NabdBFT: A Heartbeat-Driven, Human-Gated Layer-1 with Deterministic Finality for Sovereign Digital Economies

Khaled AlThameen

Abstract: **NABD is a layer-one blockchain for sovereign digital economies that require deterministic settlement, operational resilience, and accountable human authorization. Its consensus protocol, NabdBFT, runs four phases (PROPOSE, VOTE, CERTIFY, COMMIT) under a $Q = \lfloor 2N/3 \rfloor + 1$ quorum with view-scoped certify locks. A heartbeat liveness plane, decoupled from the consensus round, classifies a silent validator as suspect after $T_{\text{suspect}} = 1.5\,\text{s}$ and dead after $T_{\text{dead}} = 2.3\,\text{s}$, and starts view change on a dead leader without waiting for the round's phase timers to lapse. In laboratory leader-kill tests with these thresholds, crash failover completes in about 3.3 s. Because this figure is a function of the configured thresholds rather than of the protocol family, the paper limits the claim to crash faults, which heartbeats can observe, and compares against reactive protocols only under stated timeout configurations. On a live public testnet of eight validators hosted two per machine on four shared two-vCPU instances, hash-confirmed public-ingress finality averaged 1.214 s (p50 1.197 s, p95 1.436 s, $n = 30$). Above the consensus layer, NABD makes policy-gated human authorization a ledger primitive: biometric capture, matching, and liveness detection run inside trusted devices or oracles, and validators verify only signatures from registered attesting keys together with commitments and signed policy effects. The ledger also provides native multi-asset state, deterministic autonomous-program execution in WebAssembly, and a scheme-agile signature envelope (Ed25519, Falcon-512, hybrid) that is post-quantum-ready but not FIPS 206 validated. The network is live at testnet.nabdchain.org.**

***Index Terms*—blockchain, Byzantine fault tolerance, deterministic finality, liveness, failure detectors, self-healing networks, biometric authorization, autonomous programs, post-quantum cryptography, finality measurement, digital identity**

2.3 s	**≈3.3 s**	**1.214 s**	$N{=}8$
crash-leader detection	crash failover (lab)	mean finality ($n{=}30$)	4 hosts, live testnet

I. Introduction

Digital economies include people, organizations, public services, and a growing population of autonomous machines. The ledger underneath them has to be final, auditable, inexpensive to operate, and able to keep going when ordinary computers fail. Where the stakes are high, it also has to bind actions to a present human without turning human bodies into public data.

Classical BFT protocols meet the safety requirement. Their recovery from a failed leader, however, is driven by round timers: PBFT, Tendermint, and HotStuff begin view change only after a timeout expires, and the length of that timeout is a deployment choice that trades false leader changes against stall time. NABD adds a second, faster signal for the common failure. A heartbeat subsystem tracks every validator's health at a cadence independent of the consensus round, so a crashed or silent leader is classified as dead within a bounded T_{dead} and view change begins on that classification. The heartbeat plane never touches the commit rule: it decides when to change leaders, and quorum-backed certificates decide whether a block is safe.

This paper describes NabdBFT and the ledger built on it, and reports evidence from a live testnet. Its contributions are:

- a heartbeat liveness plane that detects a crashed leader within T_{dead} and triggers view change ahead of the round's phase timers, with an explicit failover bound $T_{\text{failover}} \le T_{\text{dead}} + \Delta_{vc} + T_{\text{phase}}$ and a laboratory crash failover of about 3.3 s (Sections IV and IX);
- a four-phase BFT path whose CERTIFY step materializes quorum evidence for catchup and view change, with a quorum-intersection safety argument, an explicit liveness argument, and bounded TLA$^+$ model checking (Section V);
- a ledger model with canonical transaction hashing, native assets, and WebAssembly autonomous programs under deterministic state rules (Section VI);
- a human-authorization primitive in which biometric matching stays outside chain state and the chain verifies attesting-key signatures over transaction-bound commitments (Section VII);
- a public-ingress finality measurement that anyone can repeat from observable protocol artifacts (Section IX).

Sections IX and X state which claims are measured, which are modeled, and which are design only.

II. Background and Related Work

NABD belongs to the state-machine-replication tradition for Byzantine environments. PBFT established the practical three-phase pattern and the quorum-intersection argument reused by later systems [1]; BFT-SMaRt turned that line into a reusable replication library with reconfiguration and production-oriented engineering [2]. HotStuff reduced leader-change cost through chained quorum certificates and linear communication [3], and Tendermint adapted BFT voting to blockchain operation with proposer rounds, validator sets, and immediate finality under partial synchrony [4]. NabdBFT claims no lower asymptotic message complexity than these protocols. Its contribution is an explicit liveness-repair plane

K. AlThameen is with the NABD Foundation. Corresponding author: khaled@nabdchain.org. December 2025.

around a conservative quorum path, together with a separate CERTIFY evidence boundary used by catchup and view-change logic.

The liveness model follows the failure-detector view of distributed computing. Chandra and Toueg showed that unreliable failure detectors suffice for consensus when their assumptions are stated precisely [5]; heartbeat-based leader detection in the same spirit is standard in crash-tolerant replication. NABD keeps its alive, suspect, dead, and recovered labels outside the commit predicate: a label can start repair or view change, but it cannot lower the quorum size or replace a signature. The safety model follows the TLA$^+$ practice of specifying invariants before making implementation claims [6], [7].

Hyperledger Fabric separates endorsement, ordering, validation, and commit, and treats deterministic validation and policy as first-class concerns [8]; its private data collections keep raw payloads off the public channel while committing hashes for audit [9]. NABD uses a different consensus engine and a single native ledger model, but it adopts the same boundary: public state carries commitments and policy effects, while private material stays behind an explicit interface.

Algorand uses verifiable random functions for cryptographic sortition [10], and RFC 9381 standardizes ECVRF [11]. NABD implements VRF leader selection but does not activate it in the current deployment; Section VIII states the gating and migration status.

For post-quantum authentication, NIST FIPS 204 standardizes ML-DSA [14], and FN-DSA, the Falcon-derived scheme with smaller signatures [12], [13], is being standardized as FIPS 206 [15]. NABD implements Falcon-512, the concrete liboqs parameter set corresponding to FN-DSA-512; its status relative to FIPS 206 is stated in Section VI-D.

Biometric authorization draws on prior work on biometric privacy and revocation: templates, feature extraction, matching, and public audit data must be separated because a compromised body signal cannot be rotated like a password [16], [17]. Autonomous programs in NABD execute in WebAssembly, following its design goal of a portable, sandboxed execution format [18].

