

The Network of Momentum (NoM) — Decoded and Expanded Whitepaper (Community Reconstruction)

Status: Community-authored reconstruction (interpretive, non-normative). Purpose: Decode and clarify the original Network of Momentum (NoM) / Zenon early whitepaper intent (circa 2020).

Reader's Guide — How to Read This Document

This document is a community-authored reconstruction of the original Network of Momentum (NoM) whitepaper released circa 2020. It is not a protocol specification, implementation guide, or amendment to the canonical text. Readers should approach it as an interpretive aid designed to clarify intent, structure, and internal consistency.

What This Document Is

- A decoding of an intentionally under-specified research draft.
- A structured attempt to make implicit architectural assumptions explicit.
- A separation of what the original paper states from what logically follows from those statements.
- A guide for reasoning about NoM without presuming its current implementation status.

What This Document Is Not

- Not a new protocol proposal.
- Not a normative specification.
- Not a claim that all described mechanisms exist today.
- Not an assertion of “official” Zenon design authority.

Where this document appears more concrete than the original draft, it does so only to explain constraints, not to introduce new mechanics.

Canonical vs Interpretive Content

Throughout the document, content is categorized into three classes:

1. **Canonical** Statements explicitly present in the 2020 whitepaper or directly paraphrased from it.
2. **Interpretive (Non-normative)** Logical elaborations intended to clarify how canonical statements can coexist coherently. These do not add new protocol primitives, actors, or guarantees.
3. **Not Specified in Draft (Open Questions)** Areas where the original draft is silent, ambiguous, or intentionally incomplete. These are framed as questions, not answers.

Readers are encouraged to distinguish these categories when forming conclusions.

How to Read Technical Sections

- Mathematical expressions in this document are illustrative abstractions, not binding specifications. They clarify relationships and invariants but should not be treated as implementation-level definitions.
- Security analysis is aligned strictly to attack classes explicitly listed in the original draft. No new threat models are introduced.
- Consensus descriptions explain how the system could reason about agreement, not how to code it.

If a mechanism appears unusually abstract or incomplete, this likely reflects faithful preservation of the original draft's level of detail.

Intended Audience

This document is written for:

- Readers seeking clarity beyond high-level summaries.
- Researchers evaluating verification-first and dual-ledger architectures.
- Developers reasoning about what NoM permits, not what it mandates.

It assumes familiarity with distributed systems and decentralized consensus but does not assume prior agreement with any specific interpretation.

How to Disagree Productively

Disagreement is expected and welcome. Readers who contest interpretations are encouraged to:

1. Point to specific canonical text that contradicts the reconstruction.
2. Identify which assumptions are interpretive rather than canonical.
3. Propose alternative interpretations that satisfy the same constraints.

This document is intended to be falsifiable through citation, not authoritative by assertion.

Comparison Index — What This Reconstruction Does Not Claim

This index clarifies common misreadings by explicitly stating positions this document does not take.

It does not claim Zenon was fully specified in 2020

The reconstruction explicitly treats NoM as a research architecture, not a finalized protocol.

It does not claim REFUSE semantics were formally defined

Three-outcome verification is presented as an interpretive model consistent with bounded verification constraints, not as canonical doctrine.

It does not claim zApps are consensus-critical

zApps are treated as downstream, optional, and execution-layer concerns. Consensus correctness does not depend on their behavior.

It does not mandate unikernels

Unikernels are presented as illustrative execution environments, not required infrastructure.

It does not introduce new cryptographic primitives

All cryptographic mechanisms referenced are already implied or mentioned in the canonical draft.

It does not redefine Zenon's economics

Incentives, proof markets, and pricing dynamics are discussed only where implied and are explicitly marked as open research areas.

It does not claim performance guarantees

No throughput, latency, or scaling figures are asserted beyond qualitative behavior.

It does not override implementation reality

Discrepancies between this reconstruction and current Zenon implementations are acknowledged as possible and expected.

Summary Statement

This reconstruction aims to make the internal logic of the Network of Momentum intelligible without asserting ownership over its evolution. It preserves ambiguity where the original draft preserves ambiguity and clarifies structure where the original draft implies structure without specification.

It is an aid to understanding — not a declaration of truth.

Preface

Canonical (from the 2020 draft)

- The canonical whitepaper presents the Network of Momentum (NoM) as a partially specified architecture rather than a finalized system.
- The document aims to communicate directional principles and invariants, not a complete execution model.
- Core ideas include a dual-ledger design, distributed actor model, and verification-driven correctness.
- The original draft distinguishes between what the architecture allows versus what is already implemented.

Reconstruction (Interpretive, Non-normative)

This paper is not a proposal for a new distributed ledger architecture, nor an announcement of a finalized system. Instead, it functions as an architectural decoding—an effort to make explicit what the early Zenon materials (circa 2020) leave implicit. Those materials outline a conceptual foundation for the Network of Momentum but intentionally omit full execution semantics, light-client rules, or heterogeneous verification mechanisms.

Goals of this reconstruction:

1. Extract architectural invariants from the canonical text.
2. Clarify ambiguous or implied aspects of consensus, verification, and actor interaction.
3. Formalize behaviors that were informally or contextually described.
4. Preserve distinction between architectural capabilities (what the system allows) and implementation status (what is built).

No new primitives, assumptions, or consensus mechanisms are introduced here. When this reconstruction appears more concrete than the original whitepaper, such details should be read as interpretive formalizations—logical completions of implied design constraints—not as additions to the canonical text.

The purpose of this reconstruction is to make NoM’s internal logic legible for:

1. Readers seeking clarity beyond high-level marketing or conceptual summaries.
2. Researchers evaluating the consistency of dual-ledger and verification-first systems.
3. Developers reasoning about the architecture’s permissible designs without presuming their current deployment.

Not Specified in Draft (Open Questions)

- Extent of canonical formalization for light clients and partial verification.
- Whether “bounded verification” was a fully intended design primitive or inferred retrospectively.
- Whether the “REFUSE” verification mode (as later discussed in community materials) was implicit or emergent.

Abstract

Canonical (from the 2020 draft)

- Describes a dual-ledger system separating consensus ordering (meta-layer) and transactional execution (block-lattice).
- Introduces multiple classes of network participants (“actors”) with different verification responsibilities.
- Emphasizes security, scalability, and decentralization as guiding design principles.
- States that the paper is a research draft—not a deployment specification.

Reconstruction (Interpretive, Non-normative)

This reconstruction clarifies the implied structure of NoM:

- A dual-ledger architecture consisting of a meta-DAG (for consensus ordering) and a block-lattice (for transactional state).
- An actor hierarchy spanning full consensus participants (Pillars), relays (Sentinels), and light verifiers (Sentries).
- Bounded verification semantics allowing nodes to maintain correctness under constrained resources, using three outcomes: accept, reject, or refuse.

These verification semantics and the explicit notion of “REFUSE” are interpretive constructs, included to explain the correctness guarantees implied by the canonical design.

