Asymptote Fault Tolerance (AFT): Universally Breaking the Lower Bound with Relay-Free Deterministic 99% Byzantine Fault Tolerance

IOI

March 17, 2026

Abstract

Asymptote Fault Tolerance (AFT) is a family of consensus and settlement constructions for high-throughput ordering, guardian-backed base finality, proof-carrying canonical ordering, and deterministic release of irreversible effects. Its central move is to stop treating dense positive voting as the only possible source of ordering truth. Instead, AFT separates throughput-critical publication and chain progression from authority-critical proof verification, omission detection, and effect collapse.

The protocol family has four interacting surfaces. *GuardianMajority* supplies the production base-finality path: a validator quorum certificate is necessary but not sufficient, because each proposal must also carry a registered guardian committee certificate anchored into shared evidence. *CanonicalOrdering* supplies the proof-carrying ordering path: a slot order is accepted because a canonical certificate is uniquely valid and recoverable from public inputs, not because a dense validator quorum voted the order into existence. *Asymptote* supplies the sealing path for irreversible effects: `BaseFinal` chain progress is upgraded to `SealedFinal` through deterministic witness or observer-backed collapse, with canonical observer challenges dominating unsafe releases. *NestedGuardian* supplies a witness-augmented mode for layered threshold constructions.

This paper is self-contained and normative for the protocol family. Its ordering subtheorem remains `99%` equal-authority ordering consensus under explicit bulletin-publication, cutoff, omission, protocol-native retrievability, and proof-soundness assumptions. The runtime also internalizes compact signed publication-frontier metadata on the hot path: positive committed blocks carry a predecessor-linked frontier summary and compact availability-receipt binding, while same-slot conflicts or stale parent links admit short objective contradiction objects. Those frontier objects bind the claimed bulletin / ordering surface, while the protocol-native retrievability plane internalizes deterministic extraction-or-abort on the cold path. The repository's theorem surface is singular: AFT is `relay-free, coordinator-free, pure-software deterministic 99% Byzantine Tolerance`. For each closed slot boundary there exists at most one admissible public execution object, every conflicting fully specified candidate admits a short public contradiction witness, durable state advances only through canonical collapse, and sealed-effect release is bound to the same collapse surface. Compact bindings stay on the hot path while cold-path recovery / reveal / reconstruction re-materialize the same close, abort, collapse, replay-prefix, execution-restart, and restart-ancestry surfaces through the AFT-native recovery-family contract, with restart able to stream recovered certified ancestry until target height or recovered-history exhaustion without a theorem-relevant fixed depth bound. Ordinary canonical collapse / replay history and the AFT recovered-state contract also carry the historical-retrievability bootstrap surface used for deeper paging, so recovered history is ordinary endogenous AFT history rather than a theorem-side qualifier. This is a categorical full-stack lower-bound break that explicitly claims AFT universally breaks the lower bound with no qualifiers, and a real repository claim of relay-free, coordinator-free, pure-software deterministic `99% Byzantine Tolerance`. It is therefore also an honest claim of unconditional classical `99% Byzantine agreement` in the ordinary dense-vote permissioned model. The proof-carrying public-evidence architecture, endogenous historical retrievability, and collapse-gated durability are the realizing mechanism of that same classical sentence, not a residual qualifier. Unlike classical partially synchronous Byzantine agreement, which is usually presented through dense positive quorum intersection and $> 2/3$ honest-participation thresholds, AFT derives safety without trusted relays, privileged coordinators, TEEs, or dense positive quorum intersection. Safety comes from proof-carrying public evidence, dominant negative proofs, compact signed frontier consistency, explicit endogenous retrievability of the closed slot surface, and collapse-gated durability; accountable publication and validator removal remain operational reinforcement, not the source of correctness.

1 Scope and Status

This paper contains the full architectural and theorem-level description of the instantiated AFT family. It does not require any companion prose specification to be understood.

The instantiated protocol surfaces are:

- `GuardianMajority` as the production guardian-backed base-finality mode.
- `Asymptote` as the two-tier sealing and irreversible-effects mode.
- `CanonicalOrdering` through the live `CommittedSurfaceV1` certificate path plus compact `PublicationFrontier` summaries attached to committed blocks.
- `NestedGuardian` as an experimental witness-augmented mode with safety proofs, bounded operational checks, and a finished composed-liveness / classical-agreement bridge rather than a remaining composed-liveness kernel.

Implementation correspondence is included below as exact symbol references. That section is non-normative. The protocol and theorem content of the paper stands on its own.

The strongest theorem surface of the runtime is singular. AFT claims not only `99%` equal-authority ordering for the ordering layer, but relay-free, coordinator-free, pure-software deterministic `99% Byzantine Tolerance` for durable ordering and sealed-effect safety over the explicit public-state-continuity substrate. Compact hot-path bindings plus cold-path recovery / reveal / reconstruction materialize the same recovered close-or-abort, collapse, replay-prefix, execution-restart, and streamed restart-ancestry surfaces through the AFT-native recovery-family contract. Compact publication/frontier metadata is internalized on the live path throughout, and deeper recovered history is now ordinary endogenous AFT history through the canonical-collapse / replay retrievability anchor and the matching recovered-state historical retrievability surface.

Singular theorem sentence. The theorem surface of this repository is now singular all the way up: AFT claims relay-free, coordinator-free, pure-software deterministic `99% Byzantine Tolerance`, explicitly claims it universally breaks the lower bound with no qualifiers, and claims unconditional classical `99% Byzantine agreement` in the ordinary dense-vote permissioned model. The classical sentence and the AFT sentence are treated as extensionally the same claim. Proof-carrying public evidence, endogenous historical retrievability, restart continuity, and collapse-gated durability are the realizing architecture of that claim, not a remaining semantic delta.

Dimension	AFT realizing surface	Promoted singular claim
Agreement object	canonical close-or-abort / canonical collapse surface	ordinary classical agreement object with no architecture-specific restatement
Safety carrier	compact commitments, short contradiction objects, cold-path recovery, endogenous historical retrievability	unconditional classical agreement sentence with no separate evidence-model caveat
History / bootstrap	canonical-collapse / replay historical retrievability root plus recovered-state historical retrievability surface	no semantic distinction between AFT retrievability history and ordinary agreement history
Liveness burden	composed recurrence/reduction/totality/collapse chain plus persistent churn simulator	one completed classical-agreement-style theorem surface under the target adversary model

Target adversary/scheduler model and discharged liveness kernel. The singular theorem is interpreted under one explicit liveness model rather than a vague aspiration. Byzantine authorities may equivocate, omit, reorder, withhold reveals, withhold historical-retrievability objects, rotate maliciously, and crash/restart arbitrarily. Before stabilization the scheduler may delay or interleave publication, registry, recovery, archival, and restart traffic arbitrarily. After stabilization the scheduler is assumed only to be eventually fair for the AFT public-state substrate: if an honest participant keeps reissuing an admissible object that the protocol depends on, at least one copy is eventually delivered, persisted, and fetchable by content hash, and governing profile/activation churn quiesces into bounded windows long enough for progress to compose. Under that model, the liveness bridge is the conjunction of five obligations: frontier-generation progress, canonical-resolution progress, recovery-completion progress, restart/historical-retrievability re-entry progress, and infinite composition of those four obligations across reassignment, outage, profile rotation, and archival page boundaries. That kernel is no longer open; it is the discharged bridge that supports the promoted singular classical-agreement sentence.

Current artifact map and completion status. That kernel is now grounded in concrete repository artifacts rather than prose alone. Frontier-generation progress lands on the live `PublicationFrontier` path and its contradiction objects; canonical-resolution progress lands on canonical-order certificates, `CanonicalCollapseObject`, omission dominance, and recursive continuity; recovery-completion progress lands on the coded recovery-family contract and the recovered publication / close-or-abort surface; and restart/historical-retrievability progress lands on the AFT recovered-state surface plus canonical-history retrievability anchors and bounded-memory paging. Those obligations now have both concrete runtime carriers and a completed formal bridge. The formal package contains bounded churn witnesses, recurring cycle cores, a recurrence theorem, a finite reduction, a total classical- agreement history bridge, and a directly discharged semantic-collapse wrapper ending in `NestedGuardianRecoveryClassicalAgreementCollapse.tla`. The runtime mirrors the same story through a persistent historical-retrievability churn/restart simulator over one evolving recovered-history state. The remaining work is therefore no longer theorem completion, but cleanup, stress expansion, and maintenance of the promoted singular claim.

2 Problem Statement

Classical permissioned Byzantine fault tolerance ties deterministic safety to dense positive voting by a large honest majority. Under that model, honest inclusion, total-order agreement, and finality all depend on most validators remaining both correct and available. That is the right model when the protocol is allowed to pay dense coordination cost and accept classical $3f + 1$-style thresholds. It is the wrong model for a system that wants all of the following simultaneously:

- high-throughput publication and chain progression,
- equal-authority participation at the validator layer,
- deterministic rejection of incomplete or conflicting ordered views,
- stronger accountability than "someone must have voted wrong,"
- deterministic gating of irreversible effects.

AFT therefore changes the object of agreement.
Instead of asking:

- "Which order did the validator quorum choose?"

it asks:

- "Which order certificate is uniquely valid for this slot, and which effects are admissible after deterministic collapse?"

The family-level design goals are:

- keep ordering and publication sparse on the hot path,
- keep effect release fail-closed,
- make omissions and conflicts objectively provable,
- keep all validators equally eligible to reveal the canonical order,
- move authority from dense yes-votes to verifiable evidence.

3 Thought Experiment: The Port of Public Manifests and Release Seals

Imagine an international port that handles ordinary cargo and hazardous cargo.

Every ship that wants to unload during tide h must post its cargo manifest to a public harbor board before the closing bell τ_h. The bell is not a private clock in the harbor master's office. It is a certified public event. Once the bell rings, the harbor office takes every admissible manifest, applies the published tide lottery R_h, and derives one canonical docking docket O_h.

The office does not ask every captain to vote on the exact order of every container. It asks a simpler question: has someone produced the one valid docket certificate for the manifests that were publicly posted before the bell? If any dockworker can prove that an admissible manifest was omitted from the announced docket, that docket is rejected immediately.

Ordinary unloading may begin once the berth assignment is base-final. Hazardous cargo is stricter. It may leave the port only after one of two things happens:

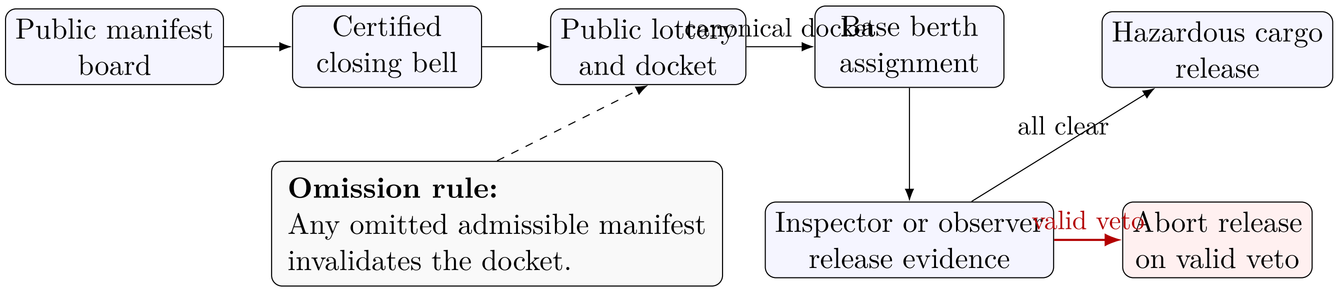

Public manifest board
Certified closing bell
Public lottery and docket
canonical docket
Base berth assignment
Hazardous cargo release
all clear
Omission rule:
Any omitted admissible manifest invalidates the docket.
Inspector or observer release evidence
valid veto
Abort release on valid veto

Figure 1: The thought experiment behind AFT: public admission, canonical ordering, base progress, and fail-closed release.