III. System and Threat Model

A. Validators and Quorum

Let N be the number of validators in the active validator set and $f = \lfloor (N-1)/3 \rfloor$ the Byzantine fault bound. The quorum size is

$$Q = \left\lfloor \frac{2N}{3} \right\rfloor + 1 = N - f, \tag{1}$$

where the identity holds for every N (write $N = 3f + 1 + k$ with $k \in \{0, 1, 2\}$ and expand). Any two quorums Q_1 and Q_2 satisfy

$$|Q_1 \cap Q_2| \geq 2Q - N = N - 2f \geq f + 1. \tag{2}$$

A block is final when the protocol observes Q valid certify messages for it under the active validator set and validity rules. There is no probabilistic confirmation depth.

B. Network Assumptions

NABD targets known-validator networks under the partial-synchrony assumption used by BFT protocols [1], [4]: messages between correct validators may be delayed arbitrarily before an unknown global stabilization time (GST), after which they are delivered within a bound δ. Safety never depends on timing. Liveness depends on timing only through the timer assumptions stated in Section V-B.

C. Fault Model

Up to f validators may be Byzantine in any round. The heartbeat plane distinguishes two fault classes that the safety argument does not need to distinguish. A crashed or partitioned validator stops sending all messages, including heartbeats, and is observable by silence. A Byzantine leader may keep sending heartbeats while withholding or delaying proposals, and is observable only through the phase timers. Heartbeat detection therefore accelerates recovery from crash faults; Byzantine slow-leader faults recover on the reactive path.

D. Host Co-location

In the current testnet two validators share each host (Section IX-A). A host failure or compromise removes or subverts two validators at once, so the effective host-level fault tolerance of the $N = 8$ deployment is one host, and the two validators on a host must be assumed to fail together. Validator keys on the same host share one trust boundary. The protocol-level figure $f = 2$ is correct, but it should not be read as tolerance of two independent host faults.

E. Biometric Oracle Trust

Biometric matching and liveness detection take place inside a capture device or oracle that the chain cannot inspect. The chain trusts the key registered for that device or oracle; compromise of the device, its key, or the governance that registers keys defeats the biometric gate for every action that key can authorize. Higher-assurance policies reduce this exposure with M-of-N attesting keys and recovery delays (Section VII-C).

IV. NabdBFT

A. Round Structure

NabdBFT orders transactions in rounds. Each round r has a view v and a leader $L(r, v)$ drawn from the active set; the current deployment uses round-robin leader selection. Fig. 1 shows one round and Fig. 2 gives the corresponding steps.

1) PROPOSE: the leader builds a candidate block for (r, v) with height, round, view, parent hash, transaction root, state precondition, and signature.
2) VOTE: each validator checks chain identity, proposal signature, parent, state precondition, transaction validity, account nonces, balances, fee policy, and asset and autonomous-program rules, then votes for the block hash $H(B)$ scoped to (r, v).

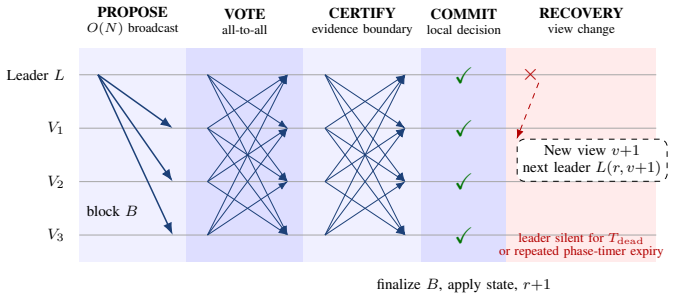

Fig. 1. One NabdBFT round over a leader and three replicas (representative of the active set). PROPOSE is a single $O(N)$ broadcast; VOTE and CERTIFY are all-to-all $O(N^2)$ phases; COMMIT is a local decision on Q certify messages. CERTIFY is an evidence boundary: it materializes the quorum that observed B so catchup and view change can verify it directly. The recovery branch is entered either on a heartbeat classification of the leader as dead or on repeated phase-timer expiry; in both cases commit still requires quorum-backed certify evidence.

Round algorithm: NabdBFT round r, view v

1) Select leader $L(r, v)$ from the active validator set.
2) L proposes block B with height, round, view, parent hash, transaction root, state precondition, and signature.
3) Each validator checks chain identity, proposal signature, parent, state precondition, transaction validity, account nonces, balances, fee policy, and autonomous-program (AP) or asset rules.
4) A valid proposal produces a signed vote for $H(B)$ scoped to (r, v).
5) A validator that observes $\geq Q$ valid votes for $H(B)$ emits a signed certify message and records a view-scoped lock.
6) A validator that observes $\geq Q$ valid certifies commits B, applies the deterministic state transition, records the post-state hash, and advances to $r + 1$.
7) If the heartbeat plane marks the leader dead, or phase timers repeatedly expire without quorum evidence, validators enter view change; the next view must preserve the highest valid lock known to quorum evidence.

Fig. 2. Round algorithm corresponding to Fig. 1. Steps 1–6 map to PROPOSE, VOTE, CERTIFY, and COMMIT; step 7 is the recovery branch.

3) CERTIFY: a validator that observes at least Q valid votes for $H(B)$ emits a signed certify message and records a view-scoped lock on $H(B)$.
4) COMMIT: a validator that observes at least Q valid certify messages commits B, applies the deterministic state transition, records the post-state hash, and advances to round $r + 1$.

Each phase carries a timer $T_{\text{phase}} = 0.75\,\text{s}$. The round rhythm is chosen to be close to human interaction time, about one second, which suits wallet payments, asset transfers, device checkpoints, agent commitments, and identity events. Chain time records quorum acceptance of an ordered event; physical-world assurance is supplied by device security, capture quality, oracle policy, and governance.

B. Why CERTIFY Is Explicit

The CERTIFY phase is an evidence boundary rather than a latency optimization. It materializes the set of validators that observed and accepted a candidate before commit, gives late or recovering validators a compact certificate to verify during catchup, and makes the view-change lock visible to the repair plane. A three-step design can merge this evidence into neighboring phases, at the cost of reconstructing during catchup and view change which validators observed which candidate. NabdBFT pays one extra quorum-observation step to keep the boundary explicit. The latency cost of that extra step has not yet been measured in isolation (Section IX-D).

TABLE I
VALIDATOR LIVENESS STATES

State	Meaning	Protocol role
Alive	Recent activity observed	Normal voting and peer health
Suspect	Silence exceeds $T_{\text{suspect}} = 1.5\,\text{s}$	Early liveness signal
Dead	Silence exceeds $T_{\text{dead}} = 2.3\,\text{s}$	View change and repair signal
Recovered	Activity resumes	Peer returns to normal operation

C. View Change

A validator enters view change for round r when either (i) the heartbeat plane classifies the current leader as dead, or (ii) phase timers expire repeatedly without the evidence needed to advance: five consecutive phase-timer failures while the leader is not heartbeat-suspect, two while it is suspect, with a 4.0 s cooldown between requests; a separate stuck detector requests view change after 60 s without progress. It broadcasts a signed view-change message (`nabd_view_change`) for $(r, v + 1)$ carrying its highest certify lock. The leader $L(r, v + 1)$ forms the new view from a quorum Q of view-change messages, carried as a new-view certificate (`nabd_new_view_certificate`) that proposals in the new view must present; an $f+1$ threshold serves only as a convergence trigger and never authorizes a view change or a proposal. The leader must re-propose the highest valid lower-view lock present in that evidence; a validator that holds a lock votes only for a proposal consistent with that lock or with higher-view quorum evidence. Phase timers are fixed and do not back off across consecutive failed views; the operating-envelope consequences are stated in Section V-B.