The document maintains a strict separation between:

- Canonical content — information directly in the 2020 draft.

- Interpretive content — logical elaborations for clarity.
- Open questions — intentionally unspecified or unresolved aspects of the 2020 design.

Not Specified in Draft (Open Questions)

- Whether “three-outcome verification” (accept/reject/refuse) was formally present in the original draft or added by later interpreters.
- How the meta-DAG’s finality interacts with partial proofs during asynchronous operation.

I. Introduction

Canonical (from the 2020 draft)

- Introduces the Network of Momentum as a decentralized ledger system combining aspects of blockchains and DAGs.
- Notes that traditional blockchains (e.g., Bitcoin, Ethereum) suffer scalability limits due to linearity and global synchronization.
- Suggests that NoM overcomes these limits via a dual-layer ledger enabling asynchronous and parallelized operations.
- States four guiding design principles:
 1. Decentralization: Trustless operation without central coordination.
 2. Scalability: High throughput and low latency across large networks.
 3. Security: Resistance to Sybil, double-spend, and denial-of-service attacks.
 4. Verifiability: Support for bounded, resource-limited verification.

Reconstruction (Interpretive, Non-normative)

Distributed ledger technology (DLT) evolved rapidly from Bitcoin’s linear proof-of-work chain to diverse architectures that seek parallelism and asynchrony. NoM belongs to this second wave—its dual-ledger structure explicitly separates global consensus ordering from local transactional execution, enabling each component to scale independently.

Under this design:

- The meta-DAG handles consensus events, ordering, and epoch transitions.
- The block-lattice provides user-level transaction histories for each account.
- Together, they yield a distributed system that can tolerate heterogeneous participants, intermittent connectivity, and variable resource capacities.

Conceptually, NoM’s position among DLT families can be seen as a synthesis:

- Like DAGs, it supports parallel updates.
- Like blockchains, it maintains strong ordering and finality guarantees.

- Like agent-centric systems, it allows local state verification and asynchronous operation.

The introduction of a verification-first architecture—where correctness is prioritized over synchrony—distinguishes NoM from throughput-optimized systems.

Context (Non-canonical background)

To situate NoM in broader distributed systems research:

- Blockchain: Linear, globally replicated history.
- DAG-based ledgers: Non-linear structures permitting concurrency (e.g., Zilliqa).
- Block-lattice: Each user maintains an account-chain (e.g., Nano).
- Agent-centric ledgers: Validation occurs per-agent rather than globally (e.g., Holochain).
- BFT consensus: Systems using Byzantine Fault Tolerance mechanisms (e.g., Tendermint).

This contextual framing is non-canonical—provided solely for comparison and not drawn directly from the 2020 draft.

Not Specified in Draft (Open Questions)

- Whether NoM assumes partial synchrony or complete asynchrony.
- How the system prioritizes between decentralization and latency under adverse conditions.
- How “heterogeneous verification” is implemented or incentivized across actors.

II. Ledger Structures and Design Rationale

Canonical (from the 2020 draft)

- Introduces several ledger archetypes: blockchain, DAG, block-lattice, and others.
- Describes a dual-ledger architecture consisting of:
 1. A meta-DAG for consensus ordering.
 2. A block-lattice for transactional storage.
- Indicates that separating ordering (meta-layer) from execution (transaction layer) provides modularity, scalability, and improved data locality.

- Highlights that consensus finality occurs in the meta-DAG, while user transactions are executed and stored in the block-lattice.

Reconstruction (Interpretive, Non-normative)

A. Blockchain

The traditional blockchain structure (as in Bitcoin, Ethereum) forms a sequential chain of blocks, each referencing its predecessor through a cryptographic hash. This structure provides immutability and security through linear ordering, but limits scalability because every node must process and store all transactions in sequence.

Canonical interpretation: The 2020 draft acknowledges the blockchain’s reliability but identifies linearity as the bottleneck preventing scalability and parallel verification.

B. Directed Acyclic Graph (DAG)

DAG-based systems attempt to solve blockchain scalability by allowing multiple branches of transactions to exist concurrently. Each transaction references multiple predecessors, forming a graph structure without cycles. This design improves throughput but increases the complexity of achieving consistent global ordering and raises the possibility of conflicting subgraphs.

Canonical reference: The draft uses DAG principles for NoM’s meta-layer but adds stake-weighted virtual voting to reimpose determinism.

C. Block-Lattice

In a block-lattice architecture, each user controls an independent account-chain recording personal transactions asynchronously. Each send or receive transaction is independent and can be appended without waiting for network-wide consensus. The block-lattice model enables high throughput, minimal contention, and localized verification—forming the foundation for NoM’s transactional layer.

Interpretive clarification: The canonical whitepaper implies that each account-chain is part of a larger mesh coordinated by the meta-DAG, which provides transaction ordering and prevents inconsistencies.

D. The Dual-Ledger Approach

NoM combines both paradigms by introducing:

1. A meta-DAG — governs ordering, epoch transitions, and consensus decisions.

2. A block-lattice — stores confirmed transactional states per account.

The interaction between these ledgers decouples agreement from execution. Consensus decisions (what transactions are valid and in what order) occur on the meta-DAG, while execution (updating balances, zApp states, etc.) occurs asynchronously in the block-lattice.

Benefits inferred from canonical structure:

- Scalability through concurrent transaction processing.
- Modular separation of consensus logic from state management.
- Improved efficiency for light clients that only need account-specific data.
- Lower synchronization costs across the network.

Ledger Type	Advantages	Disadvantages
Blockchain	Simple, secure, and widely understood	Linear bottleneck, limited scalability
DAG	Parallel transactions, high throughput	Ordering complexity, fork risk
Block-Lattice	Independent account chains, localized verification	Requires external ordering logic
NoM Dual-Ledger	Combines scalability with determinism	Coordination between layers adds complexity

Table 1: Comparison of ledger types

Not Specified in Draft (Open Questions)

- The exact interface between the meta-DAG and block-lattice (how ordering proofs are referenced in account-chains).
- Whether the two ledgers are stored separately or as an integrated data structure.
- How light clients efficiently synchronize across both layers.
- Whether epoch boundaries in the meta-DAG explicitly map to checkpoints in the block-lattice.

III. Consensus Algorithms and Protocol Families

Canonical (from the 2020 draft)

- Defines consensus as the process by which distributed participants agree on a consistent system state.
- Mentions several families of consensus mechanisms including Proof-of-Work (PoW), Proof-of-Stake (PoS), and hybrid Byzantine Fault Tolerance (BFT) systems.
- Indicates that NoM leverages hybrid principles—combining PoW and stake-based virtual voting—to balance decentralization, scalability, and energy efficiency.
- References “virtual voting” as an implicit ordering mechanism embedded within the meta-DAG.
- Notes that consensus ensures both safety (agreement among honest nodes) and liveness (continued progress despite faults).

