix A — Launch parameters 61
Appendix B — Notation 64

1 Introduction

1.1 What Anemos is

Anemos is a fair-launch, instant-final proof-of-stake layer-1 with a native, protocol-level over-collateralized stablecoin and a consensus-embedded price oracle. It is deliberately minimal — account-based, Byzantine-fault-tolerant with instant finality, VRF cryptographic sortition, **no virtual machine**, and a small fixed set of transaction types — and it is engineered so that a full staking node runs on commodity hardware, including as a full node on an Android phone. Within that lean core, Anemos adds the two things a sound on-chain dollar needs and folds them directly into consensus rather than bolting them on as smart contracts.

A fair monetary policy. Anemos has **no premine to insiders, no foundation or team allocation, no treasury drawdown, and no supply cap.** Genesis distributes a disclosed, reproducible fair-launch allocation from a public, documented seed, and the coin supply is **minted on every block** by a Kaspa-style geometrically-decaying schedule (initial reward $R_0 = 2$ ANM) that converges to a **supply-indexed perpetual tail** of $\approx 2\%$ of supply per year (§3). Indexing the tail to supply — rather than fixing it to a nominal constant — is a deliberate *security* choice: it keeps the validator security budget and the stablecoin reserve funding a meaningful share of an elastic supply for the life of the chain, so the protocol funds its own security and its own reserve forever. The whole block reward flows to those who secure and use the network — the proposer (and its optional delegate) and the stablecoin reserve — never to a privileged founder set.

A native stablecoin. Anemos's defining contribution is a Djed/Shen-style, dual-tranche, over-collateralized stablecoin implemented as **native consensus logic** — a first-class module mirroring the chain's existing bond/withdraw/sortition modules, with its own state, executors, and a merkle subtree folded into the block state root (§4). Building the stablecoin *into* the protocol, rather than as a contract on top of a VM, is what lets the block-reward split, the reserve accounting, the rebase-index interest, and the failure-handling all be **bit-for-bit deterministic** and **O(1) per block** — the properties that keep the chain fork-safe and phone-viable. Price, too, is delivered as **consensus-embedded block data**: a rotating committee subset signs the ANM/USD price each block, the proposer aggregates the signatures into one BLS aggregate carried in the block body, and the *existing* certificate attests it — so the oracle reuses the chain's own authentication with no new consensus message and no change to the riskiest vote/certificate path (§5).

Built on a clean foundation. Anemos builds on the Pactus blockchain [1], whose lean, instant-finality BFT proof-of-stake core — account-based, VM-free, VRF-sortition — is an excellent base for exactly this kind of minimal, deterministic, mobile-friendly system. We are grateful for that foundation and keep its authorship and license intact; Anemos is a respectful fork that adds a fair launch and a native stablecoin to a design we admire.

1.2 Honest framing

This paper argues the *soundness* of the above, and it does so without overselling. The genuinely hard parts of an on-chain dollar are not the engineering mechanics — those are well-trodden — but the economics, and we state them plainly: (a) the oracle's trust ceiling — a USD peg needs a USD price, and a new chain has no native source, so the ceiling is honest-majority-of-stake; (b) the *reflexivity* of a stablecoin whose reserve is denominated in its own volatile native coin; and (c) bootstrapping real value before the peg can mean anything. We make the adversarial case against our own design explicit rather than hiding it: Theorem 1 (§4.6) shows that the peg **can** fail under a sustained native-coin collapse, for *any* starting collateral ratio, and we do not claim otherwise. Anemos's answer is not a false "cannot die" guarantee but an engineered *graceful* failure — an intermediate debt-restructuring mode and, beyond it, an orderly pro-rata wind-down (§4.7–4.8) — whose rarity under realistic parameters we quantify by simulation (§7).

Two further honest disclosures, in the same spirit. **(d) The genesis lockup's scope.** The one-year lockup (§5.12) applies only to the genesis *validators'* self-bond — a small fraction of genesis supply. The remaining fair-launch distribution across the placeholder address set carries no lockup and is liquid from block 1; that set is a reproducible interim distribution derived from a public seed, slated for replacement by the real recipient set at launch. This is a governance/distribution detail outside consensus-code scope, but material to a reader evaluating the "no insider premine" claim — which remains separately true (the treasury sentinel holds zero balance and there is no team drawdown) but should not be read as implying every genesis coin is locked. **(e) Pre-launch governance centralization.** During the current pre-launch/testnet phase, genesis parameters, protocol upgrades, and the finalization of the real mainnet recipient set are controlled by the project's maintainers, not by any on-chain governance mechanism. We name this centralization dependency rather than omit it, consistent with the self-disclosure above.

---

2 Consensus Layer

2.1 Inherited guarantees

Anemos inherits Pactus's [1] consensus: a practical-BFT-family state machine (propose → prepare → precommit → commit) with **instant finality** — a committed block is final, there are no reorganisations — over 10-second rounds with at most 1000 transactions per block. Validators are selected by **VRF sortition** (Algorand-style cryptographic sortition [2]) over a proof-of-stake committee in the provable-security tradition of Ouroboros [3], and votes aggregate into a single BLS certificate per height. Under the standard partial-synchrony model, BFT consensus is **safe** with up to $f < n/3$ Byzantine voting power and **live** once the network is synchronous [4], [5]. Anemos does **not** touch the vote/certificate hot path — the riskiest code in any BFT chain — which is why the oracle is delivered as a **consensus-embedded block-body oracle section** attested by the existing precommit certificate (§5), **not** as ordinary transactions and **not** as a new consensus message.

2.2 State commitment and the determinism invariant

Every node re-executes every block and must agree on a single **state root**. Pactus folds the account and validator trees with a single hash of their concatenated roots:

$$\mathsf{avRoot} = H(\mathrm{accRoot} \,\|\, \mathrm{valRoot}).$$

Anemos extends this to commit the stablecoin module — **both** its aggregate state and per-holder ownership — and then folds in six further subtrees, for **nine merkle trees — eight subtree families — in a fixed, append-only order**. The construction follows a fixed associativity: the subtrees are first folded **into the stablecoin root**, and the account/validator root is mixed in **last**. First the stablecoin root combines the aggregate state and the per-holder ownership tree:

$$\mathsf{scRoot} = H\,\big(\underbrace{H(\sigma)}_{\text{aggregate}} \,\|\, \underbrace{\mathsf{holderRoot}}_{\text{ownership}}\big),$$

where σ is the serialized aggregate stablecoin state and $\mathsf{holderRoot}$ is the root of a persistent-merkle tree over per-holder leaves (or a fixed sentinel $H_\varnothing$ when there are no holders). The holder subtree is **always folded** — it uses the sentinel $H_\varnothing$ when there are no holders, never skipped — so the two-leaf scRoot shape is constant. Committing *ownership*, not just aggregate supply, is essential: without it a node could serve a fake per-holder balance that no honest node could detect via the root.

The stablecoin root is then extended by six more subtrees — the **oracle-deviation**, **genesis-lock**, and **pending-unbond** trees, followed by the pooled-delegation **pool** aggregate and per-delegator **position** trees and the **pending-delegation** escrow tree (the seated-validator delegate/undelegate buffer, §5.11) — **each folded in only when it is non-empty**, and **always in this fixed, append-only order** (a new family is only ever appended at the end):

$$\mathsf{scRoot}' = \mathsf{scRoot} \bowtie \mathsf{devRoot} \bowtie \mathsf{lockRoot} \bowtie \mathsf{unbondRoot} \bowtie \mathsf{poolRoot} \bowtie \mathsf{posRoot} \bowtie \mathsf{pendRoot}, \qquad \mathsf{stateRoot} = H\,(\mathsf{avRoot} \,\|\, \mathsf{scRoot}'),$$

where $A \bowtie B \equiv H(A \,\|\, B)$ when B's subtree is non-empty and $A \bowtie B \equiv A$ otherwise — and the account/validator root avRoot is combined with the extended stablecoin root in a single final hash. These two empty-handling rules are **distinct**: the holder subtree is *always* folded (with the $H_\varnothing$ sentinel when empty), whereas the six $\bowtie$ subtrees are *skipped entirely* — not sentinel-substituted — when their record set is empty. So an absent $\bowtie$ subtree contributes **nothing**. The fold order and the only-when-non-empty rule are themselves consensus contracts: folding an empty $\bowtie$ subtree, mixing avRoot in early, or folding out of order, forks the chain. The nine-tree, eight-family fold is illustrated in **Fig. 1**.

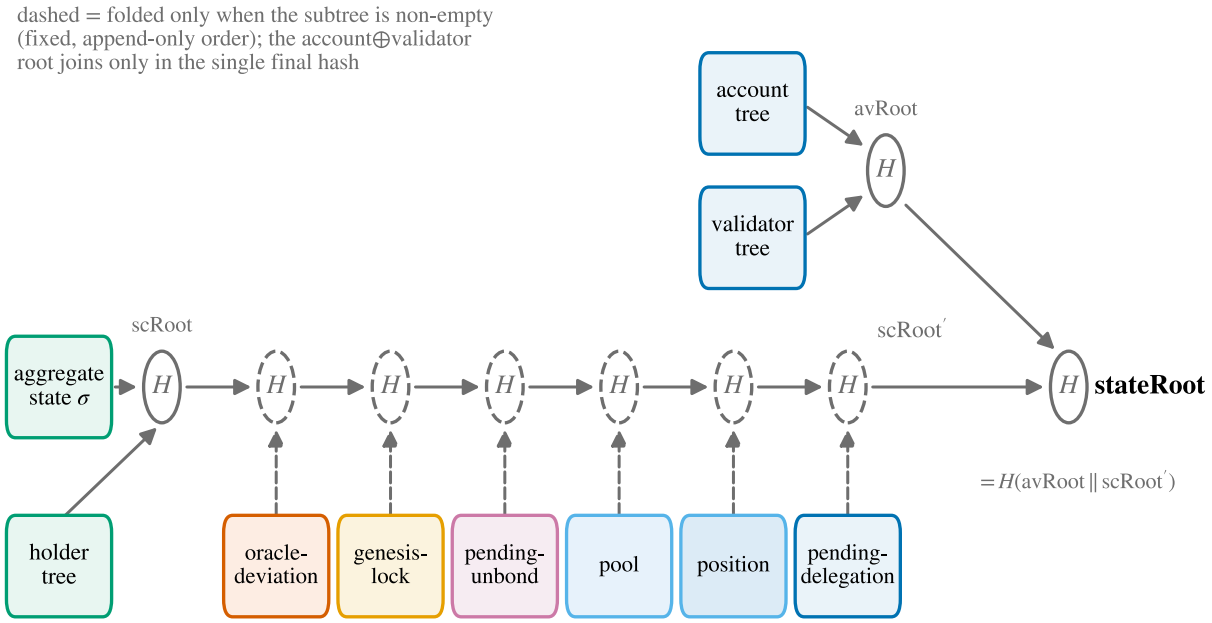

Fig. 1 — The nine-tree (eight-subtree-family) state-root fold: the six conditional subtrees extend the stablecoin root first, and the account⊕validator root is mixed in last, in the single final hash

The determinism invariant. Because the root is consensus-critical, **any** non-determinism in the module is a chain halt or fork. Anemos enforces:

- *Integer-only fixed-point math* — no floating point anywhere on the execution path (§6).
- *Canonical, fixed-width serialization* of every committed object, so $\mathsf{Bytes} \circ \mathsf{FromBytes} = \mathsf{id}$.
- *Order-independent commitment*: holder leaves are keyed by a stable index assigned in transaction order, and $\mathsf{SetHash}\ (i, \cdot)$ is commutative, so the rebuilt tree after restart equals the live tree.
- *Trusted height*: per-block emission and the reserve slice are computed from the sandbox's block height, never from proposer-supplied transaction fields.

The aggregate-state serialization is pinned by a frozen golden test vector reproduced on every supported architecture (amd64, arm64, arm32/Android); a state-root divergence is treated as a defect (§6.3).

2.3 Android-grade cost bound

A design constraint is that a full staking node must run on a phone. Every per-block addition the module makes is therefore **O(1)** and integer-only: the rebase index bump, the TWA updates, the per-block power-weighted-median computation, and the restructuring writedown are all constant-work; interest is *not* iterated per holder (§4.3).

3 Monetary Policy and Emission

3.1 No treasury, no premine, mint-on-block

Anemos runs **no treasury at all**: there is no premine, no foundation reward, and no total-supply cap. The block subsidy is **minted on demand** from the schedule rather than drawn down from a pre-funded balance, so the security budget can never exhaust a finite pool. The Treasury address is purely a *coinbase sentinel* that identifies the subsidy transaction; it holds **zero balance** and is never debited. Each block's subsidy is **minted** — its recipients are credited with no sender debit — exactly as a coinbase transaction in a UTXO chain. Genesis itself — on mainnet and testnet alike — is **deterministic and byte-reproducible**: every quantity the genesis hash depends on derives from a documented public seed with a pinned genesis timestamp, so a regenerated genesis is byte-identical to the one embedded in the released client software.

3.2 The emission schedule: decay to a supply-indexed perpetual tail

The per-block reward follows a Kaspa-style *chromatic halving* [6], [7]: a smooth geometric decay that halves every twelve months, descending from an initial reward R_0 toward a perpetual tail. **Anemos's tail is not a fixed nominal constant — it is indexed to total supply.** This is a *survival and security* mechanism, not an inflation target (see the boxed rationale below):

$$R(m) = \max\Big(R_\infty(m), R_0 \cdot \rho_{\text{dec}}^{\,m} \Big), \qquad \rho_{\text{dec}} = \Big(\frac{1}{2}\Big)^{1/12}, \qquad m = \big\lfloor\, h / B_{\text{month}} \,\big\rfloor,$$

where the per-block tail tracks a fixed fraction $\tau = 2\%$ of *total minted supply* S per year, **laddered to its leading digit** so it only steps at order-of-magnitude rungs (avoiding a jittery reflexive feedback between supply and the reward):

$$R_\infty(m) = \frac{\text{ladder}\,\big(\,\tau\, S(m)\,\big)}{B_{\text{year}}}, \qquad \text{ladder}\,(x) = \Big\lfloor\, \frac{x}{10^{\lfloor \log_{10} x \rfloor}} \,\Big\rfloor \cdot 10^{\lfloor \log_{10} x \rfloor}, \qquad B_{\text{year}} = 3,153,600.$$

Here h is the block height and $B_{\text{month}} = 30 \cdot 24 \cdot 3600/10 = 259,200$ blocks (10 s blocks). The implementation fixes ρ_{dec} to the exact rational $4053909305/2^{32}$ ($\approx (1/2)^{1/12}$); ladder $(\cdot)$ uses an **integer** $\log_{10}$ loop (never a float), so every quantity is bit-identical on every CPU and every integer width (§6.1). $S(m)$ is the **schedule supply** — genesis supply plus the pure-function cumulative emission — *not* the live total of all balances, which the stablecoin module can move sharply each block; indexing on the smooth deterministic schedule keeps the tail and the stake cap (§5.11) immune to mint/redeem whipsaw. The per-month tail and the laddered stake cap are re-computed once per month and cached in a vector, so the per-block lookup stays $O(1)$.

With launch parameters $R_0 = 2$ ANM, $\tau = 2\%$ and an $8,032,000$ ANM genesis, the geometric decay falls below the supply-indexed tail — the **handover** — at month $m^\star \approx 54$ ($\approx$ year 4.5), after which the reward latches to the perpetual tail forever (**Fig. 2**).

> **Why a *supply-indexed* tail (the whole point).** Anemos has an **elastic** supply (the native stablecoin's reflexive mint/redeem plus this perpetual emission grow supply over decades). A chain with elastic supply but *fixed nominal* parameters dies slowly: a fixed tail reward becomes a negligible fraction of a growing supply, so the **validator security budget collapses toward zero**, $\text{AttackCost} \sim \varphi^{0.5} \cdot \text{price}^{1.5} \cdot \text{stake}$ falls, and the network becomes cheap to capture. Indexing the tail to 2% of supply keeps the security budget — and the stablecoin reserve funding (§4.3, §4.8) — a **meaningful, non-vanishing share of supply in year 1, year 10 and year 50 alike**. We do not micro-optimize to hit exactly 2% (the ladder deliberately floors it to a round number); a slightly-off inflation figure is harmless, a security budget that decays to dust is fatal.

> **Be precise about what is floored: this is a share-of-*supply* (ANM-denominated) invariant, not a USD security guarantee.** The tail keeps the budget a constant-order fraction of the ANM supply; it does **not** put a floor under the *USD* cost to attack, which by §3.5/§4.6 scales as $\mathsf{AttackCost} \sim \varphi^{0.5} \cdot \text{price}^{1.5}$ and therefore collapses with the ANM price regardless of how the tail is set — no emission rule can secure a chain whose token has gone to zero. And because the realized rate traces the 2%-to-1% sawtooth described below, even the ANM-denominated share is a *band*, not a flat line. The claim here is narrow and deliberate: the tail stops the security budget from vanishing *relative to supply* (the slow-death mode of a fixed nominal tail), which is the failure it is designed to prevent — not a guarantee of any absolute USD security level.

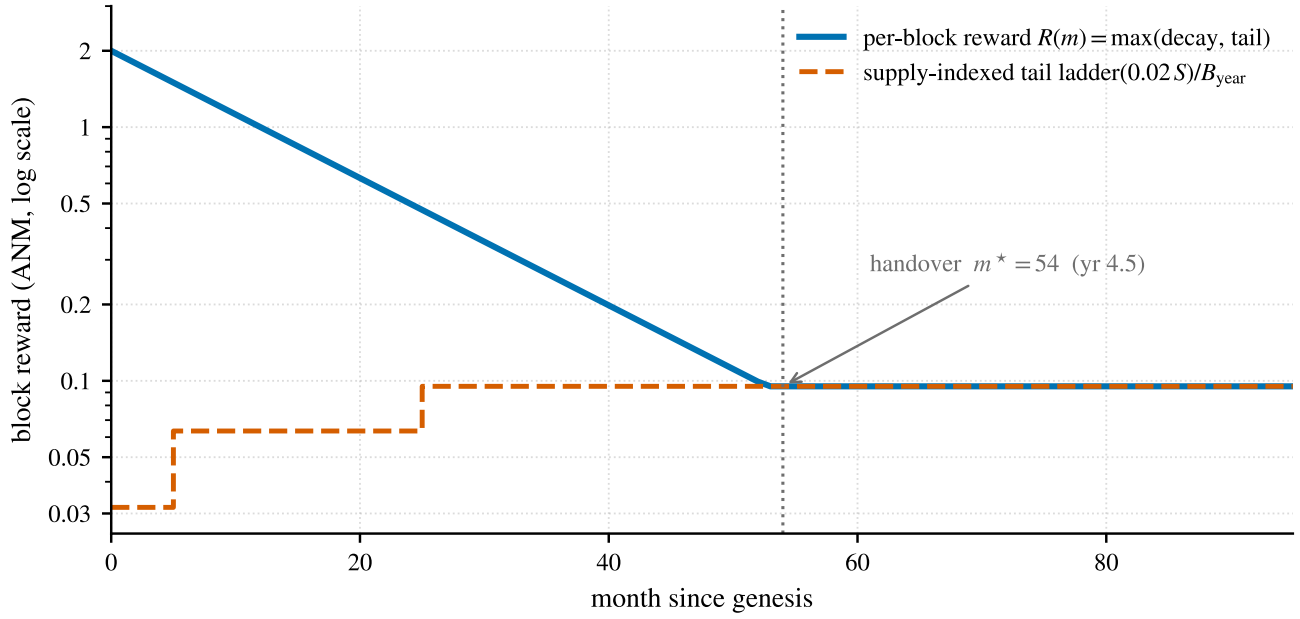

Fig. 2 — Emission schedule

The payoff of indexing the tail to supply is shown directly in **Fig. 3**. First, a precision the figure makes explicit: the 2%-of-supply tail is the **combined** budget — emission is a *single pot* split each block between validators and the stablecoin reserve (§3.4–§3.5). Validators receive only the **net complement** of the comfort-band reserve slice ($s = 25\%$), so the *validator* security budget is $0.75\times$ the gross emission share, not the full 2%; the hard $\phi_{\min} = 75\%$ floor (§3.5) guarantees validators keep at least three-quarters of every block even in the worst case — the floor and the comfort-band share coincide at 75% — and the slice tapers off in distress so the realized validator share rises back toward 100% exactly when security matters. Fig. 3 therefore plots the **validator net budget** (which is also the $\phi_{\min}$ floor) as the primary curve and keeps the gross combined-pot tail as a faint reference; both Fig. 3 and Fig. 5 treat the reserve slice as reducing the validator share. The headline result holds: the validator budget stays a **bounded, non-vanishing** share of supply over the decades under the supply-indexed tail, whereas a fixed nominal tail $R_0/10$ decays monotonically toward dust as supply grows — exactly the slow-death failure mode the indexing is designed to avoid. Be precise about the *shape*, though: because the ladder floors $0.02\,S$ to a single leading digit, the realized (combined-pot) tail-over-supply does **not** sit flat at 2% — it traces a **sawtooth that starts each leading-digit plateau at exactly 2% and decays toward $2\% \cdot d/(d+1)$ by the plateau's end (for leading digit $d \in \{1, \ldots, 9\}$), then jumps back to 2% when supply crosses the next rung**. The depth of the tooth is *not* uniform across rungs: it is deepest on the leading-digit-1 rung, which falls to $\approx 1\%$ (half of nominal), and shallower on higher-digit rungs — e.g. the leading-digit-9 rung only falls to 1.8% (90% of nominal) — because $d/(d+1)$ is closer to 1 for larger d (the validator net curve is the

same sawtooth scaled by 0.75, so a $\approx 1.5\% \to 0.75\%$ tooth on the deepest, digit-1 rung). Concretely the gross realized rate is ladder $(0.02\,S)/S$: it equals exactly 2% when $0.02\,S$ lands on a power of ten ($S = 50 \cdot 10^k$), then falls toward $\approx 1\%$ as S approaches $100 \cdot 10^k$ (the per-block tail is frozen on a rung while supply keeps growing), before the next rung snaps it back to 2% — so **the floor of the deepest (digit-1) tooth is about half the nominal rate**; other rungs bottom shallower, per the $d/(d+1)$ rule above. This is harmless to the design because the argument is a *survival floor*, not an inflation target: even at the $\approx 1\%$ bottom of the deepest (gross) tooth the validator budget is a meaningful, constant-order share of supply — categorically different from the fixed-nominal tail, which falls toward zero without bound. The same coarse leading-digit ladder governs the stake cap (§5.11), so its realized fraction has the same sawtooth character (deepest tooth $\approx$ half of nominal, on the leading-digit-1 rung) about its 0.8% nominal. (This is an asymptotic, share-of-supply argument, so the figure necessarily uses a multi-decade horizon — the only place in this paper that does; the *practically-relevant* supply trajectory is the 16-year view of Fig. 4 below.)

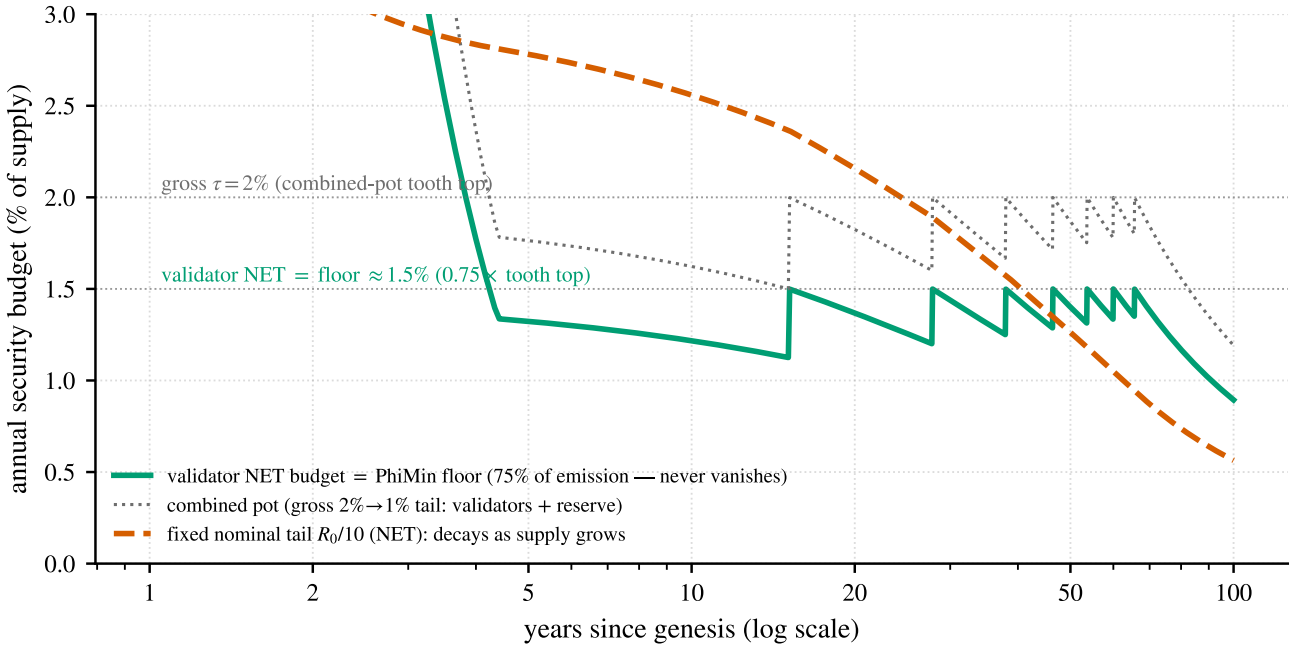

Fig. 3 — Security budget vs supply

3.3 Supply trajectory

Cumulative issuance is the sum of the schedule. During the decaying phase it is a truncated geometric series; in the perpetual tail it grows as a laddered ~2%/yr of supply:

$$S(h) = S_{\text{gen}} + \sum_{k=1}^{h} R\,\big(\,\lfloor k/B_{\text{month}} \rfloor\,\big),$$

so post-handover total supply grows **geometrically at the laddered tail rate** rather than linearly — by design, because the security budget and reserve funding must scale *with* supply, not shrink relative to it. **Fig. 4** plots the first sixteen years — the practically-relevant horizon: the front-loaded decay phase carries genesis's 8.032M ANM to $\approx$ 16.86M by the R_0-decay-to-tail handover at year ≈ 4.5, after which the gentle laddered $\approx 2\%$/yr tail funds validators and the stablecoin reserve, so golden-age yield funding (§4.3) and recovery-token buyback (§4.8) never decay to a vanishing share of supply. The same compounding continues smoothly beyond the window (the schedule is a closed-form pure function of height, e.g. $\approx$ 18.49M ANM by year 10); all money math is staged in arbitrary-precision integers and the schedule latches at a safe int64 ceiling, so supply stays representable and overflow-free for the life of the chain.

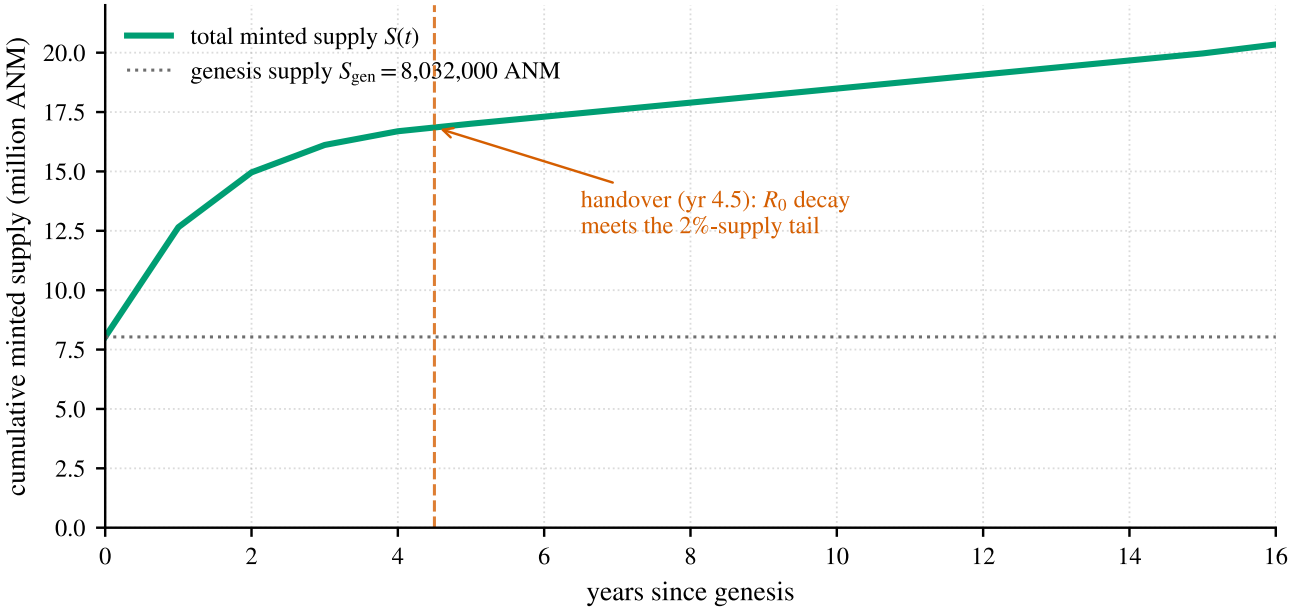

Fig. 4 — Cumulative supply (first 16 years)

3.4 Reward split and the conservation theorem

Let F be the block's accumulated transaction fees and $s(h)$ the **total stablecoin carve-out** from the proposer subsidy (§4), with $s(h) = 0$ while the module is inactive. The carve-out has **two parts** — the health-dependent reserve slice and, above the target, the capped cash-yield diversion (§3.5):

$$s(h) = \underbrace{\left\lfloor \frac{s\left(\bar{r}\right)}{\mathsf{RatioScale}} R(h) \right\rfloor}_{\mathsf{reserveSlice}\ (h)} + \underbrace{\mathsf{diversion}\ (h)}_{\text{above-target, §3.5}}, \qquad \mathsf{pool}\ (h) = R(h) - s(h)\ \left(\geq \phi_{\min} R(h)\ \right),$$

where $s\left(\bar{r}\right)$ is the **health-dependent** slice fraction defined in §3.5 and $\mathsf{diversion}\ (h)$ is the above-target 25% senior cash-yield diversion (non-zero only when the instantaneous ratio is above the 1200% target, §3.5, §4.1; zero otherwise). The validator floor $\phi_{\min} = 75\%$ binds the **combined** carve-out: the divertible amount is capped at $R(h) - \lfloor R(h)\,\phi_{\min} / \mathsf{RatioScale} \rfloor - \mathsf{reserveSlice}$, so $\mathsf{reserveSlice} + \mathsf{diversion} \leq (1 - \phi_{\min}) R(h) = 25\%\, R(h)$ and the proposer always keeps $\mathsf{pool}\ (h) \geq \phi_{\min} R(h) = 75\%\, R(h)$. (Both s_{base} and the above-target diversion are individually bounded at $1 - \phi_{\min} = 25\%$, and the two *can* apply in the same block — in the lag window where the slow ratio TWA still authorises the comfort-band slice while the *instantaneous* ratio already sits above the target, §4.3 — but the diversion is clamped to the combined cap $R(h) - \lfloor R(h)\,\phi_{\min} / \mathsf{RatioScale} \rfloor - \mathsf{reserveSlice}$ stated above, so the floor holds in every regime, the overlap window included.)