D. Heartbeat Liveness Plane

The protocol adds a liveness plane around the quorum path. Every validator emits a heartbeat every $T_h = 0.5\,\text{s}$, and any message from a peer refreshes that peer's liveness, so consensus participation doubles as a liveness signal. A peer silent for $T_{\text{suspect}} = 1.5\,\text{s}$ is marked suspect and one silent for $T_{\text{dead}} = 2.3\,\text{s}$ is marked dead; renewed activity returns it through recovered to alive (Table I, Fig. 3). The classifier changes repair behavior and view selection only. It is not an input to block validity, it does not replace signatures, and it never lowers Q (Lemma 3).

When the current leader is marked dead, the validator enters view change immediately rather than waiting for the round's phase timers. Failover time for a crashed leader is therefore bounded by

$$T_{\text{failover}} \leq T_{\text{dead}} + \Delta_{vc} + T_{\text{phase}}, \tag{3}$$

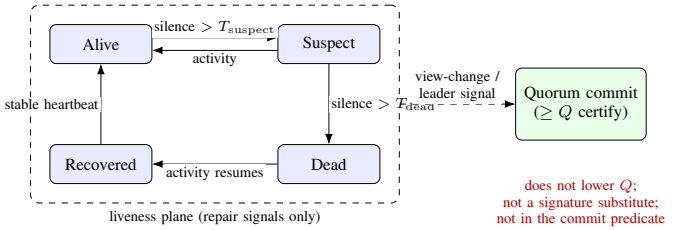

Fig. 3. The four-state liveness classifier (Table I) is a self-healing repair plane around the quorum path. It can request a view change or change leader selection, but by Lemma 3 it never lowers Q, never replaces signatures, and never enters the commit predicate.

where Δ_{vc} is view-change propagation time. With $T_{\text{dead}} = 2.3\,\text{s}$ and $\Delta_{vc} \approx 0.7$–$1.0\,\text{s}$ the bound is roughly 3.8–4.1 s; repeated laboratory leader-kill tests observed failover of about 3.3 s (Section IX-D).

Two limits should be read alongside that figure. First, the bound is a property of the chosen thresholds, not of the protocol family: a reactive protocol configured with a 2.3 s round timeout would begin recovery at a comparable time. What the heartbeat plane changes is the coupling. Detection time is fixed by T_{dead} regardless of round length, block size, or where in the round the failure occurs, and it does not grow if round timeouts are lengthened for large blocks. Second, heartbeats observe silence only. A Byzantine leader that heartbeats but withholds proposals is caught by the phase timers, so the worst-case failover for Byzantine leaders is the reactive bound. The thresholds must also stay above the post-GST message delay plus T_h (Section V-B); otherwise correct leaders are falsely suspected and view changes churn.

E. Message Complexity

For a validator set of size N, PROPOSE is one $O(N)$ broadcast plus block bytes. VOTE and CERTIFY are all-to-all, $N(N-1)$ signed messages each. COMMIT is a local decision; validators additionally broadcast commit evidence so that lagging peers can catch up, which adds at most another $N(N-1)$ messages. A round therefore costs at most $3N(N-1)$ small control messages, $O(N^2)$ overall, and $O(NB + N^2S)$ bytes, where B is block payload size and S the size of signed control evidence. Heartbeats add a separate $O(N^2)$ stream per T_h that does not enter the commit predicate. For $N = 8$ the three all-peer message types account for at most 168 messages per round before retries, catchup, or client relay traffic.

V. Safety, Liveness, and Model Checking

A. Safety

The safety argument is deliberately separated from the liveness classifier. Liveness signals may request repair or a new view, but they do not lower quorum size, change transaction validity, or authorize commit without a certificate.

Lemma 1 (Quorum intersection): Let $f = \lfloor (N-1)/3 \rfloor$ and $Q = \lfloor 2N/3 \rfloor + 1 = N - f$. Any two quorums intersect in at least $f + 1$ validators.

Proof: Two quorum sets Q_1 and Q_2 have $|Q_1| = |Q_2| = N - f$, so $|Q_1 \cap Q_2| \geq |Q_1| + |Q_2| - N = N - 2f$. Since $N \geq 3f + 1$, $N - 2f \geq f + 1$. At most f validators are Byzantine, so the intersection contains at least one correct validator. ■

Lemma 2 (View-scoped locking): A correct validator does not certify two different block hashes for the same round, including across views. When a higher view is formed, the proposal path preserves the highest valid lower-view lock known to quorum evidence at proposal time.

Argument. The implementation and the TLA$^+$ model represent correct-validator vote and certify uniqueness together with per-validator certify locks across views. A correct validator that has locked a hash releases or advances that lock only through higher-view evidence that is itself quorum-backed. Two conflicting hashes therefore cannot both collect valid certify evidence unless a correct validator violates its locking rule.

Lemma 3 (Liveness-plane safety preservation): The alive/suspect/dead/recovered classifier cannot create a conflicting commit.

Argument. Liveness classification changes repair behavior and view selection only. It is not an input to the block validity predicate, it is not a replacement for signatures, and it does not enter the commit predicate as a quorum substitute. A suspect or dead signal can start recovery, but the recovered path must still present ordinary quorum evidence before commit. The lemma holds by construction; it says nothing about progress, which is the subject of Section V-B.

Theorem 1 (Agreement for $N \geq 3f + 1$): For any active validator set with $N \geq 3f + 1$ and at most f Byzantine validators, two correct validators cannot commit different valid blocks for the same round.

Proof sketch: Assume two correct validators commit conflicting hashes h and h' for the same round. Each commit requires quorum-backed certify evidence. By Lemma 1, the two quorums intersect in at least one correct validator. By Lemma 2, that validator cannot certify both hashes in the same round and view, and cannot carry a conflicting lock into a higher view without quorum-backed evidence. By Lemma 3, liveness repair cannot bypass the certificate path. The assumption contradicts correct-validator locking and quorum intersection. ■

B. Liveness Under Partial Synchrony

The previous revision argued only that the liveness plane preserves safety. This section states the assumptions under which NabdBFT also makes progress, and the operating envelope they induce for the current fixed-timer configuration.