Reconstruction (Interpretive, Non-normative)

Consensus determines how participants in a decentralized network achieve agreement on transaction order without centralized control. NoM’s design follows a hybrid approach, blending multiple consensus paradigms to satisfy different architectural goals.

A. Consensus Properties

Core objectives for a decentralized consensus protocol include:

- **Adversary Resistance:** Tolerate Byzantine faults and arbitrary node misbehavior.
- **Sybil Resistance:** Prevent identity inflation; nodes cannot gain influence by creating more identities.
- **Accountability:** Actions are cryptographically signed and attributable.
- **Censorship Resistance:** No participant can unilaterally suppress valid transactions.
- **Denial-of-Service (DoS) Resistance:** Maintain functionality even under resource exhaustion attempts.
- **Finality:** Once finalized, a transaction remains immutable in the ledger.

Performance indicators often discussed:

- **Throughput:** Transactions processed per second.
- **Scalability:** Ability to maintain throughput as participation grows.
- **Latency:** Time from transaction broadcast to confirmation.
- **Fault Tolerance:** Maximum Byzantine fraction tolerable while maintaining correctness.

B. Proof-of-Work (PoW)

Canonical: The draft references PoW as an anti-Sybil and spam-resistance mechanism.

PoW involves solving computational puzzles whose solutions are costly to generate but cheap to verify. Nodes must expend measurable energy or computational effort to submit valid blocks or proofs.

Interpretive Clarification: In NoM, PoW has two layers:

1. Pillar PoW: Used for epoch completion and participation in consensus.
2. Sentinel PoW links: Used to throttle transaction submission and prevent spam.

This two-level PoW structure ensures that both consensus and transaction propagation are resource-bound, making attacks economically expensive.

C. Proof-of-Stake (PoS)

PoS replaces computational expenditure with economic commitment. Participants lock collateral (“stake”) to gain block production or voting rights, proportionally to their holdings.

Canonical notes: The draft references stake-weighted voting and indicates that adding more nodes does not increase voting power. Thus, Sybil attacks do not yield advantage since consensus power scales with staked value, not node count.

Interpretive: NoM applies PoS in virtual voting: each Pillar’s voting weight corresponds to its stake, and consensus decisions depend on cumulative stake-weighted support.

D. Delegated Proof-of-Stake (DPoS)

Canonical: Mentioned briefly in comparison as a scalability-oriented model using elected delegates.

Interpretive: Not adopted directly by NoM, but useful context for understanding hybridization. Delegates introduce centralization trade-offs which NoM avoids by distributing consensus responsibilities across many Pillars.

E. Proof-of-X Variants (Context)

Other mechanisms (e.g., Proof-of-Storage, Proof-of-Retrievability, Proof-of-Elapsed Time) are referenced for comparison. The 2020 draft does not indicate their direct use in NoM. They are presented to highlight potential adaptations or extensions of PoW/PoS mechanics for specialized applications.

F. Hybrid BFT Consensus

Canonical: The draft discusses combining PoW or PoS with Byzantine Fault Tolerance (BFT).

Interpretive: NoM's hybridization separates validator selection (stake-weighted identity) from consensus execution (virtual voting). BFT-like safety emerges through deterministic supermajority voting embedded in the meta-DAG structure.

Context (Non-canonical background): Systems like Zilliqa (PoW-BFT) and Tendermint (PoS-BFT) represent similar hybrid approaches but are not directly implemented in NoM.

G. Cellular Automata Consensus (Context)

Referenced as an abstract model for local update rules achieving global convergence. This concept supports the interpretive view that NoM's meta-DAG consensus can be viewed as a distributed automaton, where local interactions yield eventual global agreement without centralized orchestration.

H. Virtual Voting

Canonical (explicitly stated in 2020 draft):

- “Virtual voting” replaces explicit message-passing with inference from message histories.
- Nodes deduce votes based on observed data structures (e.g., DAG topology), minimizing communication overhead.

Interpretive: In NoM, the meta-DAG itself embodies this mechanism:

- Pillars broadcast events (transactions, PoW completions).
- Each Pillar derives a view of others' votes implicitly from received DAG links.
- Consensus is achieved when all honest Pillars converge on identical deterministic ordering derived from these shared observations.

Properties:

- Eliminates explicit vote messages.
- Scales with DAG propagation rather than quorum chatter.

Consensus Type	Advantages	Disadvantages
Proof-of-Work	High Sybil resistance	Energy cost, low throughput
Proof-of-Stake	Energy-efficient	Requires economic security assumptions
Delegated PoS	High scalability	Centralization risk
BFT	Strong consistency	High communication cost
Virtual Voting	Scalable, implicit consensus	Requires high DAG connectivity
Cellular Automata	Local rules, emergent order	Sensitive to topology and latency

Table 2: Comparison of consensus types

- Provides asynchronous BFT-like safety in open networks.

Not Specified in Draft (Open Questions)

- Exact function describing how stake weight maps to voting influence.
- Thresholds for PoW difficulty adjustment within epochs.
- How randomness (if any) influences tie-breaking in virtual voting.
- The extent to which the hybrid PoW–PoS mechanism is finalized versus conceptual in the 2020 draft.

IV. Prerequisites and Actor Model

Canonical (from the 2020 draft)

- Describes several classes of network actors—each responsible for specific roles in communication, verification, and consensus.
- Identifies the “Pillar” as the full consensus participant.
- Mentions Sentinels as intermediaries that relay data and perform partial proofs.
- Mentions lightweight observers capable of verification within resource limits.
- Describes the network as asynchronous, with no global clock or guaranteed message timing.
- Indicates that consensus safety and liveness depend on the assumption that less than one-third of total stake behaves maliciously.

Reconstruction (Interpretive, Non-normative)

A. Definitions

A *node* is any software entity executing the NoM protocol. Nodes differ in storage, computation, and connectivity capacity, leading to the hierarchical actor model summarized below.

Actor	Canonical Role	Reconstruction Description
Pillar	Consensus participant	Maintains both ledgers (meta-DAG + block-lattice); executes virtual voting; finalizes epochs.
Sentinel	Relay and proof constructor	Relays transactions, constructs PoW fragments (“links”), serves proofs to verifiers.
Sentry	Lightweight verifier	Maintains pruned ledger view, verifies locally under resource constraints.
Representative	Optional intermediary	Provides transaction relay and proof access for light clients; non-consensus participant.

Table 3: Actor roles in NoM

Each node operates within explicit resource bounds defined by:

$$R_V = (S_V, B_V, C_V)$$

where S_V is storage, B_V bandwidth, and C_V computational capacity. Verification performance and decision modes depend on these local limits.

B. Supermajority Definition

A supermajority condition is reached when the cumulative active stake of agreeing Pillars exceeds two-thirds of the total:

$$\sum w_i \geq \frac{2}{3}W_\epsilon + \delta$$

where:

- W_ϵ : total active stake in epoch ϵ
- δ : small positive margin ensuring decisiveness

This threshold ensures deterministic agreement across all honest Pillars.