There is **no foundation cut**. Before it is paid out, the proposer pool is re-partitioned once more: an **oracle-participation slice** $\mathsf{oracleSlice}\ (h) = \lfloor \sigma\, \mathsf{pool}\ (h) \rfloor$ ($\sigma = 10\%$, §5.16) is carved out and minted **at commit**, split equally among the block's correct oracle attesters (integer dust to the reserve), and the proposer keeps the remainder. The proposer's share is paid to the proposing validator-pool, where it splits between the **operator commission** $c = \lfloor \mathsf{pool} \cdot \mathsf{commissionBps}\ / 10000 \rfloor$ (to the operator's spendable account) and the **delegators' reward escrow** (the remainder, distributed by the pool's per-share accumulator, §5.11) — the escrow portion is minted directly into the pool's slash-immune reward escrow during block execution, so a pool proposer's coinbase itself carries only the operator pay plus fees; fees are paid to the proposer. All destinations are ordinary accounted balances, so the split is reward-only and conserves native units exactly. The coinbase transaction's total value is validated against

$$\mathrm{subsidyAmt}\ (h) = R(h) + F - s(h) - \mathsf{oracleSlice}\ (h),$$

with the carve-out $s(h)$, the oracle slice, and (for a pool proposer) the delegators' escrow each minted by the same pure function of committed state as the block is processed — so the block's total mint is always exactly $R(h) + F$, however it is partitioned.

> **Theorem (per-block conservation).** *For every block at height h, the net change in total native units — defined as $T = \sum$(account balances) $+ \sum$(validator stakes) $+ \sum$(pending-unbond entries) $+ \sum$(pool reward escrows) $+ \sum$(pending-delegation escrows) $+$ Reserve — equals exactly $R(h)$, regardless of the transaction mix or delegation state.*

Proof sketch. Fees are removed from payers and re-minted to the proposer inside the subsidy, so they cancel ($-F + F$). The coinbase mints $R(h) + F - s(h) -$ oracleSlice to the proposer (for a pool proposer, the delegators' portion of that amount is minted into the pool's reward escrow instead — a bucket of T); the carve-out $s(h)$ — **both** the reserve slice **and** the above-target diversion — is minted into the reserve; and the oracle slice is minted to the block's correct attesters (accounts, pools, and dust-to-reserve — all buckets of T); together $R(h) + F$. Subtracting the F removed from payers leaves $R(h)$. Unbonding only moves units between the stake, pending-unbond, and balance buckets, and a slash moves stake or pending entries into the reserve; a delegation aimed at a seated validator likewise moves balance into the pending-delegation escrow when it executes, and its activation (or refund) moves the escrow into stake (or back to balance) — all inside T. (The pending-delegation bucket holds two distinct kinds of entry: a pending **deposit** genuinely escrows real ANM value pulled from the depositor's balance, whereas a pending **undelegate** entry is a zero-value *request* — it records only the amount the delegator wants released, with no ANM moved yet; the real value stays in the validator's stake until activation moves it into the pending-unbond bucket. Both kinds are accounted for inside T either way, since an undelegate request's value simply has not left the stake bucket yet.) The above-target diversion only **re-splits** the diverted ANM between the senior interest pool (paid out by raising the senior rebase index, §4.1, §4.3) and a reserve buffer (which lifts junior residual equity) — all inside the reserve, minting no new ANM — so T is conserved. Stablecoin mint/redeem likewise only *move* units between accounts and the reserve (conserving T). The carve-out is computed from the **same trusted height** h both where it is carved out of the subsidy and where it is minted, so they cancel at month boundaries. No other mint path exists. ■

This invariant holds across the full case matrix (module on/off × {empty pool, partially-filled pool, full pool} × any commission rate); making the slice a fraction of the live reward — rather than a fixed absolute cut — is what keeps it structural at month boundaries and under any pool state.

3.5 Variable subsidy: a health-dependent reserve slice

Because the stablecoin module lives *inside* consensus, the block-reward split can read system health in real time. We therefore make the slice s **a function of the committed collateral-ratio TWA** $\bar{r}$ rather than a fixed fraction — but the shape of that function is the opposite of the naïve intuition, for two reasons our analysis (and simulation) make precise.

(i) Security. Model validator participation by the USD break-even of a marginal (Android) node with hosting cost c: a node of stake σ stays online iff $\phi R B_{\text{year}}\,(\sigma/S)\,p \geq c$, where $\phi = 1 - s$ is the validator share. With a Pareto stake tail this yields an equilibrium active stake $S^{\star}(\phi) \propto (\phi R p/c)^{(a-1)/a}$ and a constant elasticity $\varepsilon = (a-1)/a \approx \frac{1}{2}$. The cost to acquire an attacking fraction θ of stake is then

$$\mathsf{AttackCost}\,(\theta, \phi, p) = \theta\, p\, S^{\star}(\phi) \propto \phi^{\varepsilon}\, p^{\,1+\varepsilon}.$$

The **price exponent (≈ 1.5) dominates the validator-share exponent (≈ 0.5)** — and a deeper diversion is *triggered* by a low $\bar{r}$, which is a low p. So diverting **more** when the chain is most attackable is the real hazard, not reflexivity. We therefore impose a hard **validator floor** $\phi_{\min} = 0.75$: the slice can

never exceed $1-\phi_{\min}=25\%$, so validators always keep at least three-quarters of every block (BFT liveness and oracle-committee honesty). **Fig. 5** plots this asymmetry directly: cutting the validator share to 75% costs only $\approx 13\%$ of the attack cost ($0.75^{0.5}$), while halving the price costs $\approx 65\%$ ($2^{-1.5}$) — which is exactly why the control law protects the *price-sensitive* side (taper off in a falling market) under the hard $\phi_{\min}$ floor rather than leaning the slice in.

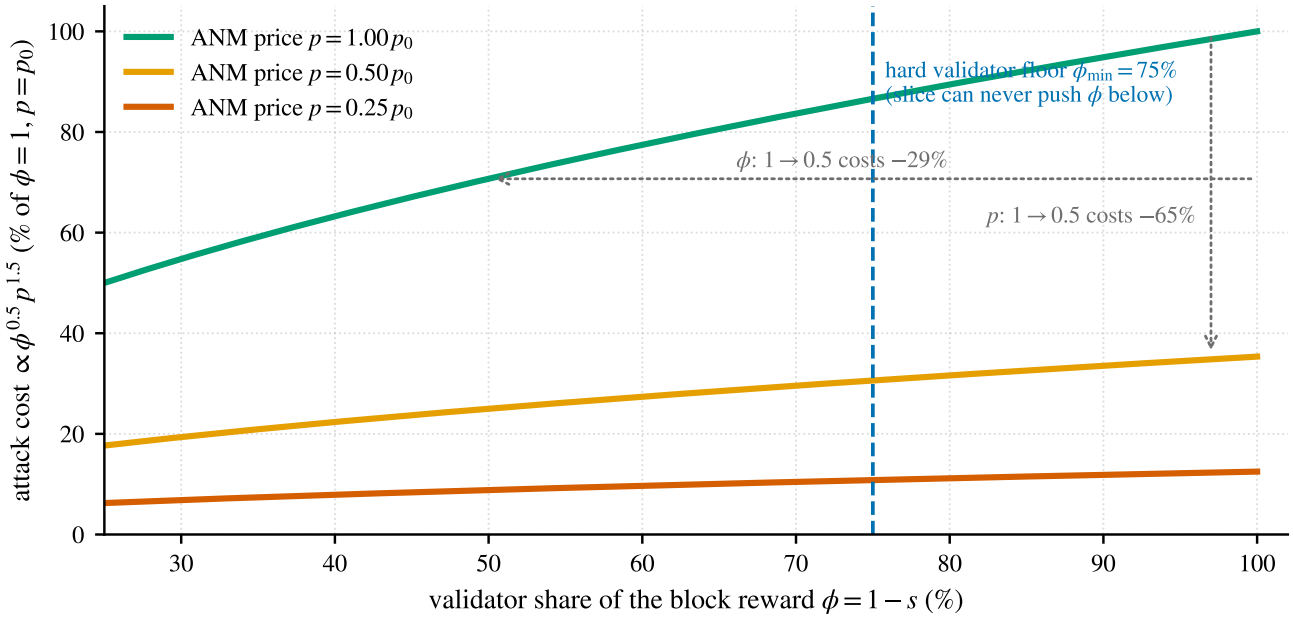

Fig. 5 — Attack-cost sensitivity: AttackCost $\propto \phi^{0.5}p^{1.5}$*, so the price exponent dominates the validator-share exponent*

(ii) Futility (Theorem 1, quantified). The reserve gains at most $s\,R$ ANM per block. Reserve *value* grows only while the fractional mint rate beats the price-decay rate:

$$\frac{d(pX)}{dt} = pX\,y + p\,sRN > 0 \Leftrightarrow \frac{sRN}{X} > |y|.$$

For any healthy reserve in a real crash ($|y| \sim 1$–3%/ day vs. an inflow of a few %/ month), the inflow is hopelessly outpaced — printing harder into a collapse cannot save the peg (the security-side echo of §4.6). Crucially, the slice is **locked in the reserve and never sold**, so unlike Terra's mint-and-dump it adds *zero* direct sell pressure; the front-running / reflexivity doom-loop does not materialise at these magnitudes.

The control law therefore does the OPPOSITE of the naïve intuition (Fig. 6): it builds the reserve buffer at a base rate in *good* times (the comfort band) and **tapers the slice toward zero as the ratio falls**, routing emission to validators (security) precisely when the token is under stress, and leaving the peg's tail risk to the restructuring and pro-rata wind-down:

$$s\left(\bar{r}\right) = \{0 \quad \text{for} \quad \bar{r} \geq 1200\% \quad \text{(ratio TWA above target: emission} \to \text{validators)}$$

5% for $800\% \leq \bar{r} < 1200\%$ (comfort: build the buffer; the marginal slice that pushes the instantaneous ratio past 1200%spills to

$$\text{taper} \quad 25\% \to 0 \quad \text{for} \quad 200\% \leq \bar{r} < 800\% \quad \text{(divert less as the chain weakens)}$$

$$0 \quad \text{for} \quad \bar{r} < 200\% \quad \text{(deep distress: emission} \to \text{security)}$$

evaluated in integer fixed-point on the *committed, pre-block* ratio TWA — identically at the propose, validate, and commit sites — so the slice subtracted from the proposer equals the slice minted into the reserve and the conservation theorem (§3.4) is preserved unchanged. $\bar{r}$ is the slow ($\alpha = 1/100$) ratio TWA, and s is a pure function of committed state, so the split is **grind-proof**: a proposer does not choose it, and moving $\bar{r}$ requires holding real collateral for ~ 100 blocks (the ratio TWA is additionally seeded

conservatively at activation so the first minter cannot land it in the slice-0 zone). The base 25% keeps the validator share at exactly the hard $\phi_{\min} = 75\%$ floor — the comfort-band share and the floor coincide.

Simulation (Fig. 7). Across crash severities, the slice policy barely changes *solvency* — at 22%/month, insolvency arrives on day 178 (fixed 25%), 196 (naïve lean-in), 168 (Anemos taper): the futility result, with the taper marginally *earlier* because it (correctly) declines to print into a fall. But it changes *security* sharply: a naïve "divert-up-to-75%" policy crushes the security margin $M = \frac{1}{3}pS/L$ ($5.19 \to 3.78$ at 5%/month), while the taper *improves* it over a flat slice ($5.19 \to 5.96$ at 5%/month; $1.76 \to 2.04$ at 10%/month) by diverting *less* from validators exactly when security is fragile. The variable subsidy's value is **the security it preserves**, not any solvency it adds — and the sound shape is counter-cyclical to the naïve "prop up harder when low" design, which the analysis and simulation both show is a trap.

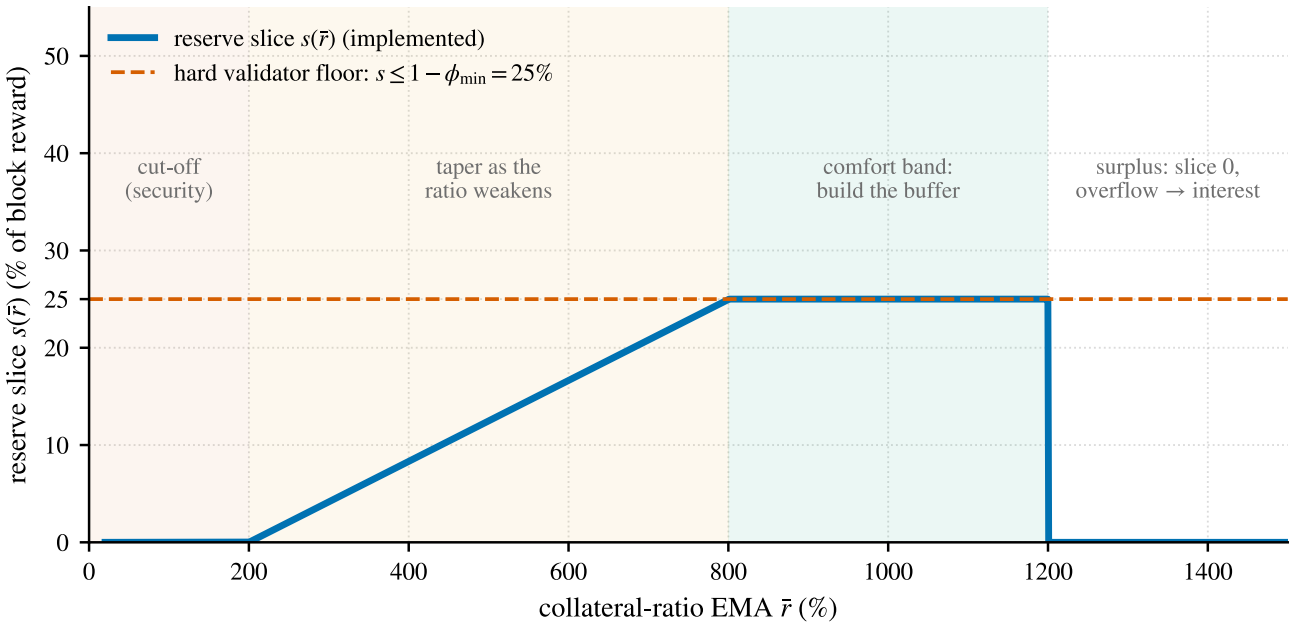

Fig. 6 — Variable slice control law

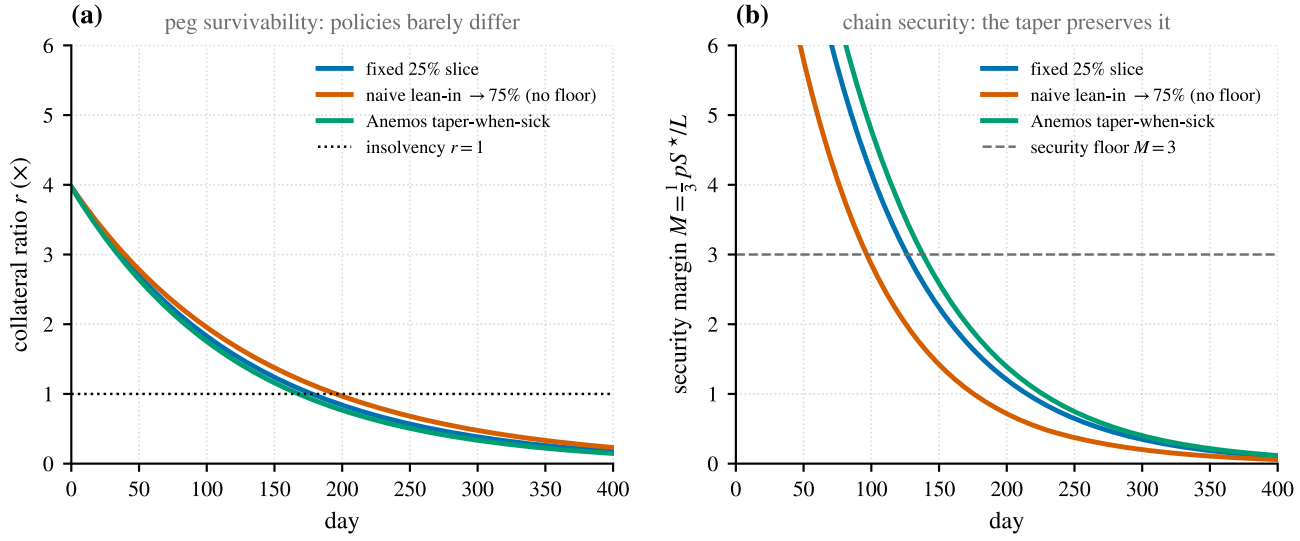

Fig. 7 — Variable subsidy scenarios

Above the target the slice is not 100%-to-validators — 25% is diverted to the senior capped, smoothed cash yield. The control law above sets $s = 0$ for $\bar{r} \ge 1200\%$, so the block reward would

otherwise go entirely to the proposer once the chain is comfortably over-collateralised. That above-target reward instead funds the **senior** golden-age yield of §4.1, §4.3: when the *instantaneous* ratio is above the 1200% target, the protocol **diverts a fixed 25% of the block reward** into the senior **interest pool**, leaving the proposer exactly the $\geq 75\%$ floor. The interest pool is drained to senior holders at a **steady, capped per-block rate** ($\leq 16\%$/yr, §4.3), and whatever the cap cannot pay this block simply **stays in the reserve as buffer** — which lifts the junior reserve coin's residual equity. The junior tranche earns **no cash yield**: its return is the residual-equity appreciation alone (§4.1). The diversion is **gated entirely above the target**: below 1200% there is no diversion at all, so validators keep their whole reward exactly when the chain is weakest — the same "build in good times, nothing in bad times" discipline as the reserve slice itself. The cost to security is small and lands where it is cheapest. At the 25% diversion the validator *budget* above target drops by 25% ($\varphi = 0.75$), but because $\mathsf{AttackCost} \propto \varphi^{0.5}$ is sublinear the **attack-cost give-up is only $\approx$ 13%**, and the hard $\phi_{\min} = 75\%$ floor is exactly where the above-target validator share sits — it is never breached. Crucially the diversion fires *only* above the target — i.e. exactly where the ANM price is highest, the safest point on the $\varphi^{0.5}p^{1.5}$ curve (the same asymmetry as Fig. 5): we divert where it is cheap (high price, deep buffer) and never where it is dangerous (sub-target, low price). **Fig. 10** shows the resulting block-reward split (validator + senior cash yield + reserve buffer $\rightarrow$ junior appreciation) across the four regimes of the control law, including the zero diversion below the target; **Fig. 11** shows that the realised senior APR is $\approx \mathsf{cap} \times f$ — the 16% headline is reached only at a near-full-year bull. The senior yield mechanism itself, its cap, and its honest limits are detailed in §4.1 and §4.3.

Honest failure modes. (1) Under a deep price collapse, $\mathsf{AttackCost} \propto p^{1.5} \rightarrow 0$ regardless of ϕ — no reward floor can secure a chain whose token has collapsed; the floor buys time and avoids *accelerating* the collapse, nothing more (the consensus analogue of Theorem 1). (2) Correlated stake-and-price flight can make the crisis elasticity exceed our calm-market $\varepsilon \approx 0.5$. (3) The cut-off is gated on the collateral-ratio TWA $\bar{r}$ as a proxy for the live USD security margin $\bar{M}$, rather than on M directly (which would use on-chain total stake); the two are correlated, and the $\bar{r}$-based curve already captures most of the benefit deterministically — a residual approximation, not a gap that affects determinism or safety.

4 The Native Stablecoin

4.1 Dual-tranche structure

Anemos uses the Djed/Shen pattern [8], echoing the over-collateralised CDP lineage of MakerDAO's Dai [9]. Two claims share one ANM reserve:

- **Senior — the stablecoin (peg token).** A claim on $1 of reserve value, paid first.
- **Junior — the reserve coin (equity).** First-loss capital; absorbs volatility and is worth $\max(0, \text{reserve value} - \text{liabilities})$ pro-rata, i.e. *worthless before the senior claim takes any loss.*

What the reserve coin returns: first-loss equity, residual-equity appreciation only — no cash yield. The junior reserve coin is **recapitalisation / first-loss capital** whose entire return is **residual-equity appreciation**: a reserve-coin deposit is tracked in the junior sub-bucket $\mathsf{RcReserve} \subseteq \mathsf{Reserve}$, and junior equity is the residual claim on over-collateralisation ($\mathsf{Equity} = \mathsf{Surplus} = \max(0, \mathsf{ReserveValue}\ (\mathsf{Reserve} - \mathsf{RecoveryFund}) - \mathsf{Supply})$ — the recovery-buyback earmark is excluded because recovery tokens rank senior to junior equity, §4.8). As the reserve appreciates — ANM price gains, the emission-funded reserve slice, *and* the above-target buffer that the senior cash-yield cap cannot absorb (§3.5) — the per-share junior equity rises. The junior tranche earns **no cash yield of its own**: there is no junior interest leg and no junior-favoured cross-fill. Every protocol cash-flow bypasses junior: ordinary mint/redeem fees are debited in native ANM and accrue to the **proposer/reserve** (a 50% slice is rebased to the **senior** index by the fee-rebase, §4.3); golden-age interest is a rebase of the **senior** index I and never touches the reserve coin (§4.3); and the entire above-target cash yield goes to **senior** (§3.5, §4.3). So the reserve coin is a pure leveraged long-ANM first-loss position, *subordinated* (it eats the senior shortfall before it is worth anything), whose upside is the residual surplus — including the buffer the senior cap leaves behind. **Fig. 8** plots the payoff: the reserve coin owns the *whole* surplus above the senior claim — leveraged upside on the ANM price, first-loss below — in contrast to a deposit-capped design that could never capture the emission-funded surplus.

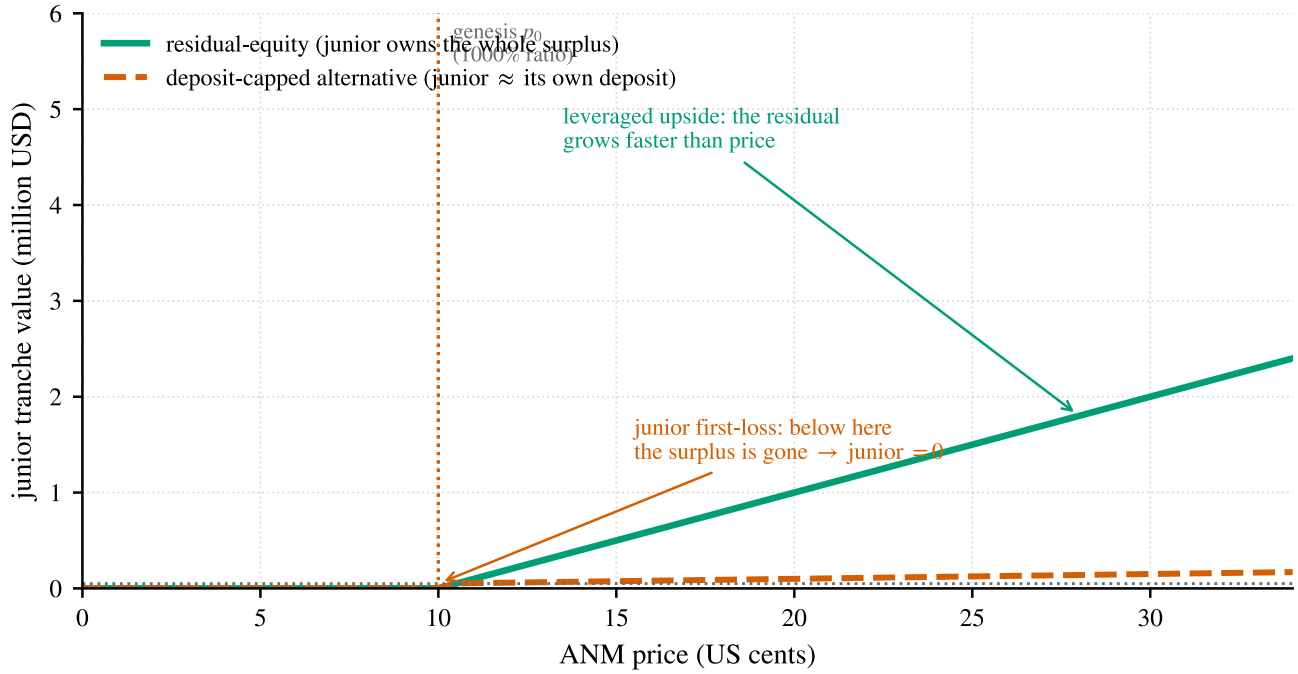

Fig. 8 — Junior residual-equity payoff vs the ANM price: the reserve coin owns the whole surplus above senior (leveraged upside, wiped first-loss below the coverage point), in contrast to a deposit-capped design that never captures the emission-funded surplus.

The honest incentive consequence for crisis recapitalisation. Because the only reward is ANM upside on the deposit — with *added* downside from the first-loss subordination — the rational time to buy reserve coins is when ANM is **cheap and the holder is bullish on ANM**, not specifically when the peg most needs recapitalising. In a drawdown the would-be recapitaliser is asked to inject capital that will first absorb the senior shortfall, for a return that is *only* the ANM rebound it could get by simply holding ANM directly. The protocol does **not** paper over this with a junior cash yield: paying first-loss *equity* a fixed *cash* coupon mis-prices the tranche and is unfundable on a thin base without starving senior. The recapitalisation incentive is therefore the honest one — residual-equity upside plus the above-target reserve buffer that accrues to junior — and the crisis-time recapitalisation premium remains structurally weak, which we state plainly rather than disguise. The reserve coin is a sound *structural* first-loss layer: it provably protects the senior peg, and a fresh minter can never capture the senior-funded buffer.

The junior-mint cap is decoupled from the interest target, so junior risk capital can thicken in a healthy chain. A junior reserve-coin deposit is **rejected once the post-deposit collateral ratio reaches the decoupled junior cap of 2000%** — **not** the 1200% senior emission/interest target, which is the threshold the *emission slice* and golden-age interest read. Because the junior cap sits well above the $\approx 1200\%$ ratio at which the per-block emission split (§3.5, §4.3) pins a mature system, **fresh junior first-loss capital can keep being added in a healthy chain** and can thicken to $\approx 150\%$ of the senior book in calm times, while existing junior holders can still redeem out (§4.2). Junior is thickest in young or stressed states (early after launch, or after a drawdown), but it is not turned away once the chain is merely healthy. The senior tranche is protected by *both* the emission-funded, protocol-owned reserve **and** a standing pool of junior risk capital that the decoupled cap lets accumulate above the interest target (**Fig. 9**).

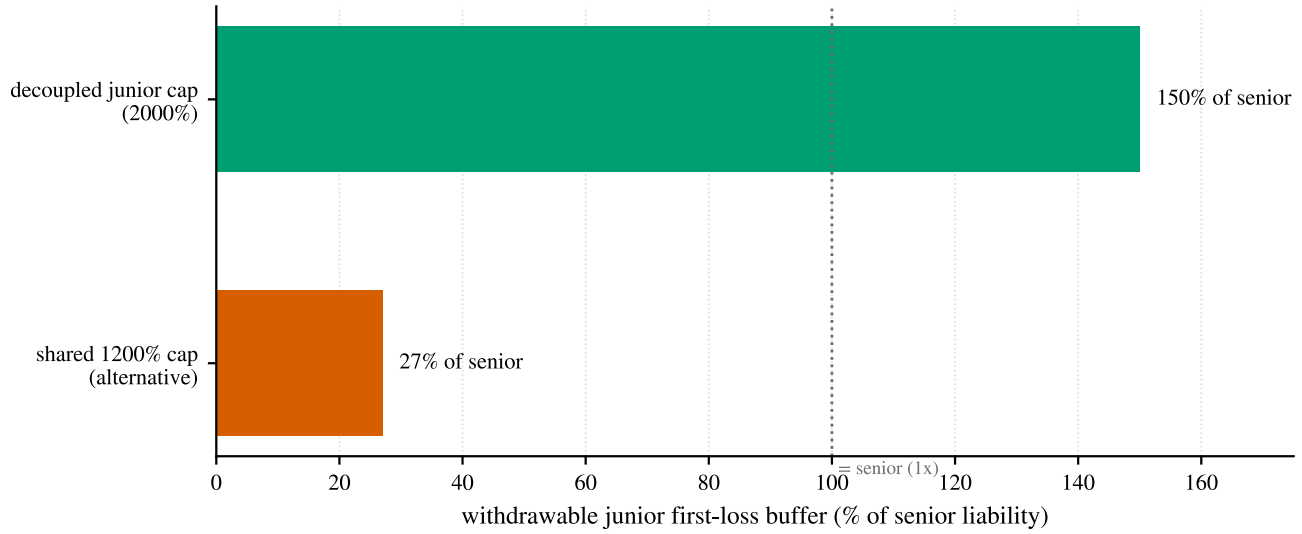

Fig. 9 — The decoupled junior-mint cap: under a shared 1200% cap the withdrawable junior first-loss buffer stalls near ~27% of the senior book; the decoupled 2000% junior cap lets it thicken to ~150% of senior while the emission/interest target stays 1200%.

