Wait, Who's Going to Maintain All This?

The Unbrowse Maintenance Network: Proof of Indexing and Bonded Accountability in a Shared Route Graph

Lewis Tham
Unbrowse AI
lewis@unbrowse.ai

June 2026

Abstract

A shared graph of reusable web routes solves discovery: an agent looks up a route someone already mapped instead of re-deriving it from a browser on every call, and the wedge paper [1] shows this wins whenever the route fee undercuts the cost of rediscovery. Discovery, though, is a one-time cost; freshness is a standing liability. Routes decay — schemas drift, auth flows mutate, fields are renamed behind cheerful `200`s — and a route that is wrong with confidence is worse than no route at all. This paper takes up the maintenance problem: who keeps the routes fresh, and how is the claim that a route is fresh made trustworthy. We frame the graph as a common-pool resource that degrades under uncoordinated use [5], show why pure usage fees structurally under-provide freshness (a non-rival public good [6] whose decay is a negative externality [7]), and make freshness a *verifiable artifact* rather than a bare assertion: a maintainer publishes a *proof of indexing* — a content-addressed attestation that a route was re-indexed against its live source, in the lineage of The Graph's Proof of Indexing [13] and Filecoin's proof that storage is held across time [14]. Collateralised accountability secures the proof: a maintainer bonds the network's security asset against it, and a proof that fails to reproduce under independent re-indexing is slashable [9], finalised by a threshold quorum [10] and adjudicated against a tamper-evident, independently auditable log [12]. We are explicit about the token this requires — why it exists strictly for the security function, why its fair distribution is a security parameter rather than a marketing choice, and why a coordination asset launched for money cannot become trustworthy — and equally explicit about the limit we have not closed, formal Sybil resistance [8]. Throughout we separate what is implemented in the product from what is available as runnable, tested reference code and what remains a design. We stop at the trust surface and do not publish the capture method itself [4].

1 Introduction: the maintenance problem, not the discovery problem

The first problem an agent has on the open web is *discovery*: finding the route in the first place is expensive, and re-deriving it on every call is wasteful. The wedge paper [1] reduces it to a single inequality — when the fee for reusing a route someone already mapped, $f_{\text{shared route}}$, is less than the cost of rediscovering it from scratch, $c_{\text{rediscovery}}$, the rational move is to share. A live route graph already does this: an intent resolves to a ranked endpoint shortlist and executes, and the cost of mapping each route is amortised across every agent that follows. That result is established and we build on it rather than restate it.

This paper is about the *second* problem, which only shows up once the first is solved. A graph with real traffic is a graph with real decay. Schemas drift; authentication flows mutate; a field is renamed, a pagination cursor changes shape, an endpoint is deprecated behind a redirect that returns a cheerful `200`. None of this is adversarial — it is simply what live systems do. The inequality that justified sharing the route says nothing about who keeps it true a month later. A route that was correct when it was mapped and is wrong when it is used is worse than no route at all, because it is wrong with confidence: the agent does not know to fall back. Discovery is a one-time cost; freshness is a standing liability.

So the question this paper takes up is narrow and unglamorous: *who keeps the routes fresh, and how is their claim that a route is fresh made trustworthy?* Maintenance is not a feature bolted onto discovery; it is the thing that decides whether the shared graph degrades into a liability or compounds into an asset. We separate, throughout, what is running from what is designed. The route graph itself — resolution, execution, the signed record of which routes were used and what they returned — is live. The accountability machinery that makes a freshness claim costly to fake — the bonding, challenge, and slashing layer described in the later sections — ships as a runnable, tested reference implementation (`reference/network/`), distinct from production deployment. We would rather ship a running reference and call it a reference than over-claim it as a deployed product.

2 The route as a common-pool resource

The shared route graph is not a public good in the textbook sense, and naming it correctly matters for the governance that follows. It is a *common-pool resource* (CPR) in the precise sense Ostrom gave the term [5]: a resource from which it is hard to exclude users, but whose value is depletable through uncoordinated use. The depletion here is not consumption of a finite stock — a route does not get used up when an agent calls it. The depletion is *epistemic*. Every agent that leans on a stale route and propagates its output as correct, every contributor who adds a route and never returns to confirm it still resolves, lowers the average trustworthiness of the graph for everyone. The classic tragedy — each participant rationally drawing down a shared resource because the cost of their own neglect is socialised — reappears exactly, with staleness in the place of overgrazing.