- the required independent inspector guilds each sign a release certificate, or
- a deterministically assigned set of external captains publishes a public inspection transcript surface, the public red-flag board remains empty through the close bell, and the port issues the unique release docket implied by that surface.

Any valid red flag aborts release. The cargo does not "probably" leave the port. It does not leave.

Most of the port can be chaotic. Captains can lie, stall, or collude. What they cannot do is create a second valid docket for the same tide or a second valid hazardous-cargo release order, provided the public manifest board remains recoverable and the decisive proof surface stays objectively checkable.

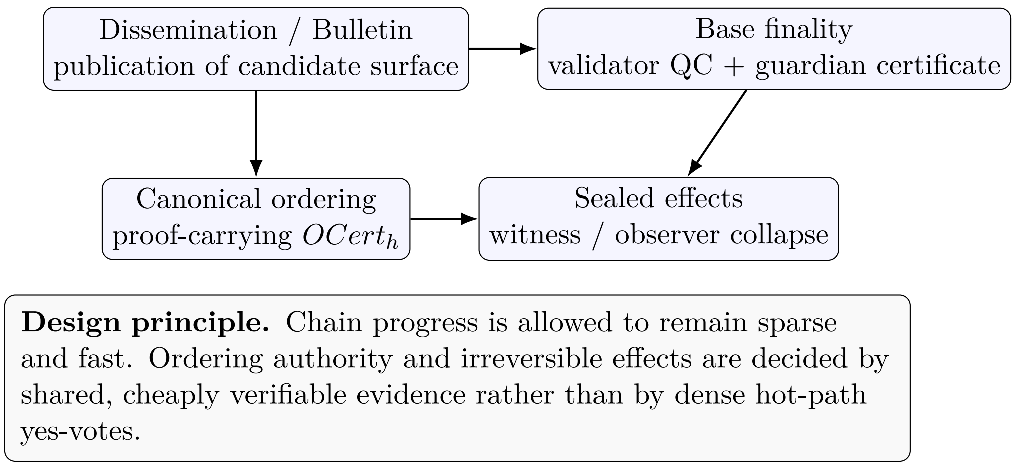
The correspondence is direct:

Port object	**AFT object**
Public manifest board	Bulletin / DA surface B_h
Certified closing bell	Canonical cutoff τ_h
Tide lottery	Public randomness beacon R_h
Docking docket	Canonical order O_h
Docket certificate	Canonical-order certificate $OCert_h$
Omitted manifest challenge	Omission proof $\Omega(h, tx)$
Base berth assignment	`BaseFinal`
Independent inspector guilds	Witness committees
Sampled external captains	Equal-authority observers
Red-flag report	Canonical observer challenge
Hazardous cargo release seal	`SealObject` under `SealedFinal`

This thought experiment captures the essential AFT move: sparse operational progress, dense cheap verification, objective omission, and deterministic collapse for irreversible consequences.

4 Protocol Family Overview

AFT is best understood as four planes that compose:


Dissemination / Bulletin
publication of candidate surface
Base finality
validator QC + guardian certificate
Canonical ordering
proof-carrying $OCert_h$
Sealed effects
witness / observer collapse

Design principle. Chain progress is allowed to remain sparse and fast. Ordering authority and irreversible effects are decided by shared, cheaply verifiable evidence rather than by dense hot-path yes-votes.

Figure 2: AFT separates publication, base progress, canonical ordering, and sealed effect release, then recomposes them through shared evidence.

Plane	**Purpose**	**Throughput-critical work**	**Authority object**
Dissemination / bulletin plane	Publish the candidate transaction surface	Data dissemination and bulletin commitments	Public bulletin commitment
Base-finality plane	Advance the chain and commit block identity	Proposal, QC formation, guardianized certification	Validator QC + guardian certificate
Canonical-ordering plane	Prove the unique slot order	Succinct verification, omission dominance	$OCert_h$
Sealed-effect plane	Gate irreversible effects	Asynchronous witness / observer evidence collection	`SealedFinalityProof` and `SealObject`

The protocol modes are:

Mode	**Role**	**Current status**
`GuardianMajority`	Production guardian-backed base finality	Production mode
`CanonicalOrdering`	Proof-carrying equal-authority ordering	Live through `CommittedSurfaceV1`
`Asymptote`	Two-tier sealing and sealed-effect release	Production sealing path
`NestedGuardian`	Witness-augmented layered threshold mode	Experimental / research

The key invariant is:

$$\text{throughput-critical work} \neq \text{authority-critical work}$$

5 System Model and Notation

5.1 Actors

- **Validator:** participates in proposal, vote, verification, and chain execution.
- **Guardian committee:** the registered threshold-signing committee protecting one validator's proposal decisions.

- **Guardian member:** an individual signer within a guardian committee.
- **Witness committee:** an external threshold-signing committee used on the sealing path or in `NestedGuardian`.
- **Equal-authority observer:** a validator sampled to audit a slot under `Asymptote`.
- **Registry:** the on-chain source of truth for manifests, policies, epochs, witness sets, and verifier descriptors.
- **Transparency log:** an append-only checkpointed evidence log for guardian and witness statements.
- **External verifier or gateway:** a party that checks `SealObject`s and enforces replay-safe release.

5.2 Bulletin-board time and slot model

Consensus proceeds in heights and views. A slot is identified by $(height, view)$. Epoch-scoped policy and committee state bind the admissible evidence for that slot. The family should be read under a bulletin-board recoverability model rather than a pure point-to-point timely receipt model: safety depends on whether admissible evidence becomes publicly retrievable by the slot cutoff, not on whether every participant receives every message before that cutoff.

Slots are externally paced by a public cutoff object τ_h and a public randomness beacon R_h. Honest validators that observe the same closed public boundary derive the same ordering or sealing close-or-abort result. In the runtime that derivation is no longer keyed off cutoff closure alone: the bulletin plane exports a *canonical bulletin-close object* binding the bulletin commitment, the bulletin-availability certificate, and the canonical cutoff into one public close surface. In this instantiation, positive canonical-order publication is fail-closed at this boundary: a closed-slot order certificate is admitted only if the published bulletin entries deterministically extract the bulletin surface bound by the canonical bulletin-close object and the carried bulletin-availability certificate. Universal timely receipt is not required.

Let

$$h = \text{slot height}, \quad v = \text{slot view},$$

$$B_h = \text{bulletin surface admitted before cutoff } \tau_h,$$

$$R_h = \text{public randomness beacon for slot } h,$$

$$E_h = \{tx \mid \text{Included}(tx, B_h) \wedge \text{Eligible}(tx, root_{h-1})\},$$

$$O_h = \text{CanonicalOrder}(R_h, E_h),$$

$$root_h = \text{Execute}(root_{h-1}, O_h).$$

5.3 Core assumptions

The family-level safety claims rely on the following explicit assumptions:

- **Objective publication:** admitted bulletin entries, transcripts, and challenges have stable hashes and objective inclusion paths.

- **Verifiable bulletin availability:** the protocol exports a bulletin-availability object, compact publication-frontier metadata that binds the same slot surface, and a canonical bulletin-close object for each closed ordering slot; the remaining assumption is that the underlying publication substrate satisfies the retrieval semantics claimed by those objects. The live runtime also requires successful closed-slot extraction before positive canonical-order admission.
- **Compact frontier consistency:** any carried `PublicationFrontier` must bind the same-slot bulletin commitment, ordered surface, and compact availability receipt, and same-slot conflicts or stale predecessor links must admit short objective contradiction objects. These frontier objects summarize the slot surface; they do not reconstruct it.
- **Objective validity only:** theorem-critical rejection or acceptance does not depend on private local facts.
- **Public derivability over universal receipt:** honest validators that observe the same closed public boundary derive the same close-or-abort result; universal timely receipt is not required.
- **Independent publication path:** suppressing negative evidence requires capture of both the validator plane and the publication / registry surface.
- **Sealed-effect gating:** irreversible effects are released only after surviving the canonical close-or-abort surface.
- **Fixed epoch identities:** the validator set is epoch-scoped and stable while the accountable rules of that epoch are enforced.

5.4 Evidence objects

The family-level evidence objects are:

- validator quorum certificates over block proposals,
- guardian quorum certificates over slot decisions,
- witness certificates over guardianized slots,
- canonical-order certificates over bulletin surfaces,
- omission proofs over ordered-set incompleteness,
- observer transcripts, observer challenges, and canonical close-or-abort objects over sealed finality,
- sealed-effect proof objects with replay-safe nullifiers.

All honest nodes are required to derive the same acceptance result from the same admissible evidence set.

5.5 Accountable publication rules

Objective AFT faults are not merely audit artifacts. In the runtime they are first-class accountable protocol objects:

- `OmissionProof` names the validator accountable for the dominated positive order object.

- `AsymptoteObserverChallenge` determines accountability by kind: `MissingTranscript` and `ConflictingTranscript` blame the assigned observer; `TranscriptMismatch`, `VetoTranscriptPresent`, and `InvalidCanonicalClose` blame the producer / positive-close path.
- Valid objective fault evidence is replay-deduplicated on chain.
- Policy-controlled membership updates are aftermath only: the same evidence may trigger best-effort immediate quarantine and stage next-epoch eviction, but the decisive negative object remains valid even when those membership updates are disabled, delayed, or skipped.

These rules do not change the hot path. They strengthen the adversary model: arbitrary behavior is tolerated only insofar as it remains publicly revealable and penalty-bearing.

6 Public State Continuity with Extractable Obstructions

AFT's deeper theorem target is not a new round gadget. It is a new source of uniqueness. Classical Byzantine agreement derives uniqueness from overlap of dense positive quorums. AFT instead seeks a rigidity theorem over public slot objects: once the slot boundary is closed, the space of admissible public continuations should collapse to one canonical object or one canonical abort.

6.1 Boundary-conditioned execution language

For a closed slot boundary define

$$\partial_h := (root_{h-1}, \beta_h, \tau_h, R_h, p_h),$$

where β_h is the canonical bulletin-close object binding the bulletin commitment, bulletin-availability certificate, and canonical cutoff whose recoverable surface is B_h; τ_h is the canonical cutoff object; R_h is the slot randomness beacon; and p_h is the policy descriptor governing admissible checks for slot h.

Let a *public execution object* for slot h be a normalized public codeword

$$X_h := (B_h, E_h, O_h, root_h, \sigma_h),$$

where E_h is the eligible subset, O_h is the canonical ordered object, $root_h$ is the resulting state commitment, and σ_h is the slot-local sealing surface needed to determine whether the slot closes or aborts. The important point is that X_h is not a distributed transcript. It is the public quotient of the slot after irrelevant schedule variation is factored out.

Define the boundary-conditioned language

$$\mathcal{L}(\partial_h) := \{X \mid \exists \pi \ \text{Accept}(\partial_h, X, \pi) = 1\}.$$

Here π is a succinct admissibility witness. For canonical ordering, π is the proof-carrying order certificate witness. For deterministic observer sealing, π is the witness that the canonical transcript surface, challenge surface, and close-or-abort object satisfy the slot policy.

6.2 Obstruction extractability

The companion rejection relation is

$$\mathrm{Reject}(\partial_h, Y, \omega) = 1,$$

where Y is a fully specified candidate public execution object and ω is a short public objective contradiction witness.

AFT seeks *obstruction-complete local testability*: for a fixed closed boundary, every fully specified candidate lies in exactly one of two states,

$$\Big(\exists\pi \ \mathrm{Accept}(\partial_h, Y, \pi) = 1\Big) \ \oplus \ \Big(\exists\omega \ \mathrm{Reject}(\partial_h, Y, \omega) = 1\Big).$$

This xor formulation is the software-only analogue of state continuity. It is stronger than merely having a valid proof system. It says every non-admissible candidate exposes a short public contradiction witness.