The above-target diversion funds a *senior-only*, capped, smoothed cash yield. The control law sets $s = 0$ for $\bar{r} \geq 1200\%$, so above the target the block reward would otherwise go $\approx 100\%$ to validators. Above the target the protocol instead diverts a fixed **25% of the block reward** into the **senior interest pool** — the proposer keeps exactly the $\phi_{\min} = 75\%$ floor — and that pool is drained to senior holders at a **steady, capped per-block rate**. There is **no junior cash leg and no cross-fill**: the *whole* diverted budget targets senior, and whatever the senior cap cannot absorb this block **stays in the reserve as buffer**, which conservation-cleanly lifts junior residual equity (it does *not* revert to validators). The senior yield is therefore:

- **One cap, ≤ 16%/yr** (`InterestCapAnnualBps` = 1600). The annual cap is converted deterministically to a per-block USD budget ($\mathsf{supply} \times \mathrm{cap} \,/\big(10,000 \cdot B_{\mathrm{year}}\big)$), so summed over a year the realised senior interest can never annualise above 16% of senior supply — the headline number *is* the maximum annualised senior APR.
- **Smoothed (accrue-to-pool → steady drain).** Above-target revenue is **lumpy** (it arrives only during bull runs). Rather than pay the instantaneous overflow the block it lands — which would give a spiky, cohort-unfair rate — the protocol accrues it into a persisted **InterestPool** and pays it out at the steady capped per-block budget (`DrainInterestPool`). A holder's realised rate is thus a smooth function of *time over target*, not of which blocks happened to overflow. The senior interest is realised by raising the senior rebase index I (§4.3); junior is never paid cash.
- **Zero below target.** The diversion — and so all senior cash yield — is **0 below the 1200% target**, preserving security funding exactly when the chain is most attackable.

Why 16% is a ceiling, not a promise (the fairness / fundability result). The realised senior APR is approximately **cap** $\times f$, where f is the **fraction of the year the chain spends over the 1200% target**. The full 16% is reached **only at $f \to 1$** — a near-full-year sustained bull — and a typical few-months-over-target year delivers far less. This is the honest, self-limiting behaviour: the cap bounds the headline, the *revenue* (how long the chain is over target) sets the realised rate. The sizing/fairness simulation confirms both halves: (a) realised APR tracks $\mathrm{cap} \times f$ across caps $\{8, 16, 24\}\%$, so a 24% cap would still need $f \to 1$ to pay 24%; and (b) the **smoothed** drain pays a **steadier** monthly rate than an unsmoothed pay-the-overflow scheme (lower per-month payout-rate variance), i.e. it is fairer across cohorts and time. Validators give up 25% of the *above-target* reward, but because $\mathsf{AttackCost} \propto \varphi^{0.5}$ is sublinear *and* the diversion happens only where the ANM price is highest (the safest point on the $\varphi^{0.5}p^{1.5}$ curve, §3.5), the **attack-cost give-up is only ≈ 13%, and the $\phi_{\min} = 75\%$ validator floor is exactly where the above-target validator share sits** ($\varphi = 0.75$ at the 25% diversion) — it is never breached. A mint→redeem round-trip cannot farm the yield (it rebases *standing* senior shares over a block, so a \$100k one-block round-trip nets ≈ −\$500 after the 0.5% round-trip fees). This defense covers only the *same-block* round-trip; it does not by itself rule out a rational actor who mints ahead of a predictable above-target period and simply *holds* through it rather than round-tripping — a materially weaker strategy, since it commits real capital and directional risk for the whole window rather than farming a single block.

Fig. 10 shows the four-regime reward split — validator pool ($\geq 75\%$), the capped+smoothed senior cash yield, and the reserve buffer that lifts junior residual equity above target; the comfort-band validator-plus-reserve split; the taper zone where the slice ramps toward zero; and the 100%-validator deep-distress regime (no slice, no diversion, no cash yield). **Fig. 11** shows the senior APR fairness result: realised APR ≈ cap × f (panel a, the 16% headline only at a full-year bull) and the smoothed drain's lower payout lumpiness vs an unsmoothed pay-the-overflow scheme (panel b).

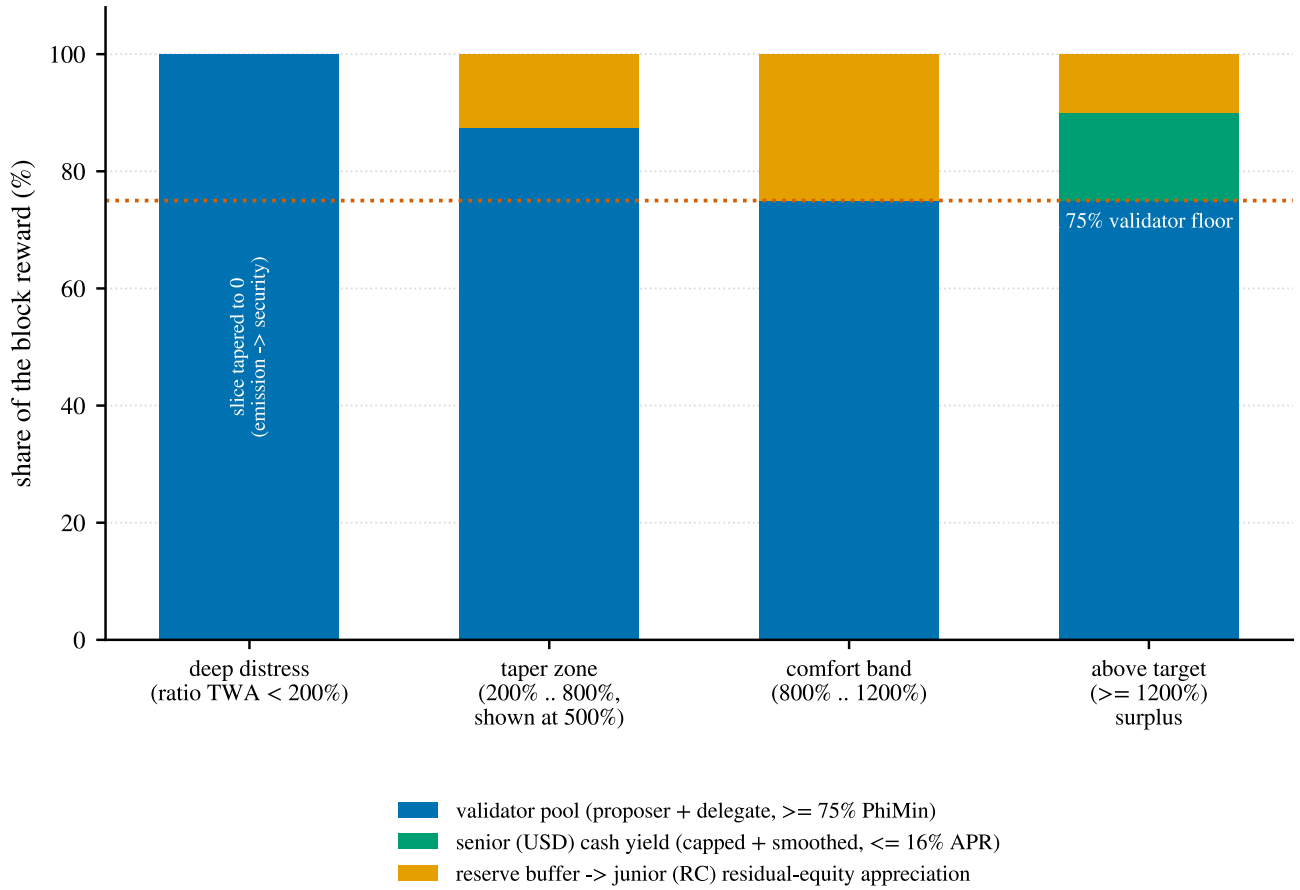

Fig. 10 — Reward-distribution waterfall across the four collateral-ratio regimes of the §3.5 control law: deep distress (ratio TWA < 200%) = 100% validator (slice tapered to zero); taper zone (200–800%, shown at its 500% midpoint) = the slice ramps 0→25%; comfort band (800–1200%) = validator 75% + 25% reserve slice; above target (≥ 1200%) = validator 75% + the 25% diversion split between the capped/smoothed senior cash yield and the reserve buffer → junior appreciation. The junior tranche earns no cash yield.

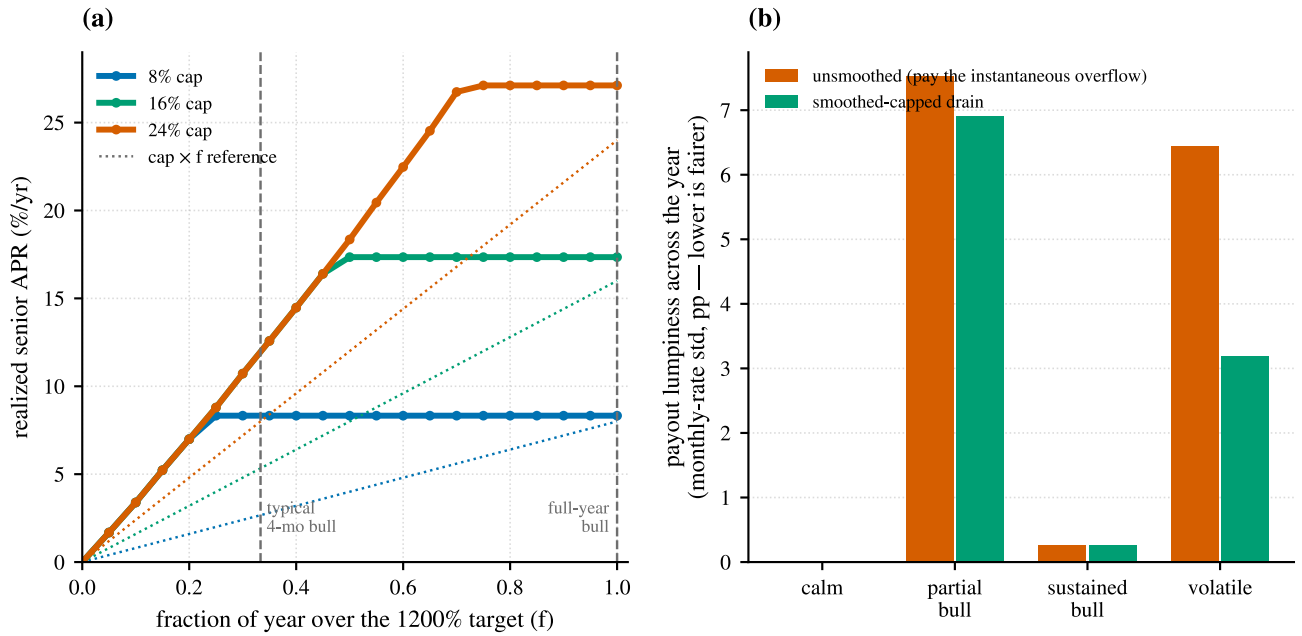

Fig. 11 — Senior APR fairness: (a) realised senior APR ≈ cap × f (fraction of year over the 1200% target), so the 16% headline is reached only at a near-full-year bull; (b) the smoothed-capped drain pays a steadier monthly rate (lower lumpiness) than paying the instantaneous overflow each block.

Let X be the reserve in ANM, p the ANM/USD oracle price (PriceScale fixed-point), and L the stablecoin liabilities in USD. The **collateral ratio** is

$$r = \frac{p \cdot X}{L} \quad (\text{in } 10^6 \text{ fixed-point}).$$

4.2 Mint, redeem, and the anti-death-spiral floor

Minting deposits ANM and credits stablecoin **only if the post-mint ratio stays above a floor** $r_{\min}$ (400%):

$$\text{mint allowed} \Leftrightarrow \frac{p\,(X + \Delta)}{L + \Delta_{\$}} \geq r_{\min}.$$

This minting-halt-below-floor rule is the formally-verified anti-death-spiral bound lifted from Djed [8]: it prevents the system from *originating* new liabilities into a thinly-collateralised state. Redemption is a senior claim and is always permitted, with a reserve-exhaustion haircut handled by the wind-down rule (§4.7). The junior tranche is recapitalisation capital. The **withdrawable** earmark is a separate sub-bucket RcReserve ⊆ Reserve: a junior redeem is hard-clamped to that earmark (the primary senior protection), so junior can never pull the senior book below the operating floor. Junior's *economic* equity, however, is the **full residual surplus** — residual-equity, Equity = Surplus = max(0, ReserveValue (Reserve − RecoveryFund) − Supply) — realised as reserve-coin share-price appreciation; the standing surplus *above* RcReserve is not held in RcReserve but accrues to the RC price (§4.1). The load-bearing invariant is that a fresh junior minter can never capture the senior-funded buffer: *mint-then-immediately-redeem at the same price returns at most the deposit.*

Scope of the junior first-loss. Junior redemption (reserve-coin redeem) is gated only at the 400% operating floor — junior may withdraw its equity freely while the ratio stays above it. The first-loss guarantee is therefore *conditional on how the decline arrives.* Against a **discontinuous gap-down** — an oracle move that drops the ratio below the floor in one step, with no intervening block at which junior could have redeemed — junior is trapped and absorbs the loss ahead of the senior tranche, exactly as marketed. Against a **gradual decline**, however, an *informed* junior holder can rationally redeem and exit while the

ratio is still between the 1200% target and the 400% floor, pre-empting the first-loss it was meant to bear; only the junior capital still in the system at the moment of breach is subordinated. (The price-TWA-smoothed oracle of §5.4 helps here: it damps per-block price moves, which both makes a true instantaneous gap-down rarer *and* widens the window in which an attentive junior could exit — these pull in opposite directions for the buffer's reliability.) So the buffer should be read as a **guaranteed** first-loss against a gap-down and a **best-effort, exit-eroded** first-loss against a slow decline — not an unconditional cushion.

Senior-side mint-lag: the price-TWA min-price guard. The committed oracle price is the slow ~ 3.5 h price TWA of §5.4, **not** the spot ANM price. That lag is exactly what makes single-block oracle manipulation worthless (§5.4–5.5), but on the *minting* side it would otherwise cut the other way during a genuine fast ANM decline: if ANM dropped sharply over a few hours the price TWA reads above spot for several hours, and valuing a deposit at the stale-high TWA would mint senior stablecoin worth more than the collateral contributed, draining real reserve value to a fresh senior claim. Anemos closes this with a **mint-guard price**: the mint path values the deposited collateral at $\mathsf{MintGuardPrice} = \min(\text{committed price TWA}, \mathsf{LastMedianPrice})$ — the fresher $\pm 10\%$-clamped median the oracle accepted this block, before the slow TWA — and the 400% floor check $\frac{p\,(X+\Delta)}{L+\Delta_{\$}} \geq r_{\min}$ is evaluated at that **same lower** price ($p = \mathsf{MintGuardPrice}$). During a fast decline the mint is therefore charged the lower, fresher median on both sides, so it cannot mint senior USD against the stale-high price TWA — the over-valuation leak is driven to $\approx \$0$, and the floor binds on the fresher price exactly when spot is falling. In a calm market the two prices coincide and the guard is inert (no honest-mint false positives). The **residual** is bounded: because the fresh reference is itself $\pm 10\%$-clamped per block (ClampMove, §5.4), a price gap larger than 10% *within a single block* still leaks within that per-block bound — but not the full ~ 3.5 h of price-TWA staleness the unguarded path would have exposed. This is the senior-side mirror of the junior-exit caveat above (junior loses first-loss capital to *exits* on the way down; senior is protected from over-valued *mints* on the way down by the guard). **Fig. 12** shows both halves: the averaging-window trade-off that motivates the ~ 3.5 h operating point, and the guard removing the crash-time bleed without touching the committed peg.

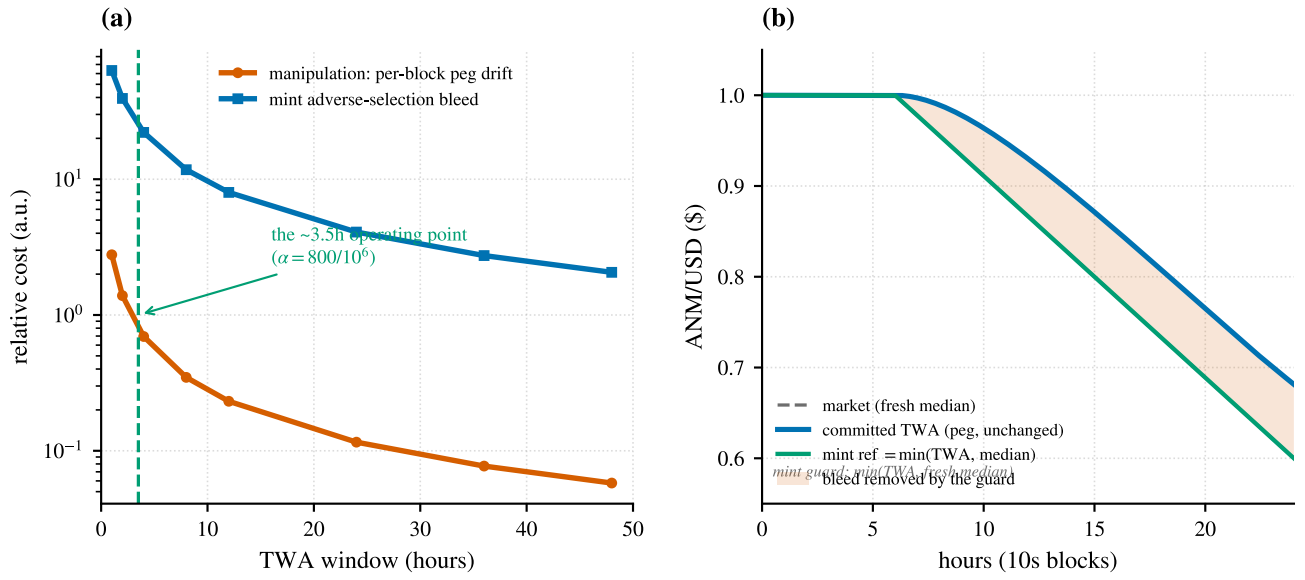

Fig. 12 — The price-TWA window trade-off and the mint-guard price: (a) a longer averaging window crushes per-block manipulation but increases the adverse-selection bleed on a genuine move — the ~3.5 h operating point balances the two; (b) during a fast decline the committed TWA lags high while the fresh clamped median tracks the market, and pricing mints at min(TWA, median) removes the bleed without touching the committed peg.

4.3 Golden-age interest via an O(1) rebase index

Stablecoin balances are stored as **shares**; a single global monotone index $I \geq I_0$ converts shares to displayed balance:

$$\text{balance} = \lfloor\, \text{shares} \cdot I/I_0 \,\rfloor, \qquad \text{mint:}\;\; \text{shares} = \lfloor\, \text{amount} \cdot I_0/I \,\rfloor\,.$$

The product shares $\cdot\, I$ is evaluated in **arbitrary-precision integers** before the division by I_0, and the result is narrowed to int64 by **deterministic saturation** (clamp to $2^{63}-1$), never a wrapping cast — so even a whale balance under a long golden-age index growth can never trigger an overflow panic that would halt the chain; interest distribution additionally *refuses* (no-op) any bump that would push I past the int64 range. Accruing interest is a **single index bump** — $O(1)$, no per-holder iteration (the property that keeps the module Android-viable). Interest is the **marginal overflow of the per-block emission slice above the 1200% target** (the *instantaneous* reserve ratio at the committed price, not the lagging ratio TWA): the block-subsidy reserve slice tops the reserve up to **exactly** $r^\star = 1200\%$, and only the part of that slice which would push the ratio *above* 1200% is diverted to holders. Concretely, with liabilities L and price p, the reserve target is $X^\star = \lfloor\, r^\star L\,\mathsf{PriceScale} \,/\, (\mathsf{RatioScale}\, p) \,\rfloor$ ANM (the fixed-point form of $r^\star L/p$, with $r^\star$ a RatioScale value and p a PriceScale value), the slice splits as $\Delta_{\text{res}} = \min(s, \max(0, X^\star - X))$ and $\Delta_{\text{int}} = s - \Delta_{\text{res}}$, and the USD value of Δ_{int} is the interest budget for the period. **Below 1200% the entire slice tops up the reserve and interest is exactly zero** — the 800% comfort-band edge plays no part in the interest decision (it only shapes the emission-slice taper of §3.5). The system must also be solvent. This is what makes the yield sustainable (the anti-Anchor property [10]) and **self-throttling**: distributing the overflow raises liabilities, lowers r back toward the target, and the slice once again lands wholly in the reserve, stopping the payments (**Fig. 13**).

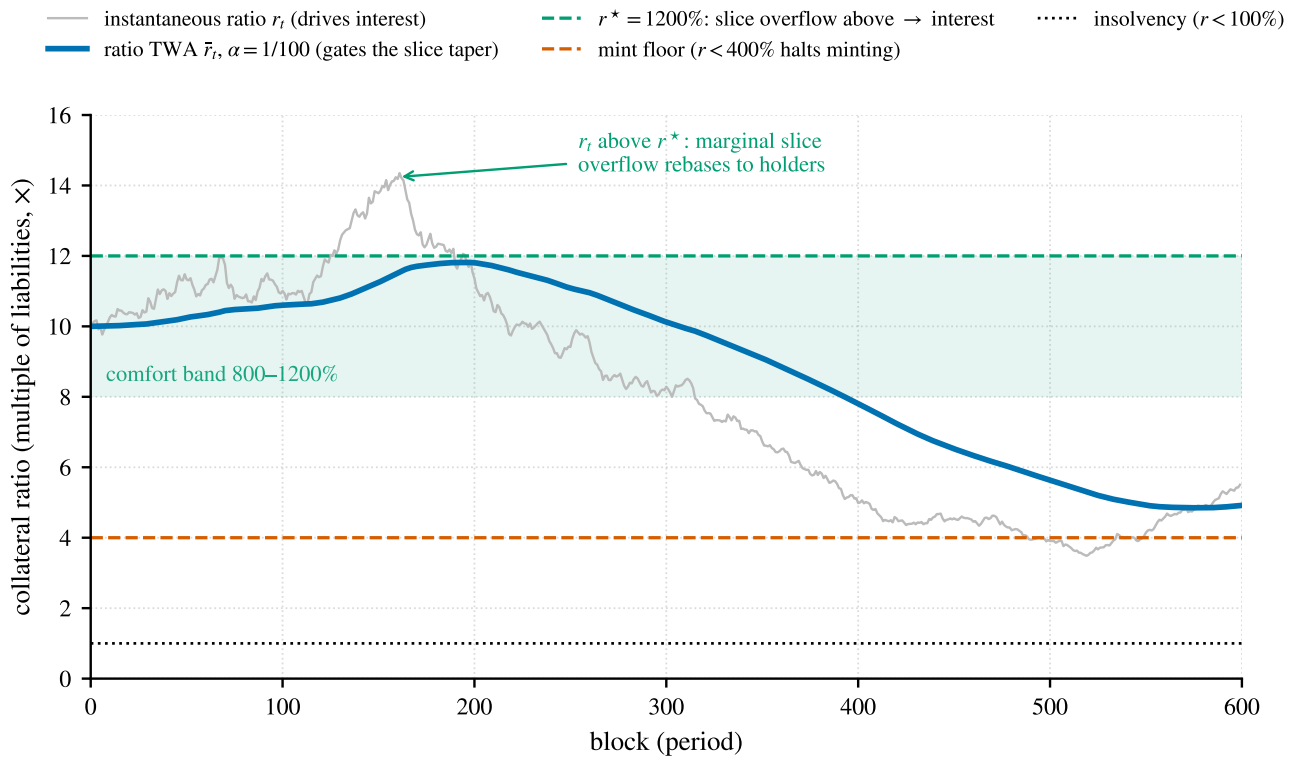

Fig. 13 — The collateral-ratio TWA and comfort band

Golden-age interest is a *transient*, not a standing APR. Two gates must hold simultaneously for a payment: the *instantaneous* ratio must be above 1200% (so the slice has overflow to divert), **and** the slow ratio TWA $\bar{r}$ must still read *below* 1200% (so the emission slice s is non-zero at all — the slice returns zero once $\bar{r} \geq 1200\%$, §3.5). Interest therefore flows only in the narrow window where spot has crossed

the target but the lagging ratio TWA has not yet caught up. In that overlap window the two policies run *simultaneously and in tension*: the slow ratio TWA still reads inside the comfort band, so the control law authorizes the full 25% buffer-building slice, while the fast *instantaneous* ratio already sits above the target, so the marginal overflow of that same slice is rebased to holders as yield — the protocol is both topping up the reserve and paying it out in the same block. This is not a leak (conservation still holds: the whole slice lands in the reserve and interest is only a USD rebase of standing reserve value, §3.4), but it is a genuine *timescale mismatch* between the slow $\sim$ 100-block ($\alpha = 1/100$, $\approx$ 17 min) ratio-TWA gate on the slice and the per-block instantaneous gate on interest, and it is exactly why this *golden-age overflow* yield is transient rather than steady. Under *sustained* over-collateralization the ratio TWA converges above 1200%, the slice shuts off, the overflow vanishes, and **this overflow interest decays to zero**. So the 16% figure is a ceiling on the senior yield, not a guarantee, and this particular *overflow* path is a *transitional* contribution paid while the reserve buffer tops up *through* the target on a lagging ratio TWA — it is **not**, by itself, a durable standing rate; the durable standing component comes from the above-target diversion below.

The above-target diversion pays a *standing*, smoothed senior cash yield. The "decays to zero" property scopes to the golden-age marginal-overflow path described here. Separately, the above-target diversion (§3.5, §4.1) feeds the **senior** interest pool through the **same** rebase index I (and the **same** single per-block budget) precisely when the instantaneous ratio sits above the 1200% target — so a sustainably over-collateralised system does **not** pay senior holders nothing: it pays a standing, capped ($\leq$ 16%/yr) above-target cash yield via the same Index. Both senior-Index paths (overflow and above-target diversion) share **one** per-block budget (no double-pay), so the combined senior interest never exceeds the 16%/yr cap. Crucially, the above-target revenue is first **accrued into a persisted InterestPool and then drained at the steady capped per-block rate** (§4.1), so the realised senior rate is a smooth function of *time over target* ($\approx \mathsf{cap} \times f$) rather than of which blocks happened to overflow — fairer across cohorts and across time (**Fig. 11**). The junior reserve coin receives **no** cash interest by either path; its return is residual-equity appreciation only (§4.1).

Golden-age interest is senior-favouring, not junior-neutral. The self-throttling framing above is correct for the *senior* tranche, but it is not neutral to the junior reserve coin. Interest raises the rebase index I, which raises the senior liability $\mathsf{Supply} = L$; the junior tranche's claimable equity is the residual $\mathsf{Equity} = \mathsf{Surplus} = \max(0, \mathsf{ReserveValue}\ (\mathsf{Reserve} - \mathsf{RecoveryFund}) - L)$. The surplus subtraction excludes the recovery earmark but **not** the junior-contributed $\mathsf{RcReserve}$, so each unit of senior interest paid raises L and **shrinks Surplus — and with it the junior equity — by the same amount**. In other words, senior interest transfers over-collateralisation buffer **to senior holders at the expense of the junior tranche's mark-to-market equity** (conservation still holds — the slice's ANM never leaves the reserve, §3.4; this is a *valuation* shift between tranches, not a leak). This is the honest cost of routing the *entire* above-target yield to senior: junior is paid **only** through whatever the senior cap leaves in the reserve buffer plus its own residual-equity appreciation. The "self-throttling" property should be read as throttling the *senior* payout, not as leaving the junior claim untouched.

The interest cap is an ANNUAL ceiling on the senior APR. A "period" in this module is **one block ($\sim$ 10 s)**, so a cap written "per period" is silently a per-*block* cap. The parameter is therefore expressed **per year**: $\kappa_{\mathrm{yr}} = 1600$ bps (= 16%, the maximum annualized senior APR, `InterestCapAnnualBps`), converted to the per-block budget $\kappa_{\mathrm{blk}} \cdot L = \left\lfloor \kappa_{\mathrm{yr}}\, L / (10{,}000 \cdot N_{\mathrm{yr}}) \right\rfloor$ where $N_{\mathrm{yr}} = 3{,}153{,}600$ blocks/yr, paid in discrete ticks. Because a period is one block ($\sim$ 10 s), the cap is expressed per year (κ_{yr}) and divided by N_{yr} into a per-block budget, so the value 1600 bps is exactly the maximum annualized senior APR — and, because the InterestPool is drained at most one such budget per block, the *combined* senior interest can never annualise above 16% of senior supply. The 16% ceiling is a generous *headline* that is only ever realised at a near-full-year sustained bull ($f \to 1$, Fig. 11); a typical few-months-over-target year delivers far less ($\approx \mathsf{cap} \times f$). It sits well clear of Anchor's fatal $\sim$ 20% APY, and a hard validation bound caps any genesis/governance value at 2000 bps (20%), so even a captured governance cannot exceed the cap. The cap is only a *ceiling*:

because the yield is self-throttling and funded solely from the per-block reserve slice's overflow plus the 25% above-target diversion (a few % of a $\sim$ 2 ANM subsidy), the realized rate is almost always far below 16% — the cap is a defence-in-depth backstop, not the binding throttle. A stablecoin mint can never push the ratio above the 1200% target (it raises the senior liability, *lowering* the ratio) — but the target is not reachable by the block subsidy alone: a junior reserve-coin deposit is capped at the *decoupled* 2000% junior cap (§4.1), not at 1200%, and a senior redemption shrinks liabilities, so junior deposits and senior redemptions can also carry the instantaneous ratio above the target and trigger the overflow/diversion mechanics — all of them conservation-clean paths into the same single capped interest budget.

The fee-rebase adds a small, durable senior floor yield. The golden-age overflow is a transient, and the above-target diversion pays only while the chain sits over target. Separately, **half of every mint/redeem fee's USD value is rebased to the senior index** through the same InterestPool and the same single per-block budget. The fee ANM has already entered the reserve when the rebase occurs, so the rebase is conservation-neutral — it raises the senior liability against backing already present — and it is gated off while the system is insolvent (a rebase must never deepen a shortfall). Scaling with turnover rather than with the ratio, it contributes a durable $\sim$ 0.2–0.8%/yr floor under the senior APR (**Fig. 14**).