C. Representative Role

Representatives serve as accessible endpoints for light clients, helping them broadcast transactions and retrieve verification proofs. Although they aid in network propagation, correctness does not depend on their

honesty—clients always verify proofs locally (or refuse when incomplete).

D. Transaction Types

Canonical transactions include:

1. Send/Receive transfers between account-chains.
2. Protocol messages marking PoW completions or epoch boundaries.
3. Smart interactions (zApps), representing application-level logic.

Each transaction contains sender/receiver addresses, balances, signatures, and PoW link data.

E. Epoch

An *epoch* is a discrete period used for grouping transactions and finalizing consensus. During each epoch:

- Pillars perform PoW work.
- Transactions propagate and accumulate.
- Once a supermajority of Pillars complete their PoW, the epoch finalizes, and the next begins.

Epochs serve as synchronization boundaries for virtual voting rounds.

F. Virtual Voting

Canonical: Described as consensus where votes are inferred, not transmitted. Each Pillar observes other Pillars' PoW completions and transaction acknowledgments within the DAG, deducing majority opinion deterministically.

Interpretive: The DAG's topology acts as a record of message causality, from which all honest Pillars independently infer identical results.

G. Broadcast Mechanism

Broadcasts disseminate:

- Epoch completion (“finishing PoW”) messages, and

- Transaction summaries for inclusion.

Reliable broadcast ensures every honest node eventually receives identical finalized data, even in asynchronous conditions.

H. Network Model

The network is open and asynchronous:

- Nodes may join or leave at any time.
- Messages are authenticated via asymmetric cryptography.
- No timing guarantees are assumed.

Safety and liveness hold under the honest-majority assumption: fewer than one-third of total stake acts maliciously.

I. Verification and Resource Bounds

Each node's verification function operates within resource constraints:

$$\text{Verify}_v(p) \rightarrow \{\top, \perp, \text{REFUSE}\}$$

Where:

- \top : Verified as valid.
- \perp : Determined invalid.
- REFUSE: Insufficient data/resources to verify.

REFUSE maintains correctness by preventing unsound conclusions when proofs are incomplete.

J. Verification Regimes

1. Regime I — Connected Verification

- Adequate connectivity and complete proof access.
- Probability of correct verification approaches 1 for honest nodes.

$$\Pr[\text{Verify}_v(p) = \top \mid \text{honest network}] \rightarrow 1$$

2. Regime II — Bounded Verification

- Limited bandwidth or connectivity.
- Nodes may output REFUSE rather than incorrect decisions.

This ensures correctness under partial knowledge.

K. Proof Availability and Incentives

Proofs represent economic goods. Pillars and Sentinels are incentivized to supply proofs to light clients. When a verifier emits REFUSE, it signals market demand for missing proofs—creating an incentive to respond for rewards or fees.

This mechanism creates an economically self-correcting proof market.

L. Consensus Independence from Verification

Global consensus progresses regardless of local verifier capability. Even if some verifiers operate under bounded verification (and refuse some proofs), global ordering and finality are unaffected. This separation ensures system-wide determinism while supporting heterogeneous devices.

M. Browser and Mobile Clients

Light clients (Sentries) often operate via browser or mobile interfaces. They use WebAssembly or similar environments to perform cryptographic verification. They can verify locally or refuse safely—preserving correctness even in intermittent connectivity.

N. Actor Responsibilities Summary

Actor	Responsibilities
Pillars	Execute consensus, maintain dual ledgers, finalize epochs.
Sentinels	Relay transactions, construct PoW links, supply proofs.
Sentries	Verify account-level transactions within resource bounds.
Representatives	Connect clients to the network; correctness remains local.

Table 4: Actor responsibilities

This hierarchical model ensures that correctness and scalability coexist—heavy nodes provide consensus capacity, while lightweight verifiers preserve universal accessibility.

Not Specified in Draft (Open Questions)

- Detailed economics of proof supply and pricing mechanisms.
- Criteria for Representative selection or reputation weighting.
- Cryptographic construction of PoW link fragments.
- Relationship between stake-weighted voting and epoch boundaries.
- Extent of fault tolerance if multiple Pillars drop out simultaneously.

V. NoM Ledger and Consensus Mechanism

Canonical (from the 2020 draft)

- Describes a dual-ledger architecture consisting of:
 1. The meta-DAG, which orders and finalizes transactions.
 2. The block-lattice, which stores and executes user-level transactions.
- Indicates that consensus proceeds in epochs, each culminating in a PoW completion broadcast by Pillars.
- States that a supermajority of Pillar stake finalizes each epoch, ensuring agreement and immutability.
- References virtual voting and shared coin rounds as the theoretical basis for finality.
- Indicates that consensus safety holds if fewer than one-third of total stake is Byzantine.

Reconstruction (Interpretive, Non-normative)

A. The Dual-Ledger Architecture

The Network of Momentum (NoM) employs two ledgers working in concert:

1. Transactional Ledger (Block-Lattice)

- Each account maintains an independent chain recording send/receive events and balance updates.
- Transactions can occur asynchronously without waiting for global confirmation.

- Each account-chain is cryptographically linked through signatures and PoW fragments.

2. Consensus Ledger (Meta-DAG)

- Maintained by Pillar nodes, it records consensus events such as epoch completions, transaction acknowledgments, and PoW commitments.
- The meta-DAG provides a consistent, deterministic ordering of all finalized transactions.

Interpretive Clarification: This architectural separation enables execution decoupling—transactions can propagate and execute locally, while consensus handles ordering and conflict resolution at the global level.

B. Ledger Query Model and Proof Retrieval

Clients query the ledger through Representatives (typically Sentinels) to obtain verifiable information, such as:

- Account balances.
- Inclusion proofs for transactions.
- Epoch and finality confirmations.

Each query returns a proof bundle:

$$\Pi(Q) = \{\text{commitments, signatures, PoW metadata, epoch references}\}$$

Verification is performed locally:

$$\text{Verify}_v(\Pi(Q)) = \top$$

if valid within resource constraints, or REFUSE if incomplete.

Proof bundles are designed to be modular—a verifier can confirm local state without downloading the global ledger.

C. Interaction Between Dual Ledgers

The meta-DAG orders transactions deterministically via virtual voting. Once consensus finalizes an epoch, all transactions decided in that epoch are appended to the block-lattice.

Let:

- D_ϵ : transactions decided in epoch ϵ .
- O_ϵ : deterministic ordering function.

Each finalized transaction is appended:

$$\forall t \in O_\epsilon(D_\epsilon) : \text{append}(t)$$

This ensures every honest node shares an identical ledger state after finalization.

D. Verification Scope by Node Type

Node Type	Verification Scope
Pillar	Full verification (signatures, PoW, voting, epoch transitions).
Sentinel	Transaction and PoW link verification; partial ledger validation.
Sentry	Account-level transaction and balance verification.

Table 5: Verification scope by node type

Resource hierarchy:

$$R_V^{(\text{pillar})} \gg R_V^{(\text{sentinel})} \gg R_V^{(\text{sentry})}$$

This allows scalability without compromising correctness.