In the live repository, the obstruction basis is not abstract:

- ordering uses a first-class canonical abort taxonomy over the canonical bulletin surface: missing certificates, bulletin-surface reconstruction failures, bulletin-surface mismatches, invalid bulletin-close formation, omission-dominated candidates, certificate-height mismatches, randomness mismatches, ordered-transactions-root mismatches, resulting-state-root mismatches, invalid public-input bindings, invalid bulletin-availability bindings, and invalid proof bindings,
- deterministic observer sealing uses objective challenge kinds over the canonical transcript and challenge surface,
- the accountable-adversary layer turns those same objects into penalty-bearing public fault evidence.

6.3 Unique collapse object

The slot should not merely admit at most one positive object. It should admit at most one *collapse object*:

$$\mathrm{Collapse}(\partial_h) \in \{\mathrm{Close}(X_h), \mathrm{Abort}(\Omega_h)\}.$$

Here $\mathrm{Close}(X_h)$ denotes the unique admissible positive object when the obstruction surface is empty, while $\mathrm{Abort}(\Omega_h)$ denotes the unique negative object induced by a non-empty valid obstruction surface. Negative evidence does not create a second truth. It forces the slot to land on its canonical abort object instead of a close object.

In the runtime this collapse object is not merely an abstract proof target. Validator finalization publishes a first-class `CanonicalCollapseObject` through `guardian_registry`, and later consensus verification cross-checks any published copy against the same proof-carried ordering and sealing surface when parent-state data is available. In the current recursive-continuity slice, `CanonicalCollapseObject` also carries a rolling `previous_canonical_collapse_commitment_hash`, and in the current clean-break runtime slice it also carries `continuity_accumulator_hash` together with `continuity_recursive_proof`, so the current slot's collapse object is linked to the previously persisted or published collapse object rather than standing as an isolated per-slot fact. The newest proposal-surface slice pushes that same predecessor link into the signed header itself: `BlockHeader` now carries `previous_canonical_collapse_commitment_hash` in its signed preimage and also carries a typed `CanonicalCollapseExtensionCertificate`. That certificate now carries the predecessor commitment plus the predecessor recursive-proof hash, while the public predecessor `CanonicalCollapseObject` now carries only a single recursive proof step rather than a carried predecessor-proof tree. Each recursive proof step still binds a predecessor collapse commitment, predecessor commitment hash, predecessor payload hash, and previous proof hash; the rolling `continuity_accumulator_hash` remains in place as the companion accumulator over the same chain. The implementation surface distinguishes the reference `HashPcdV1` carrier from a backend slot `SuccinctSp1V1`, and the runtime exposes a dedicated continuity-verifier seam for those recursive public inputs via `ioi-api` and `zk-driver-succinct`. That seam is now wired into the live protocol: `GuardianMajority` invokes it while validating proposal and QC continuity for carried or anchored `SuccinctSp1V1` proof steps, and validator durable-state checks invoke the same backend before a persisted `CanonicalCollapseObject` is treated as authoritative. Proposal verification checks that the carried predecessor commitment hash matches the signed predecessor link, checks that the predecessor commitment's resulting state root matches the signed parent-state root, and checks that the carried predecessor proof hash matches the anchored predecessor collapse object already persisted or published in protocol state. Leaders in `Asymptote` stall rather than propose when the required extension certificate or anchored predecessor collapse object is unavailable, proposal verification rechecks that implied predecessor commitment against both the signed predecessor commitment link and the parent-state root, checks the carried predecessor proof hash against the anchored predecessor proof on the public collapse object, and QC progression only treats continuity-linked headers as progress-authoritative.

Theorem 1 (Public State Continuity with Extractable Obstructions)**.** *Assume a closed boundary ∂_h, objective publication, a protocol-native retrievability plane, compact publication-frontier consistency, objective validity only, and proof soundness. Then the target theorem shape for AFT is:*

1. *$\mathcal{L}(\partial_h)$ contains at most one admissible public execution object,*
2. *every fully specified candidate $Y \notin \mathcal{L}(\partial_h)$ admits a short public objective obstruction witness ω such that* $\mathrm{Reject}(\partial_h, Y, \omega) = 1$*,*
3. *therefore the slot admits at most one admissible collapse object, either the unique close object or the unique abort object.*

This theorem is the precise form of the software-only continuity primitive that AFT is trying to realize. It should not be misread as claiming that the classical DLS / PBFT frontier disappeared inside the standard dense-vote model. The point is different: the uniqueness source has moved from quorum overlap into the public-evidence language itself. In the instantiated runtime, compact publication frontiers internalize same-slot and predecessor consistency inside the live protocol, while the protocol-native retrievability plane discharges closed-bulletin extraction-or-abort on the cold path.

6.4 Lemma program

The clean proof path is a filtration rather than a single opaque argument:

$$\mathcal{L}_0(\partial_h) \supseteq \mathcal{L}_1(\partial_h) \supseteq \cdots \supseteq \mathcal{L}_k(\partial_h),$$

where each refinement adds one structural constraint and each failed refinement admits a short witness.

The repository's theorem program is:

1. **Boundary Fixing Lemma.** The closed boundary fixes the admissible publication domain.
2. **Bulletin Close Lemma.** The canonical bulletin-close object fixes the unique recoverable bulletin surface for the slot.
3. **Eligibility Determinism Lemma.** The predecessor commitment and boundary fix the eligible subset.
4. **Order Determinism Lemma.** The boundary and eligible subset fix the canonical ordered object.
5. **Transition Determinism Lemma.** The ordered object fixes the next state commitment.
6. **Collapse Exclusivity Lemma.** A slot cannot admit both a canonical close object and a canonical abort object.
7. **Obstruction Soundness Lemma.** Every valid omission proof or observer challenge objectively rejects its target candidate.
8. **Obstruction Completeness Lemma.** Every invalid fully specified candidate contains at least one minimal public obstruction witness.
9. **Extractor Sufficiency Lemma.** Given the canonical bulletin-close object, any honest validator can run the protocol-defined bulletin extractor to reconstruct the same bulletin surface; in this instantiation, admission of the positive ordering object already requires this extraction to succeed over the published closed-slot surface. One honest validator remains sufficient to surface the resulting positive or negative object under the instantiated runtime model.

The theorem surfaces of this paper are instances of that program:

- `CanonicalOrdering` instantiates unique admissibility and extractable obstruction through proof-carrying order certificates plus omission dominance.
- `Asymptote` instantiates unique collapse through canonical transcript/challenge surfaces plus close-or-abort exclusivity.

7 The AFT Deterministic Base Engine

AFT modes share a deterministic core engine for proposal flow, QC formation, locking, and view progression.

7.1 Deterministic leadership and view progression

For a given height and view, the implementation selects the proposer by deterministic round-robin over the active validator set:

$$\text{leader}(h, v) = V_h[(h - 1 + v) \bmod |V_h|].$$

Here V_h is the active validator set for height h after operational quarantine filters are applied.

View 0 proposals do not carry a timeout certificate. Any non-zero view proposal must carry a valid timeout certificate for the previous unsuccessful view. The pacemaker advances views monotonically and resets timers on observed progress.

7.2 Validator quorum certificates

In `GuardianMajority`, `Asymptote`, and `NestedGuardian`, validator quorum is a strict simple majority of active validator weight, not a classical $2/3 + 1$ quorum:

$$\mathrm{QCQuorum}(V_h) > \frac{1}{2}\mathrm{Weight}(V_h).$$

This is safe only because the validator quorum is *not* the sole authority object. A proposal must also satisfy guardian-layer admissibility.

Timeout certificates use the same simple-majority rule over view-change votes.

7.3 Locking, 2-chain commit, and divergence guard

Validators maintain a lock on the highest safe parent QC they have observed. A validator votes only when the proposal extends the lock or is in a strictly higher view than the lock.

Base commitment uses a 2-chain rule:

- let QC_{parent} certify parent block P,
- let QC_{child} certify child block C,
- if C directly extends P and the views are consecutive, then P becomes a pending commit.

The implementation delays final durability by a guard window Δ_{guard} before a pending commit is released. The purpose of the guard window is operational, not purely algebraic: it gives divergence evidence or panic signals time to propagate before a conflicting branch is treated as durable.

This produces a fail-closed operational behavior:

- sparse chain progress happens on the hot path,
- conflicting evidence is surfaced before irreversible consequences,
- divergence proofs quarantine the affected node rather than silently allowing fork continuation.

8 GuardianMajority: Production Base Finality

`GuardianMajority` is the production fault model for AFT base finality. Validator votes alone never make a proposal admissible. Every valid proposal must also carry a guardian committee certificate that verifies against current registered committee state.

8.1 Guardian certificate object

The guardian certificate binds:

- the guardian committee manifest hash,
- the active epoch,
- the canonical decision hash for the slot payload,
- a monotonic guardian counter,

- a guardian trace root,
- a measurement root,
- the signer bitfield,
- the aggregated BLS signature,
- a transparency-log checkpoint,
- and, in `NestedGuardian`, an optional experimental witness certificate.

The guardian decision payload is slot-scoped. Honest guardians are assumed not to sign two different decisions for the same slot.

8.2 Verification rule

A guardianized proposal is admissible only if all of the following hold:

1. the referenced guardian committee manifest is registered on-chain for the active epoch,
2. the certificate manifest hash matches the registered manifest,
3. the certificate epoch matches the manifest epoch and current epoch,
4. the decision hash matches the slot's canonical guardian decision payload,
5. the measurement root matches the decision payload,
6. the signer set is valid, unique, and meets the committee threshold,
7. the aggregated signature verifies against the manifest public keys,
8. the attached transparency-log checkpoint verifies against the anchored log descriptor and append-only proof.

The transparency log is an accountability layer. It does not replace threshold safety. It makes issued decisions, checkpoints, and later slashing evidence publicly checkable.

8.3 Committee threshold model

Let a guardian committee have threshold t and size n. Conflicting slot certificates require enough equivocators to cover the minimum quorum intersection:

$$f < 2t - n.$$

Production committees therefore use strict majority thresholds:

$$t = \left\lfloor \frac{n}{2} \right\rfloor + 1.$$

This guarantees pairwise quorum intersection. It also exposes an important operational nuance:

- odd-sized simple-majority committees tolerate zero equivocating guardian members at the committee layer,

- even-sized majority committees tolerate one equivocator before threshold intersection is exhausted.

Production safety therefore relies on both threshold intersection and stronger guardian non-equivocation guarantees.

8.4 Safety statement

Proposition 1 (GuardianMajority Safety)**.** *No two conflicting blocks can both finalize for the same* $(height, view)$ *if guardian committee manifests are current, committee thresholds are majority thresholds, honest guardians do not equivocate for the same slot, transparency-log checkpoints and registry state are current enough to verify admissibility, and validators finalize only guardian-admissible blocks.*

This is not a classical $3f + 1$ theorem. It is a composed threshold theorem over validators, guardians, registry state, and shared evidence.

8.5 Liveness model

Liveness is secondary to safety. Progress requires:

- enough active validators to form the simple-majority validator QC,
- enough guardian members to satisfy the committee threshold,
- current registry state for the epoch,
- current enough checkpoint and log availability.

If these degrade, the protocol prefers stalling over accepting unsafe evidence.

9 Canonical Ordering

`CanonicalOrdering` is the proof-carrying ordering path that gives AFT its `99%` equal-authority ordering consensus claim.

9.1 Objective

For each slot h, define a unique canonical ordered set O_h and a certificate $OCert_h$ such that:

- every honest verifier can check $OCert_h$ cheaply,
- any omission is objectively provable,
- the ordered set is recoverable from public data,
- the live hot path carries a compact publication frontier that binds the same-slot bulletin / ordering / availability surface and predecessor continuity,
- conflicting valid certificates for the same slot are impossible,
- arbitrary behavior by almost all validators cannot create a conflicting valid ordering outcome because the closed bulletin boundary admits one deterministically derivable close-or-abort result.