(A1) After GST, messages between correct validators arrive within δ, and $\delta + T_h < T_{\text{suspect}}$; with the current values this requires $\delta < 1.0\,\text{s}$.

(A2) At most f validators are faulty, and leader selection is round-robin over the active set, so at most f consecutive views of a round have a faulty leader.

(A3) Phase timers are fixed at $T_{\text{phase}} = 0.75\,\text{s}$ and do not grow across consecutive failed views; view-change requests are rate-limited (Section IV-C), and heartbeat labels can accelerate but are never required for a view-change request.

Argument. After GST, (A1) implies that no correct validator marks a correct, live leader suspect or dead, because the leader's heartbeats arrive with gaps of at most $T_h + \delta < T_{\text{suspect}}$. Liveness-driven view changes are then triggered only by leaders that are in fact silent. By (A2), a correct leader is selected within $f + 1$ views. Because the timers are fixed (A3), progress additionally requires the post-GST delay to fit the phase budget: when δ plus processing stays below T_{phase}, each of PROPOSE, VOTE, and CERTIFY completes within its timer, so the correct leader's view collects view-change evidence, re-proposes the highest lock, and gathers Q votes and Q certifies from the $N - f \geq Q$ correct validators. Correct validators vote for that proposal because it carries the highest lock any of them could hold (Lemma 2), so the view completes and the round commits. Every round therefore eventually commits.

The argument is informal, and it is conditional on the fixed-timer delay envelope of (A3), which the bounded TLA$^+$ models do not capture because they model no timing. The operating envelope follows directly: sustained one-way delay above 1.0 s between validators, or a validator process that stops emitting heartbeats for that long while busy, produces false suspicion of live leaders, while sustained delay plus processing above the 0.75 s phase budget stalls rounds without any false suspicion; guaranteed progress requires both conditions to hold with margin. A second consequence concerns the event loop. Heartbeat emission shares the node's single poll loop with transaction validation, so a validation burst that occupies the loop longer than T_{suspect} delays heartbeats and can cause false suspicion; the largest end-to-end finality observed on the testnet (2.179 s, Section IX-C) is within 0.12 s of T_{dead}. Per-tick work is therefore rate-limited where the implementation permits it (commit-certificate verification and catchup process one item per tick), and keeping heartbeat emission unblocked by batch validation is an explicit engineering requirement for the current configuration.

C. Model Checking

The TLA$^+$ model inventory under `formal/tla/nabdbft/` covers a main safety model and a smaller view-reset regression model. On 2025-06-04, TLC 2.19 completed the $N = 4$, $F = 1$ baseline with $Q = 3$, `MaxRound=0`, `MaxView=1`, and two modeled block hashes. The run checked `TypeOK`, `Agreement`, `Validity`, `VotingUniqueness`, and `CorrectLeaderReproposal`, and completed without invariant violations after 2,611,170,154 generated states, 230,597,253 distinct states, depth 60, and zero states left on the queue (Table II).

The `CorrectLeaderReproposal` invariant is proposal-time based: a correct leader that observes lower-view lock evidence when it proposes must re-propose that locked hash. This avoids a false failure in the pool-free model where a lower-view certificate appears after a higher-view proposal and would otherwise retroactively invalidate an earlier correct proposal. The view-reset liveness model completed with 42 generated states, 21 distinct states, and depth 9. The expected-fail reset diagnostic violates `NoSameViewReset` when the unsafe reset behavior is enabled, which confirms that the check is live.

TABLE II
TLA$^+$ EVIDENCE SUMMARY

Model	Result	Evidence
$N = 4$, $F = 1$ safety baseline	Complete pass	2.611 B generated, 230.6 M distinct, depth 60
$N = 4$, $F = 1$ rerun (fp=64)	Complete pass	collision risk 3.5×10^{-4}
View reset liveness	Complete pass	42 generated, 21 distinct, depth 9
View reset regression	Expected fail	`NoSameViewReset` diagnostic violation
$N = 7$, $F = 2$ sweep	Bounded partial	12.28 B generated, 1.18 B distinct, depth 13, disk exhausted

These TLC runs are bounded model-checking evidence. They support the proof sketch above but do not replace a theorem-prover result for all validator-set sizes. The $N = 4$, $F = 1$ baseline was independently rerun (fp=64, seed 123456789) and reproduced the same completion signature with a reduced fingerprint-collision estimate of 3.5×10^{-4} (down from 1.4×10^{-2} in the original fp=37 run). A larger $N = 7$, $F = 2$ production-geometry sweep ran for about 10.2 hours on a lab VM and explored 12.28 billion generated and 1.18 billion distinct states to depth 13 with zero invariant violations before terminating on disk exhaustion, with the state queue still growing (about 752 million states). That run supports the claim that the production geometry admits no small-counterexample safety violation within the explored frontier; it is not exhaustive verification. Neither model includes the heartbeat plane or timers, so the liveness argument of Section V-B is currently unsupported by model checking.

VI. LEDGER AND EXECUTION

A. Accounts, Canonical Hashing, and Replay Protection

NABD uses account-based state with NABD1 account addresses. Transactions are signed over canonical payloads so that every validator verifies the same meaning. Blocks are protocol-versioned, and asset state participates in deterministic ledger state.

Transaction validation binds the signed payload to the account sequence and the chain identity. Validators verify canonical transaction meaning, account nonce, sufficient balance, transaction type, fee policy, asset rules, and signature scheme. A previously accepted nonce cannot be replayed for the same account. Cross-chain replay is blocked by signing the network identifier and genesis hash into the protocol payload: the transaction hash is derived from the signed payload, validators reject mismatched chain identity, and a valid signature from one NABD network has no standing on another network or on a fork with a different genesis identity.

B. Native Assets

Native asset operations include creation, minting, transfer, and burn. Fungible assets represent community money, credits, points, quotas, or regulated asset units. Unique units represent

certificates, tickets, access passes, inventory receipts, machine identities, or claims on physical objects. Asset rules are checked in the VOTE phase alongside ordinary transaction validity, so a block that violates an asset rule never collects a quorum.

C. Autonomous Programs

Autonomous programs execute as WebAssembly modules through the WASM3 interpreter, reached from the host layer through native bindings rather than a native-code JIT. The host layer enforces the manifest, owner, code hash, ABI hash, capabilities, storage limits, fee limits, event limits, return-size limits, and upgrade and freeze policy. Deployments and calls are signed transactions, so the ordinary nonce, replay, chain-identity, and fee checks apply before execution.

The execution model is intentionally narrow. Programs interact with ledger state through a bounded host API that covers balance lookup, transfer, storage read and write, and state recording. High-assurance deployments can require a policy grade in which machine signatures and a human biometric signature are both present before deployment, or before a critical call crosses its configured boundary (Section VII-D).