E. Proof-of-Work Links

Canonical: PoW links serve as Sybil and spam mitigation.

Interpretive Description: Each transaction accumulates PoW contributions as it travels through the network. PoW links contribute to accumulated work that helps determine transaction eligibility and ordering priority. The draft does not fully specify the exact format or cryptographic structure of these links.

Total accumulated work can be represented as:

$$W(t) = \sum_{j=1}^h w_j$$

where w_j represents individual work contributions. A transaction becomes eligible once accumulated work meets network requirements, tying resource expenditure to transaction propagation and ensuring anti-spam enforcement.

Not Specified in Draft (Open Questions)

- Detailed per-link field structure (nonces, signatures, identity binding).
- How work contributions are cryptographically verified and aggregated.
- Whether PoW links carry explicit Sentinel identity or are anonymous.

F. Conflict Resolution

When conflicting transactions arise (e.g., double-spends), the network applies predefined deterministic rules referenced earlier in the consensus mechanism to select one valid transaction and discard others.

All honest Pillars applying these rules reach identical outcomes, ensuring conflict resolution without ambiguity. The draft references these ordering rules but does not provide their complete specification.

Not Specified in Draft (Open Questions)

- Complete specification of the deterministic ordering function.
- Whether ordering considers accumulated PoW weight, timestamps, transaction depth, hash values, or other factors.
- Priority and composition of multiple tiebreak criteria if present.

G. Consensus Process Overview

Consensus proceeds through epochs. Each Pillar maintains:

- Epoch index (ϵ).
- Local transaction pool (\mathcal{T}_ϵ).
- Received PoW completions (\mathcal{F}_ϵ).

Once:

$$\sum_{i \in \mathcal{F}_\epsilon} w_i \geq \frac{2}{3} W_\epsilon + \delta$$

the epoch finalizes.

If a Pillar hasn't completed its PoW by this time, it aborts computation and synchronizes to the next epoch.

H. Knowledge Convergence Across Epochs

Due to asynchronous communication, not all Pillars see identical data instantly. Over successive epochs, transaction knowledge converges.

Let $\text{Know}(p, \epsilon)$ represent transactions known to Pillar p at epoch ϵ . A transaction becomes decidable once:

$$\sum_{i \in \mathcal{P}} \mathbf{1}[t \in \text{Know}(i, \epsilon)] \cdot w_i \geq \frac{2}{3}W_\epsilon + \delta$$

This ensures eventual convergence to a single canonical ordering.

I. Virtual Voting and Deterministic Ordering

All Pillars deterministically derive the same transaction order:

$$O_\epsilon^{(p)} = O_\epsilon^{(q)} \quad \forall \text{ honest } p, q$$

Consensus therefore requires no explicit votes—only shared observation of DAG structure.

This is what grants NoM’s consensus its “virtual” nature.

J. Shared Coin for Termination

The draft references a shared coin round as a termination mechanism if consensus stalls (e.g., due to partitions or DoS). This mechanism is mentioned as providing eventual convergence guarantees but the specific algorithmic details are not provided in the draft.

Not Specified in Draft (Open Questions)

- Detailed algorithmic steps for the shared coin mechanism.
- Randomness source and generation procedure.
- How the coin round interacts with existing consensus state.
- Conditions triggering shared coin activation.

K. Pillar Proof-of-Work and Difficulty Adjustment

Pillars complete proof-of-work as part of epoch finalization. The draft indicates that difficulty adjusts based on epoch completion times to maintain stable consensus intervals.

Interpretive Description: Difficulty adjustment follows a feedback mechanism where faster epoch times increase difficulty and slower times decrease it, stabilizing the network rhythm over time. This affects Pillar-level PoW computation requirements for subsequent epochs.

Not Specified in Draft (Open Questions)

- Whether Pillars can delegate or outsource PoW computation to pools.
- Specific difficulty adjustment formula or parameters.
- How difficulty changes propagate and achieve consensus among Pillars.
- Relationship between Pillar PoW difficulty and transaction PoW link difficulty.

L. zApps and Unikernel Execution

Canonical mention: zApps (application-layer logic) exist but are not part of core consensus.

Interpretive expansion: zApps execute in sandboxed “unikernel” environments to ensure deterministic, isolated computation. Each application state follows:

$$\text{State}(A) \in \{\text{Proposed}, \text{Provisioned}, \text{Running}, \text{Checkpointed}, \text{Terminated}\}$$

Execution costs resources according to:

$$\text{Price}(A) = (p_{\text{cpu}}, p_{\text{mem}}, p_{\text{net}}, p_{\text{disk}})$$

A gas-like fee model funds execution, ensuring bounded computation.

M. Proof-Native Verification for zApps

zApps may emit cryptographic proofs of state transitions:

$$S : (s_{\text{in}}, x) \mapsto s_{\text{out}}$$

producing proof π_S such that:

$$\text{Check}(S, \pi_S) = \top$$

Light clients can verify or refuse (REFUSE) if resource limits are exceeded—preserving local correctness regardless of resource capacity.

N. Summary

The Network of Momentum’s consensus mechanism merges:

- PoW-based Sybil resistance,
- Stake-weighted virtual voting, and
- Deterministic epoch-based ordering.

Its dual-ledger architecture provides scalable, verifiable, and fault-tolerant distributed consensus, supporting nodes with diverse capabilities.

Not Specified in Draft (Open Questions)

- Full mathematical definition of O_ϵ (ordering function).
- Randomness source for shared coin rounds.
- Specific proof format for zApps.
- Whether the meta-DAG references block-lattice hashes directly or indirectly.

VI. Security Model

Canonical (from the 2020 draft)

- The security section explicitly outlines possible attack vectors.
- The canonical assumptions include:
 - An asynchronous network (no timing guarantees).
 - Safety and liveness depend on less than one-third Byzantine stake.
 - Consensus guarantees are probabilistic but converge deterministically under normal conditions.
- Each subsection (“Possible Attacks”) corresponds to a distinct vector: double spending, forking, DNS, eclipse, Sybil, DoS, consensus delay, and majority control.
- The draft’s language focuses on qualitative resilience rather than formal proofs.

Reconstruction (Interpretive, Non-normative)

NoM’s security model combines economic disincentives (PoW), stake-weighted voting (PoS), and deterministic ordering (meta-DAG) to maintain correctness and progress. Each canonical attack class is outlined below with reconstruction commentary and open questions.

A. Double-Spending Attack

Canonical Claim: The draft indicates that conflicting transactions (“double spends”) are resolved deterministically after several epochs. All honest Pillars eventually learn both transactions and converge on a single valid one, discarding the other using predefined rules. It mentions a potential penalizing algorithm under consideration for malicious actors.

Clarification (Interpretive): Convergence depends on the supermajority condition. As both conflicting transactions propagate, every honest Pillar observes both and applies the deterministic resolution mechanism. Thus, only one transaction survives, preventing double spend.