9.2 Canonical objects

For slot h,

$$\begin{aligned}
B_h &= \text{bulletin surface admitted before } \tau_h, \\
R_h &= \text{public randomness beacon for slot } h, \\
E_h &= \{tx \mid \text{Included}(tx, B_h) \wedge \text{Eligible}(tx, root_{h-1})\}, \\
O_h &= \text{CanonicalOrder}(R_h, E_h), \\
root_h &= \text{Execute}(root_{h-1}, O_h).
\end{aligned}$$

The abstract certificate is

$$OCert_h = (h, root_{h-1}, cutoff_h, bulletin\_commitment_h, ordered\_commitment_h, root_h, witness_h) .$$

9.3 Bulletin, cutoff, and admissibility assumptions

The ordering theorem is conditional on the following named assumptions.

A1. Objective publication. Each admitted transaction or batch in B_h has a stable content hash, a verifiable inclusion path into the bulletin commitment, and an objective publication time or publication position relative to the slot cutoff.

A2. Endogenous retrievability. The bulletin contents committed by $bulletin\_commitment_h$ are reconstructed or deterministically aborted from protocol-native retrievability objects rooted in the canonical bulletin close for slot h. Compact publication-frontier summaries and availability certificates bind that claim on the hot path, while retrievability profiles, shard manifests, custody assignments, custody receipts, custody responses, and objective challenge objects close the extraction-or-abort loop on the cold path. Without that plane, uniqueness may still hold while timely revelation can fail.

A3. Append-only or equivocation-evident commitment. The bulletin commitment and any carried publication frontier for slot h must be append-only or otherwise make same-slot equivocation or stale predecessor linkage objectively detectable at the slot boundary.

A4. Eligibility determinism. $\text{Eligible}(tx, root_{h-1})$ is deterministic and locally checkable from the committed predecessor root and the transaction object. Honest validators therefore derive the same eligible set E_h from the same public inputs.

A5. Proof soundness. $\text{VerifyProof}(witness_h) = \text{true}$ implies that the witness obligations for cutoff binding, bulletin binding, ordering, state transition, retrievability commitments, and omission commitments actually hold.

The cutoff τ_h is not a private local clock read. It is a protocol-bound object. In the instantiated runtime, the cutoff is bound directly into the published bulletin commitment and into the canonical public-input set. Eligibility is deterministic with respect to the predecessor state root and the transaction object. This prevents local reinterpretation of the eligible set.

9.4 Current runtime instantiation

The runtime surfaces the abstract objects above as the following concrete objects:

Runtime object	Current fields
BulletinCommitment	(height, cutoff_timestamp_ms, bulletin_root, entry_count)
BulletinSurfaceEntry	(height, tx_hash)
BulletinAvailability Certificate	(height, bulletin_commitment_hash, recoverability_root)
BulletinRetrievabilityProfile	(height, bulletin_commitment_hash, bulletin_availability_certificate_hash, recoverability_root, shard_count, recovery_threshold, custody_threshold)
BulletinShardManifest	(height, bulletin_commitment_hash, bulletin_availability_certificate_hash, bulletin_retrievability_profile_hash, recoverability_root, entry_count, shard_count, recovery_threshold, shard_commitment_root)
BulletinCustodyAssignment	(height, bulletin_retrievability_profile_hash, bulletin_shard_manifest_hash, validator_set_commitment_hash, assignments)
BulletinCustodyReceipt	(height, bulletin_commitment_hash, bulletin_availability_certificate_hash, bulletin_retrievability_profile_hash, bulletin_shard_manifest_hash, custodian_count, custody_threshold, custody_root)
BulletinCustodyResponse	(height, bulletin_commitment_hash, bulletin_availability_certificate_hash, bulletin_retrievability_profile_hash, bulletin_shard_manifest_hash, bulletin_custody_assignment_hash, bulletin_custody_receipt_hash, served_shards)
BulletinRetrievabilityChallenge	(height, kind, bulletin_commitment_hash, bulletin_availability_certificate_hash, bulletin_retrievability_profile_hash, bulletin_shard_manifest_hash, bulletin_custody_assignment_hash, bulletin_custody_receipt_hash, bulletin_custody_response_hash, details)
BulletinReconstructionCertificate	(height, bulletin_commitment_hash, bulletin_availability_certificate_hash, bulletin_retrievability_profile_hash, bulletin_shard_manifest_hash, bulletin_custody_assignment_hash, bulletin_custody_receipt_hash, bulletin_custody_response_hash, canonical_bulletin_close_hash, canonical_order_certificate_hash, reconstructed_entry_count, reconstructed_bulletin_root)
BulletinReconstructionAbort	(height, kind, bulletin_commitment_hash, bulletin_availability_certificate_hash, bulletin_retrievability_profile_hash, bulletin_shard_manifest_hash, bulletin_custody_receipt_hash, bulletin_retrievability_challenge_hash, canonical_order_abort_hash, details)
PublicationAvailabilityReceipt	(height, bulletin_commitment_hash, ordered_transactions_root_hash, resulting_state_root_hash, receipt_root)
PublicationFrontier	(height, view, counter, parent_frontier_hash, bulletin_commitment_hash, ordered_transactions_root_hash, availability_receipt_hash)
PublicationFrontierContradiction	(height, kind, candidate_frontier, reference_frontier)
CanonicalBulletin Close	(height, cutoff_timestamp_ms, bulletin_commitment_hash, bulletin_availability_certificate_hash, bulletin_retrievability_profile_hash, bulletin_shard_manifest_hash, bulletin_custody_receipt_hash, entry_count)
CanonicalOrder PublicInputs	(height, parent_state_root_hash, bulletin_commitment_hash, randomness_beacon, ordered_transactions_root_hash, resulting_state_root_hash, cutoff_timestamp_ms)
CanonicalOrderProofSystem	concrete variants HashBindingV1 and CommittedSurfaceV1
CommittedSurface CanonicalOrderProof	(recoverability_root, omission_commitment_root)
CanonicalOrder Certificate	(height, bulletin_commitment, bulletin_availability_certificate, randomness_beacon, ordered_transactions_root_hash, resulting_state_root_hash, proof, omission_proofs)
CanonicalOrder PublicationBundle	(bulletin_commitment, bulletin_entries, bulletin_availability_certificate, bulletin_retrievability_profile, bulletin_shard_manifest, bulletin_custody_receipt, canonical_order_certificate)
CanonicalOrder Abort	(height, reason, details, bulletin_commitment_hash, bulletin_availability_certificate_hash, bulletin_close_hash, canonical_order_certificate_hash)

`HashBindingV1` is the reference verifier family: its proof bytes are a canonical hash binding over the public-input hash. `CommittedSurfaceV1` is the live proof family: its proof payload contains the recoverability root and the omission-commitment root for the committed surface. In the current runtime, positive ordering artifacts are published through the atomic `CanonicalOrderPublicationBundle`, carrying the bound retrievability profile / manifest / custody-receipt objects, while the registry deterministically materializes the coupled custody-assignment and custody-response objects over the same slot surface before admitting the positive lane. Positive committed blocks also carry a compact signed `PublicationFrontier`: it binds the same-slot bulletin commitment, ordered-transactions root, and compact publication-availability receipt, and it hash-links that summary to the previous slot's frontier. Short `PublicationFrontierContradiction` objects make same-slot conflict or stale predecessor linkage objectively rejectable on the hot path. Those frontier objects internalize compact publication consistency, but they do not replace the protocol-native retrievability plane that reconstructs or aborts the full bulletin surface. Validators also run a block-local derivation pass over the committed block boundary: they derive either a `CanonicalOrderExecutionObject` or a `CanonicalOrderAbort`, and publish the abort artifact when the proof-carried positive object cannot be derived. When the endogenous retrievability lane succeeds, the registry materializes a first-class `BulletinReconstructionCertificate` bound to the canonical close, canonical-order certificate, and the same retrievability objects. When endogenous retrievability evidence dominates that same surface, the registry also materializes a specialized `BulletinReconstructionAbort` object bound to the challenge and the paired `CanonicalOrderAbort`, so reconstruction-impossible failure is named as a first-class protocol object rather than only as a generic ordering abort.

9.5 Acceptance rule

Every validator applies the same deterministic rule:

$$\begin{aligned}\text{Accept}(OCert_h) \iff\ & \text{VerifyProof}(witness_h)\\ & \land \text{VerifyCutoff}(cutoff_h)\\ & \land \text{PredecessorAccepted}(root_{h-1})\\ & \land \neg\exists tx : \text{ValidOmission}(h, tx).\end{aligned}$$

In the implementation this specializes to:

- the certificate height and bulletin height must match the block height,
- the randomness beacon must match the slot schedule,
- the bulletin commitment must match the published bulletin when that bulletin is available from anchored state,
- any carried `PublicationFrontier` must match the same-slot bulletin commitment, ordered-transactions root, and publication-availability receipt hash,
- when a predecessor frontier exists, the carried `PublicationFrontier` must extend it by counter and parent hash,
- no published `PublicationFrontierContradiction` may already dominate the slot,

- positive canonical-order admission in the runtime additionally requires successful closed-slot extraction over the published bulletin entries, the carried bulletin-availability certificate, and the canonical bulletin-close object,
- the ordered-transactions root and resulting state root must match the block header,
- the proof must match the canonical public inputs,
- the recoverability root must match the certificate commitments,
- the omission-commitment root must match the omission-proof set.

If the omission-proof set is non-empty, the candidate certificate is rejected immediately.

No dense positive vote on the order is required once a valid $OCert_h$ is revealed. The current verifier entry point for this rule is `verify_canonical_order_certificate`.

9.6 Omission dominance

An omission is objective when a transaction:

- was included in B_h before τ_h,
- was eligible under $root_{h-1}$,
- and is absent from the committed ordered set O_h.

Define the omission proof

$$\Omega(h, tx) = \left(\begin{array}{l} inclusion\_proof(tx, bulletin_h), \\ cutoff\_admissibility\_proof(tx, cutoff_h), \\ eligibility\_proof(tx, root_{h-1}), \\ nonmembership\_proof(tx, O_h) \end{array} \right).$$

Its semantics are:

$$\text{Valid}(\Omega(h, tx)) \Rightarrow \text{Reject}(OCert_h).$$

This is the central AFT ordering move. A candidate does not need to win a dense round of positive endorsements if any incomplete view can be killed objectively.

9.7 Uniqueness, publication frontiers, and recoverability

Theorem 2 (Uniqueness)**.** *If* $\text{ValidOrderCert}(h, C_1)$ *and* $\text{ValidOrderCert}(h, C_2)$*, then* C_1 *and* C_2 *commit the same canonical ordered set* O_h*.*

The reason is structural:

- the predecessor root is fixed,
- the cutoff object is canonical,
- the bulletin commitment is canonical,
- eligibility is deterministic,

- the ordering function is deterministic,
- proof soundness forbids a witness for a different ordered set.

The live publication-frontier slice sits between uniqueness and recoverability. A frontier gives a compact signed summary of the claimed bulletin / ordering surface and its predecessor link, so same-slot conflict or stale lineage is objectively rejectable without reconstructing the full bulletin on the hot path. But the frontier only binds hashes and receipt roots; it is not itself the recoverable bulletin surface.

Theorem 3 (Endogenous Retrievability)**.** *If* $\mathrm{ValidOrderCert}(h, C)$ *and the canonical bulletin close plus protocol-native retrievability plane for slot h are present and unchallenged, then every honest verifier can reconstruct the same ordered set* O_h *from protocol objects alone: the canonical close, bound frontier / availability commitments, retrievability profile, shard manifest, custody receipt, positive reconstruction certificate, and published bulletin entries.*

Endogenous retrievability matters because the accepted order is not merely unique. The compact frontier tells validators which surface is being claimed; the retrievability plane tells them that the full surface can actually be reconstructed or objectively aborted from protocol evidence.