Determinism requirement. Execution is metered by a deterministic instruction budget: the WASM3 interpreter carries a NABD patch that charges per-instruction fuel, every call runs under a manifest-declared fuel limit, and repository tests check that the fuel accounting is reproducible across runs. The current build additionally enforces a wall-clock timeout in the forked executor as a node-local defense. A wall-clock guard is not deterministic across differently loaded hosts, so it must never decide the committed outcome of a call: today a timeout aborts the local block write, and the node recovers by retry and catchup rather than by committing a divergent result. Production hardening requires that every outcome-affecting bound be the deterministic fuel budget, with the wall-clock guard confined to node-local defense.

D. Signature Envelope and Post-Quantum Readiness

NABD uses a scheme-aware signature envelope so that transaction authorization can evolve across algorithms. The same canonical payload can be authenticated under Ed25519, Falcon-512, or a hybrid policy that carries both. The envelope keeps the ledger model stable while allowing stronger signature requirements where a community or institution chooses them (Table III, Fig. 4).

Falcon-512 support is active in the implementation through the `signatures[]` envelope. Hybrid transactions carry a Falcon-512 signature and an Ed25519 signature over the same canonical payload, preserving compatibility with existing wallets while allowing post-quantum-ready policy. Repository tests cover Falcon-512 key generation, Falcon-only transaction signing, and hybrid Ed25519 plus Falcon transaction signing.

Naming and status are kept precise. FN-DSA is the standard family name under FIPS 206; Falcon-512 is the concrete liboqs parameter set the runtime executes. At the time of writing (December 2025), FIPS 206 remains a draft standard, with final publication expected in late 2026 or early 2027 [15].

TABLE III
SIGNATURE ENVELOPE CHARACTERISTICS

Mode	Size evidence	Role
Ed25519	64 B signature	Compatibility and fast verification path
Falcon-512 (FN-DSA-512)	897 B public key; 648–662 B signature (mean 655.2 B, 1000 samples)	Post-quantum-ready authorization
Hybrid	Additive entries in `signatures[]`	Transition and high-assurance policy

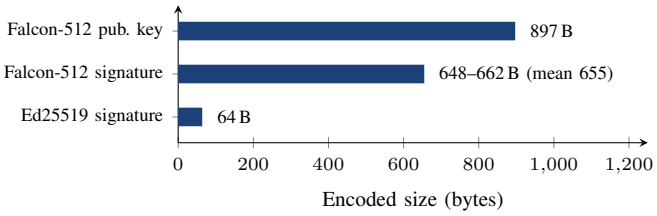

Fig. 4. Signature-envelope size evidence (Table III). Falcon-512, the FN-DSA-512 parameter set being standardized under FIPS 206, is chosen for its compact post-quantum signatures relative to other lattice schemes, at the cost of a larger public key than Ed25519. A hybrid transaction carries both an Ed25519 and a Falcon-512 entry in the `signatures[]` array.

NABD is therefore post-quantum-ready rather than FIPS 206 validated, and it will not assert FIPS 206 compliance until the final byte formats and known-answer tests are adopted by the runtime. Hybrid Ed25519 plus Falcon-512 verification is sub-millisecond on the target two-vCPU validator class; batch-verification throughput is a separate measurement tracked for hybrid-by-default operation.

VII. HUMAN AUTHORIZATION

A. What the Chain Verifies

In many blockchain systems a biometric unlocks a phone or laptop while the chain sees only an ordinary software key; if the device is lost, stolen, or destroyed, the human identity behind the account is stranded. NABD moves the biometric requirement into ledger policy, and it is important to be exact about what that means at the protocol level.

A biometric-gated action is a transaction or program call whose policy requires, in addition to the ordinary account signatures, a signature from one or more keys registered as attesting devices or oracles. The signed binding includes a fresh nonce or action hash, the target transaction or program action, a policy identifier, an expiry bound, and device or oracle attestation metadata. Validators verify these signatures and the binding. They do not, and cannot, verify that a biometric match took place: the assurance that a present human authorized the action rests on the device or oracle boundary and on the governance of the registered keys (Section III). The chain records only commitments, hashes, attestations, public keys, and signed policy results; a biometric template never appears in chain state.

B. Capture and Privacy Boundary

A multi-layer palm-vein scanner captures the biometric signal inside secure hardware or a trusted oracle, where template

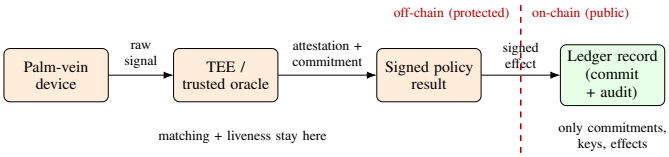

Fig. 5. Biometric privacy boundary. Palm-vein capture, matching, and liveness detection stay inside the off-chain secure boundary; only commitments, attestations, public keys, and signed policy effects cross to the public ledger. The raw biometric template never appears in chain state.

matching and liveness detection are performed (Fig. 5). Raw body data stays outside chain state. Public surfaces show what users and communities need in order to verify state, while sensitive payloads, diagnostics, secrets, raw state diffs, and unnecessary proof material remain hidden by default. Privacy in this model comes from separation: capture and matching stay in protected environments, and the ledger records commitments and signed results, so public accountability is preserved without turning the ledger into a biometric database. Biometric templates require a revocation and replacement policy outside public ledger state [16], [17].

C. Transaction Binding and Security Considerations

Attestations must be transaction-bound. A valid attestation includes a fresh nonce or action hash, the target transaction or program action, a policy identifier, an expiry bound, and device or oracle attestation metadata. Replay fails when the nonce is consumed, the transaction hash changes, or the expiry passes.

A single trusted device is not a universal trust root. Higher-assurance deployments can require M-of-N biometric oracles, independent device attestations, governance approval, or a recovery delay before a critical action executes. If biometric infrastructure is unavailable, biometric-gated actions fail closed or pause according to policy while ordinary non-gated transactions continue. Registration, rotation, and revocation of attesting keys are governance actions and should themselves be gated. In the current implementation the attesting keys are registered as operator-configured approver sets on each validator rather than as on-chain records; the attestation a validator checks is a plain Ed25519 or Falcon-512 signature from an authorized key over a typed deployment-authorization payload (deployer, contract, nonce, code hash, security grade), not a TEE quote; and rotation or revocation is an operator configuration change. On-chain registration records, quote verification, and governance-gated rotation are design targets.

Human authorization is not limited to a single emergency mode. The same mechanism lets a community require live biometric approval across a spectrum of high-value actions: authorizing treasury and asset movements, approving governance changes and program actions, gating physical access, and, at the high-risk extreme, acting as an emergency recovery circuit breaker for account recovery, suspension, or program shutdown.