Not Specified in Draft (Open Questions):

- The exact specification of conflict resolution rules.
- The “penalizing algorithm” is mentioned but undefined.
- Whether offenders lose stake, PoW credit, or reputation is unspecified.

B. Forking (Ledger Cloning) Attack

Canonical Claim: Forking is mitigated by dual PoW layers:

1. Transaction-level PoW (links).
2. Epoch-level PoW by Pillars.

The heavier ledger (with greater accumulated PoW) is considered canonical. New nodes synchronize with multiple Pillars to confirm the heaviest valid ledger.

Clarification (Interpretive): Ledger weight represents the aggregate work across Pillars and transaction links. Multi-source synchronization prevents deception because an attacker cannot easily fabricate PoW history across independent Pillar sets.

Not Specified in Draft (Open Questions):

- “Several Pillars” is undefined—minimum count unknown.
- The exact synchronization or bootstrap algorithm is not given.

C. DNS Attacks

Canonical Claim: DNS-based peer discovery can be subverted by IP injection. The draft acknowledges this as a real risk and references mitigations “existing in similar systems.”

Clarification (Interpretive): The draft treats peer discovery as out of scope for NoM-specific solutions, implying reliance on standard DNS hardening or alternative bootstrap methods.

Not Specified in Draft (Open Questions):

- Whether NoM uses DNS seeds, static lists, or DHTs for discovery.
- No canonical countermeasure is defined.

D. Eclipse Attacks

Canonical Claim: Eclipse attacks (isolating a node by controlling its peers) are unlikely for Pillars because of their connectivity and identity hardness. However, users connecting to only a few Pillars could be isolated. The draft suggests randomizing connections at startup to reduce success probability.

Clarification (Interpretive): The design assumes that Pillars maintain sufficient redundancy and diversity in connections, reducing eclipse feasibility except for isolated clients.

Not Specified in Draft (Open Questions):

- Quantitative peer connection thresholds are unspecified.
- No formal bootstrap policy described.

E. Sybil Attacks

Canonical Claim: Sybil attacks confer no advantage since consensus power is weighted by stake, not node count.

Clarification (Interpretive): This holds if stake distribution is honest and stake cannot be rapidly split or recombined. PoW link costs further discourage node inflation for spam purposes.

Not Specified in Draft (Open Questions):

- How stake is measured under churn (joining/leaving).
- Interaction between stake and PoW layers over time.

F. Denial-of-Service (DoS) Attacks

Canonical Claim: Transaction flooding (DoS) against Sentinels is mitigated by transaction fees, which impose a cost on spamming. Consensus is unaffected because Pillars process only finalized transactions.

Clarification (Interpretive): DoS resistance derives from economic friction: attackers must expend real resources to maintain spam. Even if Sentinels are overwhelmed, the consensus layer remains safe, albeit delayed.

Not Specified in Draft (Open Questions):

- Minimum fee requirements and adaptive mechanisms not defined.
- Sentinel overload behavior unspecified.

G. Consensus Delay

Canonical Claim: If an attacker interferes with Pillar communication (e.g., through DDoS), consensus may stall temporarily. However, probability of eventual supermajority approaches 1 as epochs progress. Mentions a shared coin epoch invoked after several consecutive non-finalizing epochs.

Clarification (Interpretive): The “shared coin” acts as a probabilistic termination mechanism to break tie situations. Though theoretical, it guarantees eventual convergence without central intervention.

Not Specified in Draft (Open Questions):

- How consensus delay is detected.
- Randomness source for the shared coin.
- Public auditability of randomness events.

H. Majority (51%) Attack

Canonical Claim: If an adversary controls $>50\%$ of total stake, it can include or reorder new transactions but cannot rewrite past ones. The draft explicitly states that the honest-majority assumption must hold and mentions a “hard limit condition.”

Clarification (Interpretive): This corresponds to the classical security limit in stake-weighted systems. An attacker exceeding the honest majority undermines safety but cannot retroactively forge prior epochs due to PoW commitments and finality checkpoints.

Not Specified in Draft (Open Questions):

- How long-range stake attacks are mitigated.
- Recovery protocol after majority takeover.

I. Consolidated Security Table

Attack	Canonical Claim	Clarification	Open Gaps
Double Spend	Conflicts resolved deterministically	Relies on convergence and resolution rules	Penalization undefined
Forking	Heaviest ledger rule	PoW accumulation prevents fake forks	Peer count unspecified
DNS	Peer discovery injection	Standard mitigations	Bootstrap unspecified
Eclipse	Client isolation risk	Pillars robust due to identity hardness	Peer selection undefined
Sybil	Stake-weighted protection	Node inflation yields no advantage	Stake churn rules missing
DoS	Economic friction via fees	Costs limit spam	Fee policy undefined
Consensus Delay	Shared coin ensures progress	Probabilistic termination	Detection/randomness missing
Majority	Honest majority required	Cannot alter past ledger	Recovery unspecified

Table 6: Consolidated security analysis

J. Summary of Canonical Security Assumptions

1. Asynchronous network: No timing guarantees.
2. Honest majority: Safety holds if $< \frac{1}{3}$ Byzantine stake.
3. Eventual propagation: Every valid transaction eventually reaches a supermajority.
4. PoW costs: Spam and Sybil resistance enforced economically.
5. Deterministic ordering: Virtual voting ensures convergence.

Not Specified in Draft (Open Questions)

- Detailed detection mechanism for consensus delay.
- Implementation of penalization system.
- Randomness generation for coin rounds.
- Recovery protocol for temporary 51% takeover.
- Concrete fee policy for DoS mitigation.
- Quantitative connectivity requirements for eclipse resistance.

VII. Parameters and Complexity Analysis

Canonical (from the 2020 draft)

- Analyzes overall protocol efficiency in terms of message complexity, time complexity, and verification cost.
- Indicates that during an epoch:
 - Users issue transactions through Sentinels.
 - Pillars conduct broadcasts for consensus.
 - Verification occurs locally per node.
- States that safety and liveness scale with the number of Pillars (N), assuming a supermajority of honest stake.
- Provides high-level asymptotic behaviors for message propagation and consensus communication.
- Emphasizes scalability through resource heterogeneity (light clients and full nodes).

Reconstruction (Interpretive, Non-normative)

A. Overall Complexity

Let:

- S : number of Sentinel nodes.
- N : number of Pillar nodes.
- M : number of user transactions per epoch.

Transaction dissemination follows logarithmic complexity through the Sentinel layer:

$$O(\log S)$$

Consensus broadcast among Pillars is quadratic globally or linear per node:

$$O(N^2) \text{ globally, } O(N) \text{ per Pillar.}$$

Each Pillar must send messages to every other Pillar for epoch finalization, ensuring complete state synchronization. Expected epoch duration remains approximately constant under stable conditions, with convergence time bounded by network latency rather than computation.