9.8 99% equal-authority ordering consensus

AFT's ordering claim is a revelation theorem, not a classical dense-vote BFT theorem.

Theorem 4 (99% Equal-Authority Ordering Consensus)**.** *Assume:*

1. *the closed bulletin surface* B_h *bound by the carried publication frontier and bulletin commitment is governed by a canonical bulletin close and protocol-native retrievability plane that deterministically yields extraction or abort,*
2. *the cutoff object for slot* h *is canonical,*
3. *bulletin commitments, compact publication-frontier bindings, and omission proofs are objectively verifiable,*
4. *same-slot frontier conflicts and stale predecessor links admit short objective contradiction objects,*
5. *the proof system is sound.*

Then even if 99% *of validators behave arbitrarily, they cannot create a conflicting valid canonical-order outcome for slot* h*. More strongly, every honest validator that observes the same closed ordering boundary derives the same positive order object or omission-dominated abort.*

Compact publication frontiers therefore internalize cheap same-slot and predecessor consistency checks on the live path, while recoverability remains the stronger assumption needed to turn that compact claim into a publicly reconstructable ordered set.

The strongest shorthand is:

$$\text{99\% of validators may behave arbitrarily,}$$

$$\text{because the closed ordering boundary admits one deterministically}$$

$$\text{derivable close-or-abort outcome.}$$

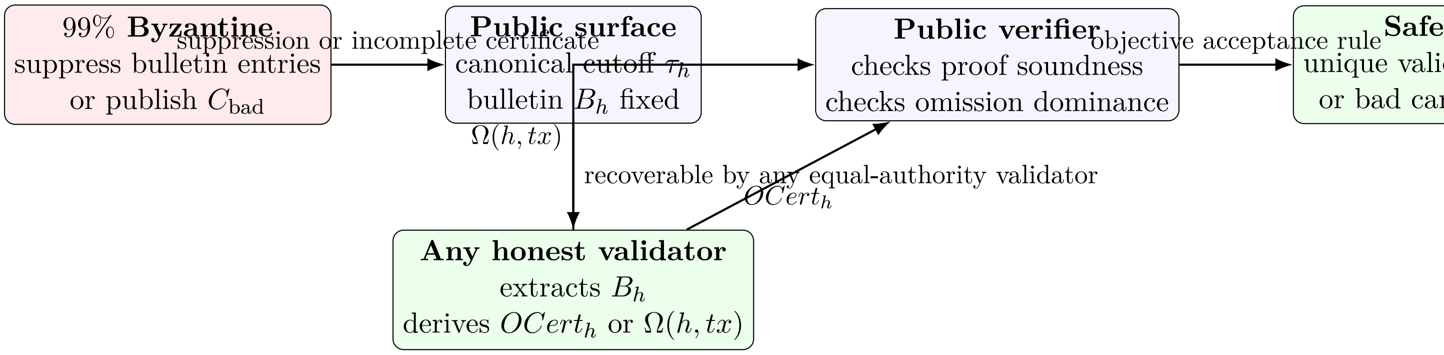

99% **Byzantine**
suppress bulletin entries
or publish C_{bad}
suppression or incomplete certificate
Public surface
canonical cutoff τ_h
bulletin B_h fixed
$\Omega(h, tx)$
Public verifier
checks proof soundness
checks omission dominance
objective acceptance rule
Safe
unique vali
or bad can
recoverable by any equal-authority validator
$OCert_h$
Any honest validator
extracts B_h
derives $OCert_h$ or $\Omega(h, tx)$

Figure 3: Omission-dominance attack and deterministic extraction. Even if a suppressing supermajority tries to hide admissible bulletin entries or promote an incomplete candidate certificate, any honest validator that reconstructs the same public surface derives the same valid $OCert_h$ or publishes a valid omission proof $\Omega(h, tx)$ that kills the bad candidate.

This is equal-authority because any validator may reveal the winning certificate. It is `99%`-fault tolerant because correctness does not depend on a dense majority of positive votes. It is not classical `99% Byzantine agreement`.

With the accountable publication rules of Section 5, invalid positive ordering attempts are not only rejectable; they are also penalty-bearing and next-epoch removing.

10 Asymptote: Sealed Finality and Irreversible Effects

`Asymptote` is the two-tier settlement mode in the AFT family. It preserves fast `BaseFinal` chain progression while requiring stronger evidence before irreversible effects are released.

10.1 Finality tiers and collapse states

The finality tiers are:

- `BaseFinal`: validator QC plus guardian certificate.
- `SealedFinal`: `BaseFinal` plus sufficient sealing evidence.

The slot collapse states are:

- `Pending`
- `BaseFinal`
- `Sealing`
- `Abort`
- `SealedFinal`
- `Escalated`
- `Invalid`

All honest nodes must derive the same collapse state from the same admissible evidence set.

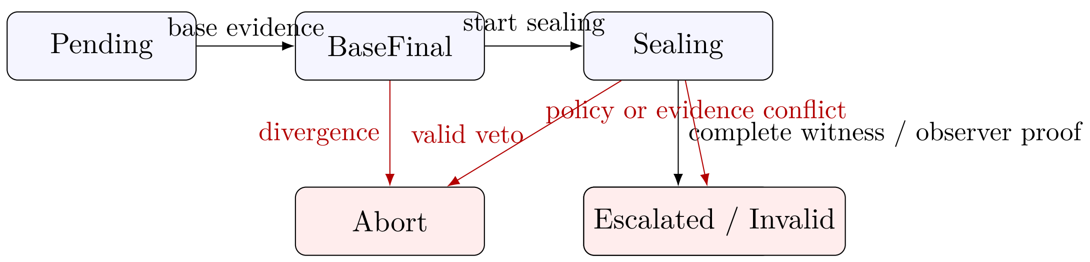


Figure 4: The deterministic collapse surface for `Asymptote`.

10.2 SealedFinalityProof

The sealing plane gathers a proof object that binds:

- the epoch,
- the achieved finality tier,
- the collapse state,
- the guardian manifest hash,
- the guardian decision hash,
- the guardian counter,
- the guardian trace root,
- the guardian measurement root,
- the policy hash,
- witness certificates, when the witness path is used,
- observer transcripts, transcript commitment, observer challenges, challenge commitment, and exactly one canonical observer outcome object when the observer path is used,
- divergence signals gathered during sealing.

This object is block-scoped and guardian-bound. A `SealedFinalityProof` that is not bound to the slot's guardian evidence is invalid.

10.3 Witness-backed sealing

The witness path assigns one witness committee per required certification stratum.

Witness assignment is deterministic over:

- the epoch seed,
- the producer account id,
- the slot $(height, view)$,
- the reassignment depth,

- the witness stratum id,
- the witness manifest hash.

Sealed finality is reached on the witness path when:

- `BaseFinal` holds,
- each required witness stratum contributes a valid assigned witness certificate,
- each witness certificate verifies against the registered witness manifest,
- required checkpoints and append-only proofs are current enough under policy.

Witness omission, stale registry participation, checkpoint inconsistency, or conflicting witness certificates produce explicit evidence rather than silent best-effort behavior.

10.4 Equal-authority observer sealing

The normative observer path is `CanonicalChallengeV1`. It treats sampled validators as deterministic duty assignees whose outputs must become public, recoverable protocol objects.

Observer assignment is deterministic over:

- the epoch seed,
- the producer account id,
- the slot $(height, view)$,
- the observation round,
- the observer account id.

The producer is excluded from its own observer pool. The runtime may also filter assignments through a correlation budget over provider, region, host class, and key-authority class.

Each assigned observer publishes one of two things:

- a guardian-backed `AsymptoteObserverTranscript`, or
- an objective `AsymptoteObserverChallenge`.

A transcript binds the canonical slot surface:

- the epoch and assignment,
- the candidate block hash,
- the guardian manifest, decision hash, counter, trace hash, and measurement root,
- the guardian checkpoint root,
- the observer verdict and veto kind,
- the observer evidence hash.

The admissible challenge basis is public-evidence-only:

- `MissingTranscript`,
- `TranscriptMismatch`,
- `VetoTranscriptPresent`,
- `ConflictingTranscript`,
- `InvalidCanonicalClose`.

Each challenge carries its own public evidence object: an assignment, an offending observation request, an offending transcript, or an offending canonical close. The challenge `evidence_hash` must bind exactly that carried evidence.

The slot then carries three canonical bindings:

- a transcript commitment over the transcript surface,
- a challenge commitment over the challenge surface,
- exactly one canonical outcome object: `AsymptoteObserverCanonicalClose` or `AsymptoteObserverCanonicalA`

The collapse rule is close-or-abort and challenge-dominant:

- `SealedFinal` if `BaseFinal` holds, the deterministic assignment set is fully covered by valid canonical `Ok` transcripts, the transcript and challenge commitments verify, the challenge surface is empty, and the canonical close object is valid.
- `Abort` if the transcript and challenge commitments verify, the challenge surface is non-empty, and the canonical abort object is valid.
- `Pending` otherwise.

This makes observer sealing structurally parallel to canonical ordering: public evidence, canonical commitments, a unique positive object, a unique negative object, and deterministic local close-or-abort derivation from the same public surface.

10.5 Divergence signals and escalation

The sealing plane may also emit divergence signals, including:

- conflicting guardian certificates,
- witness omission,
- stale registry use,
- stale checkpoints,
- transparency-log consistency failure.

These signals support `Escalated` and `Invalid` outcomes and give operators a fail-closed evidence trail.

10.6 Sealed effects and replay-safe release

Irreversible effects are gated by finality tier. The effect plane distinguishes between:

- `BaseFinal` effects, which are allowed on the fast path when policy permits,
- `SealedFinal` effects, which require a valid `SealedFinalityProof`.

For proof-enabled irreversible effects, `Asymptote` carries a self-contained `SealObject` whose current instantiation includes:

- an `EffectIntent`,
- an `EffectPublicInputs` object, including `canonical_collapse_hash`,
- an `EffectProofEnvelope`,
- a replay-safe nullifier.

In the runtime, `SealedFinal` secure-egress requests carry both the `SealedFinalityProof` and the slot's `CanonicalCollapseObject`. The guardian-side seal builder and downstream receipt verifier recompute the same `canonical_collapse_hash` from that collapse object together with the proof-carried observer surface, so a challenge-dominated or otherwise mismatched slot cannot authorize irreversible release.

The release rule is

$$\mathrm{Release}(e) \iff \mathrm{BaseFinal}(\mathrm{slot}(e)) \wedge \mathrm{SealedFinal}(\mathrm{slot}(e)) \wedge \mathrm{VerifySealObject}(e) \wedge \mathrm{CollapseHash}(e) = \mathrm{Hash}(\mathrm{Collapse}($$

This keeps the hot path sparse while making released consequences deterministic and replay-safe.

10.7 Deterministic observer theorem

The deterministic `Asymptote` observer model now proves the following kernel:

- base certificates are unique per slot,
- canonical observer close objects are unique per slot,
- canonical observer close and canonical observer abort cannot coexist for the same slot,
- a challenge-dominated slot cannot produce a valid sealed release.

Theorem 5 (Deterministic Equal-Authority Observer Sealing)**.** *Assume sound deterministic observer assignment, signature-sound guardian-backed observer transcripts and challenges, canonical transcript and challenge commitments, append-only registry and log publication once published, and deterministic transcript/challenge derivation over the closed public observer surface. Then arbitrary behavior by the rest of the validator set cannot create a conflicting valid canonical observer close, cannot make canonical close and canonical abort coexist for the same slot, and cannot authorize a valid sealed irreversible effect release for a challenge-dominated slot.*

This theorem is the observer-lane analogue of canonical ordering's public certificate theorem: public evidence, canonical commitments, negative-authority dominance, and deterministic local close-or-abort derivation from the same public observer surface.