D. Biometric-Gated Autonomous Programs

A central application of the primitive is autonomous-program policy. A program can run its normal rules automatically while reserving high-risk actions for biometric authorization, verifying before a configured critical boundary that the protocol has received the required attestation evidence. Examples include a community treasury that releases emergency funds only after approved human biometric signatures; a robot controller that requires human presence before changing safety limits; a public-service workflow that requires a live person before issuing a credential; an asset program that pauses minting, burning, or transfer during a risk event until biometric approval is provided; and a recovery program that restores access when ordinary devices or keys are lost. Biometric-only accounts are possible in controlled deployments; autonomous programs provide the broader programmable pattern.

E. Implementation Status

The evidence classes in Section X do not yet include a dedicated biometric test class. Implemented and tested today are the policy evaluation of typed machine and human-biometric attesting signatures with authorized-approver binding for autonomous-program deployment, enforced in transaction validation and covered by repository tests, and legacy biometric enrollment and verification records behind operator-gated API routes. No validator-side capture, template matching, or liveness detection exists, no capture-hardware integration is deployed, and the TEE attestation flow, the on-chain attesting-key registry, and the richer authorization envelope of Section VII-C remain design work.

VIII. Implementation

The autonomous-program host layer and the Ed25519 wallet-compatible signing path are implemented in Perl; Falcon-512 is reached through native XS bindings to liboqs, and autonomous programs run in the WASM3 interpreter through the same binding mechanism. Production operation uses round-robin leader selection with strict signatures and certify locks in public hardening mode.

VRF leader selection is implemented with a native ECVRF-EDWARDS25519-SHA512-TAI suite as specified in RFC 9381 (suite 0x03) [11], validated byte-for-byte against the RFC 9381 Appendix B test vectors; it is hard-gated out of the current deployment pending an election-agreement fix, and activation is planned after that remediation. ECVRF is not resistant to quantum attacks. The suite identifier is declared in genesis configuration and bound into the chain identity hash, and the migration path to a post-quantum VRF, once one is standardized, is a coordinated network upgrade with per-epoch VRF key rotation. Consensus votes and finality certificates are signed with Ed25519 today; the scheme-agile envelope (Section VI-D) already carries Falcon-512 and hybrid signatures for transactions and governance, and migrating consensus signatures to ML-DSA-65 (FIPS 204) [14] is on the roadmap so that block finality will not depend on elliptic-curve security.

Release integrity is enforced by gates on chain identity, image content, dependency security, and configuration lock; a durability drill covers backup, restore, quick-check, and ledger-count consistency (Table IX). The consensus engine

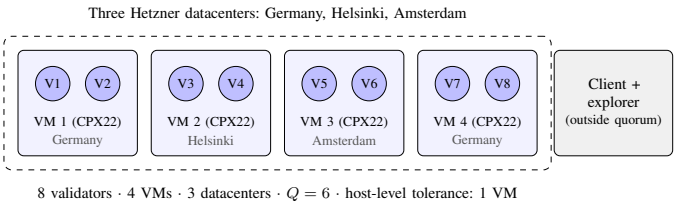

Fig. 6. Current $N = 8$ operating topology on the public testnet. Two validators share each VM, so $Q = 6$ tolerates the loss of any single validator VM with six validators remaining and no spare. The validator VMs are distributed across three Hetzner datacenters (Germany, Helsinki, Amsterdam); operating across multiple providers is a planned extension.

and the networking layer are implemented in Perl 5.42, with networking over ZeroMQ; native C/XS components provide the signature bindings (Ed25519 via libsodium 1.0.18, Falcon-512 via liboqs 0.12.0, BLAKE3 1.8.5) and the WASM3 0.5.1 interpreter with NABD's fuel-metering patch. The core implementation comprises approximately 48,600 lines of Perl (about 16,800 consensus, 4,500 networking, and 15,900 ledger and execution lines) plus about 1,500 lines of XS glue and 12,300 lines of vendored WASM3 C, alongside a test suite of about 28,700 lines. The implementation, the TLA$^+$ models of Section V-C, and the test suite are maintained at https://gitlab.com/ktham33n1/nabd_chain.

IX. Evaluation

The evidence in this section consists of a live public-ingress finality sample, laboratory leader-kill failover runs, and a modest single-wallet throughput check on the current testnet. The paper does not claim baseline throughput or failover figures for PBFT, HotStuff, or Tendermint from this deployment; the comparison in Section IX-F is qualified accordingly.

A. Testbed and Operating Topology

NABD runs a live public testnet at testnet.nabdchain.org. The current operating sample uses eight validators. Four Hetzner CPX22 virtual machines (2 shared vCPU, 4 GB RAM, AMD EPYC) run two validators each, and a fifth virtual machine hosts client and explorer services outside the consensus quorum (Fig. 6, Table IV). The validator machines are spread across three Hetzner datacenters in Germany, Helsinki, and Amsterdam, so that host-, rack-, and site-level faults are geographically decorrelated; operating across multiple cloud providers is a planned extension.

For $N = 8$ the quorum rule gives $Q = \lfloor 16/3 \rfloor + 1 = 6$. The network continues through the loss of any single validator VM, because six validators remain, but with no spare: after one host loss the set sits exactly at Q, and any further validator stall halts finality until the host returns (Table V). At the host level the deployment therefore tolerates one host fault, the same as a four-validator set with $f = 1$. Larger validator sets and one-validator-per-host placement add capacity above the quorum threshold and are the natural growth path.

This configuration is a deliberate floor rather than a recommended ceiling. Two validators share each two-vCPU, 4 GB instance, so the active set runs on modest, contended hardware with no headroom. The finality figures below are what NabdBFT delivers at its most constrained; dedicated CPUs, more memory, one-validator-per-host placement, and faster storage can only improve tail latency and throughput. Running a deterministic layer-1 with its load balancers and explorer on this class of hardware is part of the point: low-cost sovereignty means a community can own its settlement layer.

TABLE IV
Validator Deployment Mapping

VM	Datacenter	Spec	Validators / role
VM 1	Germany	CPX22 (2 vCPU, 4 GB)	V1, V2
VM 2	Helsinki	CPX22 (2 vCPU, 4 GB)	V3, V4
VM 3	Amsterdam	CPX22 (2 vCPU, 4 GB)	V5, V6
VM 4	Germany	CPX22 (2 vCPU, 4 GB)	V7, V8
VM 5	Germany	4 vCPU (client tier)	Client + explorer

Public addresses are omitted here and retained in an internal audit appendix.

TABLE V
Quorum Resilience Across Network Sizes

N	f	Q	After 1 failure	After 2 failures
5	1	4	4/4 exact quorum	3/4 halted
7	2	5	6/5 one spare	5/5 exact quorum
8	2	6	7/6 one spare	6/6 exact quorum
9	2	7	8/7 one spare	7/7 exact quorum
11	3	8	10/8 two spare	9/8 one spare

Failures counted per validator. In the current topology one host failure counts as two validator failures.