The reason this framing is useful, rather than merely a tidy metaphor, is that the CPR literature does not stop at diagnosing the tragedy. Ostrom's central finding is that durable commons are neither privatised nor centrally administered; they are governed by the appropriators themselves under a small set of recurring design principles [5]. Three of those principles map directly onto the layers this paper builds. *Clearly defined boundaries* — who is entitled to claim a route is fresh, and who may draw on it — become the trust-tier and identity machinery: a freshness claim is not anonymous, it is attached to an accountable party. *Graduated sanctions* — penalties that scale with the severity and repetition of a violation rather than a single all-or-nothing punishment — become the slashing tiers, where a first low-confidence miss is treated differently from a repeated or high-value false claim. *Nested enterprises* — governance organised in layers, smaller units inside larger ones — become the trust tiers themselves: open routes governed lightly for low-risk traffic, sitting inside more tightly governed high-trust routes for authenticated or high-value paths.

We borrow the template, not the institution. Ostrom studied irrigation systems, fisheries, and forests — human appropriators with faces and long memories. A route graph traversed by agents has neither, which removes some of the informal social enforcement her commons relied on and forces the corresponding work onto explicit, mechanical accountability. The economic case for why that explicit machinery is necessary at all — why pure usage fees under-provide freshness, and why a bond is the corrective rather than an embellishment — is taken up next; the cryptographic

and settlement primitives that make the sanctions enforceable, including the micropayment rail [3] over which usage is settled and against which maintenance must be funded, are developed in the sections that follow. What the CPR lens fixes, here at the outset, is the shape of the problem: the graph degrades under uncoordinated use, and a graph that degrades needs governance, not just a price.

3 Why pure usage under-provides freshness

A route fee is a usage fee: it is paid by the agent that calls a route, in proportion to how often it calls. That is exactly the right instrument for *rationing* a route — whoever uses it more pays more — and exactly the wrong instrument for *funding the freshness of the graph as a whole*. The maintained graph is a non-rival good: my reading a fresh route does not consume it, and a thousand other agents can read the same fresh route at no marginal cost to me [6]. It is also close to non-excludable in the dimension that matters — once a route is verified fresh, the benefit of that freshness accrues to every agent that touches it, not only to the one whose fee happened to trigger the check. A good that is non-rival and hard to exclude is the textbook public good [6], and the textbook consequence follows: each agent prefers to let *someone else's* fee pay for the maintenance and then free-ride on the result. Sum that preference over a whole population of rational callers and maintenance is systematically under-supplied. Usage fees alone do not fail because the price is wrong; they fail because freshness is the wrong kind of good to price by the call.

The mirror image of the under-supplied public good is the over-supplied externality. When a maintainer lets a route rot — a schema drifts, an auth flow mutates, a parameter is silently renamed — the cost of that decay does not land on the maintainer who neglected it. It lands on every downstream agent whose call now fails, retries, or quietly returns wrong data. That is a negative externality in the precise Pigovian sense: a private action (neglect) whose social cost (aggregate failure across all callers) is borne by parties the actor never compensates [7]. Pigou's prescription for an uninternalised externality is not exhortation but a price on the harmful act, set so the actor faces the cost it imposes on others [7]. In a route graph the corresponding instrument is a bond the maintainer posts against its own freshness claim, slashable when the claim turns out false — the Pigovian corrective rendered as collateral rather than a tax. Staleness stops being a free option and becomes a liability the claimant actually carries.

These two failures point at the same structural conclusion, and it is the reason the security function cannot ride on the usage rail. The usage rail — stablecoin settlement bound to each `execute` over x402 [3] — is built to meter and route payment for calls that happen; it is excellent at exactly that and should stay narrow. But under-provision of a public good and the pricing of a negative externality are both *accountability* problems, not metering problems: they need an asset that can be bonded, staked, and slashed against the integrity of the graph itself, whose value is tied to that integrity rather than to call volume. That is the job we reserve for FDRY, kept strictly separate from the USDC usage rail for the same reason one does not pay a deductible out of the till: the thing that funds maintenance and the thing that prices its failure must be the asset whose worth rises and falls with the network being worth trusting at all. The bond-and-slash corrective ships as a runnable, tested reference (`reference/network/bonding.py`), not yet a production mechanism; the usage rail it sits beside is live [3].

4 Proof of indexing: making freshness a verifiable artifact

The previous section establishes *why* the graph needs accountability; this one supplies the primitive the accountability is about. A bond, by itself, only punishes a freshness claim after it has been found false — it presupposes some way to find it false. We want more than a slashable assertion: we want freshness to be a *verifiable artifact* a stranger can check, not a promise they must take on faith. That artifact is a *proof of indexing*.