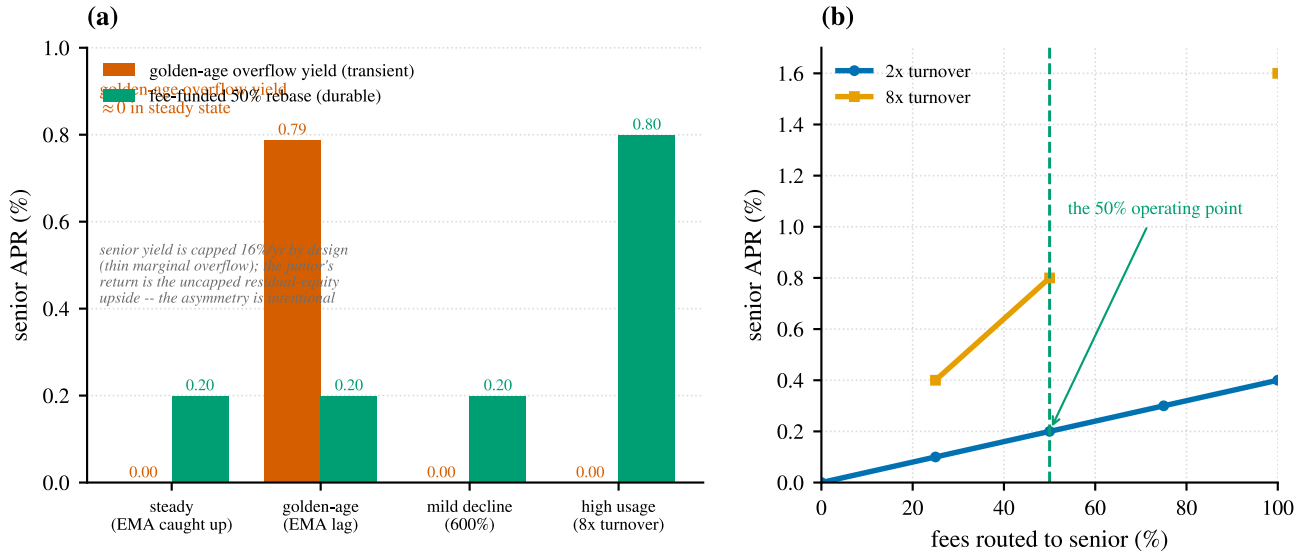

Fig. 14 — Senior yield composition: (a) the golden-age overflow is ≈0 in every durable steady state — the fee-funded 50% rebase is the durable ~0.2–0.8%/yr component, scaling with turnover; (b) the fee-share sweep at 2× and 8× turnover, with the 50% operating point marked.

4.4 The collateral-ratio TWA

The instantaneous ratio is noisy and grindable; gates use a slow time-weighted average (the **ratio TWA**), an $O(1)$ integer recurrence

$$\bar{r}_t = \bar{r}_{t-1} + \alpha \left(r_t - \bar{r}_{t-1} \right), \qquad \alpha = \frac{1}{100},$$

computed in integer fixed-point. Moving the *trigger* by manipulation requires holding an extreme ratio for $\sim 1/\alpha$ consecutive blocks — which itself means depositing real collateral, i.e. doing exactly what the interest is meant to reward.

4.5 Circuit breaker

A per-period cap bounds the **net** (mint − redeem) flow, limiting how fast a mania or a run can move the system and capping the volume an oracle-bias attacker can misprice. With period-flow accumulators (M, Rd) and supply L:

$$\text{AllowMint}\ (\Delta) \Leftrightarrow M + \Delta - Rd \leq \max\big(\ \lfloor c\,L \rfloor, c_{\min}\ \big),$$

where c is the per-period cap fraction (5%) and $c_{\min}$ an absolute floor that lets a young system bootstrap (otherwise $L = 0 \Rightarrow \text{cap} = 0$ bricks the first mint). The breaker is **net**, not gross, so self-cancelling churn cannot exhaust it, and it bounds an oracle-bias attacker's per-period profit by $S \leq c\,L\,\varepsilon_{\max}$, where $\varepsilon_{\max}$ is the maximum mispricing the oracle permits within tolerance.

Two breakers: a per-block spike limiter and a rolling-window run brake. Anemos has **both** limiters. (a) The **per-block** breaker above is a single-block spike limiter: it caps the net flow in any one block at $\max(5\%\,L,\, c_{\min})$, where $c_{\min} = \$100,000$. (b) A **rolling-window** breaker (a decaying accumulator) caps the *net redeem* over a trailing 8,640-block ($\sim$ 1-day) window at $\max(20\%\ \text{of supply}, \$100,000)$. It is an $O(1)$ decaying accumulator — each redeem adds its value, each mint subtracts (floored at zero, so it is a net cap), and the commit hook decays it once per block ($w -= \lfloor w/\ \mathsf{windowBlocks} \rfloor$) — with no per-block scan. So a determined run that the per-block breaker alone would let drain the book in ~ 20 blocks is throttled to $\sim 20\%$ of supply *per day*, i.e. stretched to **~days**, while honest $\leq$cap/day flow never trips it (**Fig. 15**). A shared cap is also, unavoidably, a shared griefing surface: an attacker can "pin" the redemption window by placing a large legitimate redemption first, crowding out later honest redeemers until the window decays. This costs the attacker real capital (the redemption itself), so it is not a free denial-of-service, and its worst case is a delay, not a loss of funds — but it is an acknowledged, low-severity griefing vector rather than a fully closed one.

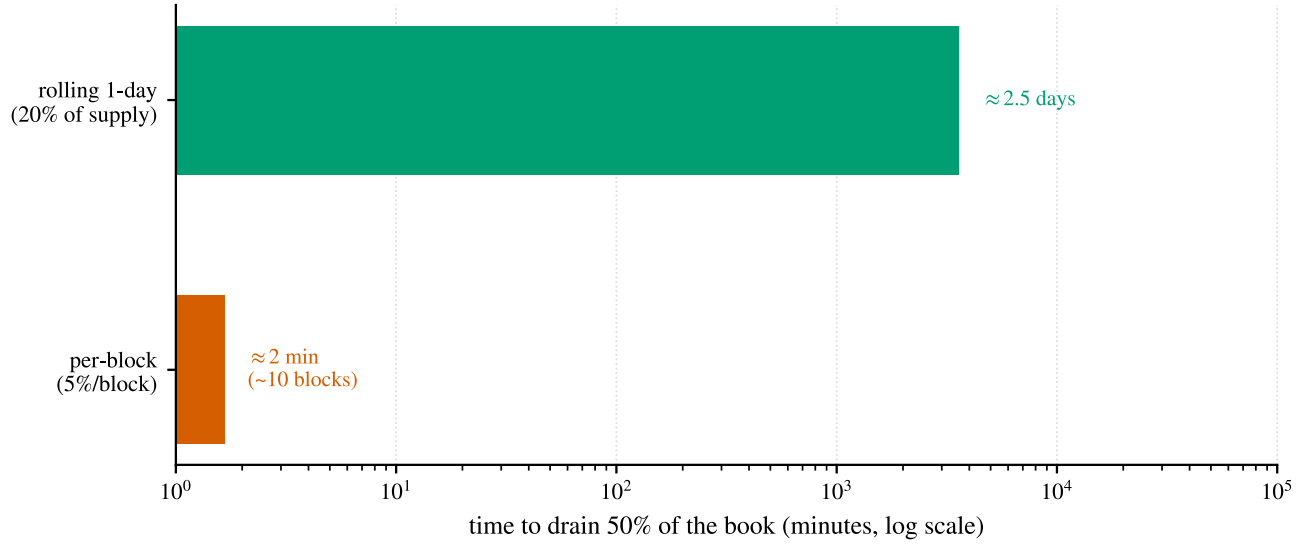

Fig. 15 — Time to drain half the senior book: the per-block breaker alone is a spike limiter (a determined run clears 50% in minutes), while the rolling 1-day 20%-of-supply window stretches the same run to days.

The per-block floor $c_{\min}$ dominates below a $\sim$ \$2M book. For the per-block breaker the cap is the *larger* of $c_{\min}$ and 5% of the senior book L; the two are equal at $L \approx \$2,000,000$. Below that crossover the $\$100,000$ floor binds and the per-block fractional limit is inert (a flat $\$100,000$/block, $\sim$ \$864M/day, far larger than a young book) — a deliberate bootstrap accommodation so the floor clears a usefully large first mint. This static $c_{\min}$ floor is therefore a real early-life gap in the *per-block* breaker; the rolling-window breaker's own $\$100,000$ floor is commensurate (it is never tighter than the per-block breaker for a young/low-supply system). The load-bearing early defences are the rolling-window brake, the slow price-TWA oracle, and the wind-down ρ.

Period scope. The per-block accumulators reset every block (period $\equiv$ block ≈ 10 s), so c and the per-block profit bound $c\,L\,\varepsilon_{\max}$ are **per-block**: if only the per-block breaker existed, net redemption could still drain the book in $1/c = 20$ blocks (~ 3.5 min). Under the rolling-window cap, however, the actual bound is $\sim 20\%$ of supply per day, so a determined run is stretched to **~days**, not minutes. The breakers compose

with (i) the slow price-TWA oracle, which caps any sustained mispricing an attacker could mint/redeem against (§5.2), and (ii) the pro-rata wind-down ρ of §4.7, which removes the first-mover advantage *within* a settlement period regardless of how many blocks a run spans.

4.6 Reflexivity and the impossibility theorem

The reserve is denominated in ANM, whose USD value tracks ANM's market price; emission-funding mints *more of the same volatile asset*. Ratio floors **dampen** but cannot **remove** this feedback.

> **Theorem 1 (no unconditional solvency).** *Consider any reserve X (ANM) backing liabilities L (USD) at price p, with $r_0 = pX/L$. If the price follows a path $p_t = p_0\, e^{Y_t}$ with $Y_t \to -\infty$ (a sustained collapse), then for **every** finite starting ratio r_0 there exists a time τ with $r_\tau = r_0\, e^{Y_\tau} < 1$ — i.e. insolvency. No minting-halt or band rule prevents this, because they constrain new liabilities, not the value of the existing reserve.*

Proof. Minting halts and interest gates only ever *reduce* L or hold X; they cannot increase pX. Since $r_\tau = r_0 e^{Y_\tau}$ and $Y_\tau \to -\infty$, r_τ crosses 1 in finite time for any r_0. ■

The proof treats X as held constant, but in reality the emission reserve slice and slash forfeits genuinely *add* to X over time (§3.5, §5.3) — the theorem survives this only because §3.5's separate quantitative futility argument shows that inflow rate is bounded and vanishingly small relative to a real collapse's price-decay rate, so it cannot outrun $Y_t \to -\infty$; the constant-X proof above is the clean asymptotic statement, and §3.5 is what closes the gap for the realistic case where X is slowly growing.

Theorem 1 is the honest refutation of any "cannot die" claim: a stablecoin backed solely by its own volatile coin **cannot** be made unconditionally solvent. Anemos's response is not denial but **graceful failure** — restructuring for a moderate shortfall (§4.8) and an orderly pro-rata wind-down for the catastrophic tail (§4.7) — with quantified rarity (§7).

4.7 Orderly wind-down

When the system is insolvent ($r < 1$) and the shortfall is too deep to restructure (§4.8), redemption switches from first-come-full-payout (a bank-run race) to a **pro-rata settlement**: every redeemer in the period receives the same fraction

$$\rho = \min(1, r) = \min(1, pX/L)$$

of their claim's value, burning their full shares while paying only ρ of the value. Here p in terminal mode is the price-TWA min-price guard min(committed price, LastMedianPrice), so terminal recovery recognises a crash from the fresher reference rather than the lagging committed price. Because $\rho \le r$, the aggregate payout never exceeds the reserve, and — crucially — the *last* redeemer receives the same ρ as the *first* (**Fig. 16**). This removes the first-mover advantage that turns a depeg into a run.

This equality holds **at a fixed price**, so ρ (in ANM terms) is preserved for the holders who remain. There is a residual limit, though: ρ is recomputed each block from the *then-current* oracle price, and the per-block circuit breaker (§4.5) forces a large run to settle over many blocks. If the price keeps falling *across* those blocks (the very Theorem-1 collapse that triggered terminal mode), an early redeemer settling at price p_t receives a higher ρ than a late one settling at $p_{t+k} < p_t$. So a *residual cross-block first-mover advantage survives*, bounded by the per-block price decay × the run length; the slow price TWA damps the per-block price move that drives it. Within a single block, ρ is *approximately* equal across redeemers rather than exactly identical: the executor recomputes ρ per-transaction against the live, sequentially-mutating reserve and supply, so at a fixed price a later same-block redeemer sees a very slightly higher ρ than an earlier one (a late-mover advantage, the mirror image of the cross-block first-mover effect above). The residual intra-block variance is bounded by the per-block circuit breaker's flow cap — a genuine but low-severity

design property, not the unbounded cross-block price-decay drift the paragraph above describes. Entry to the terminal regime is **instant** on the first insolvent period (debouncing entry would leave the run window open); exit is **debounced** over k consecutive solvent periods. A solvent-but-thin book — ratio below the 110% **pre-terminal guard** — is flagged in the committed mode as *pre-terminal*: a soft, observable warning classification (surfaced to clients and monitoring) that gates nothing by itself; minting is still governed by the §4.2 floor and terminal entry by insolvency alone. The recovery factor ρ is a *separate multiplier*, never a downward mutation of I, so the load-bearing invariant $I \geq I_0$ — where I_0 is the unit floor (the rebase index's identity value 1.0; the index can never rebase below 1.0) — survives both entry and the reversible exit. The unpaid difference simply *remains in the reserve* — it is never lost or double-credited, so ρ conserves ANM exactly. **Minting is halted throughout the terminal regime** — not by an explicit mode flag but by the §4.2 collateral floor: while insolvent ($r < 1$) every prospective mint leaves the post-mint ratio below the floor and is rejected, so no new senior claims can be issued into a failing system (Theorem 1).

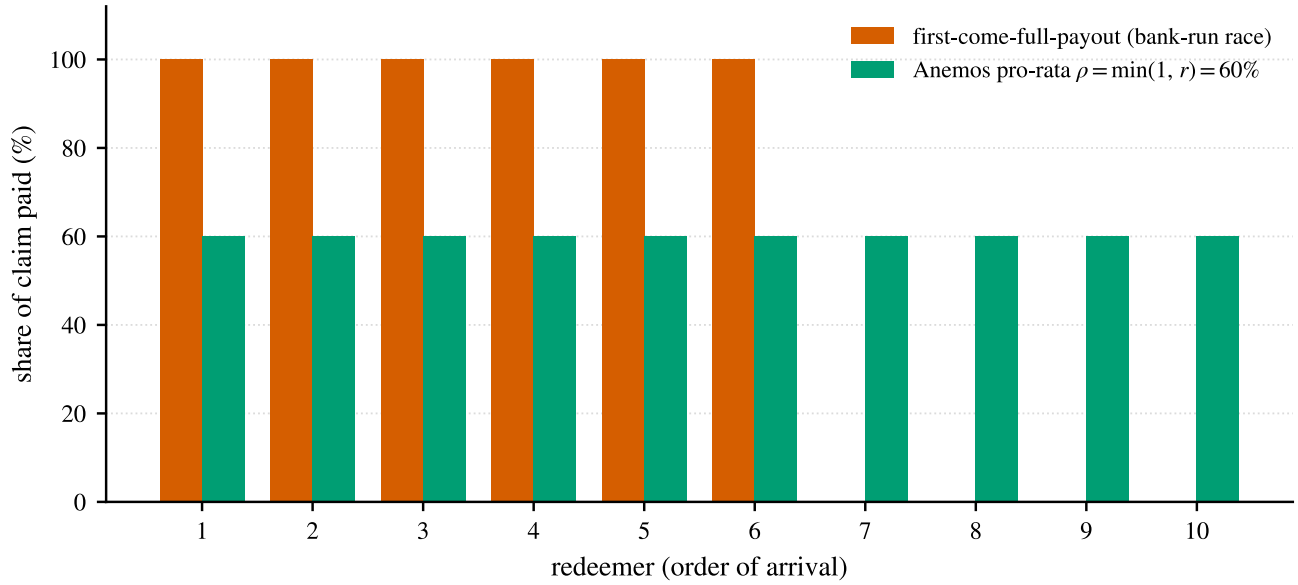

Fig. 16 — Wind-down fairness

4.8 Restructuring with recovery tokens

The wind-down of §4.7 is *terminal*: it settles holders pro-rata and closes the senior tranche. Theorem 1 says some failure state is unavoidable for *any* native-backed peg, so a graceful terminal mode is the right floor — but it need not be the *only* response to a mild shortfall. Anemos adds an intermediate **restructuring mode** that converts an insolvency event into a *deferred* claim rather than an immediate haircut.

The mechanism. Suppose collateral falls to a ratio $r = 0.8$ (< 1 but well above zero). Instead of paying every holder $\rho = 0.8$ and burning the rest, the protocol writes the senior face *down* to the backed value (restoring $r = 1$) and issues the shortfall $1 - r$ as a **recovery token**. The writedown is a single cumulative multiplier $W \leq 1$ applied to the rebase index — the *effective senior index* is $\mathsf{seniorIdx} = \lfloor I \cdot W / \mathsf{RatioScale} \rfloor$, so every senior balance shrinks uniformly in $O(1)$ while the up-only interest index I (and its overflow guarantees) stays intact. A holder of 100 stablecoins is left with senior tokens worth 80 plus 20 of recovery face. Recovery tokens are *deferred senior claims*: they carry no peg of their own, but they are first in line to be **bought back at par** from future protocol cash-flows — the perpetual emission tail (§3), mint/redeem fees, and reserve growth — before the junior reserve coin earns anything again. Unlike the senior stablecoin and the junior reserve coin, the recovery token is **not** peer-to-peer transferable — it is redeemable only via the dedicated recovery-redemption path, not sent directly between holders — since its claim is inseparable from the per-holder issuance checkpoint above. The chain stays *open for business* (mint/redeem continue

against the restored $r = 1$ book) instead of latching terminal. A minter who arrives *after* the writedown buys senior at the written-down index and is granted no recovery: issuance is settled against a per-holder checkpoint of a global *issuance accumulator* **before** any share change, so only the holders who held senior *through* the crisis receive recovery (no windfall to post-crisis entrants).

The hard case: overlapping restructurings. The sharpest question is *what if a second shortfall hits before the first has finished repaying?* Tracking per-vintage claims would be $O(\text{vintages})$ state and a fairness minefield. The clean answer is a **single fungible recovery pool**: one global outstanding face F (NanoUSD) and one earmarked buyback fund Φ (Gust, carved from the reserve). Every recovery token — from *every* restructuring, past and future — is a claim on the *same* (F, Φ) pool; a new restructuring simply adds its shortfall to F. Redemption pays the fund's current **coverage ratio** $\min\big(1, \mathsf{value}\ (\Phi)/F\big)$ pro-rata, capped at par, burning only the par-equivalent actually paid (the uncovered remainder stays redeemable as Φ grows — no forfeit). Because every token redeems against the same coverage at any instant, all vintages are **pari passu** (equal-rank) in $O(1)$ — exactly as a sovereign collective-action clause or an FDIC receivership pools claimants of the same seniority — with no per-event bookkeeping and no ordering to game.

Where it sits in the waterfall. The result is a strict three-layer settlement order, enforced at *redemption time and at the current price* (not merely at buyback time):

$$\underbrace{\text{senior stablecoin (peg)}}_{\text{paid first; can claw } \Phi} \succ \underbrace{\text{recovery tokens } (F, \Phi)}_{\text{deferred senior, pari passu}} \succ \underbrace{\text{junior reserve coin}}_{\text{absorbs first loss, wiped before recovery pays}}.$$

Concretely: a fresh senior shortfall dissolves the recovery earmark back into senior backing ($\Phi \leftarrow 0$, the ANM never leaving the reserve) before writing senior down further; recovery redemption is gated to the genuine surplus *above full senior backing at the current price* (with a one-period oracle-move discount, §5.2, so a proposer who knows the pending price cannot extract reserve the senior tranche needs); and junior equity redemption is excluded from Φ entirely. Buyback fills Φ only from surplus that keeps senior at or above a buffer ratio, and the restructuring lifecycle **terminates** — returning to normal mode — once Φ fully covers F (every recovery token redeemable at par on demand). **Fig. 17** draws the full capital stack: the loss-absorption order on one side, the payout priority on the other, with the operating-floor gate that stops junior redemption from ever pulling senior below the floor.

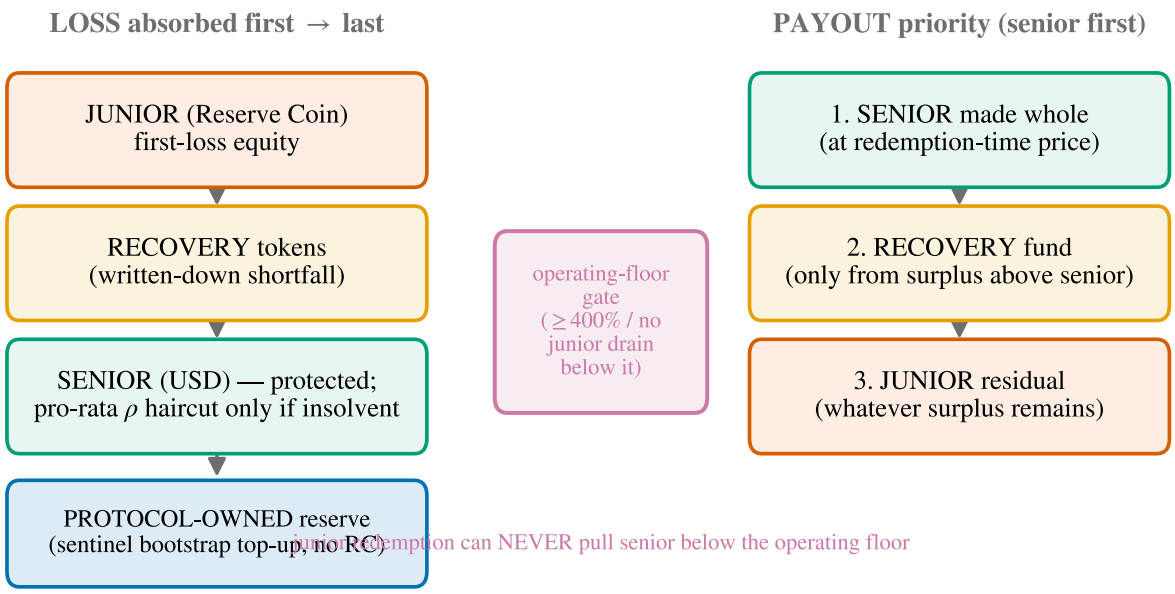

Fig. 17 — The loss and payout waterfall: junior absorbs first loss, recovery tokens are deferred senior claims bought back from surplus, senior is paid first; the operating-floor gate stops junior redemption from ever pulling senior below the floor, and the protocol-owned (sentinel) bootstrap reserve sits beneath the stack.

Honest framing. This does **not** repeal Theorem 1: if ANM's value collapses *sustained*, even recovery tokens cannot be made whole and the system still tends to the terminal wind-down — a cumulative writedown past the floor $W < W_{\min}$ (a catastrophic single-period collapse) falls through to §4.7's pro-rata settlement. What restructuring buys is *optionality on time*: it converts a sudden, racing insolvency into an orderly, fungible, buy-back-over-time obligation, and the protocol's **own perpetual emission is the credible buyback source** that most fiat-backed designs lack. The precedent base is deep — contingent-convertible (CoCo) bonds, FDIC receivership, and sovereign collective-action clauses all convert "default now" into "deferred, equal-rank claim." It is a strict improvement over a hard terminal latch for *mild* shortfalls. All recovery accounting is integer-exact with overflow-guarded saturating accumulators, so it preserves the consensus-determinism contract of §6.

Simulation (Fig. 18). A scripted moderate crash (a 38% price drop on a 130% book — statically $0.62 \times 1.3 \approx 0.81$, cushioned in the simulation by the emission-tail inflow accrued before the trough) pushes the senior tranche below par: the protocol writes the senior face down by $\approx 9\%$ cumulatively, issues that shortfall as recovery face, and stays open. As the price recovers and the emission tail accrues, buyback fills the recovery fund; coverage climbs $0 \to 100\%$ over roughly three months, after which the lifecycle returns to normal mode — the deferred claim is repaid in full at par, never having forced a terminal wind-down.

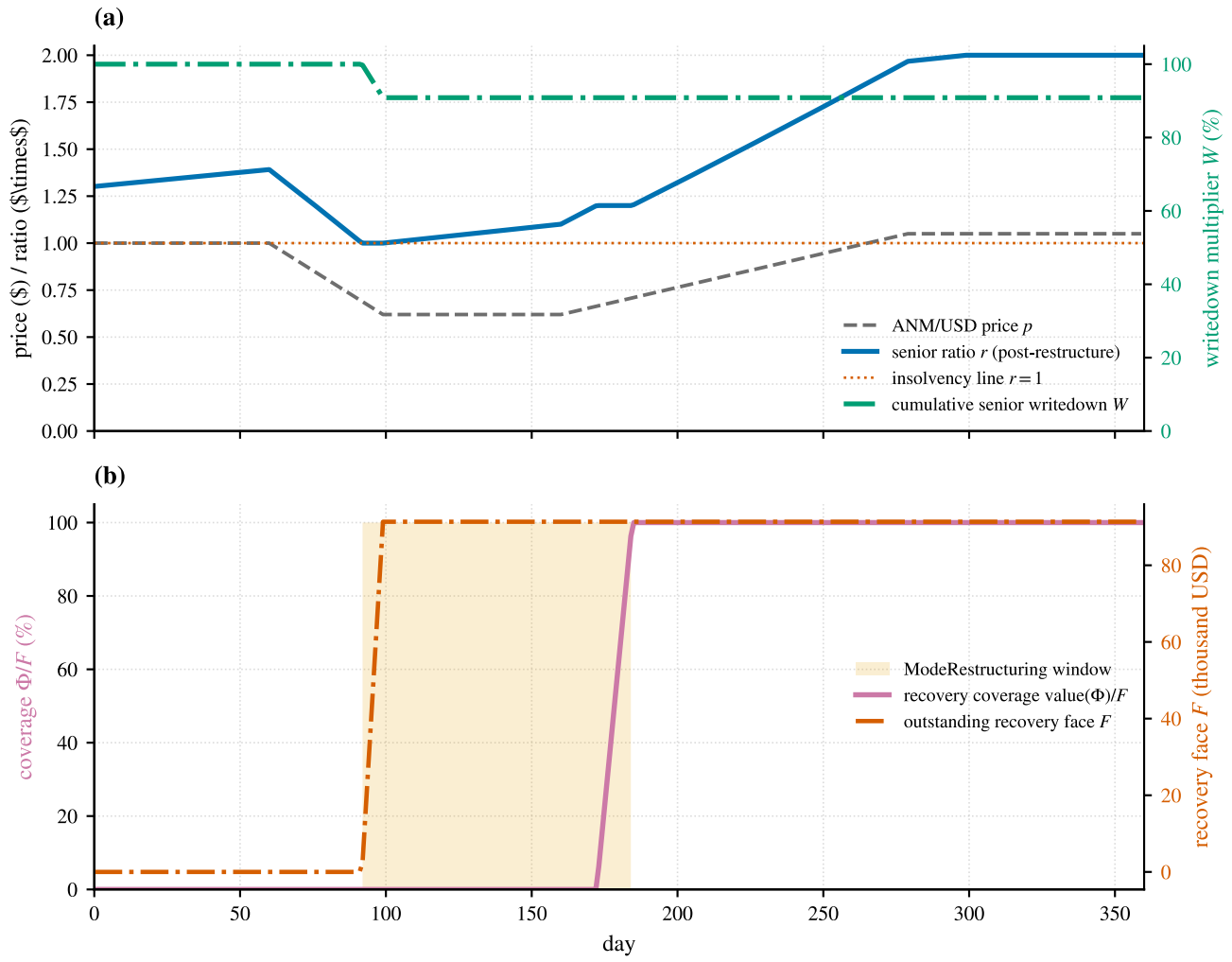

Fig. 18 — Restructuring lifecycle

4.9 Launch: the two-step reserve bootstrap

On a fresh network the stablecoin faces a chicken-and-egg deadlock. There is no reserve yet, so the **first** stablecoin mint is impossible: a full-value mint lands the post-mint collateral ratio near 100%, far below the 400% floor of §4.2, and minting halts. The obvious fix — fund the reserve first by minting reserve coin (junior equity) — is blocked as well: while there is **no senior supply at all**, the system is by definition infinitely over-collateralised, so a normal-phase equity mint is rejected outright (there is no senior tranche to protect and no residual to price the equity against), and once senior supply exists the **junior equity-mint cap** of §4.1 rejects equity injection once the ratio reaches the 2000% junior cap: an over-capitalized system needs no more buffer. The guardrails that make the steady state safe thus jam the launch.

Anemos resolves this with an **explicit, capped, two-phase launch** governed by one genesis parameter T, the **bootstrap reserve target** (**mainnet 600,000 ANM, testnet 75,000 ANM**), and one **latching** state bit, the bootstrap-done flag. Let the protocol be *bootstrapping* iff $T > 0$ and the bootstrap-done flag is unset.

- **Phase 1 — fund the reserve (minting halted).** While bootstrapping, a reserve-coin deposit is a **protocol-reserve topup**: it **bypasses the anti-hype cap entirely** so anyone may fund the reserve toward T regardless of ratio or supply, while stablecoin minting is **halted** (rejected at the transaction check *and* on block re-execution, so it cannot fire through any path). The deposited ANM is added to the reserve, and the matching reserve-coin shares are minted to a **non-redeemable bootstrap sentinel**

(the all-zeros Treasury address, with no key) — **not** to the depositor and **not** into the junior reserve-coin earmark, so the bootstrapper receives no redeemable equity. The shares are seeded **1 : 1 with ANM** for the whole of Phase 1: because stablecoin minting is halted throughout bootstrapping, there is **no senior liability** (Supply = 0) to price the equity against, so the entire reserve is protocol-owned and the sentinel's share count Q **mirrors the total ANM deposited** regardless of any standing reserve — including ANM that arrives *without* a matching mint, e.g. an oracle-deviation **slash forfeit** routed into the reserve while bootstrapping (the consensus oracle runs from genesis). Pricing off the standing surplus in that window would mint *fewer* than the deposited ANM of shares and compound per topup, ratcheting the per-reserve-coin value above 1 : 1; anchoring on the raw ANM basis while there is no senior claim keeps the reserve coin opening at **exactly 1 : 1 with ANM at bootstrap-done**. This makes the seeded surplus protocol-owned and closes the first-minter surplus-capture: once Phase 2 begins and a genuine senior liability exists, a later genuine junior is priced pro-rata against the standing equity rather than inheriting it. Every overflow/saturation check still rejects rather than wraps — no value is created from nothing.

- **The latch.** Whenever the committed reserve first reaches $\geq T$ the protocol sets the bootstrap-done flag. The latch is re-checked in **two deterministic places**, both pure functions of (committed reserve, genesis target) and therefore identical on every node: after a reserve-coin deposit, **and** in the per-block emission commit step — because the per-block emission slice also grows the reserve, so a chain can cross T via emission alone, with no reserve-coin deposit. Both checks are the same monotone pure function, so the property is simply *latched deterministically wherever the committed reserve first reaches T*. The latch is **monotone**: once set it never reverts, even if the reserve later falls below T through redemptions or a haircut.
- **Phase 2 — normal operation.** With the bootstrap-done flag set the steady-state rules of §4.1–§4.2 resume unchanged: reserve-coin minting is governed by the anti-hype cap again, and stablecoin minting is enabled subject to the 400% floor. The bypass cannot leak into this phase — it is gated solely on the bootstrapping predicate, which is permanently false once latched.