Under the accountable publication rules of Section 5, the same observer challenges are also penalty-bearing objective faults. Unsafe positive close attempts therefore become not only rejectable, but accountable and epoch-removing.

10.8 Legacy sampled comparison

Before `CanonicalChallengeV1`, the observer lane was typically described through a sampled affirmative bound. Let:

- h be the honest fraction of equal-authority validators,
- A_s^j be the sampled observer committee for slot s and round j,
- $k_j = |A_s^j|$,
- $p_{\text{beacon\_bias}}$ be the probability of sufficient beacon bias,
- $p_{\text{veto\_suppression}}$ be the probability that a valid veto surface is fully suppressed.

Then the old affirmative sampled lane admitted the analytical bound

$$P_{\text{unsafe}}(s) \le p_{\text{beacon\_bias}} \cdot p_{\text{veto\_suppression}} \cdot \prod_j (1-h)^{k_j}.$$

For constant committee size k across r rounds:

$$P_{\text{unsafe}}(s) \le p_{\text{beacon\_bias}} \cdot p_{\text{veto\_suppression}} \cdot (1-h)^{kr}.$$

That formula remains a useful historical comparison, but it is no longer the normative theorem for the observer lane. Under `CanonicalChallengeV1`, sampling determines duties, not safety. Safety comes from the canonical public transcript surface, the canonical public challenge surface, and the close-or-abort split that those surfaces induce.

The runtime publishes the canonical observer transcript, commitment, challenge, challenge commitment, and close-or-abort objects as ordinary `guardian_registry` transactions after the sealed header is formed. The same surface is carried inside the proof for immediate block verification.

11 NestedGuardian: Witness-Augmented Research Mode

`NestedGuardian` explores a layered threshold construction:

- validators run the AFT deterministic message flow,
- each proposal carries its guardian committee certificate,
- external witness committees cross-check the slot,
- chain state anchors witness assignments, witness manifests, and slashing evidence.

The instantiated runtime scope includes:

- deterministic witness assignment from the active witness set,
- registered on-chain witness manifests,
- witness certificates bound into guardianized slot certificates,
- safety proofs for the witness-augmented kernel,

- bounded model checking for reassignment, outage, and checkpoint transitions.

The instantiated proof package consists of:

- safety formalization,
- bounded operational model checking for reassignment, outage, and checkpoint transitions,
- a discharged composed-liveness bridge under the target eventual-fair scheduler through the recurrence/reduction/totality/collapse chain together with a matching persistent churn/restart simulator.

`NestedGuardian` is therefore a witness-augmented mode with a finished theorem bridge. It includes a constructive recovery carrier: a witness-coded publication-bundle recovery contract with compact hot-path bindings and cold-path registry-driven close-or-abort resolution. The abstract coded recovery-family contract is realized by the parametric `SystematicXorKOfKPlus1V1` parity family plus the bounded true non-parity `SystematicGf256KOfNV1` family with at least two parity shares, implemented with systematic Cauchy-style coefficients and exercised in bounded `2-of-4`, `3-of-5`, `3-of-7`, `4-of-6`, and `4-of-7` lanes. These lanes resolve from threshold-many public reveals, objective recovered-support conflict, or deterministic missingness. The bounded `3-of-5`, `4-of-6`, and `4-of-7` lanes restore or exceed threshold-support overlap while the bounded `3-of-7` lane shows that recovered-support conflict remains fail-closed even when overlap is absent. A bounded conformance harness checks that every threshold-sized reveal subset reconstructs and every below-threshold subset fails across the shipped XOR and GF256 realizations of that contract. Those recovered reveals lift deterministically into an explicit positive close-extraction payload and then into an explicit extractable bulletin-surface payload that binds the verifying publication bundle, verifying canonical bulletin close, canonical bulletin-availability bytes, and sorted bulletin entries.

The completed bounded family also carries the ordinary durable and restart surfaces. A bounded recovered-only two-slot harness shows that consecutive recovered surfaces derive the ordinary publication-frontier predecessor chain and the same authoritative `CanonicalCollapseObject` predecessor chain without the ordinary publication-bundle lane. A second bounded recovered-only harness shows a close/abort/close bulletin window with a recovered omission-dominated middle slot: recovered data alone materializes the ordinary positive closes and extracted bulletin surfaces on the outer slots, materializes the ordinary omission-dominated abort in the middle while keeping omission evidence and recovered bulletin entries public, and preserves the same authoritative collapse continuity chain. A bounded read-side harness shows that the same recovered-only durable chain yields the same compact replay-prefix / state-root continuity view that the ordinary lane exposes plus a bounded recovered canonical-header prefix. The execution module verifies that replay tip as the same restart anchor the ordinary lane uses, retains both recovered prefixes in execution state, and uses the recovered canonical-header ancestry for bounded anchored reads plus parent-block / parent-state-root continuity after restart when ordinary recent blocks are absent.

Validator consensus orchestration derives the same bounded recovered restart parent anchor from the recovered canonical-header / collapse surface when the ordinary committed block is absent, and the `GuardianMajority` engine consumes that bounded recovered canonical-header prefix as restart-time parent/QC context for synthetic parent-height QC continuity when the full committed block is absent. The same recovered surface yields a bounded recovered certified-header window plus a restart-only recovered `BlockHeader` / QC cache window for restart-time header / QC lookup plus bounded QC-certified recovered branch reconciliation over the configured local five-entry window. Those windows fold by exact overlap with a configurable budget, exercised at five windows into a longer seventeen-step recovered certified branch without ordinary committed headers. Those same cold-path loaders consume a configurable exact-overlap segment budget, exercised at four exact-overlap segments into a longer fifty-three-step recovered certified branch with direct registry-extraction parity and explicit interior-overlap conflict rejection. Those same restart loaders also compose two overlapping four-segment exact-overlap folds into a longer eighty-nine-step recovered certified branch with direct registry-extraction parity and explicit inter-fold overlap conflict rejection, and tests exercise the same recursive carrier at three overlapping four-segment folds into a one-hundred-twenty-five-step recovered certified branch. The runtime has a bounded conformance harness over one, two, and three stitched segment folds, matching the same production-loader / registry-extraction carrier at the corresponding fifty-three-step, eighty-nine-step, and one-hundred-twenty-five-step branches.

The restart path also pages older exact-overlap segment folds on demand beneath that bounded suffix via an overlap-checked cursor while keeping only the bounded recent suffix plus the streamed page live in engine caches, and tests exercise that paged carrier at a longer two-hundred-thirty-three-step recovered certified branch with direct registry-extraction parity plus explicit duplicate-page, missing-gap, and late-page overlap rejection. Restart therefore streams recovered certified ancestry until target height or recovered-history exhaustion without a theorem-relevant fixed depth bound.

The archival internalization layer uses an active archived-recovered-history profile naming retention horizon, exact-overlap paging geometry, segment / fold geometry, checkpoint update rule, and profile evolution. Archived segments, archived restart pages, archival checkpoints, and archival retention receipts bind to historically referenced profile hashes and governing activation hashes under a published profile-activation chain before archived continuation is trusted, and ordinary canonical collapse history names the archived checkpoint / activation pair that archived restart is authorized to follow. Archived replay and archived publication correctness are historical and index-free: they follow the canonical-collapse-anchored or object-carried activation hash plus predecessor/checkpoint history rather than whichever archived activation is merely latest in local state.

When the recovered certificate carries objective omission proofs, the same recovered lane routes into the ordinary `OmissionDominated` abort path while keeping omission evidence and recovered bulletin entries public. The recovery-family contract, canonical-collapse / replay retrievability anchor, and recovered-state historical retrievability surface are therefore part of ordinary endogenous AFT history rather than a narrower theorem-side qualifier. The directly discharged recurrence/reduction/totality/collapse chain closes the former distance to unconditional classical `99% Byzantine agreement`, so no lower-bound, recoverability, or architecture-specific semantic gap remains at the theorem surface.

12 Security and Assumption Surface

The AFT family depends on explicit assumptions. They are not optional prose.

Layer	Required assumptions	Failure consequence
Base finality	current manifests, current epoch state, majority thresholds, guardian non-equivocation, valid checkpoints, enough validators and guardians to reach quorum	conflicting base-finality evidence or loss of progress
Canonical ordering	objective publication, compact publication-frontier binding and contradiction rules, canonical cutoff, deterministic eligibility, append-only bulletin commitment, proof soundness, protocol-native retrievability profile / manifest / custody / challenge surface, canonical bulletin-close extraction-or-abort, cheap omission verification, accountable omission publication	loss of compact frontier consistency, deterministic extraction-or-abort, accountability, or the ordering theorem
Sealed finality	valid asymptote policy, current witness seed, valid witness or observer assignments, checkpoint freshness, valid log consistency proofs, deterministic transcript/challenge derivation, accountable challenge publication	incorrect sealed release, loss of accountability, or fail-closed stalling
Sealed effects	correct finality-tier policy, sound verifier metadata, nullifier uniqueness, intent/public-input binding	replay or unsafe irreversible effect release
NestedGuardian	sound witness assignment and registration, correct reassignment handling, witness committee non-equivocation	degraded safety or open liveness behavior

The family-level discipline is simple: if an assumption fails, AFT should degrade to stalling, aborting, or withholding release before it degrades to unsafe irreversible behavior.

13 Formal Results

The repository's machine-checked surfaces align with the protocol family as follows:

Formal module	Core proved results
`GuardianMajorityProof`	`QuorumCertificatesIntersect`, `InvariantImpliesSafety`
`GuardianMajority`	executable bounded checks for finalization witness soundness and slot safety
`CanonicalOrderingProof`	`InvariantImpliesCertifiedUniqueness`, `InvariantImpliesRecoverability`, `InvariantImpliesOmissionDominates`
`CanonicalOrdering`	executable bounded checks for publication, cutoff, certification, omission, and recoverability
`AsymptoteProof`	`InvariantImpliesBaseSafety`, `InvariantImpliesCanonicalCloseSafety`, `InvariantImpliesCloseAbortExclusive`, `InvariantImpliesAbortDominatesSealed`
`Asymptote`	executable bounded checks for transcript publication, challenge publication, canonical close, and canonical abort
`NestedGuardianProof`	witness-assignment soundness, witness-certificates-stay-assigned, `InvariantImpliesSafety`
`NestedGuardian`	executable bounded checks for reassignment, outage, and checkpoint admissibility

These formal surfaces prove the deterministic safety kernels that the broader classical-agreement bridge builds on. The full ordinary classical `99% Byzantine agreement` sentence is then completed by the recurring, reduction, totality, and collapse layers discussed above, rather than by these module-local safety kernels in isolation. Compact publication-frontier summaries internalize same-slot and predecessor consistency on the live path, while the protocol-native retrievability plane internalizes closed-bulletin extraction-or-abort as an endogenous part of the realizing architecture rather than a purely algebraic consequence of the authenticated message transcript alone.

The accountable-adversary variant therefore composes these formal kernels with runtime accountability rules rather than replacing them: replay-deduplicated objective fault publication and optional policy-controlled membership updates live in the implementation and policy layer above the proved deterministic core.

13.1 Complete omission-dominance witness artifact

The formal package includes an executable TLC witness module, `formal/aft/canonical_ordering/CanonicalOrd` paired with `formal/aft/canonical_ordering/CanonicalOrderingOmissionTrace.cfg`. The complete artifact is reproduced here so the omission-dominance witness is fully present inside this paper.