A proof of indexing is what a maintainer produces by actually re-indexing a route against its live source: a content-addressed attestation, committed to the hash-chained ledger, that the route still resolves and that its observed shape — the schema, the fields an agent depends on, the auth handshake — matches what the proof commits to, at a stated time. The proof is the freshness evidence. Anyone can verify the commitment against the ledger, and any independent party can re-index the same route and check that they reconstruct the same proof. Freshness stops being a maintainer's say-so and becomes a checkable object.

This is not a new invention, and we are careful not to dress it as one. The Graph protocol already uses exactly this primitive under exactly this name: an Indexer that has indexed a subgraph submits a *Proof of Indexing* — in the protocol's own words, "a cryptographic proof that an Indexer has correctly indexed a subgraph deployment up to a given block" [13] — is paid for it, and remains liable to a dispute that slashes its stake if the proof is wrong. A route maintainer's proof of indexing stands in precisely that position: produced by real indexing work, accepted optimistically, and disputable on independent re-indexing.

Freshness, crucially, is not a property a route has once; it is a property it has *over time*, and a single proof certifies only a single moment. The right analogy is Filecoin's separation of two proofs: a Proof-of-Replication that the work of storing specific data was actually done, and a Proof-of-Spacetime that lets a provider "prove they have stored some data throughout a specified amount of time" [14]. A route is the same: one proof of indexing shows the indexing work was done; *repeated* proofs, on a cadence the trust tier sets, are what show the route is being *kept* fresh rather than indexed once and left to rot. Maintenance is the spacetime property, not the point one.

We stop, deliberately, at *what* the proof attests and never publish *how* the index is produced. The proof commits to observable facts about the route — it resolves, its schema matches, its fields are present — which is all a verifier needs. The private capture surface that does the indexing stays outside the public artifact [4], for the abuse reasons stated there; the boundary is the difference between giving agents a checkable freshness proof and publishing the capture method. The indexing capability the proof of indexing would certify is the private capture surface, while the cryptographic proof itself ships as a runnable, tested reference (`reference/network/proof_of_indexing.py`) — a content-addressed, signed, hash-chained attestation a stranger re-derives against the real content — and the bonded economics that secure it ship as their own reference; production deployment is the remaining step.

5 Trust tiers

Not all traffic needs the same assurance, and pretending it does is its own kind of dishonesty — it taxes a price-check that can fail harmlessly at the same rate as a funds-moving call that cannot. So the graph is stratified into tiers. *Open* routes serve low-risk, idempotent, read-mostly traffic: a public listing, a schema lookup, a price quote, anything where a stale or wrong answer is cheap to detect and cheaper to retry. These carry no bond and make no guarantee beyond best effort; they are the fast, free majority of the graph and are honest about being best-effort. *Higher-trust* routes

sit above them for authenticated, high-value, or side-effecting paths — moving money, mutating an account, anything where a silent failure is expensive and a wrong route is dangerous. The escalation is the point: an agent pays for assurance only on the calls that actually warrant it, and the open tier is never asked to pretend it is something it is not.

The load-bearing rule across all tiers is what *ranks* a route, and the answer is route quality — observed freshness, success rate, faithful replay — not the size of anyone's stake. We are emphatic about this because it is the exact line where a trust network degrades into a pay-to-win one: bonding buys a maintainer *eligibility* to make a slashable claim on a high-trust route, and it buys nothing further [1]. It does not buy placement, it does not buy a rank, and it does not let a well-capitalised maintainer outbid a better one for the top of a shortlist. The moment stake purchases ranking, the graph is selling position rather than reporting quality, and a quality signal you can buy is not a quality signal. Capital gates entry to the bonded tier; merit orders within it.

A trust tier needs a way for a stranger to evaluate a maintainer it has no prior reason to trust, and we reuse the ERC-8004 "Trustless Agents" model rather than inventing one [11]. It defines three registries that map cleanly onto a route maintainer: a portable **Identity** that follows the maintainer across contexts, a **Reputation** record of signed feedback that accrues from real outcomes, and a **Validation** surface where an independent party re-executes or otherwise checks a claim before trusting it. An Unbrowse maintainer is precisely one of these trustless agents — an identity that can carry reputation and be verified by someone who does not yet trust it — which is what lets the higher tier be evaluated on portable, checkable history rather than on assertion. The ERC-8004 binding ships as a runnable, tested reference (`reference/network/erc8004.py`) that produces and verifies the Identity, Reputation, and Validation records, each wallet-signed [11]; binding to the deployed on-chain registry is integration work, and what ships in the product today is the open graph and its usage settlement, with the bonded tier behind it.