Setting $T = 0$ **disables** the bootstrap phase: the system behaves as already-bootstrapped from genesis (stablecoin minting enabled, reserve-coin minting capped). The bootstrap-done bit is part of the serialized stablecoin state and therefore of the block state root, so the phase transition is itself a consensus fact every node agrees on. This is a bounded, auditable mechanism: the reserve is funded to a known target before the peg is ever offered, exactly the "bootstrap value before the peg can mean anything" concern raised in §1.2.

The target is a fixed ANM quantity, so its real adequacy is a launch-price assumption. Unlike the emission tail and the stake cap, which are **supply-indexed** (§3.2, §5.11), the bootstrap target T is a **fixed nominal quantity of ANM** baked into the genesis module parameters: 600,000 ANM on mainnet, 75,000 ANM on testnet. It is *not* supply-indexed and *not* USD-indexed. Consequently the **USD adequacy of the launch buffer depends entirely on the ANM price at launch**: 600,000 ANM is a meaningful reserve only to the extent ANM is worth something when the peg is first offered — the same "ANM is worth ~nothing at genesis" reflexivity of §1.2/§4.6 that no nominal target can escape. The 600,000 figure is a *chosen nominal*, not the output of a USD-backing calculation, and operators should treat the moment they choose to latch the bootstrap — i.e. the price at which 600,000 ANM is judged a sufficient buffer — as a launch-timing decision, not a protocol guarantee (the monotone latch never re-gates, as stated above).

5 The Oracle

5.0 Activation height: the oracle is disabled until a fresh coin has a price

A price feed can only report a price that *exists*. At genesis a brand-new coin trades on no market, so signing, aggregating, or slashing against an ANM/USD "price" would be signing against a meaningless number — and a stale/undefined price at launch can mis-drive the reserve (e.g. deviation-slash forfeits crediting the reserve on a price that never meant anything). Anemos therefore gates the entire oracle behind a single genesis parameter, the **oracle activation height** H_0. For every block height $h < H_0$ the oracle is **disabled**: the proposer emits no `OracleData` section, a block that injects one is rejected as invalid, the commit-time price aggregation and deviation slashing are skipped, and the committed price stays 0. At $h \geq H_0$ the oracle turns on and runs exactly as described in the rest of §5. The gate is keyed on the trusted block height (never a transaction locktime) and applied identically at propose, validate, and commit, so every node agrees on the activation boundary deterministically; the price section's presence byte in the block hash already makes an absent section well-defined, so a pre-activation block hashes and is certificate-attested with no change to the certificate format.

Because the committed price is 0 during the pre-activation window, minting stablecoin is halted there (the mint path fails closed on a non-positive price, §4.2) — a priceless coin cannot be pegged to a dollar it has no exchange rate for. The **reserve can still be built**, however: the two-step reserve bootstrap (§4.9) funds the reserve at the \$1.00/ANM opening-price fallback while the price is undefined, so a fresh chain accumulates over-collateralization during its warm-up and switches minting on cleanly once the oracle activates and the committee first reaches price quorum. Setting $H_0 = 0$ makes the oracle **active from genesis** (the localnet/devnet default). Networks that launch a genuinely priceless coin set $H_0 > 0$: the public testnet uses 500 blocks (≈ 1.4 h), and mainnet's value (100,000 blocks, ≈ 11.5 days) is an indicative default — the operative mainnet value is a launch-time governance decision made once a credible ANM/USD market and a live feeder committee actually exist.

5.1 Consensus-embedded oracle: a rotating committee subset, attested by the block certificate

A USD peg needs a USD price, and a new chain has no native source. Anemos delivers price as **consensus-embedded block data**, not as standalone transactions and not as a consensus vote, following the committee/median data-feed tradition of SchellingCoin [11] and the deviation-reported aggregation of decentralized oracle networks [12]. It keeps the riskiest code (vote/certificate path) untouched, reuses the existing block-certificate authentication, and stores the oracle history compactly (§5.10). The mechanism is a per-block pipeline:

1. **Deterministic per-block subset.** A fixed subset size (**11**) of committee members is selected each block by ranking the committee on $\mathrm{Hash}(\mathrm{seed} \,\|\, \mathrm{valNum})$ and taking the lowest-ranked members. The seed is the **committed $H-1$ sortition seed** (a VRF-evolved BLS value already in the header) — fixed before block H is proposed, so the proposer cannot bias the draw, and every node re-derives the identical subset with **zero stored data**. Selection is uniform (not stake-weighted).
2. **Selected members sign a price.** A selected member's off-chain feeder (§5.9) fetches ANM/USD and signs the canonical, domain-separated message $\mathrm{H}(\mathrm{height} \,\|\, \mathrm{valNum} \,\|\, \mathrm{price})$ with its validator BLS key, producing a gossiped **price vote**. Members *not* in the subset stay silent.
3. **The proposer aggregates — as a dumb collector.** The block proposer collects the subset's valid price votes from its oracle pool and folds them into a new block-body **oracle section** = $\{\mathrm{entries}\ [(\mathrm{valNum}, \mathrm{price})], \mathrm{aggregateSig}\}$, where the single 48-byte BLS aggregate signature is the sum of the members' individual signatures over their distinct messages. Below the subset quorum the proposer includes **no section**.
4. **The existing certificate attests it.** The oracle section's hash is folded into the block hash unconditionally, and the committee's aggregated precommit **certificate already signs that block hash** —

so $\geq 2/3$ of the committee committing the block also commits the oracle data, **with zero certificate-format change**.

5. **Commit-time re-verification (every node, deterministic).** On commit each node independently re-derives the subset from the same $H-1$ seed, checks every signer is in the subset, and enforces the subset quorum — the **primary** accept rule is the $2\not\geq 3$-of-**power** supermajority ($2\lfloor(\text{subsetPower}-1)/3\rfloor+1$, equal to the precommit certificate's power threshold), and the head-count $\lfloor 2\,|\text{subset}|/3\rfloor+1$ is a **secondary** sample-size floor; it verifies the aggregate against the signers' committee public keys over each member's $\mathrm{H}(\text{height}\,\|\,\text{valNum}\,\|\,\text{price})$. It then feeds the attested prices through the **same power-weighted** median $\rightarrow \pm 10\%$ clamp $\rightarrow$ price-TWA pipeline (§5.2, §5.4) and **slashes** any signed price outside the deviation tolerance (§5.3).

The proposer is a **dumb collector, never a trusted aggregator**: it cannot **forge** or **misattribute** — a lying, non-subset, or forged section is an invalid block every other node rejects. It *can*, however, choose **which** $\geq q$ subset of the votes it holds to include: commit-time re-verification checks only that the included signers are in the subset, that there are $\geq q$ of them, and that the aggregate verifies — it does not, and cannot, require *all* available votes, and omission is never slashed (§5.3). This **omit-to-skew** is real but **bounded, and does not lower the trust ceiling**: the selected median can only land *within the range of the genuine signed votes* (no fabrication), the $\pm 10\%$ clamp plus the slow price TWA caps the committed move at $\alpha\,\delta \approx 0.008\%$ that block (§5.2, §5.4) regardless of which $\geq q$ subset is chosen, and biasing the price TWA in a sustained direction requires proposing a **sustained majority of blocks** in that direction — i.e. a sustained majority of stake-weighted sortition proposers, which is precisely the **honest-majority-of-stake** ceiling the oracle already rests on (§5.5). This bound is exercised by a skew-maximizing subset choice that holds the per-block committed move within $\alpha\,\delta$ for both the high-skew and low-skew selection. Because the accepted price is a **pure function of the in-block signed prices and committed state**, a node that was offline and re-syncs derives the *identical* price, with no attestation-replay ordering surface. There is no oracle transaction at all: the price rides in the block body, not in a tx. The five stablecoin operations (mint/redeem stable, mint/redeem reserve, redeem recovery) are carried by a **single conversion transaction** whose payload bears an operation enum {mint stable, redeem stable, mint reserve, redeem reserve, redeem recovery} — keeping the protocol's small, fixed set of transaction types while preserving each operation's exact on-chain semantics. Partial unbonding is an optional amount on the unbond transaction, and the price rides in the block body rather than as a transaction. The taxonomy is **11 contiguous types, 1–11, with no reserved gaps**: transfer (1, multi-asset) · bond (2, also the operator pool registration/self-bond) · sortition (3) · unbond (4) · withdraw (5) · batch-transfer (6) · convert (7) · coinbase (8) · delegate (9) · undelegate (10) · claim-reward (11). The pooled-delegation delegate/undelegate/claim-reward types (9–11) are account-signed deposits, withdrawals, and reward claims against an operator pool (§5.11). The batch-transfer is a user-facing signed send-to-many that debits the sender (a multi-recipient transfer, conservation-safe) — it can **never mint**. The block-subsidy mint is isolated in its own dedicated **coinbase** type — the chain's only minting path — which has no real sender (its signer is the keyless Treasury sentinel) and which mints with no sender debit. The coinbase type is internal-only: the mempool rejects any coinbase transaction, block validation accepts it only as the first transaction with the exact emission amount, and the Treasury sentinel holds no key, so a forged mint can enter neither the mempool nor a committed block. Every other type, the batch-transfer included, always has a real, balance-checked, debited sender — defense-in-depth that puts coin creation behind a single protocol-only type.

5.2 The power-weighted median and the manipulation threshold

At each block the module aggregates the subset's attested prices into a price using a deterministic **power-weighted median** (each attestation weighted by its validator's stake). For an even count N it is the **integer floored average** of the two central order statistics, computed overflow-safely as

$$\text{med} = A_{\lfloor N/2\rfloor} + \left\lfloor \left(A_{\lfloor N/2\rfloor+1} - A_{\lfloor N/2\rfloor}\right)/2 \right\rfloor = \left\lfloor \left(A_{\lfloor N/2\rfloor} + A_{\lfloor N/2\rfloor+1}\right)/2 \right\rfloor,$$

a *symmetric* tie-rule. The sort uses a **value-based, stable** comparator — never an index/address-dependent tie-break — so the sorted sequence, and therefore the median, is byte-identical on every node and compiler even when several validators submit the exact same price; the slashing-deviator list is likewise returned sorted by validator number, so the committed slashing set is order-independent. The key security fact:

> **Moving a single block's median requires controlling a strict *majority* of that block's subset's power.** The aggregation is the **power-weighted** median, so the precise requirement is a strict majority of subset *power*; the head-count statement "≥ 6 of 11" is the uniform-power case (the max-stake cap keeps any one validator below that power threshold). With an 11-member subset, a uniform-power attacker needs $> |\text{subset}|/2$ (≥ 6 of 11) of *that block's* selected members. A 1/3-of-committee adversary lands a subset majority only occasionally, and even then the move is bounded (below) and the next block draws a fresh, independent subset. The integrated, sustained ceiling is *honest-majority-of-committee* (§5.5), the same family as consensus itself.

Quorum is the subset quorum, computed from the subset size, not a constant. A new price is accepted in a block only when the subset reaches **both** quorums: the **primary** $\geq 2/3$-of-**power** supermajority of the subset (equal to the precommit certificate's power threshold — the actual security threshold), **and** the secondary head-count floor $q = \lfloor 2\,|\text{subset}|/3 \rfloor + 1$ (the sample-size floor shared by propose, validate, and commit). The basis is the **subset** size (capped at the live committee size), not the full committee, because the price comes from the ~11-member subset. This holds at every committee size, and an underfull or seat-flooded committee cannot reach the subset quorum from outside the subset. Below quorum the proposer includes **no oracle section**; at commit the **last price is carried forward** (never trust a thin sample) and a staleness counter advances (§5.4). A median is naturally robust to missing data points, which is what lets us never punish absence (§5.3).

Per-period move cap (defeats a *transient* committee majority). Even an attacker who briefly holds a submitting majority cannot jump the peg: the accepted price is clamped to move at most $\delta = 10\%$ per period. A single period's median, however manipulated, can pull the accepted price by no more than δ; an attacker would have to *hold* the majority for $\Theta(\log_{1+\delta}(\text{target} \,/\, \text{spot}))$ consecutive periods to drag the peg to a target, by which point deviation slashing (§5.3), committee rotation (§5.6), and a halted/wound-down module have all engaged. The clamp composes with the price-TWA lag (§5.4): transient control buys a bounded, slow, *and* self-punishing move — never an instantaneous one. On genuine bootstrap, when there is no prior price, the first quorum-backed median is adopted directly, so the clamp never strands the system at zero. This same one-period discount gates recovery-token redemption (§4.8) so a proposer's foreknowledge of the next price cannot extract reserve.

5.3 Omission vs. commission — the rule that protects honest nodes

- **Absence (omission) is *never* slashed.** A validator offline for minutes simply didn't submit; it pays only a soft availability-score dip with grace windows. This is what makes a flaky or Android node safe to run, and it is consistent with the BFT liveness assumption (silence $\neq$ fault).
- **Deviation (commission) *is* slashed, and the slash is *magnitude-fair* and *escalates* for repeat offenders.** Submitting a price outside the tolerance band $\left[\text{med}\,(1-\theta), \text{med}\,(1+\theta)\right]$ ($\theta = 10\%$) is penalised at commit. The penalty is **graduated on BOTH the repeat count and the MAGNITUDE** of the lie. How far outside the band a price falls ($m = |p - \text{med}|/\,\text{med}$) picks a tier — **jitter** ($\leq 20\%$), **suspicious** ($\leq 50\%$), **egregious** ($> 50\%$) — and the per-period fraction is base $\times\, 2^{\min(c,6)+\,\text{shift [tier]}}$ with base $= 1\,\text{bp}$, shift $= \{0, 3, 6\}$ ($\times 1/ \times 8/ \times 64$) and c the validator's decayed escalation level. A **GRACE** rule makes a *first small* deviation (jitter, $c = 0$) cost **zero** — a flaky feed never bleeds stake — while still recording the count so a *repeat* escalates. An honest small/sustained deviation tracks the

gentle base ladder (the $1 \to 64$ bp doubling); a wild or repeated lie is $8\times$ harsher (egregious one-off = 64 bp, egregious repeated rising toward the egregious ceiling). The fraction is **clamped to per-tier ceilings** (0.64%/6%/25%) and a **hard 25% per-period cap** — sized so a *proven* egregious case routes commensurate forfeit into the reserve while an honest single bad feed can never be zeroed. An **egregious** deviation also advances the stored count by 2, so the ban threshold (6) is reached in 3 egregious deviations instead of 6 — a validator signing wildly wrong prices is removed roughly twice as fast. The whole curve is integer shifts + floored mul-div + min clamps, deterministic, protocol-favouring; a degenerate parameterisation (equal ceilings, zero shifts, no grace) collapses it to a single flat count-only ladder. The escalation level **decays** on a lazy integer half-life: the effective count read at height H is $\mathsf{Count} \gg \lfloor (H - \mathsf{LastDeviationHeight}) / \mathsf{DecayPeriods} \rfloor$ (1000 periods $\approx$ 2.8 h at 10 s blocks), so after sustained clean behaviour a one-time offender decays back to the first-offence base. The decay is computed at read time — a pure integer, $O(1)$, no per-block sweep, no float, no wall-clock — so every node (including a cold-reloaded or re-synced one) computes the identical effective level. The validator's committee power drops by the slashed amount, so sustained manipulation bleeds stake at an accelerating rate on top of being out-voted by the median. The forfeited ANM is *moved into the reserve* it endangered — a conservation-preserving transfer, not a burn. A split-committee guard suppresses slashing when deviators are $\geq$ half the attesters, so a 50/50 split cannot be weaponised to slash the honest half.

What the slash does *not* deter — the coordinated within-tolerance bias. Deviation is measured against **this period's power-weighted median** of the subset's attestations, *not* against an external truth or the lagging price TWA. So a cartel that controls a subset majority and clusters its prices **within the ±10% tolerance band of the median it is itself setting** moves the median *with* its own votes, and **no attester ends up outside tolerance — so nobody is slashed at all.** Even when an attacker *is* caught as a power-minority outlier, the per-period penalty is bounded by the per-tier ceilings (up to the hard 25% cap). The deviation slash is therefore the deterrent for an **uncoordinated minority** feeding outliers into an honest median — it is genuinely effective there — but it is **not** the defence against a *coordinated stake majority* biasing the price within tolerance. For that case the sole protection is the **honest-majority-of-stake assumption enforced by the slow price-TWA lag** (§5.4): a within-tolerance bias still moves the committed peg by at most $\alpha\,\delta$ per block, so dragging it any meaningful distance requires *sustaining* a stake majority across thousands of blocks — exactly the ceiling stated in §5.5. The slash does not deter a coordinated majority; the price TWA does. **Fig. 19** makes this calibration explicit: the binding cost of a price-moving majority is acquiring the stake itself (the split-guard-exempt majority pays no slash), which dwarfs the clamp/TWA/breaker-bounded extractable value; the slash curve's job is the uncoordinated-minority case, lenient on honest jitter and far harsher on egregious lies.

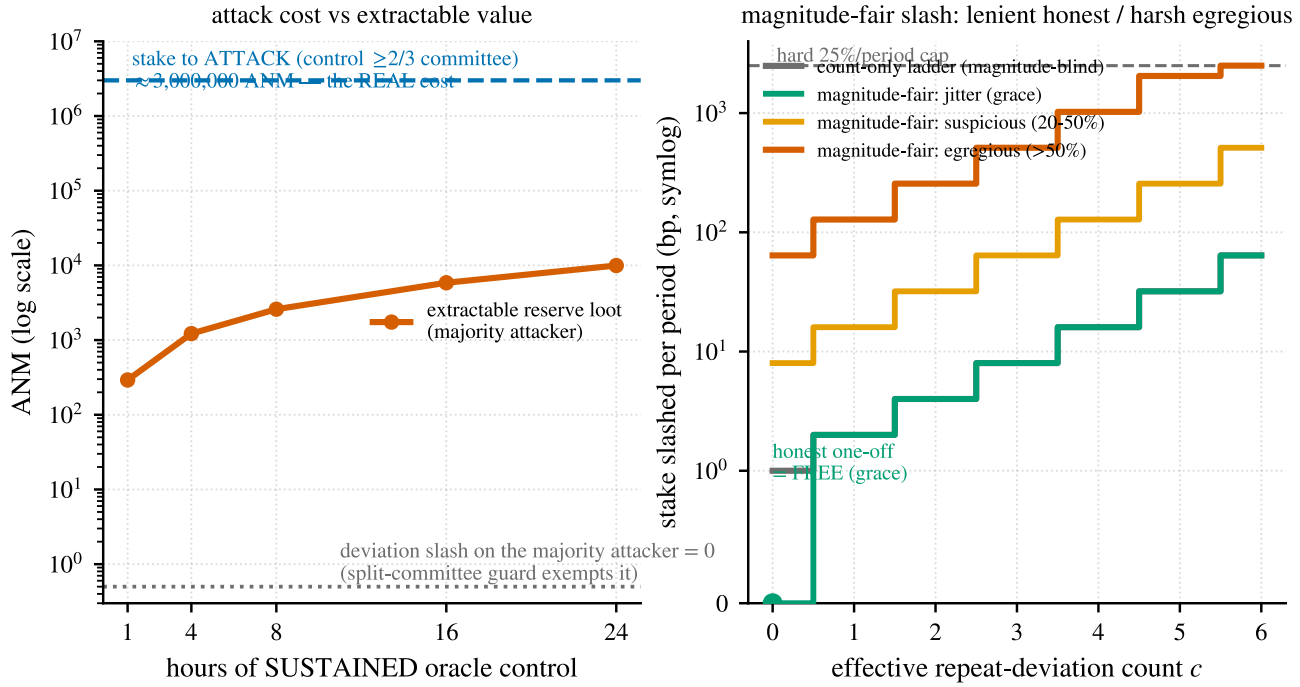

Fig. 19 — Slash calibration: (a) the deviation slash is not the majority-attack defence — a power-majority that moves the price is split-guard-exempt, so the binding cost is acquiring the stake itself, which dwarfs the clamp/TWA/breaker-bounded extractable value; (b) the magnitude-fair curve against a flat count-only ladder — lenient on honest jitter, far harsher on egregious lies.

The per-validator escalation state (validator number, stored count, last deviation height, **freeze-until height, ban flag**) lives as a leaf in the **stablecoin** module's merkle subtree, *not* on the validator object, and an **absent record is exactly count 0 / not-frozen / not-banned**, contributing nothing to the subtree root. Because a deviation is *provable* — a signed bad price the certificate attests — escalating its slash is fully trustless: no proposer or committee can fabricate a deviation against an honest validator.

- **Graduated *freeze* and dynamic *ban* (escalation beyond the stake slash).** On top of the per-period slash, a validator that keeps signing provably bad prices is progressively **removed from price aggregation** so a persistent deviator cannot keep feeding an outlier into the median at all. Two thresholds on the stored escalation count, evaluated in the same commit step that writes the deviation record: once the count reaches 3 the validator is **frozen** until a height that *ladders* with the count ($H + \mathsf{FreezeBasePeriods} \cdot (c - \text{threshold} + 1)$, 200 periods), and once it reaches the egregious count of 6 the validator is **permanently oracle-banned** (a state-committed *dynamic* ban, distinct from the static config ban; it never reverts). While a validator is frozen or banned its price **entry is dropped before the median**: the proposer omits it when building the oracle section, validation *rejects* a section that includes it, and the commit aggregation re-derives and drops it — all three reading the **same committed pre-block records at the same block height**, so the frozen/banned set is byte-identical at propose/validate/commit and on a re-syncing node (integer-only, no float, no map-iteration-order, no wall-clock). Dropping a deviator's entry only removes an *outlier*: the tiered/expanding quorum (§5.8) still requires an honest majority of the remaining subset, so the freeze/ban cannot be used to *manipulate* the price, and an attacker cannot freeze honest nodes (a freeze is set only on a provable deviation vs the period median, split-committee-guarded). The freeze **expires on its own** at its height while the count keeps decaying, so a validator that stops deviating recovers automatically; the freeze base period is deliberately shorter than the decay half-life so that an egregious offender that keeps deviating each time its freeze lifts ratchets the count up to the permanent ban. The freeze/ban fields default to the zero value and a record exists only after a real slash, so an unslashed genesis carries no such records.

The full escalation ladder — the magnitude-tiered per-period stake slash (the $1 \to 64$ bp doubling base, the $\times 8/\times 64$ magnitude shifts, and the 0.64%/6%/25% per-tier ceilings under the hard 25%/period cap), the grace on a first small deviation, the freeze at count 3, and the permanent ban at count 6 (reached in three egregious deviations), together with the lazy half-life decay that returns a reformed validator to the base tier — is shown in **Fig. 20**.

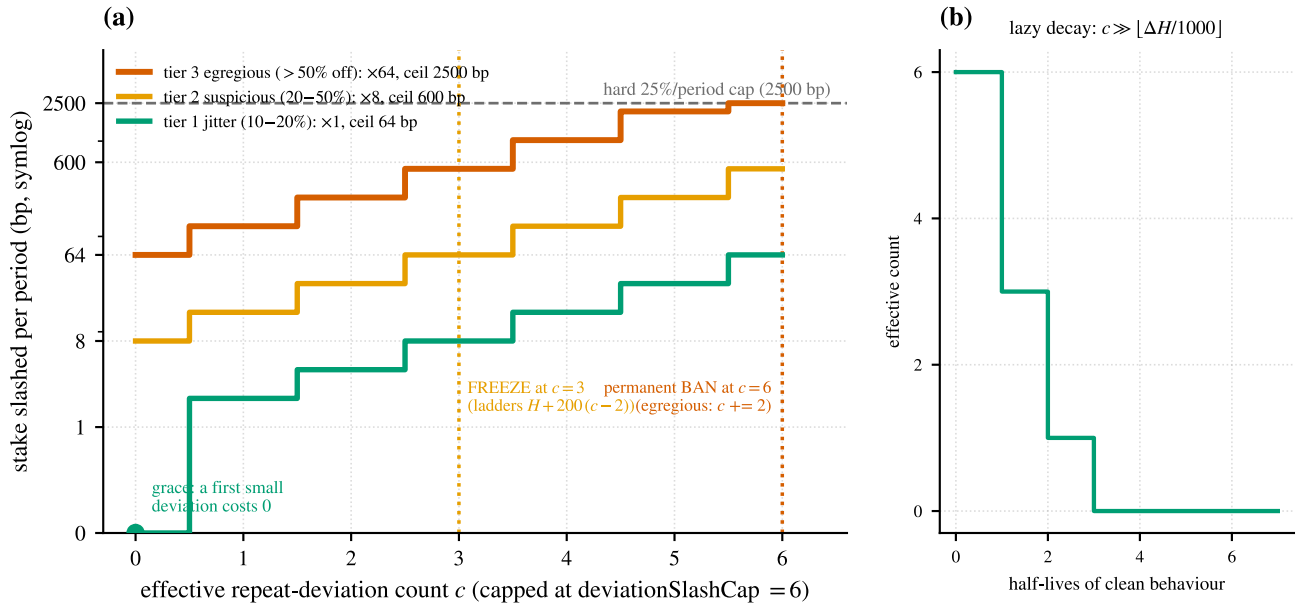

Fig. 20 — Oracle-deviation escalation ladder: the magnitude-fair graduated slash (base 1 bp doubling per repeat; ×8/×64 magnitude shifts for suspicious/egregious lies; per-tier ceilings 0.64%/6%/25% under the hard 25%/period cap; grace on a first small deviation), freeze at count 3, permanent ban at count 6 (an egregious deviation advances the count by 2), with the lazy half-life decay inset

5.4 Staleness and the price TWA

The accepted price is a **time-weighted average (price TWA)**, computed as follows each period: the committee attestations are reduced to a median; that median is **bounded to $\pm\delta$ (the per-period move cap, $\delta = 10\%$) of the running price** so no single period can inject an unbounded outlier; and the accepted price is then **advanced by a slow exponential moving average** $P_t = P_{t-1} + \alpha\,(\hat{m}_t - P_{t-1})$, where $\hat{m}_t$ is the bounded median and α ($= 800/10^6 = 1/1250$) is the TWA smoothing factor. With ~ 10 s blocks, $1/\alpha = 1250$ blocks is **$\approx$ 3.5 hours**, so the recurrence's exponential time constant *is* a **$\sim$ 3.5-hour price-TWA window**: the accepted price converges toward the market over $\sim 1/\alpha$ periods (a few hours) rather than tracking each period's median — single-block manipulation cannot move the peg, the same defence Uniswap-style oracles use against flash manipulation [13]. This is an $O(1)$ integer recurrence, **not** a stored 1250-sample moving average — the α-blended recurrence with a ~ 3.5 h time constant is the equivalent rolling average while preserving the Android $O(1)$/low-storage constraint. The committed peg tracks a genuine market move within hours ($\approx 63\%$ in ~ 3.5 h, $\approx 90\%$ by ~ 8 h); the security cost of a short integration window is paid back by the **committee term limit** (§5.6) — capping each member's tenure at ~ 8 h forces continuous rotation, so an attacker has far less time to assemble and hold a subset-majority within the window. The $\pm 10\%$ per-block move cap is a load-bearing constant of this security argument. Crucially, a *transient* committee majority moves the peg only $\alpha\,\delta \approx 0.008\%$ per period (not a full δ), and a *sustained* real step is absorbed $\approx 63\%$ $(1 - 1/e)$ over one time constant (~ 3.5 h) and $\approx 90\%$ over ~ 8 h, so dragging the peg any meaningful distance still requires a **sustained** majority over **thousands** of periods — the honest-majority-of-stake ceiling, enforced by lag. **Fig. 21** plots both behaviours from the recurrence ($\alpha = 800/10^6$, $\delta = 10\%$): a *genuine* $+10\%$ market step is tracked to $\approx 63\%$ over the following ~ 3.5 h ($\approx 90\%$ by ~ 8 h), while a *single-block* adversarial spike moves the committed peg by at most $\alpha\,\delta \approx 0.008\%$

— visually flat. **Fig. 22** quantifies the oracle-speed / committee-rotation tradeoff. A **staleness cap** halts mint/redeem after N (default 6, ≈ 1 min) consecutive periods without a fresh quorum, converting a silent-withholding attack into a safe halt rather than a stale-price exploit. Only this quorum-gated price TWA — never a spot price — is fed into the mint/redeem/emission gates. This is deliberate manipulation resistance. It would otherwise carry a cost on the *mint* side — a fast ANM decline would let a minter deposit collateral at the stale-high price TWA — but the **mint-guard price of §4.2** closes that leak: solvency-affecting ops are valued at min(price TWA, LastMedianPrice) (the fresher ±10%-clamped median), so the slow price TWA protects the peg from fast manipulation while the guard protects the reserve from over-valued mints during a fast genuine decline. Both follow from the same price TWA; the residual mint exposure is bounded by the ±10% per-block clamp on the fresh reference (§4.2).