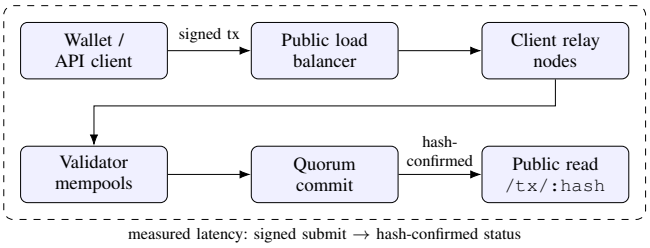

Fig. 7. Public-ingress measurement path. Latency is timed end to end from a client's signed submission to hash-confirmed visibility through `/api/v1/transactions/:hash`, so it captures load balancing, relay, mempool admission, quorum commit, and public read visibility rather than an internal validator timer.

B. Measurement Methodology

NABD measures finality from the perspective of a wallet or application rather than from an internal validator timer. The primary latency metric is the elapsed time from signed transaction submission at public ingress to hash-confirmed status through `/api/v1/transactions/:hash` (Fig. 7). The metric captures load balancing, client relay, validator mempool admission, quorum commit, and public read visibility. Table VI lists the protocol metrics used for evaluation.

C. Finality Results

Table VII and Fig. 8 summarize the public-ingress finality sample: 30 transactions submitted, 30 confirmed, with a mean of 1.214 s, p50 of 1.197 s, p95 of 1.436 s, and a maximum

TABLE VI
PROTOCOL METRICS USED FOR EVALUATION

Metric	Definition
Hash finality	Signed submit to confirmed transaction hash
Block freshness	Age of latest committed block
Quorum availability	Healthy validators over active set size
Peer completeness	Validator peers over expected mesh size
Queue drain	Mempool and relay queues after smoke traffic
Chain identity	Network identifier, genesis hash, fingerprint

TABLE VII
AGGREGATE PUBLIC-INGRESS FINALITY SAMPLE

Metric	Observed value
Samples	30 transactions
Mean latency	1.214 s (post-fix tx-gossip)
Median (p50)	1.197 s
95th percentile (p95)	1.436 s
Maximum latency	2.179 s
Confirmation	Transaction-hash lookup

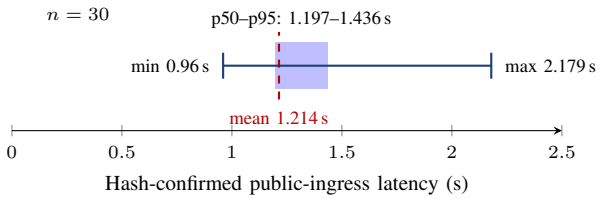

Fig. 8. Profile of the hash-confirmed public-ingress finality sample (Table VII): $n = 30$, with the p50–p95 inter-percentile band shaded, the mean dashed, and the observed minimum and maximum at the whiskers. The maximum (2.179 s) lies within 0.12 s of T_{dead}, which is why heartbeat emission must be independent of batch validation (Section V-B).

of 2.179 s. A transaction-gossip fix deployed on 2025-06-06 reduced mean finality from 3.091 s to 1.214 s; the residual latency is batch-validation overhead on under-provisioned validator containers rather than a protocol cost. The sample is small and was taken under non-saturating load; it is operating evidence for the measurement path on the current testnet, not a long-horizon service-level objective. The 30-transaction sample used direct signed submission to the public API of one validator (validator 1), with hash confirmation observed from two remote observer validators; it therefore exercises validator ingress, quorum commit, and public read visibility, but not the load-balancer and relay tier of Fig. 7.

D. Leader Failover

Repeated laboratory leader-kill tests with $T_{\text{dead}} = 2.3$ s and $\Delta_{vc} \approx 0.7$–1.0 s observed failover of about 3.3 s, inside the bound of (3). Fig. 9 places that figure against configured timeout ranges for reactive protocols. A sample size, distribution, and per-phase breakdown for the failover runs have not yet been measured and are future work; no claim in this revision depends on them. The isolated latency cost of the CERTIFY phase (an A/B experiment against a merged three-step variant) is likewise not reported.

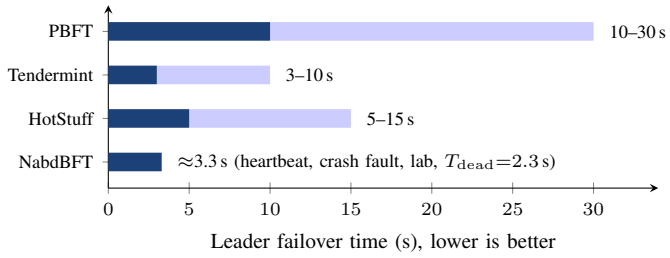

Fig. 9. Leader-failover time. NabdBFT's heartbeat detection gave a consistent ≈3.3 s crash-fault recovery in laboratory tests at $T_{\text{dead}} = 2.3$ s. The ranges shown for PBFT, Tendermint, and HotStuff are illustrative timeout-configuration ranges (light bars mark the upper end of each range), not intrinsic protocol constants; each would shrink under a shorter configured timeout, and NabdBFT's figure would grow under a longer T_{dead}. See Section IX-F.

E. Throughput

A controlled single-wallet ingress check on the live $N = 8$ testnet (one funded wallet, fixed batch size 100, live fee floor 1000, direct validator ingress, 2025-06-04) sustained 130 confirmed TPS over 36.1 s with 4,700 submitted and 4,700 confirmed transactions across 84 committed blocks. The figure is deliberately modest: it is an acceptance check on contended two-vCPU hosts under ordered single-sender load, not a saturation ceiling. The theoretical capability of the protocol follows from the round structure rather than from control traffic: at $N = 8$ a round costs at most 168 small control messages (Section IV), so with the one-second round rhythm, sustained throughput is bounded by the configured block payload and per-transaction validation cost rather than by consensus messaging.

F. Comparison With Reactive Protocols

Table VIII places NabdBFT next to PBFT, Tendermint, and HotStuff. The defining difference is the detection signal for a crashed leader: the three reactive protocols begin recovery when a round or pacemaker timer expires, while NabdBFT begins recovery when the heartbeat plane classifies the leader as dead, independently of round length. NabdBFT shares the $O(N^2)$ messaging of PBFT and Tendermint, which is acceptable at the target permissioned-to-sovereign scale (N in the single to low double digits) and to which threshold-signature aggregation could later be applied.