6 Bonded, slashable maintenance

The trust tiers of the previous section raise an obvious question: what makes a higher-trust claim actually trustworthy? Proof of indexing answers *what* a fresh route looks like as evidence; bonding answers *why* that evidence is honest. A route's freshness, even with a proof attached, is still an assertion that the proof was produced faithfully — and an assertion is only as good as what the asserter has to lose if it is wrong. So a maintainer who wishes to certify a route at a trust tier posts a proof of indexing and *bonds* FDRY against it: the bond is collateral posted into the open, escrowed against that specific proof. A proof that later fails to reproduce — an independent re-index does not reconstruct it — or that conflicts with another bonded proof about the same route, is *slashable*, and the maintainer forfeits stake. This is exactly the discipline The Graph applies to its own indexers, whose Proof of Indexing is accepted optimistically but slashed if a dispute resolves against it [13]: the maintainer is paid to be right and made to pay for being wrong, so certifying freshness stops being free talk and becomes a position with skin in it.

The shape of this is not novel and we do not claim it as such; it is exactly the *accountable-safety* discipline of proof-of-stake finality. Casper the Friendly Finality Gadget [9] secures a chain not by trusting validators but by defining slashing conditions: a validator who signs two conflicting checkpoints, or otherwise violates the protocol's safety rule, can be proven to have done so and has its deposit destroyed. A maintainer who signs "this route is fresh" over a route that is in fact stale — or who signs two contradictory freshness claims — is in precisely the position of a validator violating a slashing condition: the misbehaviour is attributable to a specific bonded identity, and the penalty falls on that identity's stake rather than on the network at large. The accountability

is local, provable, and costly, which is the only combination under which a stranger is rational to rely on a claim made by someone they have never met.

Finalising a trust-tier claim is not the act of one confident wallet. A single bonded maintainer asserting freshness is a single witness, and a single witness is not corroboration. The settlement of a tier claim is therefore a threshold event: FROST [10], a flexible round-optimised Schnorr threshold signature scheme, lets a quorum produce one valid signature only when t of n independent signers participate, with no single party able to forge the result alone. A trust-tier freshness claim is finalised by such a t-of-n quorum, so a tier does not advance on one operator's say-so and one dishonest or compromised maintainer cannot settle a claim by itself. The quorum is the structural form of the two-corroborator rule applied to finality.

We are deliberately precise about what the bond purchases. Bonding buys *credibility* — the right to make a claim a counterparty has reason to believe — and *eligibility* to participate at a trust tier. It does *not* buy ranking. The moment stake purchases placement in results, the search surface becomes pay-to-win and the asset securing trust degrades into the asset buying attention; the two roles are incompatible and we keep them apart by construction. Ranking is grounded in route quality (Section on trust tiers); the bond is a security deposit against a freshness assertion, nothing more. The bond and the slashing conditions ship as a runnable, tested reference (`reference/network/bonding.py`); the threshold finalisation it composes with — FROST t-of-n, modelled on Shamir (t, n) secret sharing where any t signers finalise and any $t-1$ cannot — ships as its own runnable, tested reference (`reference/network/frost.py`). The open and quality-ranked tiers do not depend on either, and we will not present a reference implementation as a deployed product; on-chain threshold-signature deployment is the remaining step.

7 Challenge and dispute

A bond gives a false freshness claim something to lose, but it does not by itself decide *whether* a claim is false — a slashing condition is only as useful as the procedure that proves it was met. We resolve this the way the rest of the stack resolves trust: not by an operator's adjudication, but by independent corroboration. A challenger who believes a bonded freshness claim is wrong opens a dispute by staking against it. The challenger's stake is itself collateral, so a dispute is not a free heckle: a frivolous challenge loses its stake to the maintainer, and a sound one is rewarded from the slashed bond. Both sides now have skin in the same question, which is the precondition for the question being answered honestly rather than loudly.

Resolution is by independent re-indexing. The disputed claim is a claim about an observable fact — does re-indexing this route reproduce the bonded proof? — so it is settled by independent parties who do not trust either side re-running the index and comparing what they observe against the committed proof of indexing. This is the role The Graph assigns to its *Fishermen*, who may dispute a published Proof of Indexing and, where the dispute resolves against the indexer, trigger a slash of its stake [13]. It is the two-corroborator principle made operational: a proof stands only on independent corroboration, never on one party's assertion (neither the maintainer's original proof nor the challenger's accusation is self-validating). ERC-8004's Validation registry [11] is the surface for exactly this — it defines independent re-execution, and where applicable ZK or TEE checks, as the mechanism by which one agent's work is verified by others who have no reason to favour it — so a freshness dispute is adjudicated by the re-indexing evidence, not by a trusted referee.

