

ENS Routing as a Market Mechanism

Provably fair, economically enforced compute routing on AGI.Eth

identity → market → proof → settlement → governance

*.alpha.club.agi.eth

*.alpha.agent.agi.eth

*.alpha.node.agi.eth

*.alpha.agi.eth

Executive charter

IDENTITY → EVIDENCE → SETTLEMENT →
GOVERNANCE

Identity

Roles bound to keys (ENS/DID)
(ENS/DID)
Accountability by default

Evidence

Deterministic runs emit proof
proof bundles
Replay beats rhetoric

Settlement

Escrow releases only after
validation
Receipts are settlement-grade

Governance

Policy gates + upgrades +
emergency brakes
Autonomy is bounded

Invariant no value without evidence · no autonomy without authority · no settlement without validation

Thesis

Routing becomes an open market (bids) + a public proof system (commit-reveal) + an enforcement game (bonded challenges).

The router stops being a “trusted market maker” and becomes a verifier-friendly mechanism.

One rule + scoped meaning

`<entity>.(<env>.)<role>.agi.eth`

`role` ∈ {club, agent, node} • `env` ∈ ENV_SET (optional)

Example (env = alpha)

Validator (club): `alice.club.agi.eth` | `alice.alpha.club.agi.eth`

Agent: `helper.agent.agi.eth` | `helper.alpha.agent.agi.eth`

Node: `gpu01.node.agi.eth` | `gpu01.alpha.node.agi.eth`

Scope rule `entity.env.role.agi.eth` is recognized & developed only within `env.agi.eth`

Official environment package (issued by AGI King)

`env.agi.eth` • `env.agent.agi.eth` • `env.node.agi.eth` • `env.club.agi.eth`

Role gating used in routing + validation:

- only `*.club.agi.eth` can attest/validate
- only `*.node.agi.eth` can execute
- only `*.agent.agi.eth` can bid as agents

Mechanism (high-level)

- 1) Job arrives (demand)
- 2) Eligible nodes/agents submit bids (supply)
- 3) Deterministic winner selection
- 4) Receipt emitted → metrology → settlement

Example scoring (lower is better)

$$\text{price}_i = \alpha \cdot \text{latency}_i - \beta \cdot \log(\text{stake}_i) + \gamma \cdot \text{failureRisk}_i$$

Stake acts as a risk buffer. Latency acts as the immediate cost.

A rational node bids to maximize long-run profit subject to slashing risk.

Protocol guarantees

- Bids are EIP-712 signed by ENS controllers
- Eligibility is ENS-gated (club/agent/node)
- Epoch roots anchor commitments, reveals, receipts
- Auditors can replay winner selection deterministically

Physics intuition (operational)

Minimize coordination “free energy”:

- fewer on-chain writes
- fewer governance touch-points
- push churn into resolvers (CCIP-Read), not naming

Make the simple path the dominant strategy.

Commit window

Workers submit
 $\text{commit} = H(\text{bid} \parallel \text{nonce})$

Reveal window

Reveal bid + nonce
(router can verify commit)

Deterministic selection

winner = $\text{argmin}(\text{price})$
from revealed bids + seed

Public seed

$\text{seed} = \text{keccak}(\text{epochId} \parallel \text{sealedSeed} \parallel \text{blockhash})$

Any verifier can recompute the winner:

- validate each reveal matches its commitment
- compute price_i for each bidder
- apply deterministic tie-break: $\text{keccak}(\text{bidder} \parallel \text{seed})$

Why commit-reveal matters

- Reduces herding and last-second bribery
- Makes routing verifiable without trusting the router
- Creates clean dispute objects (commit, reveal, seed)
- Enables optimistic slashing with minimal on-chain compute

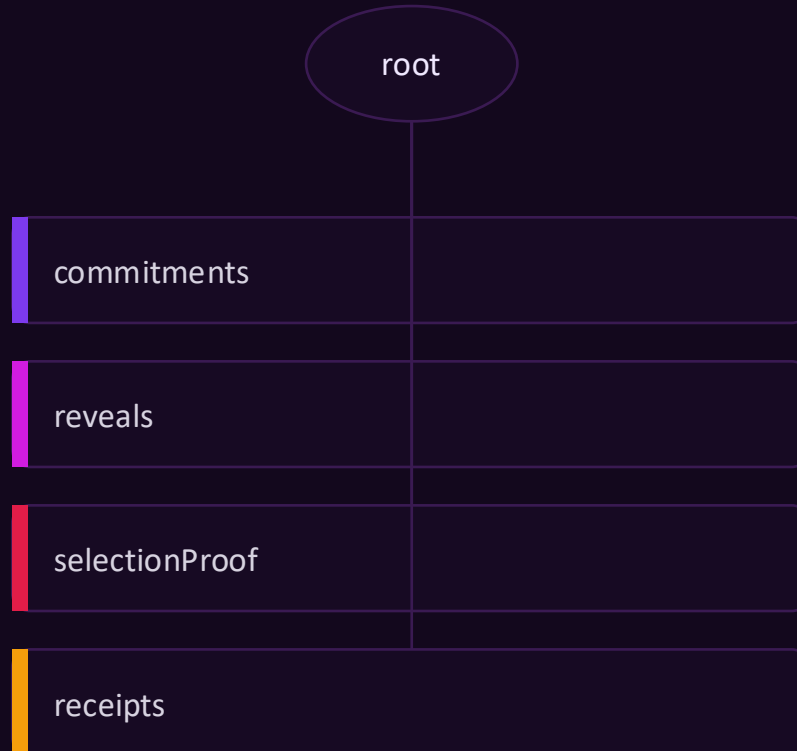
Outcome

“We say it’s fair” → “the network proves it’s fair.”