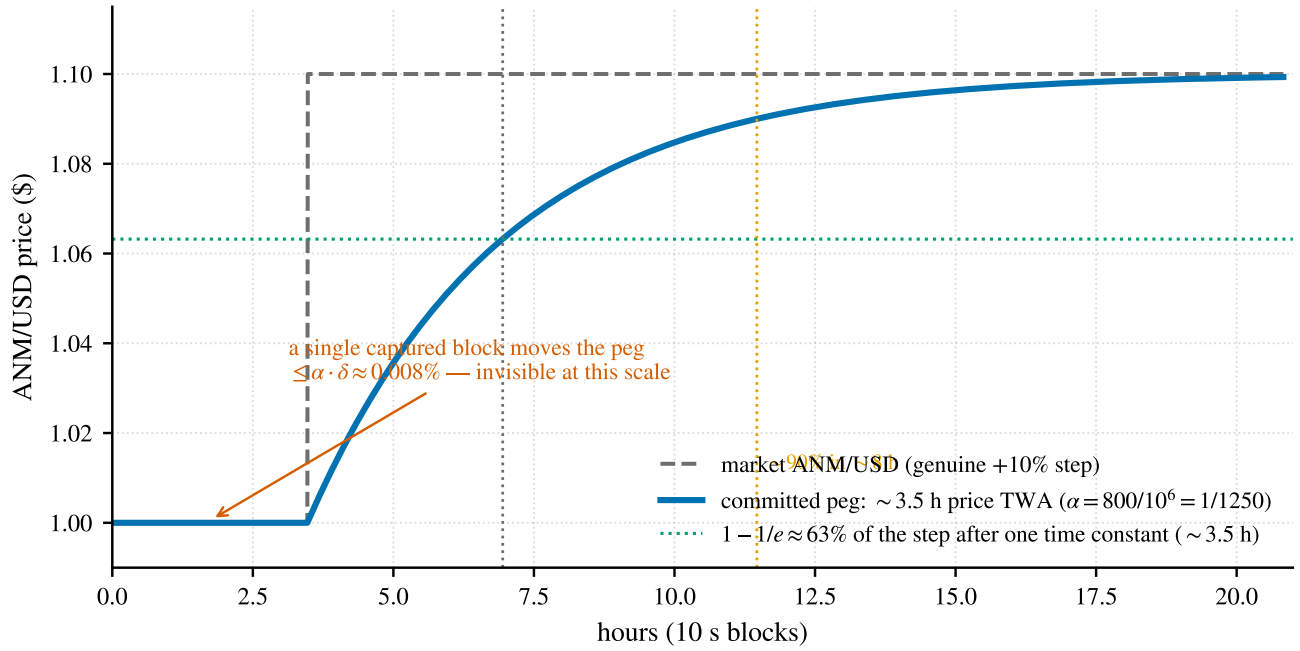

Fig. 21 — Oracle ~3.5 h price-TWA convergence

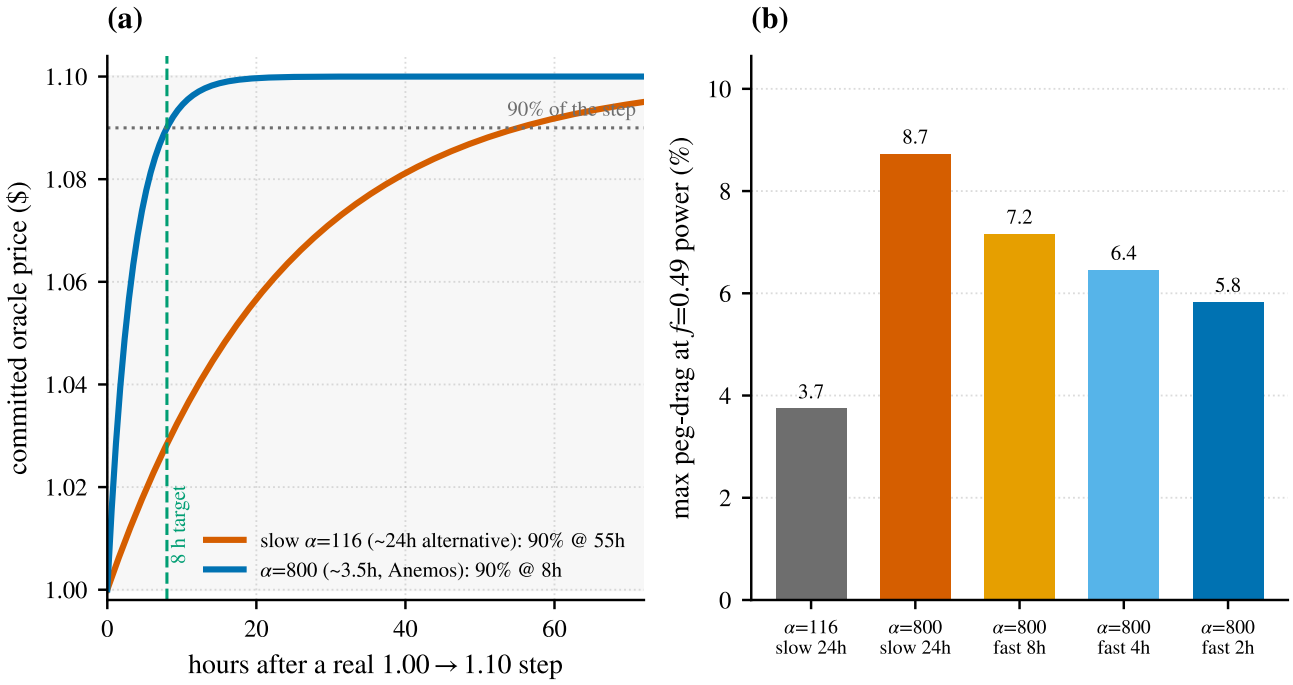

Fig. 22 — Oracle speed vs committee rotation: the ~3.5 h price TWA is a deliberately short integration window, and the ~8 h committee term limit is the rotation compensator that keeps the honest-majority-of-stake ceiling intact.

Atomic resume, no front-running window. Mint/redeem in block h execute against the *previously committed* (lagged) price, never an in-flight attestation in h's own mempool. The period aggregation runs in the block's commit hook **after** transaction execution and writes the new accepted price into committed state *before* the wind-down/restructuring mode transition is computed — so the price write and the mode change are committed **atomically in the same block**. Consequently, if a staleness halt resumes and the first fresh median is insolvent ($r < 1$): (i) no redeem ran at the stale-high price in the resume block, because mint/redeem are halt-gated through it; and (ii) the price-plus-mode transition is set in the same commit, so the next block's redeems are already restructured or pro-rata (§4.7–4.8). The transition is never a separate, front-runnable transaction — it is a deterministic function of committed attestations.

5.5 Trust ceiling (stated plainly)

The security argument composes three layers — per-block, single-block, and integrated-over-the-window:

- **Per block:** the price is the median of the ~11-member subset, so moving it needs a strict majority (≥ 6 of 11) of *that block's* selected, colluding members. A 1/3-of-committee adversary draws a subset majority only occasionally.
- **Bounded single-block capture:** even a fully captured subset moves the *committed* price by at most $\alpha\,\delta \approx 0.008\%$ in that block, because of the $\pm 10\%$ per-period clamp plus the slow price TWA (§5.4). One block of capture is essentially worthless.
- **Honest-majority-of-committee over the window:** the next block draws a fresh, independent subset. The price TWA's $\sim 1/\alpha$ ($\approx$ 1250-block, $\approx$ 3.5h) memory integrates $\approx 1250 \times 11 \approx 14,000$ attestations across many independent random subsets — i.e. the *whole* committee sampled many hundreds of times — so the smoothed price reflects the honest majority of the **full committee**. To drift the peg any meaningful distance an attacker must capture a *majority of subsets over many consecutive blocks*, which requires controlling $\approx$ honest-majority of the full committee — and committee seats are stake-weighted VRF sortition, bond-lagged, $< 1/3$-rotation-per-block, **and term-limited to ~ 8 h (§5.6)**, so that ceiling is **honest-majority-of-stake**, the same assumption as consensus itself.

Deviation slashing punishes a *minority*; a colluding subset majority *is* the agreed price for that block and would mis-slash the honest minority, which is exactly why the split-committee guard (§5.3) suppresses slashing when deviators are $\geq$ half the attesters. We lead the defence with **economic security, source diversity, the price TWA, the per-block move cap, subset rotation, and the subset quorum**, and treat the oracle subset size and committee size as moderate levers: a larger subset is more robust per block but costs more aggregation latency within the 10 s budget, and the Android constraint caps how large the committee should grow.

This headline sampling-diversity guarantee — "many independent random subsets integrate to the whole committee" — assumes the **mainnet target committee size (75)**. On a smaller committee, such as the current testnet's 11, the oracle subset already spans the *entire* committee every block, so there is no cross-subset averaging left to dilute a colluding minority's influence: the security argument on a committee that small reduces to trusting that single committee's honest majority directly, block by block, rather than the many-subsets argument above. The mainnet argument stands at its target size; this is a disclosed scope limit, not a retraction.

5.6 Can an attacker fast-activate a committee to control the price?

A natural worry: if someone activates many validators in quick succession, can they pack a committee and dictate the median? On vanilla Pactus a single round of committee control is economically harmless; once a committee can move a *price*, the question has teeth. On Anemos the answer is **no — not without an honest-majority *stake* break**, for four compounding reasons.

1. **Committee seats are won by stake-weighted VRF sortition, not arrival order.** Seat selection is a verifiable random draw weighted by bonded stake, so "activate 100 validators in a row, fast" does not deterministically place them on the committee — it places stake-proportional *probability* there. Buying a committee majority therefore costs a *stake* majority, the same resource consensus safety already assumes is honest.
2. **Bonding is rate-limited and lagged.** A newly bonded validator cannot join the committee instantly: a bond interval (≈ 360 blocks, ≈ 1 h) and a per-block rotation cap (strictly less than 1/3 of committee power admitted per block) bound how fast committee composition can change. A flash-activation burst cannot turn into a committee majority within a period.
3. **Even a captured subset is rate-limited by §5.2 and rotates next block.** Should an attacker *still* assemble a transient subset majority, the per-block move cap and price-TWA lag bound the damage to a slow, bounded, deviation-slashed drift ($\alpha\,\delta \approx 0.008\%$ that block, the same figure as §5.4/§5.5), and the *next* block draws a fresh independent subset (§5.1) — not a one-block peg seizure. This is exactly why a single round of committee control is *also* tolerable here despite the price mattering.
4. **Committee term limits force continuous turnover.** A member cannot hold a seat indefinitely: the longest-serving committee member is rotated out after a bounded tenure ($\sim 2,880$ blocks ≈ 8 h), so even a slowly-assembled majority cannot *hold* its seats long enough to bias the price across the integration window. This is the deliberate compensator for the ~ 3.5 h price TWA (§5.4): a short integration window is paired with mandatory rotation, so the time an attacker would need to control a sustained subset-majority is itself capped (**Fig. 22**).

In short, the oracle inherits consensus's own honest-majority-of-stake assumption and adds four independent rate limiters (sortition randomness, bond lag, move cap, and term-limited rotation) on top. The committee-to-price attack reduces to "break consensus," which is out of scope for *any* mechanism on the chain.

5.7 Oracle-availability exclusion, not an oracle-absence slash

§5.3 deliberately **never slashes absence**, because oracle absence is *not trustlessly attributable*: the proposer builds the oracle section from the price votes it actually received, and a node can never *prove* it received a vote that the proposer omitted (the speaker/listener equivalence of asynchronous gossip). A slash for "you failed to attest" could therefore be weaponised by a malicious proposer to frame an honest attester. So

repeated oracle absence is met not with a stake slash but with a **recoverable exclusion from proposing** — a sibling of the chain's existing consensus-availability gate.

A deterministic **oracle-availability score** is maintained over the same rolling window as the consensus score, but the obligation unit is **oracle-subset membership** rather than committee membership: for each block that committed an oracle section, a validator is *obligated* if it was in that block's subset and *present* if its (valNum, price) entry appears in the committed section; absent = obligated − present, and score = 1 − absent / obligated (1.0 if never obligated in the window). **A frozen or banned validator is *not* exempt from the obligation.** It ranks into the subset like any member, but the protocol drops its entry from the oracle section, so it is *obligated* yet never *present* — counted **obligated-and-absent** every subset block, collapsing its availability score below the gate. Counting a banned validator as obligated-and-absent is what makes a ban costly: exempting it from the obligated set would leave it a 1.0 score and its full **proposing income forever** while doing zero oracle duty — a ban is never a free opt-out. A ban therefore costs that income too — the offender is excluded from proposing in addition to forfeiting the oracle-participation reward slice (§5.16, it is never a correct attester) and the stake slash + freeze/ban of §5.3. The three penalties are *complementary*, not a double-count of one offence: the stake slash punishes the *provable signed bad price*, while the propose exclusion + reward forfeiture are the natural consequence of a banned validator *no longer attesting at all* — exactly the absence axis this gate governs. The obligation feed is applied **identically** on the live commit feed and the warm-replay feed and is a pure function of the committed pre-block records and height, so every node (including a re-syncing one) derives the byte-identical obligated set. At propose time, if a proposer's oracle-availability score is below the **minimum oracle-availability score** (**0.5**, set looser than the 0.667 consensus threshold because oracle obligations are sparser and each miss weighs more), the change-proposer phase **skips it** — exactly as the consensus gate does — until it participates again. The penalty is **exclusion, recoverable by participating**, never stake loss, so the framing limit of an asynchronous absence signal is harmless: a malicious proposer-minority cannot push an honest validator below the threshold over the window (honest-majority-bounded), and the worst a frame achieves is excluding a node that is already absent.

Three properties make this safe to gate live block production on a secondary signal. First, an obligation is recorded **only for blocks that actually committed oracle data** — a block with no oracle section (oracle inactive, no feeders, or below quorum) records no obligation, so when the oracle is *down network-wide* every score stays 1.0 and the gate never fires. Second, because the score is *recomputed from committed history* and is read **only** by the propose gate (a liveness decision that change-proposer resolves) and the read API — never by any hashed or state-root computation — a slightly different warm-up across nodes cannot fork the chain.

Third — and this closes the one gap the first property does not cover — the gate (together with its consensus-score sibling) is **bounded by two deterministic liveness escapes**, so a proposer quorum always exists. The first property protects the oracle-*down* case, but a *degraded* oracle (thin sections still committing) can legitimately drag many scores below the thresholds at once; because the scores are pure functions of *committed* history, they can only heal when new blocks commit, so an unbounded gate that excluded every seated member simultaneously would deadlock the chain **permanently and restart-proof** (nothing proposes → nothing commits → the scores never change). The escapes: **(i) an exclusion cap** — the availability gates together may exclude at most $f = \lfloor (n-1)/3 \rfloor$ of the n seated members, so at least $n - f \geq 2f + 1$ eligible proposers always remain; when more members are sub-threshold, the *highest-scored* gated members are kept eligible (deterministic tie-break by validator number), preserving the gate's pressure on the worst offenders — and **(ii) a full-rotation escape** — once a height has churned round $\geq n$ change-proposer rounds without a commit, the score gates are ignored entirely (liveness beats exclusion). Both escapes are **propose-local**: the scores are never consulted when validating a received proposal or block, so differing local decisions cannot fork the chain — the change-proposer sub-protocol resolves them, exactly as it resolves the gates themselves.

5.8 Tiered, expanding quorum — a fresh price instead of carry-forward

The single-shot part of the oracle is the fixed 11-member subset (§5.1). When too many of those 11 are offline, tier 0 cannot reach its quorum and §5.4 would carry the stale price forward. A **tiered, expanding quorum** ("rounds") recovers a *fresh* price in that case instead. Tier 0 is the base subset (size 11); the later tiers widen the pool to 17, then 25, then the **whole committee** — each a longer **prefix of the same** Hash(seed ‖ valNum) ranking, so a higher tier's number-set is a strict **superset** of every lower tier. Every tier size is **capped at the live committee**, so the final tier is always the whole committee, however large it is configured. The proposer walks the tiers from 0 upward and includes the **first** tier that reaches its quorum; the chosen tier is recorded in the block as a single byte, and the validate/commit paths re-derive the *same* tier from the same committed $H-1$ seed. A tier is accepted only when it reaches **both** its head-count quorum **and** the $2\not{4}$3-of-power quorum of §5.1, the same two-gate rule the per-block subset uses. Tier 0 is byte-identical to the single-subset case; widening costs only a few extra signatures, and only when the small tier is thin.

Widening is cheap precisely because the **off-chain feeder caches its price** (§5.9): a feeder polls its external sources at most once every few seconds and *re-uses* the cached integer across attestation attempts, so every committee member can sign from cache without hammering any API — a thin tier-0 can be widened to a larger tier without a new round of price fetches. The median, the per-block move cap, and the slow price TWA are the **same regardless of tier**; the only thing that varies is *how many* genuine signed votes the median is taken over, and tier expansion is fully deterministic, so the omit-to-skew residual stays the same bounded one (§5.1).

The nested tiers and their per-tier acceptance quorum $\lfloor 2k/3 \rfloor + 1$ (for an effective tier size k: ~~81~~ ~~17~~, ~~25~~ 17, and the top tier = the whole committee) are shown in **Fig. 23**. Every tier size is **capped at the live committee** ($k = \min(\text{tier size}, |\text{committee}|)$), so the largest tier is always every committee member (≤ 75 on mainnet, 11 on testnet) and its quorum is two-thirds of the *actual* committee — e.g. 51 at the 75-validator mainnet committee; on testnet, where the committee is 11, tier 0 already **is** the whole committee, so the wider tiers coincide with it.

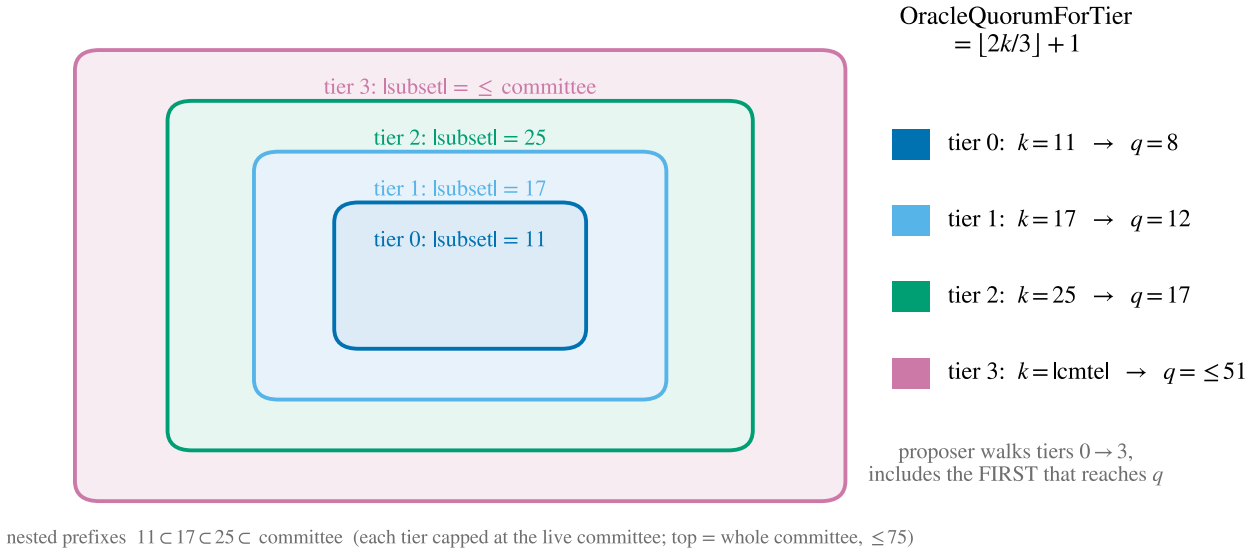

Fig. 23 — Tiered, expanding oracle quorum

5.9 The off-chain price feeder

The on-chain side (§5.1–5.6) defines *how* attestations are aggregated and policed; it does not say *where* a validator gets the price. That is the job of an **off-chain feeder** that each committee validator runs.

The reference implementation pulls the ANM/USD price (for testing, the PAC/USD price) from several independent public market-data APIs — CoinGecko, CoinPaprika, and CoinMarketCap — takes a **robust median** of whichever sources answered (requiring a minimum number of successful sources, default 2), converts it to the integer price-scale unit, and — **only when this validator is a committee member eligible to attest** (it checks membership at the max tier — i.e. the whole committee — over the same committed $H-1$ seed the commit path uses, so the proposer can deterministically widen a thin tier-0 subset to quorum, §5.8) — signs and **gossips a price vote** (§5.1). A non-committee validator stays silent; a committee member outside tier 0 still signs (its higher-tier vote is simply ignored when tier 0 already reaches quorum). The price never rides in a transaction. Each source is fetched concurrently under a timeout; a failed, hostile, or non-finite response is dropped, and the loop tolerates transient failures.

Crucially, the feeder is **not consensus-critical**. It may use floating point and wall-clock — the only value that crosses into consensus is the single integer it signs into a price vote, which the on-chain layer then re-medians *across the subset* with the subset quorum, per-block clamp, price-TWA lag, and deviation slashing of §5.2–5.5. A validator whose feeder is wrong, slow, or briefly fooled by one bad source contributes one attestation that the on-chain median absorbs; it cannot move the accepted price. The feeder is therefore an *availability and convenience* component layered on the on-chain trust model, not a trust root — consistent with the honest-majority-of-stake ceiling. Source endpoints and the priced ticker are configuration, so the same feeder retargets to ANM's own market by configuration.

5.10 Storage cost (why consensus-embedded, not per-validator transactions)

A per-validator price-attestation transaction approach would store **one signed attestation per committee member per block, forever** — at a 75-member committee, $\approx 75 \times 150\,\mathrm{B} \approx 11\,\mathrm{KB}$/block, i.e. $\approx 35\,\mathrm{GB/yr}$ of pure attestations on an archival node. The consensus-embedded design stores instead a single ~11-entry oracle section with **one 48-byte BLS aggregate signature**: $\approx 11 \times 12\,\mathrm{B} + 48\,\mathrm{B} = 180\,\mathrm{B}$, plus the section's varint count and tier-byte framing, $\approx 185\,\mathrm{B}$/block — a **~58× reduction**, while remaining fully trustless and slashable (the section is re-verified by every node, §5.1). The chain **prunes**: a non-archival node deletes blocks older than the retention window (default/min 10 days $\approx$ 86,400 blocks), so a staking / Android node only ever carries a bounded rolling window regardless of design; the ~58× saving is largest on archival nodes, where it cuts $\approx 35\,\mathrm{GB/yr}$ to $\approx 0.6\,\mathrm{GB/yr}$. The aggregate BLS verify is an $(N+1)$-pairing check $e\big(\sigma_{\mathrm{agg}}, g_2\big) = \prod_i e(\mathrm{H}(m_i), \mathrm{pk}_i)$ over the subset's distinct $\mathrm{H}(\mathrm{height} \,\|\, \mathrm{valNum} \,\|\, \mathrm{price})$ messages, with rogue-key resistance from using only pre-registered committee keys and per-message identity binding.

5.11 Pooled delegation and slashing — who bears the penalty

A validator is a **pool**, run by an **operator** that holds the BLS consensus key (the key that signs price votes and runs the off-chain feeder). Many holders delegate principal into one pool, so a natural concern is *who actually pays* when the pool is slashed for deviation (§5.3) — the operator, the delegators, or both. The pool's accounting makes the answer unambiguous, and it does so without changing any consensus weight: **a validator's power is its stake, exactly as before.**

- **A validator's stake is the pooled principal.** validator.Stake equals the operator's self-bond plus the sum of every delegator's principal, and Power () = Stake is **unchanged** — pooling does *not* subdivide power, mint per-delegator validators, or alter sortition, committee selection, or the oracle's power-weighted median. A pool is one committee identity whose weight is its total principal; the protocol's power-bearing object is untouched, so the pool model adds no new surface to the consensus, sortition, or oracle proofs (§5.1–§5.10).
- **The pool fills incrementally within the stake range.** A pool's stake grows as delegators join, bounded to [MinimumStake, MaximumStake] (the supply-indexed range below). The operator opens the pool with a self-bond of at least OperatorMinSelfBond (a tunable genesis parameter, launch 2,000 ANM) — set deliberately *above* both the launch minimum stake and the per-delegator minimum so the operator

always has meaningful skin in the game. Each delegator adds at least PoolMinDelegation (500 ANM) per position. The floor is sized so a **full** pool of the minimum-count (32) delegators still contributes only $32 \times 500 = 16,000$ ANM of delegator principal — comfortably above dust yet each individual position stays *below* the 1,200-ANM validator MinimumStake (retail stays retail) and equals exactly one quarter of the 2,000-ANM OperatorMinSelfBond (the operator always has $\geq 4\times$ skin in the game relative to any one delegator). Once the pool reaches MaximumStake it is **full** and further deposits are rejected; a whale who wants more weight must run more pools, each capped — the same decentralization brake as before.

- **Two balances per position, and only one is slashable.** Each delegator position carries a *principal* (staked ANM, part of validator.Stake, slashable) and a separate *claimable reward* balance that accrues in a **non-staked reward escrow** and is **never** part of validator.Stake. This split is what makes rewards slash-immune: a slash can only reach staked principal, so it can never reach the escrow (formalized below).
- **Per-share accounting, integer and $O(1)$.** A pool tracks total principalShares (one share per Gust of principal at deposit), a rewardPerShare accumulator, and a lossPerShare accumulator. Each block the pool's delegator-reward portion bumps rewardPerShare **once**, and a slash bumps lossPerShare **once** — never a per-holder loop, so a pool of any size costs the same $O(1)$ per block (the Android-viability invariant, §1). A position's claimable reward is shares $\times$ (rewardPerShare $-$ rewardDebt) and its realizable principal is shares less its settled share of lossPerShare, each settled **lazily** on deposit/withdraw/claim against a per-position debt baseline (the same lazy-settlement template as the recovery-token ledger, §4.8) — so a holder who joins after a slash or a reward inherits neither.
- **Commission is charged on rewards only.** The operator sets a commission in $[0, 2000]$ bps (cap 20%); each block the pool's reward P pays the operator $\lfloor P \cdot \text{commissionBps} \,/10000 \rfloor$ into its spendable account and bumps rewardPerShare with the remainder. Commission never touches principal, so a delegator's realized yield is the gross staking yield scaled by $(1 - \text{commission})$ (**Fig. 26a,b**) — independent of how full the pool is, since reward accrues pro-rata to principal. A commission *increase* takes effect only after a notice window (CommissionIncreaseCooldown, an anti-rug delay $\approx$ the unbond interval); a *decrease* is immediate.

The laddered cap ladder ($0.008\,S$) is a monotone rising step (**Fig. 24**): it launches at **60,000 ANM** ($S =$ 8M) and steps up only when supply crosses a leading-digit rung, so the decentralization brake scales with the elastic supply instead of decaying to a dust fraction of it — the same survival logic as the supply-indexed tail (§3.2). The integer leading-digit ladder makes it indifferent to the immaterial 32,000-ANM genesis auto-bond delta (§5.12): both $0.008 \times 8,000,000$ and $0.008 \times 8,032,000$ ladder to the same 60,000-ANM launch cap. As with the tail (§3.2), this coarse ladder means the cap *as a fraction of supply* is not flat at the nominal 0.8% but has the same halving sawtooth **character** as the tail (§3.2): ladder $(0.008\,S)/S$ rides from $\approx 0.8\%$ of supply (realised launch fraction $\approx 0.75\%$, since the ladder snaps 64,000 down to 60,000 at $S = 8$M) just after a rung down toward $\approx 0.4\%$ before the next rung snaps it back. The brake is therefore guaranteed to bound concentration only at the *floor* of each tooth (about half the nominal cap fraction), which is the conservative side — it constrains a single validator's stake more tightly between rungs, never less.

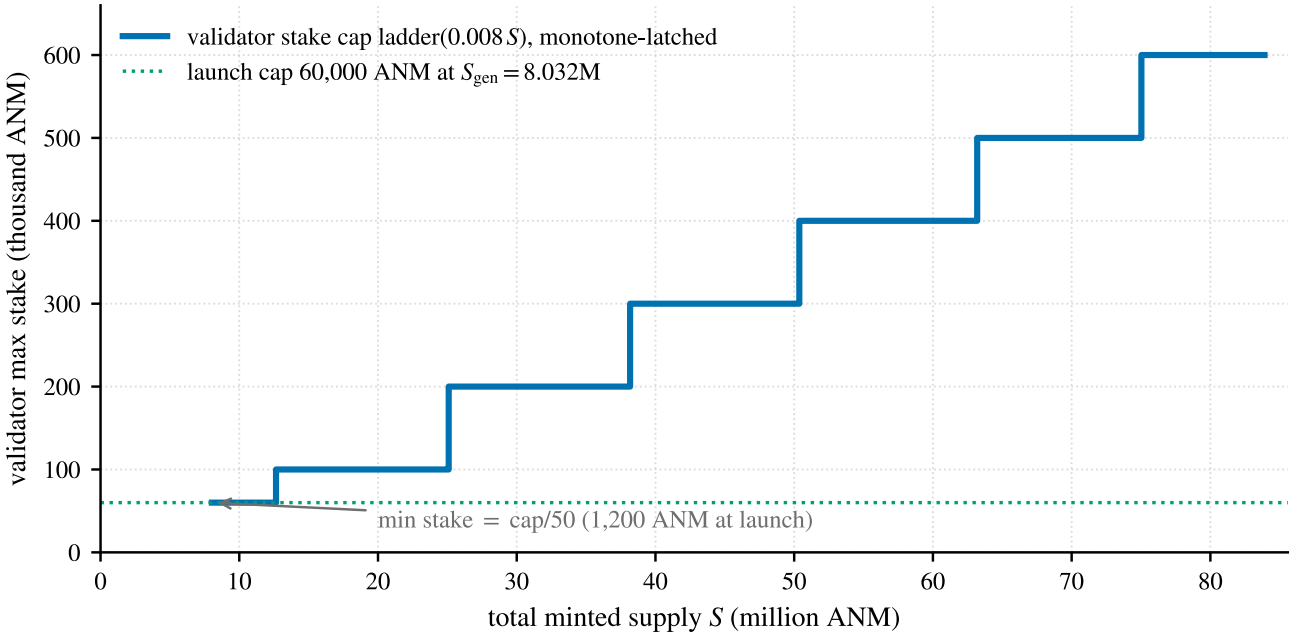

Fig. 24 — Supply-indexed laddered stake cap

The cap:min range is 50:1. The launch minimum stake is cap /50 (`capMinDivisor` 50; launch **1,200 ANM** mainnet / **800 ANM** testnet). A high floor keeps dust validators off the committee and compresses the per-seat power range, so committee power is distributed more **uniformly** — no single large-but-under-cap validator dominates a thin tail of minimum-stake seats. Because committee power enters the oracle (power-weighted median) and consensus (power-weighted quorum), a more uniform power distribution raises the cost of a power-majority capture for the same total stake — the security gain quantified in **Fig. 25**.

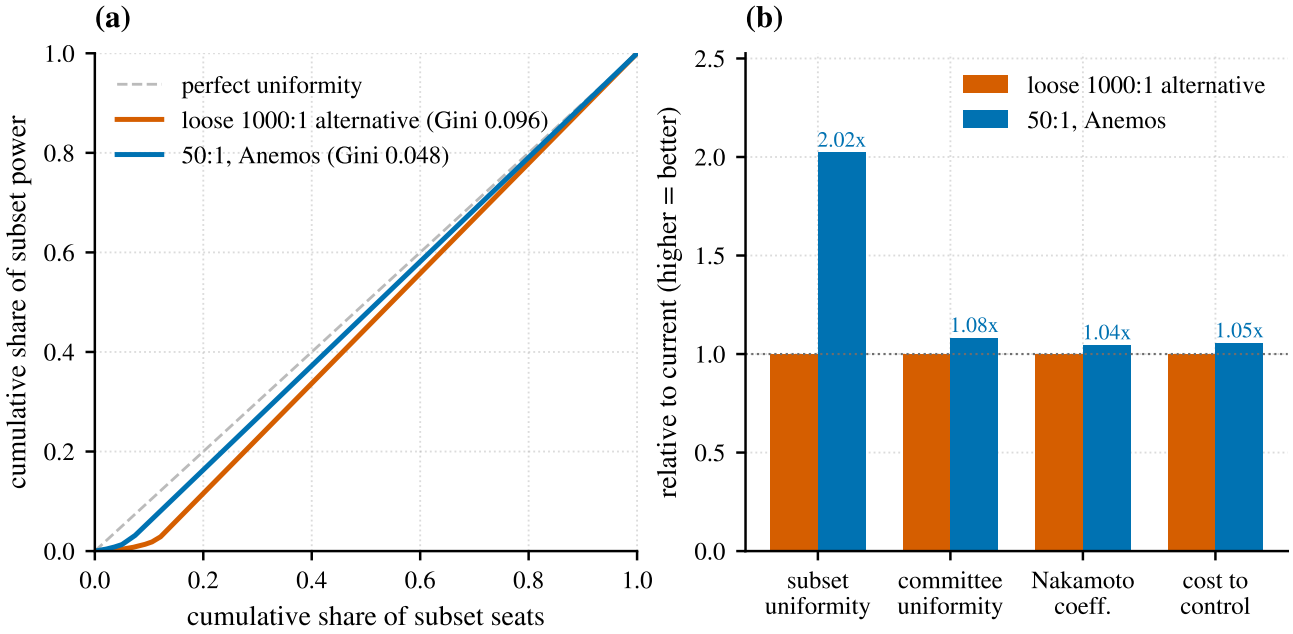

Fig. 25 — The 50:1 cap:min stake range: the high minimum-stake floor compresses the validator power distribution toward uniformity, raising the cost of a power-majority capture for the same total stake (a loose 1000:1 range is shown for contrast).