Interpretive Clarification: This analysis assumes all Pillars maintain authenticated communication channels and use deterministic message processing (virtual voting). Even with $O(N^2)$ message exchanges, the system achieves scalability by limiting consensus participation to a finite Pillar set while Sentinels and Sentries handle user interactions.

B. Verification and Asymptotic Behavior

Verification complexity for a light verifier is approximately:

$$O(\log n + m)$$

where:

- n = total number of transactions in the ledger.
- m = number of proofs retrieved for the specific account.

This logarithmic relation ensures light clients can validate their data efficiently using compact proof bundles.

Interpretive Note: Asymptotic behavior indicates that verification costs grow sublinearly with ledger size, supporting scalability for resource-constrained devices. Actual throughput depends heavily on network topology, propagation delays, and implementation details not specified in the 2020 draft.

C. Representative Selection and Network Robustness

Users select multiple Representatives (Sentinels) to connect for redundancy and reliability. Let:

- f : fraction of malicious Sentinels.
- $k = \lceil \log S \rceil$: number of Representatives contacted.

The probability that a user connects exclusively to compromised Representatives:

$$P_{\text{fail}} = f^k$$

As k increases, P_{fail} decays exponentially, ensuring strong resilience even if a significant portion of Sentinels are malicious.

Interpretive Note: This probabilistic connectivity model emphasizes fault tolerance through multi-peer randomness rather than centralized trust.

D. Epoch Management and Difficulty Adjustment

Epoch duration depends on PoW difficulty (D_e) and broadcast efficiency. The draft indicates that difficulty adjusts based on actual versus target epoch completion times to maintain stable consensus intervals.

Interpretive Description: If epochs complete faster than the target, difficulty increases; if slower, difficulty decreases. This feedback mechanism stabilizes network rhythm over time and prevents excessive variance in consensus intervals. The adjustment occurs deterministically and affects Pillar-level PoW computation requirements for subsequent epochs.

Not Specified in Draft (Open Questions):

- Specific difficulty adjustment formula, parameters, or convergence rate.
- Whether difficulty adjustment applies symmetrically to both PoW layers (Pillar vs. transaction).
- How difficulty changes propagate and achieve consensus among Pillars.

E. Cryptoeconomic Incentives

The cryptoeconomic layer balances incentives for participation, ensuring fairness and availability across actors.

Role	Reward Source	Behavior Incentivized
Sentinels	Transaction fees, PoW link rewards	Relay transactions, supply proofs
Pillars	Epoch PoW completion rewards	Maintain consensus integrity
zApp Executors	Gas-like resource pricing	Provide deterministic computation

Table 7: Cryptoeconomic incentives

Interpretive Clarification: If a Pillar fails to complete its PoW before supermajority finalization, it forfeits epoch rewards. This creates competition to maintain network liveness and discourages idle or faulty Pillars.

F. Proof-of-Stake Integration in Consensus

The 2020 draft integrates stake-weighted voting directly into virtual voting. A transaction t is finalized when the observed cumulative stake supporting it meets the supermajority threshold:

$$V(t, \epsilon) \geq \frac{2}{3}W_{\epsilon} + \delta$$

where $V(t, \epsilon)$ is the total observed stake of Pillars that have acknowledged transaction t .

Not Specified in Draft (Open Questions):

- How stake is locked or committed for consensus participation.
- Whether delegation mechanisms exist for smaller stakeholders.
- Economic penalties or slashing conditions for misbehavior.
- Stake update frequency and withdrawal procedures.

G. Managing Consensus Under Network Degradation

If network disruption prevents finalization after several epochs, the shared coin mechanism guarantees eventual progress probabilistically:

$$P_{\text{finish}} \rightarrow 1$$

as epochs increase.

Interpretive Clarification: The shared coin acts as a randomized mechanism ensuring consensus termination without introducing central control. While rarely invoked, it provides theoretical completeness—proof that the system cannot remain indefinitely stalled.

Not Specified in Draft (Open Questions)

- Quantitative upper limits for epoch durations or block sizes.
- Specific distribution ratios for Pillar and Sentinel rewards.
- Actual network performance metrics (throughput, latency).
- Detailed conditions for invoking the shared coin round.
- How Representative nodes are elected or retired.

VIII. Conclusions and Future Work

Canonical (from the 2020 draft)

- The draft concludes by reaffirming NoM’s design as a hybrid, verification-first distributed ledger.
- It highlights three architectural pillars:
 1. Dual-ledger structure — separating ordering (meta-DAG) from execution (block-lattice).
 2. Hybrid consensus — combining proof-of-work pacing with stake-weighted virtual voting.
 3. Bounded verification — ensuring correctness even with limited connectivity or resources.
- Emphasizes NoM as an ongoing research effort rather than a finalized protocol.
- Encourages further formalization, performance evaluation, and implementation testing.

Reconstruction (Interpretive, Non-normative)

The Network of Momentum (NoM) represents a conceptual evolution in distributed ledger architecture. It achieves secure, scalable consensus without sacrificing accessibility or correctness by decoupling verification, ordering, and execution into distinct layers.

Key Interpretive Insights:

1. **Verification-first paradigm:** NoM prioritizes correctness over throughput—ensuring that nodes can refuse unverifiable data rather than propagate uncertainty.
2. **Dual-ledger architecture:** The meta-DAG ensures global order and finality; the block-lattice supports parallel user transactions. This division simplifies scaling and improves modularity.
3. **Economic and cryptographic equilibrium:** Proof-of-work maintains Sybil resistance, while stake-weighted voting enforces fairness and convergence. The system’s cryptoeconomic model aligns incentives for all participants (Pillars, Sentinels, verifiers).
4. **Heterogeneous participation:** By accommodating light clients (Sentries) alongside full Pillars, NoM democratizes participation without compromising safety.
5. **Future adaptability:** Its design leaves space for integrating new verification methods, zero-knowledge proofs, or future consensus optimizations.

Together, these properties establish a foundation for a self-sustaining, trust-minimized network capable of operating securely across heterogeneous environments.

A. Canonical Future Work

The canonical draft mentions several open research directions:

- Formal verification of NoM’s correctness properties under partial synchrony.
- Quantitative evaluation of throughput and latency at scale.
- Development of formal specifications for proof markets and bounded verification.
- Exploration of “proof-carrying execution” for distributed applications (zApps).

B. Reconstruction (Interpretive, Non-normative)

The community recognizes several ongoing research priorities derived from the canonical intent:

- 1. Proof Market Dynamics:** Investigate how bounded verification naturally creates an economic marketplace for proof availability. Study equilibrium behaviors between verifiers (demand) and proof suppliers (Sentinels/Pillars).
- 2. Formalization of REFUSE Semantics:** Develop a rigorous logical model for three-outcome verification (accept/reject/refuse) ensuring safety and composability.
- 3. Proof Dissemination Optimization:** Design caching and relay strategies to minimize latency in delivering proof bundles to bounded verifiers.
- 4. Cross-Ledger Interoperability:** Explore how the dual-ledger model might connect with external consensus systems through shared verification interfaces.
- 5. zApp Infrastructure and Execution Proofs:** Advance research on verifiable computation and proof-carrying execution, extending NoM’s verification-first principles into the application layer.
- 6. Security Evaluation and Incentive Modeling:** Formalize adversarial models for DoS, stake manipulation, and consensus delay, analyzing resilience and incentive compatibility.