The router publishes a selectionRoot; verifiers independently recompute winners.

Mismatch becomes an objective slashable event.

EpochRoot = MerkleRoot(...)



What each leaf contains

commitments:

bidder, epochId, H(bid||nonce)

reveals:

bidder, bid, nonce

selectionProof:

seed, winner, tieBreak, recomputation hash

receipts:

jobId, winner, outcome, latency, signatures

Verifier-friendly property

Anyone can rebuild the epoch and arrive at the same winner winner set.

If the published root disagrees with recomputation, the the discrepancy is provable.

Bot pipeline (public)

Ingest on-chain events

CommitBid / RevealBid
SeedPublished / RootPublished

Recompute winners

Verify commits
Apply deterministic selection

Compare selectionRoot

Match → ok
Mismatch → generate proof

Publish proof package

JSON + Merkle proof
Tx links + replay script

Public Challenge Board

A mismatch triggers an objective dispute:

- proof package posted (CID + hash)
- merkle path to the published root
- deterministic recomputation transcript
- tx links for commits/reveals/seed

Unstoppable property

Verification is automatic.

No committee can “look away.”

Next version: verifier can post a bonded challenge on-chain to slash cheaters.

Bonded optimistic slashing game

GAME-THEORETIC ENFORCEMENT

Rules (simple + sharp)

- Router posts a slashable bond (ETH or \$AGIALPHA)
- Anyone may challenge with a bond
- If challenge is correct → router bond slashed, challenger rewarded
- If challenge is false/spam → challenger bond slashed
- On-chain work stays minimal (optimistic verification)

Payoff intuition

Cheating becomes negative-sum:

- Honest router: no slash risk → stable profits
- Dishonest router: high expected loss (anyone can challenge)

Spam becomes negative-sum:

- False challenger loses bond

On-chain flow (minimal)

Router posts EpochRoot + bond



Dispute window opens



Challenger posts proof + bond



If valid → slash router, reward challenger



If invalid → slash challenger

When routing is enforced by contract, reliability becomes composable.



Non-performance becomes objective

Timeout → slash stake → refund escrow → chronicle receipt
Reliability is not a promise. It's a consequence.

Policy gates (bounded autonomy)

- ENS role gating (agent/node/club)
- dispute windows + bonded challenges
- fail-closed: pause + emergency brakes

Leaderboards

Nodes (*.node.agi.eth)

- latency p50 / p95
- success rate
- α -WU / epoch
- dispute rate

Validators (*.club.agi.eth)

- honest reveal rate
- SLO scoring accuracy
- dispute participation
- slashing history

Agents (*.agent.agi.eth)

- bid competitiveness
- task completion quality
- reputation under disputes

Metrology (α -WU)

α -WU = signed telemetry $\times$ difficulty tier $\times$ quality score

Canonical, hardware-normalized work unit for settlement.
settlement.

Fails acceptance/SLO $\rightarrow$ 0 credit.

Epoch signals:

- validated α -WU / epoch
- dispute rate
- SLO drift

Actuators:

- fee splits
- quorum + slashing parameters
- tier multipliers

Olympics outcome

The market discovers who is fast, honest, and reliable —
reliable — under public proof.

Pilot

- Launch auditable router
- Publish epoch roots + receipts
- Open read-only scoreboard
- Run verifier bot in shadow mode

Output: auditability

Harden

- Add commit-reveal bidding
- Deterministic winner proofs
- Public challenge board
- Bonded optimistic slashing

Output: provable fairness

Scale

- RoutedJobManager (on-chain assignment)
- Slashing for non-performance
- Validator quorums
- Expand to multi-region nodes

Output: true enforcement

Build the Cathedral of Verifiable Work

Autonomy measured • Work proven • Value settled

Next move (Alpha): launch ENS Routing Olympics with automatic enforcement

- 1) Publish season manifest to `olympics.alpha.agi.eth`
- 2) Deploy EpochRegistry + (bonded) ChallengeRegistry
- 3) Run `fairness01.alpha.agent.agi.eth` (verifier bot)
- 4) Open `challenge.alpha.agi.eth` public board
- 5) Graduate to RoutedJobManager (on-chain assignment + slashing)

Institutional invariant

No payout without validated proof. No fairness without public verification.

`agi.eth` • ENS Routing as a Market Mechanism