Consequently, a deviation slash subtracts from the pool's **principal** and moves the forfeited ANM into the reserve (§5.3). It is **socialized pro-rata** across the whole pool — operator and delegators alike, with **no first-loss buffer**: the slash bumps lossPerShare once, so every share of principal loses the same fraction,

and the operator (whose self-bond is principal too) is penalized on equal terms with its delegators. There is no co-staker who escapes the penalty and none who bears more than its pro-rata share — the "everyone on a bad pool is penalized equally" property holds *by construction* from the single accumulator bump. The slash base includes any principal sitting in live pending-unbond entries, so a position cannot dodge an impending slash by undelegating the block before (the same no-evasion invariant as §5.14). Rounding favors the protocol: the per-share loss settles upward, so positions can never collectively realize less loss than the pool forfeited, and the sub-Gust remainder stays as pool cushion — never an over-claim.

Rewards are slash-immune (the load-bearing property). Because claimable rewards live in the reward escrow and are *never* part of validator.Stake, the slash — which only reduces staked principal — cannot reach them. A delegator's accrued rewards are unaffected by any number of slashes on its pool; they remain fully claimable to spendable ANM at any time. Principal, by contrast, exits only through the **full validator unbond period** (UnbondInterval): an undelegate converts shares to ANM at the loss-adjusted value and parks it in a pending-unbond entry that the holder can withdraw only at/after its release height, reusing the same pending-unbond machinery as a partial unbond (§5.14). The reward/principal asymmetry is deliberate — rewards are liquid because they were never at risk; principal is illiquid because it is the security bond. **Fig. 26c** traces a single position through a deposit, reward accrual, an oracle-deviation slash, and a full-period exit: its realizable principal steps down at the slash and holds, while its cumulative claimable reward rises monotonically straight through the slash, untouched. The accompanying deterministic integer simulation asserts, bit-for-bit, that total ANM is conserved across the lifecycle (a slash is a conservative transfer principal → reserve), that the sum of realizable principal never exceeds the pooled stake, and that the slash leaves every claimable balance exactly unchanged.

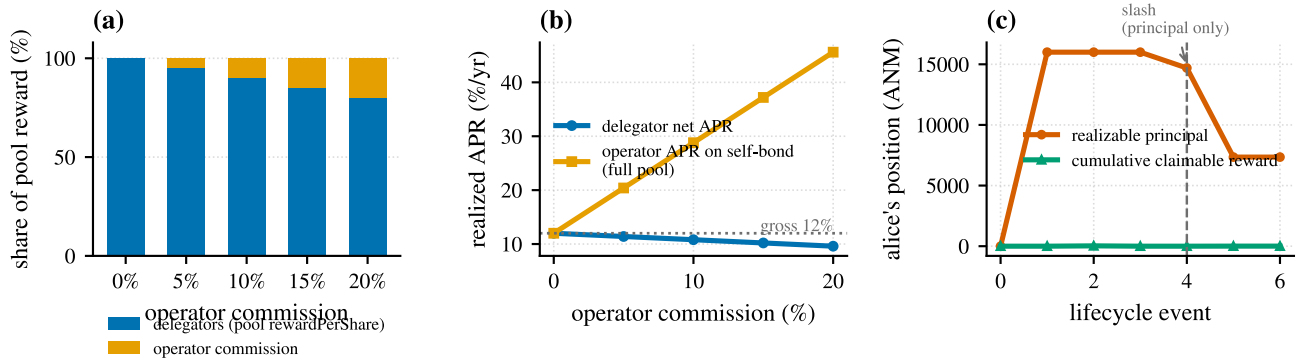

Fig. 26 — Pooled delegation economics: (a) the pool-reward split between delegators and operator commission; (b) realized delegator net APR (gross yield scaled by $1-$*commission, fill-independent) versus the operator's effective APR on its self-bond (which rises with pool fill as commission revenue spreads over a fixed self-bond); (c) the slash-immune-rewards proof — a position's realizable principal steps down at an oracle-deviation slash while its cumulative claimable reward rises monotonically straight through it.*

Delegating to a seated validator — the activation buffer. A pool's stake is its committee power, and a seated validator's stake is **frozen for its committee term**: changing it mid-term would desync the committee power a running node holds from the power a re-synced node rebuilds from the store, breaking the power threshold the precommit certificate attests (the same committee-power-consistency constraint that governs a partial unbond, §5.14). So a delegate or undelegate aimed at a validator that is currently seated in the committee (or joins it at the next height) is **not rejected — it is escrowed**. The request is recorded in a pending-delegation entry and its funds are **locked immediately**: a deposit debits and parks the delegator's principal at request time, while an undelegate locks only the transaction fee (its principal is already staked). The share and stake change then **activates** once the validator next rotates out of the committee, on the already-safe non-seated path — a buffered deposit mints principal shares at the pool's then-current loss-adjusted price and grows the validator's stake, and a buffered undelegate burns the requested shares and parks the released principal in a slashable pending-unbond entry for the delegator.

An escrowed deposit whose pool has **closed by activation is refunded** to the delegator, so no funds are ever stranded (conservation). At most one live buffered request per delegator per pool is allowed, bounded by a small per-validator cap, and a seated validator's buffered deposits count against the pool's fill cap while they wait, so an escrow can never overflow MaximumStake at activation. Each entry is its own merkle leaf that is empty on a fresh chain, so the buffer is genesis-hash-neutral. The user experience is simply that a delegation to a pool that happens to be serving in the committee *activates a little later*, when that pool rotates out — which it does within a committee term — rather than failing.

5.11.1 Operator participation controls

The pool model so far lets *anyone* deposit any amount at or above the global minimum into any pool. That is permissionless, but it is not free of cost to the pool's existing members, and it leaves the operator no way to shape the pool it is responsible for. Anemos therefore gives each operator four **participation controls**, all set in the same pool-registration `Bond` that carries the commission and self-bond, and all enforced by the protocol rather than by off-chain etiquette:

- **Private** — a boolean, **public by default**. A private pool admits principal **only from the operator's own account**; every `Delegate` from any other account is rejected, including an existing delegator topping up. Privacy is a strict, immediate toggle with no notice window, because it carries no fund-safety risk: delegators may **always** undelegate, so making a pool private can never trap an existing position.
- **MinDelegation** — the operator's per-position minimum, in Gust. Zero means "use the network-wide `PoolMinDelegation` floor" (launch 500 ANM); a set value must be **at least** that floor and is resolved at runtime as max(MinDelegation, PoolMinDelegation). The minimum gates **every** non-operator deposit — a new position *and* a top-up of an existing one — not just the first deposit, for the churn reason below. The operator's own deposits are exempt (its binding floor is the separate OperatorMinSelfBond).
- **MaxDelegation** — an optional per-position cap, in Gust (0 = no cap). A delegate that would push a single non-operator position's principal above the cap is rejected, on a top-up as well as a first deposit, so a single whale cannot dominate one pool. The operator self-bond is exempt.
- **MaxDelegators** — a cap on the number of *concurrent* non-operator positions. Zero means "use the genesis-wide `DefaultPoolMaxDelegators`" (launch 32); both the operator value and the default are bounded by the protocol hard ceiling MaxPoolMaxDelegators = 32. The operator self-bond never consumes a slot. The pool carries a live DelegatorCount — bumped by exactly one step per code path when a non-operator position is first minted or fully exits — so the cap is an $O(1)$ check.

Why a minimum, and why it applies to every deposit. A pool's stake is its committee power, and a *seated* validator's stake is frozen for its committee term (§5.11, §5.14). So a deposit into a pool whose validator is currently producing blocks does not take effect immediately; it is escrowed and activates only when that validator next rotates out (the activation buffer of §5.11). Activation grows the validator's stake, which forces a re-bond and a fresh sortition: the pool briefly leaves the active committee and must be re-selected (a `BondInterval` lag plus the expected re-sortition wait). During that reactivation window the pool proposes no blocks and signs no oracle votes, so it earns essentially no block or oracle reward — and because rewards are socialized pro-rata across the whole pool (one rewardPerShare bump, §5.11), the reward lost to that churn is borne by **every** member, not by the joiner who caused it. Churn is a negative externality. A larger minimum collects the same capital in fewer, larger deposits, and a delegator-count cap hard-bounds the number of churn events; both reduce the reward lost to reactivation. Crucially the minimum is checked on **top-ups too**: a dust top-up triggers exactly the same reactivation as a fresh deposit, so exempting top-ups would leave the churn channel wide open.

The commission-tier justification. Because a tighter pool loses less reward to churn, it delivers a higher net yield per delegated ANM, which in turn lets its operator sustain a **higher commission** while still clearing the market's net-APR floor; a churny, wide-open pool cannot. The operator's control over these ranges is therefore what makes a spread of commission tiers economically rational rather than arbitrary. A

deterministic integer simulation sweeps the (MinDelegation, MaxDelegators) policy grid against **one** seeded candidate-demand stream — every policy cell filters and clamps the same stream, so any difference between cells is attributable to the policy alone, not to randomness — and computes each cell's churn loss by an exact sweep-line over the union of the per-deposit reactivation windows. Tightening a pool from wide open (minimum 100 ANM, cap 32) to a representative tight setting (minimum 1,000 ANM, cap 16) cuts the fraction of reward lost to churn from roughly 16% to roughly 9%, lifts the net delegator APR, and raises the commission the pool can sustain from under 1% to about 8% (**Fig. 27**). The gains show clear diminishing returns, and over-tightening trades them against slot utilization — a high floor turns small delegators away and a low count cap leaves the pool underfilled — so the controls are a tunable knob, not a free lunch.

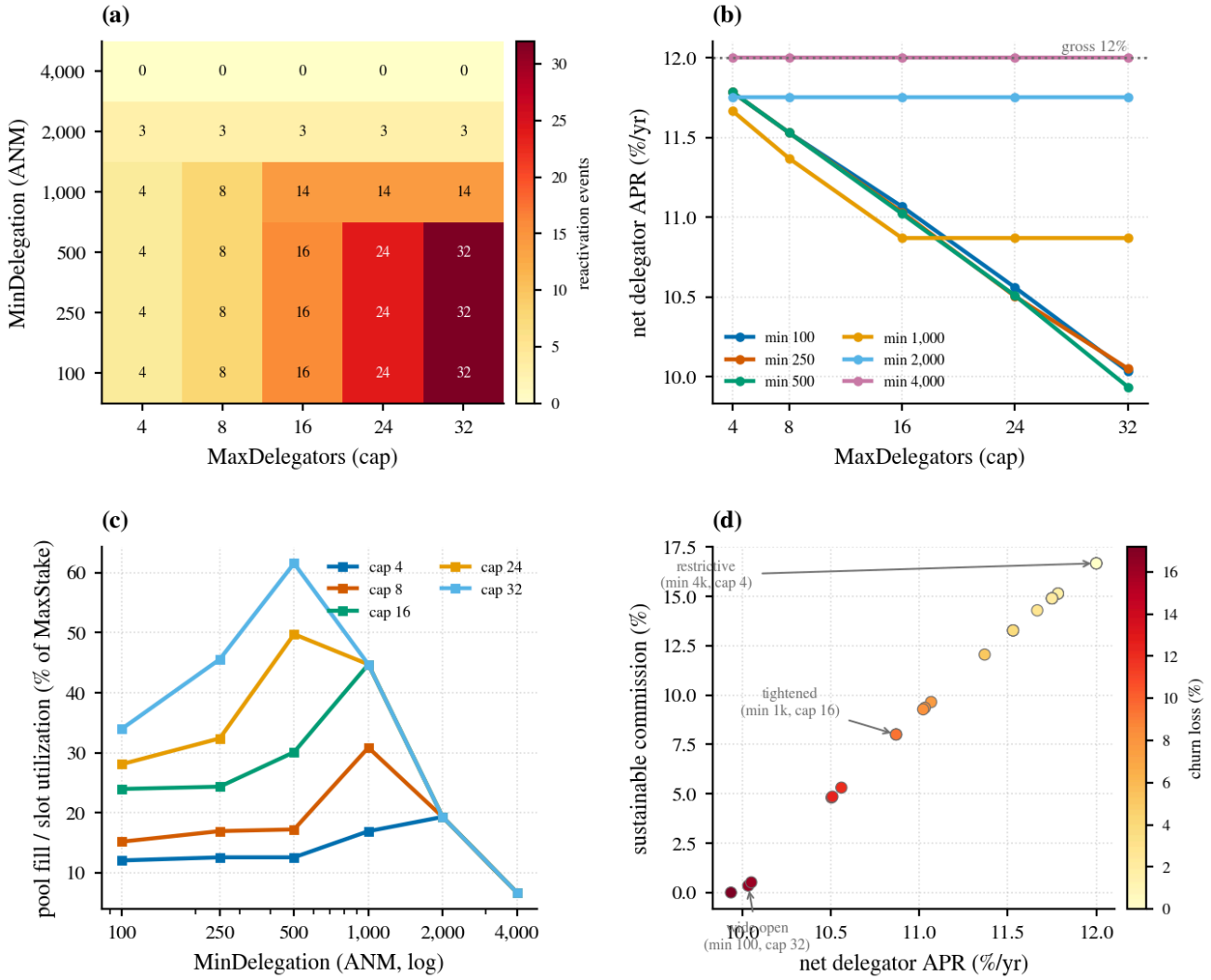

Fig. 27 — Operator participation controls and pool churn: (a) the reward lost to validator reactivation falls as the per-position minimum rises and the delegator-count cap tightens, with diminishing returns; (b) the net delegator APR and the commission a pool can sustain while still clearing the market net-APR floor both rise as churn falls; (c) the utilization trade-off — too tight a floor or cap leaves the pool underfilled.

Safety and determinism. What these controls *do* is bound churn and let an operator shape its pool; what they do **not** do is defend against a malicious operator or alter any slashing or consensus rule — privacy only restricts who may deposit, and a private pool's operator is slashed for deviation exactly like any other (§5.3, §5.11). Every control is an integer comparison evaluated identically on every node, folded into the fixed-width pool leaf, and the per-block cost stays $O(1)$ (the count cap is a single counter; the only scan, used to project a seated pool's pending new positions against its cap, is bounded by the small per-

validator pending-delegation cap). Both enforcement points agree: the direct `Delegate` path checks the controls before minting, and the **seated-escrow activation path re-checks the same controls and *refunds* the escrowed principal** (rather than rejecting, since the transaction has already committed and paid its fee) whenever a deposit would violate a control that has changed while the pool was seated — the activation-path-parity property that keeps a buffered deposit from ever bypassing a control. The four controls live in the pool leaf, which is empty on a fresh chain and therefore genesis-hash-neutral; the only genesis input is the `default_pool_max_delegators` launch parameter (Appendix A).

5.12 Genesis auto-bond and the one-year lockup

At genesis the mainnet committee is seeded by **auto-bonding a fair-launch allocation into each genesis validator**: every one of the four genesis validators carries an 8,000-ANM stake (not an idle account balance), so the network starts with its security budget already staked rather than depending on holders to bond after launch. These four validators are **separate genesis entities** — their keys are derived from their own deterministic, network-domain-separated seed labels, **additive to** the 1,000 placeholder account holders, not carved out of them. The four $\times$ 8,000 = 32,000 ANM of auto-bonded stake is therefore minted **on top of** the $1,000 \times 8,000 = 8,000,000$ ANM holder allocation, raising the mainnet genesis supply to **8,032,000 ANM** (the Treasury sentinel still holds zero — no premine). That genesis-bonded **principal is locked for one year** — genesisHeight $+ 3,153,600$ blocks at 10 s blocks. The lock is a committed per-validator record (validator number and a stake floor) carried in its own merkle subtree (modeled on the oracle-deviation subtree, §5.3): a validator cannot unbond, partial-unbond, or withdraw its stake **below that floor** until the unlock height, after which the floor is zero and the stake is free. **Rewards are always free** — they accrue to a separate reward account and are never folded into the locked principal, so the lockup never traps emission income. An *absent* record means "not genesis-locked" and contributes nothing to the subtree, so a localnet/test genesis with no auto-bond keeps an empty subtree; a mainnet genesis with auto-bonded validators carries these records.

The floor is enforced at **unbond time in both unbond branches**: a full unbond of a locked validator is rejected outright (it would drop the remaining stake to zero, below the floor), and a partial unbond may reduce stake **down to exactly the floor** but no further. Withdraw needs no separate check and contains none — a locked validator can never fully unbond (so it can never reach the unbonded state required to withdraw below the floor), and a partial unbond already cleared the floor before parking the funds in a slashable pending entry, so no withdraw path can release stake below the floor. Withdraw is therefore *structurally* protected, not actively checked. The floor expires deterministically at the global unlock height (pure integer height comparison, no wall-clock — a re-synced node computes the identical floor).

5.13 Genesis multi-validator split

A holder's whole allocation is auto-bonded, but the **dynamic stake cap** $C(h)$ (§5.11) bounds how much one validator may hold. When an allocation A exceeds the launch cap, it is split across $K = \lceil A/C \rceil$ validators: base $= \lfloor A/K \rfloor$ to each, with the integer remainder distributed one Gust at a time to the first $\text{rem} = A - \text{base} \cdot K$ validators, so the K shares sum **exactly** to A and every share is $\leq C$. The holder derives K validator keys from its HD seed (incrementing the index of one HD derivation path). At the launch cap (60,000 ANM) a standard 8,000-ANM holder has $A < C \Rightarrow K = 1$, so the split is **dormant** for standard holders and activates only for aggregated funders above the cap.

5.14 Partial unstaking — provably no weaker than all-or-nothing unbond

The unbond transaction carries an **optional amount**. A zero amount is the **all-or-nothing unbond**: the validator gives up its whole stake (zeroing its power) and withdraws it after the unbonding lag. A positive amount is a **partial unbond**: the validator reduces its stake by that amount **while it keeps validating** — there is no separate transaction type. The withdrawn amount is moved out of the live stake (so committee power drops by exactly that amount) into a **pending-unbond entry** (an entry index, an amount, and

a release height) that can be withdrawn to an account only at/after that entry's own release height. A validator may hold **up to 16 concurrent pending withdrawals** — each a separate entry with its **own amount and release height, releasing on its own independent schedule** — so a user is never forced to wait the full 21-day unbonding interval before unstaking more; a new partial unbond is rejected **only** once the validator already holds 16 live entries (a state-bloat / dust-griefing cap no real user reaches, **not** a "one at a time" wait). The entire security claim is that partial unstaking is **provably no weaker than all-or-nothing unbonding plus its lags**, regardless of how many entries a validator parks. Each attack vector and its mitigation:

- **Pre-slash unstake evasion (the load-bearing invariant).** Could a validator dodge an impending deviation slash by partial-unbonding its stake into one or more entries the block before it is caught? **No** — **every** pending entry **stays slashable for the full unbonding lag**. The oracle-deviation slasher computes its forfeit on the base of (live stake + the sum of all live pending entries), takes it from live stake first, then **drains across the entries in deterministic entry-index order** until the forfeit is satisfied; each drained portion is routed to the reserve exactly like slashed stake (conservation preserved). With all-or-nothing unbonding the whole stake stays slashable through the lag; the full set of pending entries reproduces exactly that property, so partial unstaking is **not weaker** even when the stake is split across many entries. This is tested end-to-end (a seated validator with several live entries is slashed on its `stake + Σentries` base, the overflow drains across more than one entry in order, and a cluster-wide re-sync agrees).
- **Sortition grinding / stake-bouncing.** Could a validator rapidly partial-unbond and re-bond to churn power and re-roll sortition? **No.** A partial unbond stamps the validator's last-bonding height to the current height, which (i) re-arms the **sortition bond-lag** (no sortition for one bond interval — the same lag that stops fast committee activation, §5.6) and (ii) triggers an explicit **re-bond cooldown**: while **any** pending entry is live, a bond to that validator is rejected for one bond interval. The bounce loop is closed.
- **State bloat / parallel drains.** At most **16 live pending entries per validator** — a 17th concurrent partial unbond is rejected. State per validator is bounded by a small constant, the slasher's drain loop is bounded to that constant (O(1) per block), and a validator cannot open an unbounded number of slashable-but-shrinking entries.
- **Shortening the lag via the transaction.** Each entry's release height is derived from the **trusted block height** (current height plus the unbonding interval), never from the transaction's lock time, so an attacker cannot shorten the unbonding lag by crafting the tx.
- **Committee-power consistency.** A partial unbond is rejected for a current committee member or a next-height joiner in strict mode (mirroring bond/unbond), because mutating a seated member's stake mid-block would make the committee update panic on an inconsistent power. When a slash does touch a seated member, the power/committee caches are refreshed in lockstep with the store exactly as the deviation slasher does (§5.3), so a running node and a cold-reloaded one never fork.
- **Floor interactions.** The remaining stake must stay $\geq$ the laddered minimum stake and $\geq$ the genesis lock floor (§5.12), so partial unstaking can neither create a dust validator nor evade the lockup.
- **Conservation.** The stake decrement equals the pending-entry increment exactly; withdraw moves a matured entry $\rightarrow$ balance 1:1 (and never touches the validator's other entries); a slash moves entries $\rightarrow$ reserve 1:1. The invariant $\Sigma\,\text{stake} + \Sigma\,\text{pending} + \Sigma\,\text{balance} + \text{reserve}$ is conserved across the full lifecycle — across multiple concurrent entries, partial withdrawals, and slashing (golden-vector tested).

Each pending entry is its own leaf in a dedicated merkle subtree, keyed by (validator number, entry index) with a dense leaf index (modeled on the oracle-deviation subtree). The subtree is **empty until the first partial unbond**, so all new committed state is deterministic and resync-identical.

5.15 Under-minimum validators are excluded from sortition (recoverable)

The laddered minimum stake (§5.11) **ratchets up** as committed supply grows — cap/50, resolved per height from the same monotone-latched supply ladder as the cap and the perpetual tail (§3.2). A validator bonded above the minimum at one height can therefore fall **below** the minimum at a later height purely because the minimum rose past it (the minimum is generous, and ratchets only at supply-ladder rung crossings — years apart at the ~2% tail — so this is rare). When that happens the validator is **excluded from committee eligibility**: it is no longer eligible to join the committee via sortition.

This is a **recoverable EXCLUSION, not a slash and not a forced unbond.** The validator's bonded stake stays exactly where it is and remains unbond/withdraw-able under the normal rules (§5.14, §5.12); it simply does not *validate* while under-minimum. To regain eligibility it tops its bond back up to $\geq$ the current minimum, at which point sortition is accepted again. The rationale is **anti-dust / set-bloat safety**: without this gate, a rising minimum would leave a growing pile of sub-minimum validators in the active set — a Sybil/ bloat surface and a cheap way to dilute the committee — even though the minimum that admitted them has since moved on. It mirrors the **availability-exclusion pattern** of the oracle gate (§5.7): penalise by *recoverable exclusion*, never by an unrecoverable stake loss, when the validator has done nothing provably wrong.

Enforcement is at the **single sortition eligibility gate**: a sortition transaction whose validator's stake is below the height-resolved minimum is rejected on the authoritative propose/validate path. Because it reads the **same height-resolved minimum** (resolved from committed pre-block supply) that the rest of the staking system uses, the accept/reject decision is **identical at propose and validate**, and **deterministic across nodes and on a re-synced node**: commit re-executes the already-validated, certificate-attested block without re-running the strict gate (the strict eligibility check fires only on the propose/validate path), so it trusts what propose and validate already enforced. A validator **already seated** when the minimum ratchets past it stops re-winning eligible sortition and therefore **rotates out naturally** as fresher validators join, subject to the existing `<1/3`-power committee-change-per-block limits, so even a rung crossing that excludes several validators at once rotates them out gradually rather than breaking the committee invariants. A genesis-locked validator (§5.12) that fell under a far-future minimum is excluded from validating while its principal stays locked — the two rules compose without contradiction (the minimum is far below the genesis bond, so this is only a far-future edge case). The gate is **pure runtime eligibility logic — it touches no committed state.**

5.16 Oracle-participation reward — making attestation income-rational

§5.3 slashes a *bad* price but, by itself, pays *nothing* for a *good* one — so without a participation reward, a validator that does not care to propose would have **no income reason to attest at all**, and a banned deviator (§5.7) would keep **full income with zero oracle duty forever**; oracle liveness would degrade because attesting would never be individually rational. Anemos therefore **redistributes a small slice of the *existing* block reward to the validators that actually attest**, withheld from those that do not — revenue-neutral, no new transaction, no new per-block storage.

Per block the validator pool (the proposer's share after the reserve slice and the above-target diversion, $\geq \phi_{\min} R$) is re-partitioned: an **oracle slice** $\sigma = 10\%$ of that pool is carved out and the proposer keeps the rest. The slice is then distributed **EQUALLY among the block's correct attesters** — the committed oracle section's signers **minus** the deviators (§5.3), absentees, and the frozen/banned (§5.7). The would-be shares of the excluded are absorbed by the correct attesters; the integer-floor remainder (bounded dust) folds into the reserve, so the slice is fully paid out and conservation is exact ($\text{reserve} + \text{proposerPool} + \sum \text{shares} + \text{dust} = R(h)$ bit-for-bit). Because the slice lives *inside* the validator pool, the **$\phi_{\min}$ = 75% validator floor is untouched** (the attesters are validators too). A pool attester's share flows through the same split as its

proposer reward — operator commission plus the slash-immune delegator escrow — so delegators share it pro-rata and it can never be slashed.

EQUAL per attester, not power-weighted, is the load-bearing choice: oracle-subset membership is already stake-correlated (stake-weighted sortition into the committee, uniform hash-rank into the subset), so EQUAL yields realized oracle income that tracks stake with the *same* shape as the existing block reward (decentralization-neutral), whereas power-weighting would compound a *second* stake factor ($\approx$ stake-squared, centralizing). Attesting becomes the **dominant strategy**: a free-rider that proposes but never attests earns 0 of the slice while its honest peers absorb its share; a deviator earns 0 **and** is slashed; a banned validator forfeits the slice **and** — because it is now counted obligated-and-absent (§5.7) — loses proposing too. At $\sigma = 10\%$ the forfeited slice alone is an ongoing $\approx$1.3%/yr (NPV $\approx$ 12% of stake) on top of the one-time deviation slash, and with the proposing exclusion a ban costs roughly the validator's entire franchise. **Ban is never a free opt-out, and oracle participation is individually rational** — except under the bounded liveness escapes of §5.7, which deliberately trade a narrow amount of deterrence for guaranteed chain liveness: the exclusion cap never lets more than $\lfloor (n-1)/3 \rfloor$ committee members stay gated at once (the highest-scored gated members, including a low-scoring banned validator, stay eligible to propose by rank once the cap binds), and the full-rotation escape ignores the score gates entirely once a height has churned round $\geq n$ proposer rounds without a commit. So the deterrence guarantee is **probabilistic under contention, not absolute** — a banned validator can be re-admitted to propose under these bounded conditions, though it never recovers the forfeited oracle-reward slice or the underlying slash. **Fig. 28** shows the mechanism end-to-end: the slice's place in the block-reward split, the decentralization-neutrality of the equal split, the per-role payoffs (honest / free-rider / deviator / banned), and the cost of a ban.

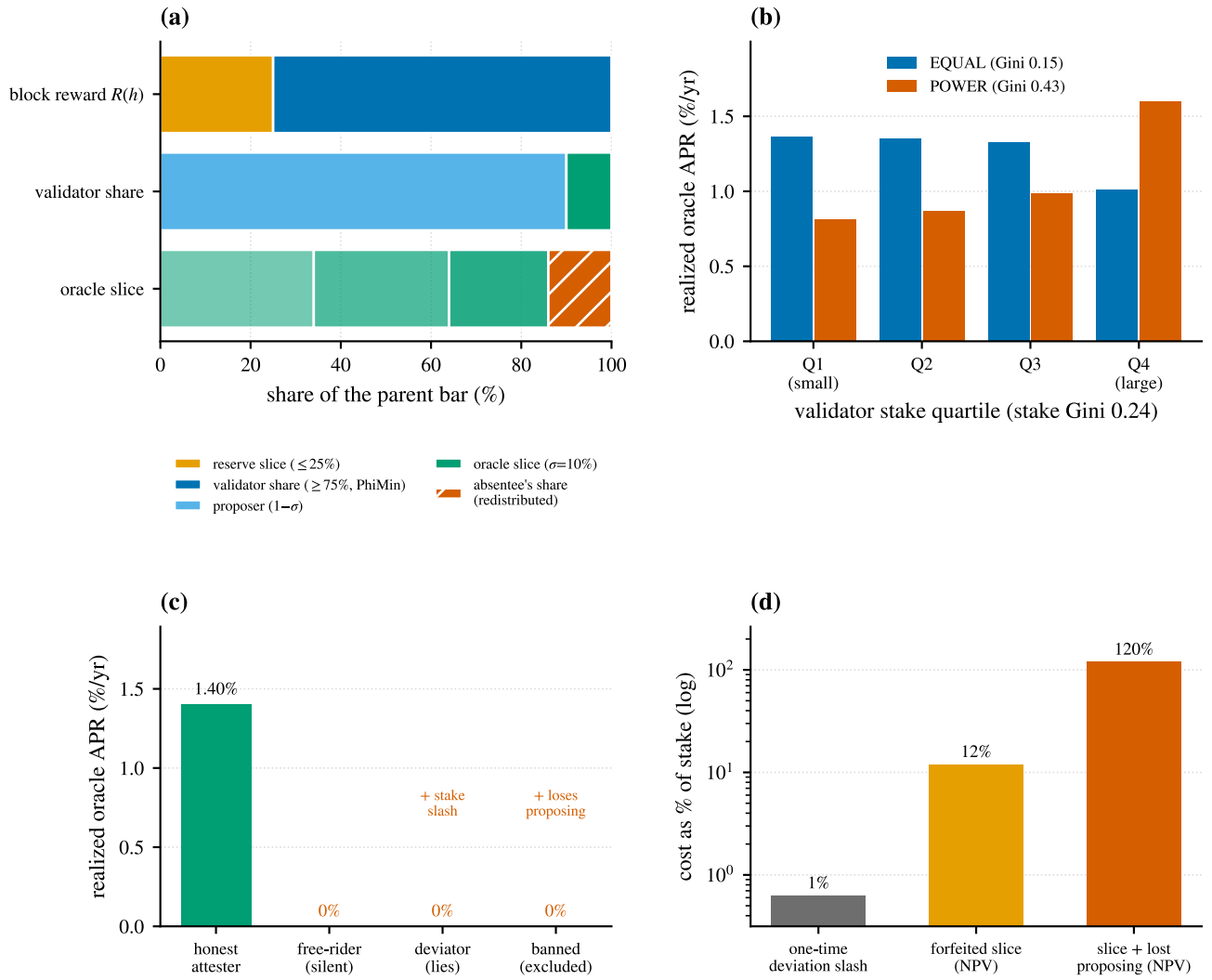

Fig. 28 — The oracle-participation reward: (a) the σ=10% slice carved from the validator pool and split equally among correct attesters; (b) equal-per-attester tracks stake like the block reward (power-weighting would compound stake-squared); (c) per-role realized oracle income — attesting is the dominant strategy; (d) the cost of a ban: the forfeited slice plus lost proposing dwarfs the one-time slash.