Not Specified in Draft (Open Questions)

- How proof markets might be priced or stabilized economically.
- Whether the REFUSE condition can integrate with cross-chain verifications.
- Specific interoperability protocols between external ledgers.
- Implementation constraints for zApp proof generation.
- Trade-offs between proof latency and network decentralization.

Appendix A — Traceability Table

Claim ID	Claim Summary	Draft Section	Reconstruction Reference
C1	Double-spend resolution through deterministic convergence	Possible Attacks §A	VI.A
C2	Fork resistance via dual PoW and heaviest-ledger rule	Possible Attacks §B	VI.B
C3	DNS bootstrap vulnerability noted; mitigations unspecified	Possible Attacks §C	VI.C
C4	Eclipse risk minimal for Pillars; randomized peers recommended	Possible Attacks §D	VI.D
C5	Stake-weighted voting prevents Sybil advantage	Possible Attacks §E	VI.E
C6	Transaction fees limit DoS effectiveness	Possible Attacks §F	VI.F
C7	Shared coin ensures consensus termination under delay	Possible Attacks §G	VI.G
C8	Majority control can reorder future transactions only	Possible Attacks §H	VI.H

Table 8: Claim traceability to canonical draft sections

Appendix B — Non-normative Analysis Notes

This appendix contains interpretive analytical observations that are not part of the canonical 2020 draft. They represent community-derived reasoning about the system’s theoretical properties and should be treated as speculative illustrations rather than verified theorems.

Note 1 — Transaction Propagation Observation

Interpretive claim: If a transaction is valid and broadcast by its Representatives, all honest Pillars will eventually observe it over time. Absence beyond reasonable periods may indicate network fault or malicious omission, detectable through querying multiple Representative sets.

Note 2 — Conflict Resolution Observation

Interpretive claim: Given two conflicting transactions, the draft indicates that honest Pillars should converge on one transaction through deterministic rules. The losing transaction would be discarded, preventing double-spend inclusion. Specific convergence time and conditions are not provided in the draft.

Note 3 — Knowledge Convergence Pattern

Interpretive claim: The asynchronous nature of the network means Pillars operating in any given epoch may have incomplete knowledge of earlier transactions. Over successive epochs, transaction knowledge propagates and converges toward completeness across the Pillar set.

Not Specified in Draft (Open Questions):

- Specific convergence bounds or timing guarantees.
- Whether convergence can be formally proven under stated assumptions.
- Quantitative relationship between epoch count and knowledge completeness.

Note 4 — PoW Inclusion Pattern

Interpretive claim: Once a Pillar completes its PoW and broadcasts it, the draft suggests other honest Pillars will include it in their consensus view, contributing to epoch finalization and fair participation.

Note 5 — Message Complexity Observation

Interpretive claim: Message complexity per consensus round appears to grow with the product of transactions and logarithm of relay nodes, potentially growing sublinearly as the network scales if relay topology is optimized.

Not Specified in Draft (Open Questions):

- Exact complexity bounds and their dependence on network parameters.
- Whether sublogarithmic scaling holds under adversarial conditions.

Note 6 — Asymptotic Scaling Pattern

Interpretive claim: The separation of consensus (Pillars) from transaction relay (Sentinels) suggests that transaction processing could scale more favorably than traditional single-layer architectures, though actual

performance depends on implementation details not provided in the draft.

Appendix X — zApps and Intentional Under-Specification

Canonical (from the 2020 draft)

- Mentions zApps as a potential application layer on top of the base ledger.
- Indicates explicitly that execution semantics are “intentionally under-specified.”
- Clarifies that consensus, correctness, and verification are defined independently of application execution.

Reconstruction (Interpretive, Non-normative)

This appendix interprets the rationale for under-specification: Zenon’s architecture prioritizes verification over execution. By not coupling correctness to application runtime, the protocol remains open-ended and future-proof.

X.1 Purpose

zApps are not required for consensus; they exist downstream of it. Ordering and correctness remain valid even if zApps fail or diverge.

X.2 Execution Constraints

1. **Non-Interference:** Execution outcomes cannot alter consensus state.
2. **Optional Participation:** Nodes may ignore zApps without affecting correctness.
3. **Bounded Verifiability:** Execution verification must permit REFUSE.
4. **Asynchronous Compatibility:** Verification may be delayed or partial.
5. **No Implicit Trust:** zApps cannot demand global acceptance.

X.3 Unikernels as Illustration

Unikernels serve as examples of lightweight, bounded execution environments aligning with verification-first design. They enable reproducible computation, explicit resource control, and isolation—but are not mandated by protocol.

X.4 Proof-Carrying Execution (Non-normative)

Future zApps may provide cryptographic proofs of computation, enabling verifiers to check correctness locally. Refusal to verify never compromises ledger safety.

Not Specified in Draft (Open Questions)

- Formal proof-of-execution format for zApps.
- Whether unikernels are canonical or illustrative only.
- Economic incentives for zApp execution or proof publication.
- Integration pathways between zApp outputs and block-lattice states.

Appendix Y — Unikernels (Illustrative, Non-normative)

Y.1 Definition

A unikernel is a single-purpose machine image bundling an application and minimal operating system components into one verifiable binary artifact. It reduces complexity, improves isolation, and allows explicit resource bounding.

Y.2 Role in Verification-First Design

Unikernels fit the NoM philosophy by supporting:

- Small trusted computing base.
- Deterministic execution boundaries.
- Explicit CPU, memory, and I/O limits.

Y.3 Deployment Considerations

Compiled as immutable images with hash-based identity:

$$\text{ID}(U) = H(\text{unikernel_image_bytes})$$

Can be distributed and referenced by digest rather than mutable server state.

Y.4 Operational Notes

- Reduced debugging visibility but higher determinism.
- Must remain optional and correctness-neutral.
- Should expose explicit, auditable input/output interfaces.

Y.5 Relationship to Bounded Verification

Unikernels must allow verifiers to refuse unverifiable outputs. They enhance modularity but do not replace local verification.

Y.6 Summary

Unikernels illustrate how execution can remain bounded, verifiable, and independent of consensus correctness. Their inclusion reflects architectural flexibility rather than protocol dependence.

Final Summary

The Network of Momentum (NoM) unites dual-ledger structure, hybrid consensus, and bounded verification to form a verification-first architecture. It preserves decentralization, scalability, and correctness under asynchronous conditions. Its modular design ensures adaptability for evolving proof systems and verifiable computation, establishing NoM as both a theoretical framework and an evolving experimental architecture.

END OF DOCUMENT — Community Reconstruction (Interpretive, Non-normative)