The failover ranges in the table are illustrative timeout-configuration ranges, not intrinsic protocol constants, and they were not measured in this environment. A reactive protocol configured with a 2.3 s timeout would recover in a time comparable to NabdBFT's 3.3 s, and NabdBFT configured with a longer T_{dead} would recover later. The previous revision's statement that NabdBFT fails over "three to ten times faster" is therefore withdrawn: it held only against the upper end of PBFT's configured range and did not hold against Tendermint's lower end. What survives comparison is the structural property: detection time is fixed by T_{dead} rather than by the round timeout, so it does not grow with block size or round length, and it does not depend on where in the round the failure occurs. A same-environment, same-timeout

TABLE VIII
PROTOCOL COMPARISON SUMMARY

Protocol	Msg. compl.	Phases	Crash detection	Failover (illustrative configured ranges)
PBFT [1]	$O(N^2)$	3	Round timer	10–30 s
Tendermint [4]	$O(N^2)$	3	Round timer	3–10 s
HotStuff [3]	$O(N)$	3	Pacemaker timer	5–15 s
NabdBFT	$O(N^2)$	4	Heartbeat (T_{dead}) + phase timer	$\approx$3.3 s at T_{dead}=2.3 s (lab)

TABLE IX
ASSURANCE EVIDENCE USED TO SUPPORT PROTOCOL CLAIMS

Evidence class	Claim supported
Formal model	Quorum intersection, view-scoped voting, and agreement safety
Consensus regression	No-reorg behavior, leader recovery, and fork resilience
Transaction tests	Nonce, replay, overspend, batch, asset, and AP validation
Public ingress sample	End-to-end hash-confirmed finality path from user entry
Release gates	Chain identity, image content, dependency security, and config lock
Durability drill	Backup, restore, quick-check, and ledger-count consistency
Biometric policy tests	Partial: deployment-gating signature tests only; no dedicated evidence class yet (Section VII)

comparison against at least one reactive implementation is the appropriate way to quantify that property and is future work.

X. ASSURANCE EVIDENCE AND SCOPE

NABD separates its claims into auditable evidence classes: formal safety models, deterministic validation tests, public-ingress operating samples, release integrity gates, and durability checks (Table IX). These classes support the properties stated in this paper and make clear which claims are architectural, which are measured, and which require deployment-specific governance.

A. Scope Boundaries

The present implementation does not claim deployed cryptographic confidential transfers, validator-side biometric matching, ISO 20022 payment-network compliance, or root-to-tenant checkpoint anchoring. The privacy model is data minimization, disclosure control, and off-chain biometric matching with on-chain commitments and policy effects. Confidential transfer circuits and cross-network anchoring are compatible extension paths but are not needed for the deterministic finality and human-authorization model described here.

B. Known Limitations

The following limitations are stated so that the evidence is not over-read.

- Leader-failover evidence is laboratory-only and lacks a reported sample size and distribution.
- The liveness argument (Section V-B) is informal and holds only inside the stated fixed-timer delay envelope; the TLA$^+$ models contain no timing.
- The heartbeat plane accelerates crash-fault recovery only; Byzantine slow leaders recover on the reactive path.
- Two validators per host reduce host-level tolerance to one host and place two validator keys inside one trust boundary.
- Finality evidence is a 30-transaction sample under non-saturating load; no latency-under-load curve or fault-injection results are reported.
- Autonomous-program calls are fuel-metered deterministically, but the additional wall-clock guard can still abort a local block write; confining it to outcome-neutral, node-local defense is required before production (Section VI-C).
- The biometric authorization path has no dedicated evidence class yet, and the chain verifies attesting-key signatures, not biometric matching.

XI. DEPLOYMENT AND GOVERNANCE

A useful layer-1 can operate without large infrastructure. NABD is intended for modest, reproducible operation: explicit genesis state, deterministic serialization, conservative services, clear validator roles, and plain operational surfaces, so that a community, company, school, cooperative, or public-good network can own its settlement layer rather than rent trust from a single platform. The protocol favors components that are understandable and replayable (canonical payloads, deterministic state, explicit configuration, small trusted surfaces, and reproducible release artifacts), because long-lived systems survive by being rebuildable rather than by depending on opaque infrastructure.

Validator membership, biometric policy, asset authority, and emergency actions are governance questions, and NABD makes them explicit. A network defines who proposes changes, who approves them, when they activate, and which authorization proofs are accepted for critical actions, including the registration and revocation of attesting keys.

XII. CONCLUSION

NABD's central idea is to make crash-fault liveness proactive. By decoupling validator health monitoring from the consensus round, a dead leader is classified within $T_{\mathrm{dead}} = 2.3$ s and replaced in about 3.3 s in laboratory tests, while a four-phase quorum path with an explicit CERTIFY evidence boundary keeps finality deterministic at a mean of 1.214 s on two-vCPU shared hardware. The failover figure is a consequence of the configured thresholds and applies to crash faults; the structural gain is that detection time no longer depends on round length. On that base, NABD adds policy-gated human authorization with biometric matching kept outside chain state, native multi-asset state, deterministic autonomous programs, and a scheme-agile signature envelope. Section X lists what remains to be measured, modeled, or built. The network runs today at testnet.nabdchain.org.

REFERENCES

[1] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. 3rd Symp. Operating Systems Design and Implementation (OSDI)*, 1999.

[2] A. Bessani, J. Sousa, and E. E. P. Alchieri, "State machine replication for the masses with BFT-SMaRt," in *Proc. DSN*, 2014.

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

[4] E. Buchman, J. Kwon, and Z. Milosevic, "The latest gossip on BFT consensus," arXiv:1807.04938, 2018.

[5] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *J. ACM*, vol. 43, no. 2, 1996.

[6] L. Lamport, *Specifying Systems: The TLA$^+$ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.

[7] Y. Yu, P. Manolios, and L. Lamport, "Model checking TLA$^+$ specifications," in *Proc. CHARME*, 1999.

[8] E. Androulaki *et al.*, "Hyperledger Fabric: A distributed operating system for permissioned blockchains," in *Proc. EuroSys*, 2018.

[9] Hyperledger Fabric Documentation, "Private data," Linux Foundation, accessed 2025.

[10] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling Byzantine agreements for cryptocurrencies," in *Proc. SOSP*, 2017.

[11] S. Goldberg, L. Reyzin, D. Papadopoulos, and J. Včelák, "Verifiable random functions (VRFs)," RFC 9381, 2023.

[12] P.-A. Fouque *et al.*, "Falcon: Fast-Fourier lattice-based compact signatures over NTRU," NIST Post-Quantum Cryptography Standardization Project.

[13] National Institute of Standards and Technology, "Post-quantum cryptography standardization," NIST Computer Security Resource Center.

[14] National Institute of Standards and Technology, "Module-lattice-based digital signature standard," FIPS 204, 2024.

[15] R. Perlner *et al.*, "Status update on the FN-DSA standard (draft FIPS 206)," NIST PQC status update, 2025.

[16] N. K. Ratha, J. H. Connell, and R. M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Systems Journal*, vol. 40, no. 3, 2001.

[17] ISO/IEC 24745:2022, "Information security, cybersecurity and privacy protection: Biometric information protection," International Organization for Standardization, 2022.

[18] A. Haas *et al.*, "Bringing the web up to speed with WebAssembly," in *Proc. PLDI*, 2017.