6 Determinism and Mathematical Foundations

6.1 Fixed-point scales (no floating point)

All economics use integer fixed-point with explicit scales, and every overflow-prone product uses arbitrary-precision integers internally:

Quantity	Symbol	Scale
Rebase index	I	$I_0 = 10^{18}$
Collateral ratio	$r, \bar{r}$	10^6
Oracle price	p	10^9
TWA smoothing	α	10^6
Fee / interest numerators	—	$10^6/10^9$

The reference rounding primitive is floor-division, $\mathsf{mulDivFloor}\,(a, b, c) = \lfloor ab/c \rfloor$ computed in big-integers, and **rounding always favours the system**: a redeemer never receives more than the burned stablecoin is worth, and a sub-unit redeem burns at least the value withdrawn via ceil-rounded shares (so dust can only ever accrue to the protocol, never drain it).

6.2 Conservation and overflow

The supply-conservation theorem (§3.4) is the global invariant. Locally, every mutation pairs a credit with a debit except the two intended emission mints. A defensive non-negativity guard fails *loud* (panics) rather than silently minting via an integer underflow, and every accumulator that could grow large (the cumulative emission, the rebase index, the share and recovery totals) is computed in big-integers and either saturates deterministically or refuses the operation rather than wrapping. The emission schedule is **float-free** and therefore byte-identical on amd64/arm64/arm32.

6.3 Golden vectors

A frozen golden test vector pins the serialized aggregate-state hash after a fixed operation sequence; any change that alters it is a deliberate protocol change, never an accidental drift. Cross-node determinism is a hard requirement: a state-root divergence is a fault.

---

7 Risk Analysis and Ruin Probability

Theorem 1 says ruin is *reachable*; the engineering question is *how probable* under realistic ANM dynamics. We model the ANM/USD price as a geometric Brownian motion $p_t = p_0 \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right)t + \sigma W_t \right)$ and treat insolvency as the absorbing event $r_t = r_0\, p_t/p_0 < 1$ over a one-year horizon, by Monte-Carlo (4000 paths, $\sigma = 90\%$/yr, a crypto-typical volatility). **Fig. 29** plots the one-year ruin probability against annualised drift for three starting ratios:

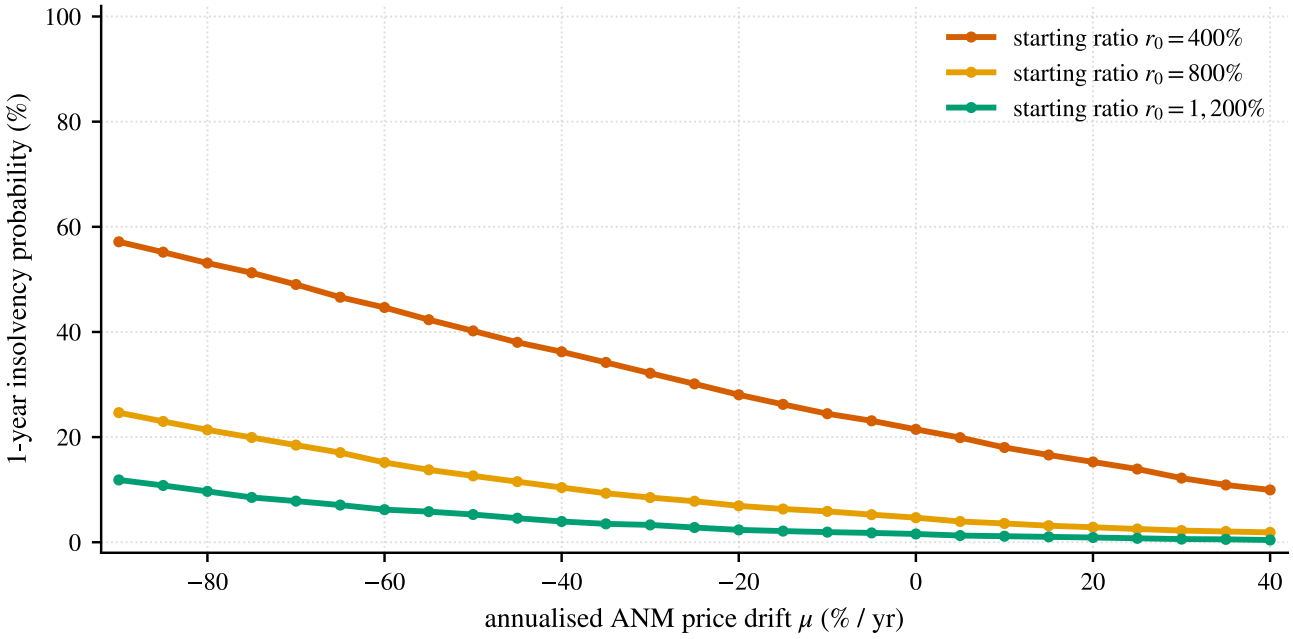

Fig. 29 — Ruin probability

The qualitative result matches the bounded-safety analysis: with a healthy starting buffer ($r_0 \geq 800\%$) ruin is negligible specifically near $\boldsymbol{\mu \approx 0}$ (flat-to-mildly-adverse drift); it is **not** negligible across the whole "non-catastrophic" range some readers might assume — at $r_0 = 800\%$ under a sustained, plausible (not exotic) negative drift of roughly −30% to −50%/yr, the one-year Monte-Carlo ruin probability rises to the **single-digit-to-low-teens percent** range, materially above zero though still far from certain. Under a severe, sustained decline ($\mu \ll 0$) ruin approaches certainty for *every* finite buffer — exactly Theorem 1. Read **Fig. 29**'s actual curve for the drift you consider plausible rather than relying on a single adjective. Over-collateralisation buys **time and probability**, not impossibility. This is why Anemos pairs a high floor with *graceful* failure modes — restructuring then orderly wind-down — rather than promising a peg it cannot guarantee.

Beyond the closed-form price model, a deterministic full-protocol simulation drives the entire module — mint/redeem flow, the emission slice, interest, both circuit breakers, restructuring, and wind-down — through benign, crash, demand-shock, oracle-manipulation, and combined worst-case scenarios over one-to-ten-year horizons. **Fig. 30** summarises the outcome matrix: the system holds peg and survives every benign and single-stress scenario and restructures and recovers through the engineered depeg; the combined worst-case and demand-collapse scenarios instead drain the senior book toward near-total collapse via mass redemption (a solvency-preserving but severe outcome — every redeemer is still paid pro-rata, §4.7) without necessarily entering the formal terminal wind-down mode within the simulated horizon — see the honest-limits discussion above and §4.6's Theorem 1, which is what guarantees terminal entry remains reachable given a long enough or deeper adverse path.

	survived	held peg	min ratio	snr coll%	jnr EV% (total)	jnr CAGR %/yr	jnr cash APR%	snr APR%	mode reached
steady flat	yes	yes	7.22	0	910	58.8	360.1	0.00	normal
bull goldenage	yes	yes	5.34	0	2480	91.6	617.1	0.27	normal
decade steady	yes	yes	7.72	0	6096	51.1	191.6	0.31	normal
usd demand shock	yes	yes	3.71	0	1380	1379.9	10000.0	0.00	normal
flash crash recover	yes	yes	2.77	0	322	321.6	2048.4	0.08	normal
oracle manipulation	yes	yes	8.89	0	3022	3021.8	496.5	0.36	normal
bear slow bleed	yes	yes	5.10	0	-61	-17.2	1783.3	0.00	normal
reflexive distress	yes	yes	3.44	19	497	496.8	10000.0	0.15	normal
depeg restructure terminal	yes	no	1.00	84	-100	-99.8	10000.0	0.00	restruct
demand collapse run	no	yes	9.10	100	2098	2098.4	428.7	4.12	normal
prolonged depression	no	yes	4.32	100	-88	-34.1	10000.0	0.81	normal
combined worst case	no	yes	3.53	100	158	157.7	10000.0	0.00	normal

green=holds / red=fails; survived := no terminal, no insolvency, senior book not wiped (<90% collapse); held peg := no haircut / no write-down

jnr EV% = TOTAL mark-to-market return over the run (NOT an annual rate); jnr CAGR%/yr (paper appreciation) · jnr cash APR% (withdrawable floor-yield) · snr APR% (senior, capped 16%/yr) are ANNUALIZED. Junior is uncapped first-loss equity, so it earns far more than senior by design -- not a shortfall.

Fig. 30 — Cross-scenario outcome matrix from the deterministic multi-year full-protocol simulation: survival, peg integrity, minimum ratio, senior collapse, junior return (total and annualised), senior APR, and the mode reached, across benign, stress, manipulation, and combined worst-case scenarios. In this simulation run, the combined worst-case and demand-collapse scenarios drain the senior book to ~100% collapse via mass redemption without the sim ever recording `mode == terminal`*; restructuring (recovery tokens, §4.8) triggers in the engineered depeg scenario. Treat these as one seeded sample of a stochastic model, not an exhaustive proof that terminal mode is unreachable under worse or longer-horizon paths.*

8 Comparison to Prior Systems

Anemos sits in a well-populated design space, and it is most honestly understood by where it agrees and disagrees with each of its peers. We compare against seven reference designs spanning the three dominant families of decentralized stablecoin: the **over-collateralized CDP** line (MakerDAO/Sky's Dai [9], Liquity's LUSD [14], and the non-pegged outlier RAI [15]); the **dual-tranche reserve-coin** line that Anemos belongs to (Djed/Shen on Cardano [8] and Zephyr on a Monero base [16]); the **fractional-algorithmic** line (Frax [17]); and the **pure-seigniorage** failure case (Terra/UST [10]). Two tables summarise the axes that matter for soundness — the first sets Anemos against the native-coin designs it descends from (and the pure-seigniorage endpoint of that line); the second against the externally-collateralized CDP and fractional designs. The paragraphs below add the per-system detail.

Anemos vs the native-coin line — the dual-tranche peers and the seigniorage endpoint.

Property	Anemos (ANM)	Djed/Shen [8]	Zephyr (ZSD) [16]	Terra/UST [10]
Design & venue	**Native dual-tranche module in consensus** (no VM)	Dual-tranche; Plutus contracts (Cardano)	Dual-tranche; in consensus (Monero fork)	Pure seigniorage; Cosmos-SDK module
Collateral & backing	**Native ANM, over-coll.** — honest, reflexive native-cap backing	ADA (host coin) — host-coin cap	ZEPH (host coin) — host-coin cap	**None** (LUNA mint/burn) — backed by nothing
Peg target	\$1, soft (price TWA)	\$1	\$1	\$1
Min collateral ratio	400% mint floor; 800–1200% bands	400–800% band	≥ 400% at mint	0% (uncollateralized)
Anti-spiral rule	Mint-floor + net circuit breaker	Mint floor (formally verified)	Reserve-ratio bounds	— (reflexive by design)
Reserve funded by	**Three on-chain sources** (below) — perpetual-emission self-funding	Mint/redeem **fees only** (usage-dependent)	Fees + block emission (partly self-funding)	—
Holder yield	**Surplus-only, self-throttling rebase**	none	none (ZYS is separate)	**Fixed 20%** (Anchor) — fatal
Oracle	**Consensus-embedded subset median, deviation-slashed, ~3.5h price TWA**	external feed	external feed	external (manipulated in collapse)
Failure mode	**Restructuring → orderly pro-rata wind-down**	mint halt	mint halt	**death spiral**

Anemos vs the externally-collateralized designs — the CDP line and the fractional-algorithmic line.

Property	Anemos (ANM)	MakerDAO/ Sky Dai [9]	Liquity LUSD [14]	RAI [15]	Frax [17]
Design & venue	**Native dual-tranche module in consensus** (no VM)	Multi-collateral CDP; EVM contracts	ETH CDP; EVM contracts	ETH CDP (non-pegged); EVM contracts	Fractional-algorithmic; EVM contracts

Property	Anemos (ANM)	MakerDAO/ Sky Dai [9]	Liquity LUSD [14]	RAI [15]	Frax [17]
Collateral & backing	**Native ANM, over-coll.** — honest, reflexive native-cap backing	ETH, wstETH, **USDC, RWAs** — mixed, incl. off-chain	ETH (single asset) — ETH market cap	ETH — ETH market cap	USDC + FXS → 100% / T-bills — off-chain
Peg target	$1, soft (price TWA)	$1	$1, hard (redeem arb)	**Floating** (PID, not $1)	$1
Min collateral ratio	400% mint floor; 800–1200% bands	per-vault liq. threshold	**110%** (capital-efficient)	~145% min	100% (≈1:1)
Anti-spiral rule	Mint-floor + net circuit breaker	Liquidation + DSR/PSM	Liquidation + Stability Pool	PID damps, liquidation	Collateral ratio + AMO
Reserve funded by	**Three on-chain sources** (below) — perpetual-emission self-funding	Stability fees, RWA/T-bill yield, PSM (revenue-funded)	**Stability-Pool depositors** (external LUSD)	Borrower fees (SF)	AMO DeFi yield + collateral
Holder yield	**Surplus-only, self-throttling rebase**	**DSR** (external T-bill/fee income)	none	none (redemption-rate only)	sFRAX (external)
Oracle	**Consensus-embedded subset median, deviation-slashed, ~3.5h price TWA**	external (Maker oracle module)	external (Chainlink + Tellor)	external (med + delay)	external
Failure mode	**Restructuring → orderly pro-rata winddown**	vault liquidation / global settlement	recovery mode / redemptions	re-leverage via rate	de-risk to collateral

The dual-tranche peers (Djed/Shen, Zephyr). These are Anemos's closest relatives: a senior peg coin and a junior reserve/equity coin sharing one over-collateralized reserve, with a 400–800% ratio band and a formally-verified mint-floor that halts new senior issuance below the floor — the exact anti-death-spiral bound Anemos adopts (§4.2). Djed/Shen [8] runs as Plutus contracts on Cardano and funds its reserve **only from mint/redeem fees**, so the buffer grows only with usage. Zephyr [16] ports the Minimal-Djed design into a Monero fork and is the one prior system that, like Anemos, **also taps block emission** to grow the reserve; it has held its peg through severe ZEPH volatility (a 52× rally and a subsequent ~90% drawdown). Anemos's advances over this line are (i) a **richer, multi-source self-funding reserve** (below), (ii) a **native consensus-embedded, deviation-slashed oracle** rather than an external feed, and (iii) a **recovery-token restructuring layer** (§4.8) that converts a mild insolvency into a deferred, fungible, buy-back-over-time claim instead of an immediate mint-halt.

The CDP line (Dai, LUSD, RAI). MakerDAO/Sky's Dai [9] is the over-collateralized baseline, but it has migrated toward **off-chain backing** — USDC via the Peg-Stability-Module and tokenized US Treasuries dominate the balance sheet, and its Dai Savings Rate is paid out of that external (stability-fee + T-bill) income. Liquity's LUSD [14] is the purest on-chain CDP: ETH-only, interest-free, with an aggressive **110% minimum** ratio made safe by a Stability Pool of LUSD depositors who absorb liquidations and a hard redemption arbitrage that pins $1 — but it has **no protocol reserve and pays holders no yield**, and

its backstop is external depositor capital rather than protocol-owned. RAI [15] is the instructive outlier: it **abandons the $1 peg entirely**, letting an on-chain PID controller float a redemption price — the closest precedent for Anemos's *control-law* monetary policy (the §3.5 variable subsidy and the ratio-TWA gates), though Anemos keeps a $1 target rather than floating it. Against all three, Anemos's distinguishing choices are native (no-VM, deterministic, O(1)) execution, a **protocol-owned self-funding reserve**, and a senior/junior tranche structure that none of the single-token CDPs have.

Anemos's reserve has more than one income source. Unlike Djed (fees only) or the CDPs (no protocol reserve at all), the Anemos reserve is **self-funding from three independent on-chain streams**, all conservation-preserving and all routed by the same collateral-ratio control law: (1) the **health-dependent block-emission slice** — up to ~25% of every block's perpetual-tail reward in the comfort band, tapering to zero as the chain weakens (§3.5), the structural source that makes the reserve grow with the chain's own security budget rather than with user demand; (2) **forfeited stake from oracle-deviation slashing** — a deviator's penalty is *moved into the reserve it endangered* (§5.3), not burned, so attacks on the price feed recapitalize the very buffer they target; and (3) **mint/redeem activity and reserve growth**, the surplus that funds golden-age interest (§4.3) and recovery-token buyback (§4.8). Because the dominant stream (1) is the perpetual, supply-indexed emission tail (§3.2), the reserve is **self-funding for the life of the chain** — it does not depend on usage fees, an external treasury, or off-chain RWA income to stay capitalized, which is the structural difference from every fee- or revenue-funded design in the tables.

The honest cost (Theorem 1). The flip side of native backing is **reflexivity**: the reserve is denominated in ANM, so its USD value falls with ANM's price (§4.6). Djed and Zephyr share this — they back a senior coin with their own host coin — whereas Dai, LUSD, and Frax dilute or remove it by holding a *different* asset (ETH, USDC, T-bills). We are explicit that native backing is the weaker solvency assumption and engineer for it: a high mint floor, a counter-cyclical emission slice, and the two-stage graceful-failure path. Terra/UST [10] is the cautionary endpoint of abandoning collateral entirely — a reflexive mint-and-burn with **no** reserve and a fatal fixed-20% Anchor yield that guaranteed the death spiral. Anemos shares Terra's *only* property of being self-bootstrapping from a native coin while differing on every load-bearing axis: it is over-collateralized, formally mint-floored, surplus-only in its yield, and it fails *gracefully* rather than catastrophically.

---

9 Conclusion

Anemos integrates an over-collateralised dual-tranche stablecoin into a deterministic instant-final BFT chain, funded by a fair, no-premine, perpetually-tailed emission that mints supply on every block and routes a health-dependent slice to the reserve under a hard validator floor, and priced by a committee oracle that slashes provable price deviation but never penalises absence (only a recoverable exclusion), fed by an off-chain multi-source feeder. We proved per-block conservation of the native unit, gave the fixed-point determinism rules that keep the module fork-safe and Android-viable, and stated honestly that no native-backed peg can be unconditionally solvent (Theorem 1) — handling the unavoidable failure with an intermediate recovery-token restructuring and, beyond it, an orderly pro-rata wind-down whose rarity we quantify.

The on-chain oracle aggregation, the variable subsidy, the restructuring mode, and the off-chain feeder are exercised by extensive deterministic and adversarial testing; deviation slashing applies a real stake penalty and **escalates to a graduated freeze and, for egregious repeat deviation, a permanent dynamic oracle ban** (§5.3) that drops a persistent deviator's price from aggregation — all re-sync-deterministic. The module parameters are frozen into genesis and validated at startup. Cold-start safety is enforced by the two-step launch of §4.9: minting is provably inert until the reserve is bootstrapped to the target (mainnet 600,000 ANM / testnet 75,000 ANM) and the latching bootstrap-done flag flips.

10 References

The bibliography is rendered below by the PDF build; the in-text bracketed markers above are the corresponding citations.

[1] Pactus Project, "Pactus: A Lightweight, Sortition-PoS, Instant-Finality Blockchain." [Online]. Available: https://pactus.org/

[2] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine Agreements for Cryptocurrencies," in *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP)*, ACM, 2017, pp. 51–68.

[3] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol," *Advances in Cryptology — CRYPTO 2017*, vol. 10401, pp. 357–388, 2017.

[4] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI)*, USENIX Association, 1999, pp. 173–186.

[5] E. Buchman, "Tendermint: Byzantine Fault Tolerance in the Age of Blockchains," 2016.

[6] Kaspa community, "Kaspa: Chromatic Halving and the Deflationary Emission Schedule." [Online]. Available: https://kaspa.org/

[7] Y. Sompolinsky, S. Wyborski, and A. Zohar, "PHANTOM GHOSTDAG: A Scalable Generalization of Nakamoto Consensus." [Online]. Available: https://eprint.iacr.org/2018/104

[8] J. Zahnentferner, D. Kaidalov, J.-F. Etienne, and J. Díaz, "Djed: A Formally Verified Crypto-Backed Pegged Algorithmic Stablecoin." [Online]. Available: https://eprint.iacr.org/2021/1069

[9] MakerDAO, "The Dai Stablecoin System." [Online]. Available: https://makerdao.com/whitepaper

[10] Various, "The Fall of Terra/UST: Post-Mortems of the May 2022 Collapse." 2022.

[11] V. Buterin, "SchellingCoin: A Minimal-Trust Universal Data Feed." [Online]. Available: https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal-data-feed

[12] L. Breidenbach *et al.*, "Chainlink 2.0: Next Steps in the Evolution of Decentralized Oracle Networks," 2021. [Online]. Available: https://chain.link/whitepaper

[13] G. Angeris and T. Chitra, "Improved Price Oracles: Constant Function Market Makers," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies (AFT)*, ACM, 2020, pp. 80–91.

[14] Liquity, "Liquity: Decentralized Borrowing Protocol (LUSD)." [Online]. Available: https://docs.liquity.org/

[15] Reflexer Labs, "RAI: A Non-Pegged, ETH-Backed Stable Asset with a PID-Controlled Redemption Price." [Online]. Available: https://docs.reflexer.finance/

[16] Zephyr Protocol, "Zephyr Protocol: An Over-Collateralized, Algorithmic Stablecoin Built on a Private Reserve." [Online]. Available: https://www.zephyrprotocol.com/

[17] Frax Finance, "Frax: A Fractional-Algorithmic Stablecoin Protocol." [Online]. Available: https://docs.frax.finance/

Appendix A — Launch parameters

Parameter	Symbol	Launch value
Genesis holder allocation	—	8,000,000 ANM (1,000 × 8,000)
Mainnet genesis validators (auto-bonded)	—	4 × 8,000 = 32,000 ANM (additive)
Mainnet genesis total supply	S_{gen}	**8,032,000 ANM** (8,000,000 holders + 4 × 8,000 validators)
Initial block reward	R_0	2 ANM
Perpetual tail (supply-indexed)	R_∞	ladder $(\tau S)/B_{\text{year}}$, $\tau = 2\%/\text{yr}$
Tail inflation fraction	τ	2 % of supply / yr
Blocks per year	B_{year}	3,153,600
Monthly decay	ρ_{dec}	$4053909305/2^{32}$ ($\approx (1/2)^{1/12}$)
Blocks per month	B_{month}	259,200
Validator max stake (laddered)	—	ladder $(0.008\,S)$ — launch **60,000 ANM**
Validator min stake	—	max-stake / 50 — launch **1,200 ANM** (mainnet) / **800 ANM** (testnet)
Pool min delegation (global floor)	—	500 ANM (`PoolMinDelegation`; §5.11, §5.11.1)
Operator min self-bond	—	2,000 ANM (`OperatorMinSelfBond`; above both floors; §5.11)
Operator commission cap	—	2000 bps = 20 % (`OperatorCommissionCapBps`; §5.11)
Commission-increase notice	—	~21 d ≈ `UnbondInterval` (`CommissionIncreaseCooldown`, anti-rug; §5.11)
Default / max pool delegators	—	32 (`default_pool_max_delegators`; hard ceiling `MaxPoolMaxDelegators` = 32; §5.11.1)
Bond interval	—	360 blocks ≈ 1 h (`BondInterval`; the bonding rate-limit/lag, §5.6)
Unbond interval	—	181,440 blocks ≈ 21 d (`UnbondInterval`; the pending-unbond release delay, §5.11, §5.14)
Reserve bootstrap target (mainnet / testnet)	—	600,000 / 75,000 ANM
Reserve slice (comfort band)	s_{base}	25 %
Validator-share floor	$\phi_{\min}$	75 %
Slice taper window	—	200 % – 800 % ratio TWA
Mint collateral floor	$r_{\min}$	400 %
Golden-age target (interest)	$r^{(\bar{\ })\star}$	1200 %
Max-collateral (anti-hype, senior emission/interest target)	$r_{\max}$	1200 %

Parameter	Symbol	Launch value
Junior equity-mint cap (decoupled)	—	2000 % (`JuniorMaxCollateralRaw`; §4.1, §4.9)
Circuit-breaker net cap	c	5 % / **block** (per-period ≡ per-block; §4.5)
Circuit-breaker absolute floor	$c_{\min}$	$100,000 / block (§4.5)
Rolling-window redeem cap	—	20 % of supply / ~1-day (8,640-block) trailing window; $100,000 floor (§4.5)
Mint fee	—	0.2 % (`MintFeeNum`; §4.2)
Redeem fee	—	0.3 % (`RedeemFeeNum`; §4.2)
Senior interest cap (max APR, smoothed)	κ_{yr}	**1600 bps/yr = 16 % max APR** (`InterestCapAnnualBps`; annual, ÷ B_{year} → per-block; accrue-to-pool → steady drain; hard bound 20 %; §4.1, §4.3)
Above-target emission diversion (senior)	—	**25 %** of the block reward above the 1200 % target (§3.5, §4.1)
Junior cash yield	—	**none** (junior earns residual-equity appreciation only; §4.1)
Senior fee-rebase share	—	50 % of mint/redeem fees → the senior index (`SeniorFeeShareNum`; §4.3)
Interest tick	—	10^{-9} of supply (finest distributable quantum)
Collateral-ratio TWA	α_r	1/100 (~17 min; emission-slice taper, §3.5)
Oracle price TWA	α_p	~1/1250 (~3.5 h; §5.4)
Committee term limit	—	~2,880 blocks ≈ 8 h (rotation compensator for the price TWA; §5.6)
Committee size	C	75 (mainnet) / 11 (testnet)
BFT consensus quorum	—	$> \frac{2}{3}$ of voting **power** (§5.2)
Oracle subset size	$\lvert \text{subset} \rvert$	11
Oracle subset quorum	q	$\lfloor 2 \mid \text{subset} \mid /3 \rfloor + 1 = 8$ of 11
Oracle tolerance	θ	10 %
Oracle per-period move cap	δ	10 %
Oracle staleness halt	—	6 quorum-less periods (≈ 1 min) → mint/redeem halt (`OracleMaxStale`; §5.4)
Pruning retention floor	—	default / minimum 10 days ≈ 86,400 blocks (`RetentionDays`; §5.10)
Oracle-participation reward slice	σ	10 % of the validator pool, equal-split to the block's correct attesters (§3.4, §5.16)
Oracle deviation base slash	—	1 bp
Deviation repeat-escalation cap	—	escalation level capped at 6 (tier-1 base ladder 1 → 64 bp)

Parameter	Symbol	Launch value
Deviation magnitude tiers	—	tier 2 at $> 2\times$ the tolerance band ($> 20\%$), tier 3 at $> 5\times$ ($> 50\%$)
Deviation magnitude shifts	—	$\times 1$ / $\times 8$ / $\times 64$ (tier 1 / 2 / 3)
Deviation per-tier slash ceilings	—	0.64 % / 6 % / 25 % per period
Deviation hard per-period cap	—	25 % (validation bound $\leq$ 50 %)
Deviation first-offence grace	—	a first small (tier-1, count-0) deviation costs 0
Egregious count increment	—	2 (the ban threshold 6 is reached in 3 egregious deviations)
Deviation escalation half-life	—	1000 periods
Oracle freeze threshold	—	count ≥ 3
Oracle freeze base / ladder	—	200 periods $\times (c-2)$
Oracle dynamic-ban threshold	—	count ≥ 6
Oracle tier sizes	—	$\{11, 17, 25,$ whole committee$\}$
Min oracle-availability score	—	0.5
Pre-terminal guard (soft)	—	110 % (`PreTerminalRaw` ; a committed mode flag, gates nothing by itself; §4.7)
Wind-down debounce	k	6 periods
Restructuring writedown floor	$W_{\min}$	5 %
Recovery buyback buffer	—	120 %

B_{year} uses a 365-day year and B_{month} a 30-day month; they index independent schedules (supply-indexed tail inflation vs the geometric decay), so $B_{\text{year}} = 12\,B_{\text{month}}$ need not hold ($3,153,600 \neq 12 \times 259,200$).

Appendix B — Notation

h block height · m month index · $R(\cdot)$ per-block reward · s total stablecoin carve-out (reserve slice + above-target diversion) · $\phi = 1 - s$ validator share · X reserve (ANM) · L stablecoin liabilities (USD) · p ANM/USD price · $r, \bar{r}$ collateral ratio and its TWA · I rebase index · W senior writedown multiplier · F, Φ recovery face and buyback fund · ρ wind-down recovery factor (the emission geometric-decay factor is the distinct subscripted ρ_{dec}, §3.2) · θ oracle tolerance · δ oracle move cap · C committee size · q oracle subset head-count quorum · κ senior interest cap (16 %, smoothed) · c breaker cap · α_p price-TWA factor ($\approx 1/1250$, ~3.5 h) · α_r collateral-ratio TWA factor (1/100, ~17 min) — two distinct averages, see §3.5/§5.4. $\phi_{\min} = 0.75$ validator-share floor.

Anemos is a respectful fork of Pactus. We keep Pactus's name on Pactus's work and claim only our own additions. Nothing here is financial advice.