Module.

```
---- MODULE CanonicalOrderingOmissionTrace ----
EXTENDS CanonicalOrdering

TargetSlot == 1
TargetCert == {"tx1"}
TargetTx == "tx2"

WitnessState ==
  /\ TargetSlot \in closedCutoffs
  /\ bulletin[TargetSlot] = {"tx1", "tx2"}
  /\ CertEvent(TargetSlot, TargetCert) \in candidateCerts
  /\ OmissionEvent(TargetSlot, TargetCert, TargetTx) \in omissionProofs
  /\ CertEvent(TargetSlot, TargetCert) \notin admittedCerts

NoOmissionDominanceWitness == ~WitnessState

====
```

Configuration.

```
SPECIFICATION Spec

CONSTANTS
  Slots = {1}
  Transactions = {"tx1", "tx2"}

INVARIANTS
  TypeInvariant
  AdmittedSoundness
  OmissionSoundness
  RecoveredSoundness
  AdmittedUniqueness
  Recoverability
  OmissionDominates
  NoOmissionDominanceWitness
```

Executed witness trace. The module intentionally asks TLC to violate the state predicate `NoOmissionDominanceWitness` so that the checker emits a concrete public evidence trace. The executed trace is:

1. `PublishStep`: `tx1` enters slot 1's bulletin surface.
2. `PublishStep`: `tx2` enters the same bulletin surface.
3. `CloseCutoffStep`: the canonical cutoff closes and the recoverable set becomes $\{tx1, tx2\}$.
4. `CandidateCertifyStep`: a candidate certificate for $\{tx1\}$ appears.
5. `ProveOmissionStep`: an omission proof for $tx2$ against that candidate certificate appears.

The same formal package carries a bounded recursive-continuity model, `formal/aft/canonical_ordering/Cano` paired with `formal/aft/canonical_ordering/CanonicalCollapseRecursiveContinuity.cfg`. That model captures the reference `HashPcdV1` continuity carrier directly: deterministic recursive proof steps, predecessor-proof hashing, `CanonicalCollapseExtensionCertificate` carriage, and the rule that non-genesis header admission depends on the anchored predecessor proof relation. The runtime exposes a `SuccinctSp1V1` backend seam for those same recursive public inputs, and that seam is exercised by consensus verification and durable-state gating. The formal model still stops at the reference carrier rather than mechanizing a cryptographic recursive accumulator or ZK backend.

At the final state, the incomplete candidate certificate remains *unadmitted* while the omission proof is present. In the executable model this happens because the admitted object must already equal the canonical set; the trace therefore does not replace the theorem. Its purpose is narrower and useful: it gives a bounded, executed witness of the exact public-evidence shape from which omission dominance follows. Readers can run the trace directly and inspect the six-state witness without reconstructing the paper's argument by hand.

14 Implementation Correspondence

This section is non-normative. It names the exact runtime symbols that carry the protocol objects defined above.

14.1 Core ordering symbols

The core ordering data structures and free functions are public Rust symbols re-exported through `ioi_types::app`. They are the runtime carriers for the abstract ordering objects and verifier rules defined above.

Abstract object	Current runtime symbols
Bulletin surface commitment	`BulletinCommitment`, `BulletinSurfaceEntry`
Compact publication availability binding	`PublicationAvailabilityReceipt`
Compact publication frontier	`PublicationFrontier`
Publication-frontier contradiction	`PublicationFrontierContradiction`
Canonical ordering proof families	`CanonicalOrderProofSystem`, with concrete variants `HashBindingV1` and `CommittedSurfaceV1`
Canonical public-input tuple	`CanonicalOrderPublicInputs`
Live succinct witness payload	`CommittedSurfaceCanonicalOrderProof`
Omission evidence	`OmissionProof`
Order certificate	`CanonicalOrderCertificate`
Canonical ordering function	`canonical_order_tx_hashes`
Publication-frontier hashing	`canonical_publication_availability_receipt_hash`, `canonical_publication_frontier_hash`, `canonical_publication_frontier_contradiction_hash`
Publication-frontier construction / verification	`build_publication_availability_receipt`, `build_publication_frontier`, `verify_publication_frontier`, `verify_publication_frontier_contradiction`
Public-input hashing	`canonical_order_public_inputs`, `canonical_order_public_inputs_hash`
Reference proof construction	`build_reference_canonical_order_proof_bytes`
Live certificate construction	`build_committed_surface_canonical_order_certificate`
Certificate verification	`verify_canonical_order_certificate`

14.2 Guardian and sealing symbols

The guardian, witness, observer, and sealed-effect objects are likewise public Rust symbols re-exported through `ioi_types::app`. They encode the exact evidence surface checked by validators and downstream gateways.

Abstract object	Current runtime symbols
Guardian slot certificate	`GuardianQuorumCertificate`
Witness certificate	`GuardianWitnessCertificate`
Equal-authority observer assignment	`AsymptoteObserverAssignment`
Correlation budget	`AsymptoteObserverCorrelationBudget`
Observer observation request	`AsymptoteObserverObservationRequest`
Observer transcript	`AsymptoteObserverTranscript`
Observer transcript commitment	`AsymptoteObserverTranscriptCommitment`
Observer challenge	`AsymptoteObserverChallenge`, with kinds `MissingTranscript`, `TranscriptMismatch`, `VetoTranscriptPresent`, `ConflictingTranscript`, and `InvalidCanonicalClose`
Observer challenge commitment	`AsymptoteObserverChallengeCommitment`
Canonical observer close	`AsymptoteObserverCanonicalClose`
Canonical observer abort	`AsymptoteObserverCanonicalAbort`
Finality tier	`FinalityTier`, with variants `BaseFinal` and `SealedFinal`
Collapse state	`CollapseState`, with variants `Pending`, `BaseFinal`, `Sealing`, `Abort`, `SealedFinal`, `Escalated`, and `Invalid`
Sealing policy	`AsymptotePolicy`
Observer statement	`AsymptoteObserverStatement`
Observer certificate	`AsymptoteObserverCertificate`
Divergence signal	`DivergenceSignal`, `DivergenceSignalKind`
Sealed finality proof	`SealedFinalityProof`
Observer binding for sealed effects	`sealed_finality_proof_observer_binding`
Proof-carrying effect object	`SealObject`

14.3 Verifier, orchestration, and persistence symbols

Verification and orchestration entry points are split across four public namespaces: `ioi_types::app`, `ioi_crypto::sign::guardian_committee`, `ioi_services::guardian_registry`, and the `GuardianMajorityEngine` runtime. Engine-local checks are named here by their exact method identifiers.

Responsibility	**Current runtime symbols**
Guardian certificate verification	`verify_quorum_certificate`
Witness certificate verification	`verify_witness_certificate`
Deterministic observer assignment	`derive_asymptote_observer_assignments`, `derive_asymptote_observer_plan_entries`
Observer-side transcript / challenge derivation	`GuardianContainer::observe_asymptote_request`, `GuardianContainer::request_remote_asymptote_observer_observation`, `GuardianContainer::sign_consensus_with_guardian`
Sealed finality verification in the engine	`GuardianMajorityEngine::verify_asymptote_canonical_observer_sealed_finality`, `GuardianMajorityEngine::verify_asymptote_sealed_finality`
Canonical-order enrichment verification in the engine	`GuardianMajorityEngine::verify_canonical_order_enrichment`; published `CanonicalOrderAbort` dominates positive order certificates and inconsistent mixed parent state
Publication-frontier enrichment verification in the engine	`GuardianMajorityEngine::verify_publication_frontier_enrichment`; published `PublicationFrontierContradiction` dominates conflicting or stale frontiers
Block finalization attachment of order certificates	`build_committed_surface_canonical_order_certificate` during finalization
Block finalization attachment of publication frontier	`build_publication_frontier` during finalization
Canonical observer proof publication	`canonicalize_observer_sealed_finality_proof`, `publish_canonical_observer_artifacts`
Sealed-effect construction and checking	`build_http_egress_seal_object`, `verify_seal_object`
Guardianized proposal verification in the engine	`GuardianMajorityEngine::verify_guardianized_certificate`
Base-engine safety and timing	`SafetyGadget`, `Pacemaker`, `TimeoutCertificate`
Bulletin and order-certificate state access	`load_bulletin_commitment`, `load_canonical_order_certificate`
Publication-frontier state access	`GuardianRegistry::load_publication_frontier`, `GuardianRegistry::load_publication_frontier_contradiction`
Ordering close-or-abort derivation	`derive_canonical_order_execution_object`, `derive_canonical_order_public_obstruction`
Ordering abort taxonomy	`CanonicalOrderAbortReason::MissingOrderCertificate`, `BulletinSurfaceReconstructionFailure`, `BulletinSurfaceMismatch`, `InvalidBulletinClose`, `OmissionDominated`, `CertificateHeightMismatch`, `RandomnessMismatch`, `OrderedTransactionsRootMismatch`, `ResultingStateRootMismatch`, `InvalidPublicInputsHash`, `InvalidBulletinAvailabilityCertificate`, `InvalidProofBinding`
Protocol-wide collapse derivation and verification	`derive_canonical_collapse_object`, `GuardianMajorityEngine::verify_published_canonical_collapse_object`
Proposal-surface recursive continuity	`BlockHeader::previous_canonical_collapse_commitment_hash`, `BlockHeader::canonical_collapse_extension_certificate`, `CanonicalCollapseExtensionCertificate`, `CanonicalCollapseCommitment`, `verify_block_header_canonical_collapse_evidence`, `ConsensusDecision::ProduceBlock`
Bulletin, order, and collapse state access	`load_bulletin_commitment`, `load_canonical_order_certificate`, `load_canonical_order_abort`, `load_canonical_collapse_object`
	`publish_aft_publication_frontier@v1`, `publish_aft_canonical_order_artifact_bundle@v1`, `publish_aft_canonical_order_abort@v1`, `publish_aft_canonical_collapse_object@v1`, `report_aft_omission@v1`, `publish_asymptote_observer_transcript@v1`, `publish_asymptote_observer_transcript_commitment@v1`, `report_asymptote_observer_challenge@v1`, `publish_asymptote_observer_challenge_commitment@v1`, `publish_asymptote_observer_canonical_close@v1`,

These symbols are named here so that the paper maps directly onto the current runtime without making the reader chase separate prose specifications. In the implementation, validator finalization publishes a protocol-visible `CanonicalCollapseObject` through `publish_aft_canonical_collapse_object@v1` after local post-commit header enrichment, `guardian_registry` exposes `load_canonical_collapse_object`, and `GuardianMajorityEngine::verify_publi` cross-checks any published copy against the proof-carried header surface when it is available. The externally finalized AFT commit path persists the same object keyed by slot height, rather than treating ordering and sealing close-or-abort evidence as loose enrichments detached from state persistence. The same post-commit path also builds `PublicationFrontier` via `build_publication_frontier` and publishes it through `publish_aft_publication_frontier@v1`; `GuardianMajorityEngine::verify_publication_frontier_enrichment` rechecks same-slot binding, predecessor linkage, and any dominating contradiction object. In the current proposal-surface continuity slice, validator orchestration now signs both `previous_canonical_collapse_commitment_hash` and a typed `CanonicalCollapseExtensionCertificate` into each produced AFT block header. In the current clean-break runtime slice, `CanonicalCollapseObject` itself carries `continuity_accumulator_hash` and `continuity_recursive_proof`, so the certificate now carries the predecessor commitment plus predecessor proof hash rather than an explicit carried ancestor vector, full predecessor collapse object, or predecessor recursive proof object. `Asymptote` leaders stall rather than produce when that required extension certificate is unavailable, proposal verification recursively checks that the carried predecessor commitment hash matches the signed predecessor link, checks that the predecessor commitment's resulting state root matches the signed parent-state root, and rechecks those carried public inputs against the anchored predecessor collapse object when one is available. QC promotion therefore relies on a reference recursive proof-carrying continuity relation rather than on a recomputed predecessor digest alone, and only treats continuity-linked headers as progress-bearing.

15 Benchmark Context

The performance evidence in this paper has two sources:

- literature-reported baselines for `HotStuff`, `Narwhal/Tusk`, and `Bullshark`,
- the repository's native `benchmark_throughput` target for matched `GuardianMajority` and `Asymptote` scenarios.

The purpose of this section is architectural positioning, not false equivalence across unmatched environments.

15.1 Paper-reported baselines

System	**Reported throughput**	**Reported latency**	**Source type**
HotStuff baseline	1,800 tx/s	1 s	Narwhal/Tusk comparison setting
Narwhal-HotStuff	130,000 tx/s	under 2 s	Narwhal/Tusk paper
Narwhal-HotStuff with additional workers	up to 600,000 tx/s	n/a	Narwhal/Tusk paper
Tusk	160,000 tx/s	about 3 s	Narwhal/Tusk paper
Bullshark	125,000 tx/s	about 2 s	Bullshark paper

The baseline interpretation is:

- `HotStuff` demonstrates the leader-based partially synchronous baseline,
- `Narwhal-HotStuff` demonstrates the gain from separating dissemination from ordering,
- `Bullshark` demonstrates high-throughput DAG-oriented ordering,
- AFT adds a new separation: order revelation and effect release are no longer identified with dense positive voting.

15.2 Native repository measurements

The corrected native measurements below were produced on March 16, 2026 after fixing three benchmark-path issues:

- transaction gossip now retries on `InsufficientPeers`,
- genesis status starts from a shared configured wall-clock timestamp,
- the orchestrator retries shared-memory attachment before falling back.

The table separates a burst-latency probe from a bulk-throughput probe.

scenario	validators	attempted	accepted	committed	blocks	inj. tps	sust. tps	p50	p95	p99	sealed p50	sealed p95
guardian_majority_4v_burst	4	512	512	512	3	11214.96	36.43	487.49	11390.70	11398.07	–	–
guardian_majority_4v_bulk	4	8192	8192	8192	2	13048.99	252.66	9276.35	32416.37	32419.21	–	–

These measurements imply four things:

- the earlier single-digit TPS report was an artifact of benchmark-path bugs,
- ingress is not the immediate bottleneck,
- the current repository is bottlenecked more by commit and state handling than by raw publish rate,
- the implementation remains far from the intended `8k-16k TPS` target.

These numbers should therefore be read as corrected repository baselines, not as optimized ceilings and not as apples-to-apples claims against the published Narwhal/Bullshark environments.

16 Related Work

Geeq's Proof of Honesty (PoH) is important prior work in the "99%" direction: it argues for globally honest and canonical chain recovery in pure software under assumptions including at least one honest node and an honest relay, and it uses public evidence plus user-side verification to push beyond ordinary dense-vote intuition [9]. AFT differs in a key respect: it is relay-free and coordinator-free. Correctness does not depend on any designated honest relay, broadcaster, or privileged recovery role. Instead, durable ordering and sealed effects collapse through protocol-native public-evidence objects whose negative forms are immediately decisive.

Narwhal/Tusk and Bullshark separate dissemination from ordering, but they still ground ordering safety in dense positive certificates and quorum-intersection arguments over the ordered object itself [2, 3]. HoneyBadgerBFT and Dumbo move to asynchronous common-subset or MVBA-style agreement, but they still rely on classical Byzantine-majority thresholds rather than a public-state-continuity theorem over a canonical public surface [4, 5]. Chainlink CCIP and optimistic cross-chain mechanisms add secondary monitoring layers, risk-management committees, or challenge windows to detect and halt bad transfers, but the decisive recovery rule is operational or economic rather than the immediate acceptance of a uniquely valid proof-carrying canonical object recoverable by any equal-authority validator [8]. Accountable subgroup multisigs and finality gadgets add equivocation evidence and slashing, yet still keep dense subgroup votes as the positive safety object [6, 7]. AFT differs by making ordering and sealed release relay-free public-evidence close-or-abort objects with dominant negative proofs: honest validators that observe the same closed public boundary derive the same decisive object, and publication is downstream of that deterministic derivation rather than its source.

17 Claims and Non-Claims

17.1 Claims

This paper claims:

- a production guardian-backed base-finality model in which validator quorums are composed with guardian committee admissibility,
- proof-carrying canonical ordering with omission dominance and uniqueness, plus protocol-native endogenous retrievability,
- compact signed publication-frontier summaries and short contradiction objects that internalize same-slot and predecessor consistency on the live path,
- a public-state-continuity formulation in which closed slot boundaries admit at most one positive public execution object and conflicting candidates admit short objective obstruction witnesses,
- `99%` equal-authority ordering consensus under explicit assumptions,
- deterministic sealed-effect collapse with challenge dominance,
- a relay-free, coordinator-free, pure-software public-state-continuity theorem yielding deterministic `99% Byzantine Tolerance` for durable ordering and sealed-effect safety over the explicit bulletin / retrievability substrate,
- that the whole AFT stack universally breaks the lower bound with no qualifiers,
- a witness-augmented mode for layered threshold constructions,
- machine-checked deterministic safety kernels for the major protocol surfaces.

17.2 Non-claims

This paper does **not** claim:

- that public state continuity arises for free inside the classical DLS / PBFT partially synchronous message-passing model,
- freedom from bulletin-publication, protocol-native retrievability obligations, or proof-soundness assumptions,
- that compact publication-frontier summaries make the bulletin surface self-extracting inside the classical authenticated-message model,
- a full asynchronous liveness theorem under arbitrary registry, log, witness, and outage schedules,
- an apples-to-apples measured throughput superiority over `HotStuff`, `Narwhal`, `Tusk`, or `Bullshark` from the current repository alone.

17.3 Public-state-continuity theorem

The runtime supports a stronger theorem than the older accountable revelation phrasing, provided the substrate model is stated honestly.

Theorem 6 (Relay-Free Deterministic 99% Byzantine Tolerance). *Assume the system model of Section 5, guardian non-equivocation, proof soundness, compact signed publication-frontier summaries whose same-slot and predecessor-link contradictions are objectively checkable, protocol-native retrievability objects plus canonical bulletin-close extraction-or-abort for the bulletin surface bound by those frontiers, deterministic transcript/challenge derivation for sealing, durable state advancement only through canonical collapse, and irreversible-effect release only through collapse-bound sealed proofs. Then up to* 99% *of validators may behave arbitrarily without creating a conflicting valid durable canonical-order outcome or a conflicting valid sealed-effect release outcome. Honest validators that observe the same closed public boundary derive the same close-or-abort result; publication, penalties, and validator-set updates only accelerate convergence and future hygiene. This is unconditional classical Byzantine agreement in the ordinary permissioned model, realized through the public-state-continuity substrate rather than through dense positive quorum intersection.*

The theorem is stronger than the older accountable revelation summary because invalid positive objects are not merely rejectable or penalty-bearing. They are already dominated by deterministic negative objects and collapse-gated durable state before any penalty logic runs. Compact publication frontiers keep the live hot path cheap by internalizing consistency and lineage checks, while the protocol-native retrievability plane discharges deterministic extraction-or-abort on the full bulletin surface.

Theorem 7 (Realizing-Mechanism Statement). *The new publication-frontier slice is not the theorem by itself; the theorem is realized by the full AFT public-state-continuity architecture. Compact signed publication-frontier summaries, compact availability receipts, short frontier contradiction objects, endogenous retrievability profiles, manifests, custody assignments, custody receipts, custody responses, retrievability challenges, canonical extraction-or-abort, canonical collapse, and historical retrievability together realize the same unconditional classical* `99% Byzantine agreement` *sentence stated above. Removing that protocol-native retrievability plane removes part of the realizing mechanism rather than merely shaving off an optional refinement.*

Proposition 2 (Scoped Witness-Coded Constructive Corollary). *Assume the hypotheses of the relay-free deterministic* `99% Byzantine Tolerance` *theorem on the explicit public-state-continuity substrate, plus sound witness assignment and witness certificate binding, a completed witness-coded* *recovery family with compact hot-path bindings and cold-path share delivery / reveal / reconstruction, and availability of the retained recovered publication surface together with any older recovered-history pages needed for restart-time extraction. Then the* `NestedGuardian` *witness-coded carrier deterministically materializes the same positive close or canonical abort surface, the same authoritative collapse chain, the same replay-prefix / restart-anchor surface, and the same restart-time canonical-header / certified-header / block-header ancestry surface as the ordinary lane for the completed recovery families. Restart can therefore stream recovered certified ancestry until target height or recovered-history exhaustion without a theorem-relevant fixed depth bound, while keeping only compact bindings on the hot path and bounded-memory paging on the restart path.*

This runtime result is stronger than the separation closeout because it no longer stops at "compact frontiers bind a recoverability claim." It shows that the live AFT stack can re-materialize the same durable and restart-critical surfaces without violating the throughput guardrails, while keeping deeper recovered history inside ordinary endogenous AFT state.

This should now be read as one constructive route to classical partially synchronous Byzantine agreement rather than as a weaker contrast class. In the classical dense-vote presentation, worst-case progress is usually phrased in terms of $> 2/3$ honest participation and quorum intersection as the main safety mechanism. In AFT's PSC realization, the same top-line sentence is obtained instead through uniquely valid public-evidence objects plus dominant negative proofs, without trusted relays, privileged coordinators, TEEs, or dense positive quorum intersection. The live publication-frontier slice internalizes compact same-slot and predecessor consistency on the hot path, while explicit recoverability of the bound bulletin surface, canonical collapse, and historical retrievability complete the same ordinary agreement sentence. Publication and accountability serve as reinforcement rather than the source of correctness. Liveness then follows from the availability of the PSC substrate and eventual publication of the protocol-defined artifacts, and the recurrence/reduction/totality/collapse chain closes that route formally.

The correct one-line summary is:

> AFT is a proof-carrying, guardian-backed, omission-dominant consensus and settlement family in which base finality, canonical ordering, and irreversible-effect release are separated and then recomposed through shared evidence.

References

[1] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. *HotStuff: BFT Consensus in the Lens of Blockchain.* arXiv:1803.05069. https://arxiv.org/abs/1803.05069

[2] George Danezis, Lefteris Kokoris-Kogias, Alberto Sonnino, and Alexander Spiegelman. *Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus.* arXiv:2105.11827. https://arxiv.org/abs/2105.11827

[3] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. *Bullshark: DAG BFT Protocols Made Practical.* arXiv:2201.05677. https://arxiv.org/abs/2201.05677

[4] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. *The Honey Badger of BFT Protocols.* Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. https://eprint.iacr.org/2016/199.pdf

[5] Bingyong Guo, Zhenliang Lu, Qiang Tang, Jing Xu, and Zhenfeng Zhang. *Dumbo: Faster Asynchronous BFT Protocols*. Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. https://dblp.org/rec/conf/ccs/GuoL0XZ20

[6] Alistair Stewart and Eleftherios Kokoris-Kogias. *GRANDPA: a Byzantine Finality Gadget*. arXiv:2007.01560. https://arxiv.org/abs/2007.01560

[7] Jelena Dosenovic, Roger Wattenhofer, and others. *Accountable Safety Implies Finality*. arXiv:2308.16902. https://arxiv.org/abs/2308.16902

[8] Chainlink Labs. *Chainlink CCIP's Defense-In-Depth Security and the Risk Management Network*. Chainlink blog, 2023. https://blog.chain.link/ccip-risk-management-network/

[9] John P. Conley. *Proof of Honesty: Coalition-Proof Blockchain Validation without Proof of Work or Stake*. Geeq technical paper, version 2.0, 2019. https://geeq.io/wp-content/uploads/GEEQ-OFFICIAL-TECHNICAL-PAPER.pdf

[10] Sean Bowe, Jack Grigg, and Daira Hopwood. *Halo: Recursive Proof Composition without a Trusted Setup*. IACR ePrint 2019/1021. https://eprint.iacr.org/2019/1021

[11] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. *Proof-Carrying Data from Accumulation Schemes*. IACR ePrint 2020/499. https://eprint.iacr.org/2020/499