For the dispute history to be auditable, the record of claims and challenges must itself be tamper-evident rather than a log a single operator could quietly rewrite. We therefore keep the sequence of freshness claims, challenges, and resolutions in an append-only, hash-chained log in

the Certificate Transparency tradition [12]: each entry commits to its predecessor, so the history and its ordering cannot be altered after the fact without breaking every subsequent root, and any independent party — not just the platform — can verify that the log is append-only and consistent. The point of the CT analogy is precisely that correctness does not rest on trusting the operator: a misissued or back-dated claim is detectable by anyone auditing the log, exactly as a misissued certificate is detectable in CT. The challenge mechanism supplies the economic incentive to look, and the transparent log makes what they find undeniable. This challenge-and-dispute layer ships in the same runnable reference as bonding (`reference/network/bonding.py`): a test adjudicates a challenge by re-derivation against the content-addressed truth, slashing a lying maintainer and forfeiting a spurious challenger's stake while conserving total stake on every path. It is the accountability complement to the bond — running and tested, not yet deployed.

8 Attribution and the Sybil limit

A maintenance economy that pays for contribution has to answer a sharper question than "who touched this route?" — it has to answer "how much better is the route *because* you touched it?" We answer it with delta-based attribution: a contributor is rewarded in proportion to the marginal improvement their work makes to a route, not for mere presence on it. Tightening a schema so that fewer calls fail, adding an endpoint that was missing, repairing an auth flow that had drifted — each is scored by the difference it makes to the route's measured quality, read off the freshness, coverage, and faithful-replay deltas between successive proofs of indexing, so the reward tracks the increment, and a no-op edit on top of someone else's work earns nothing. This keeps the incentive pointed at real maintenance rather than at land-grabbing a popular route and collecting rent on another party's labour.

This reward, and the savings it sits beside, are made legible to the contributor without exposing the chain underneath: the CLI surfaces a wallet-bound contribution ledger — money earned (the owner/indexer share that settled to you), money saved (the cost the cached route avoided), and the tokens, speed, and wall-clock time saved — read off the local impact log and your on-chain dashboard. The on-chain identity is the root the ledger binds to; the figures are a Web2-shaped view wrapped over it, so a contributor who never wants to think about wallets still sees what their maintenance produced. The wallet is read without minting, so merely checking the numbers never creates key state.

Delta-based attribution prices contribution honestly, but it does not, by itself, solve who is making the contribution — and here we have to be plain rather than reassuring. The marginal-contribution score is computed per identity, and in an open system identities are cheap. Douceur's result is the one that governs this [8]: without a central authority to certify that distinct identities correspond to distinct entities, a single entity can always forge many identities, so Sybil attacks are, in the general case, essentially always possible. An adversary who can mint identities for free can in principle split one contribution into many, or stand up a crowd of identities to manufacture the appearance of independent improvement. Bonding is our mitigation: requiring stake to participate in the attributed tier raises the *cost* of each forged identity, and graduated stake makes a large Sybil cohort expensive rather than free. But raising a cost is not closing a gap. Bonding makes the attack more expensive; it does not formally guarantee that distinct attributed identities are distinct entities, which is exactly the guarantee Douceur shows cannot be had without a trusted certifier.

We are precise about what is and is not closed. The Sybil *mitigation* ships as a runnable, tested reference (`reference/network/sybil.py`): attribution is stake-weighted and *split-invariant*

— a test proves that splitting one identity's stake across any number of Sybils yields exactly zero additional influence, because influence is $\text{stake}_i / \sum \text{stake}$ and the split conserves the numerator, so the only lever on influence is acquiring more stake at real cost. That is the bound Douceur's result [8] permits, and it is now concrete code rather than a promise. What it is *not* is a closed solution to the impossibility Douceur proves: full formal Sybil resistance — a guarantee that distinct attributed identities are distinct entities without a trusted certifier — remains open, and the honest statement is that stake-weighting narrows the economically rational attack surface, it does not eliminate the attack.

9 The token: why FDRY, and why fair-launched

This section makes one argument, and it is an argument about security, not about markets. The conclusion is uncomfortable for a token paper to state plainly: *a coordination asset launched for money cannot become trustworthy, so the only way to make it trustworthy was to launch it for something else and keep it clear of the money-motive at the root.* That is not a marketing posture dressed as virtue — it is a property the system needs in order to function, and the rest of this section derives it.

One asset, one job. The first discipline is to refuse to conflate two surfaces that the rest of crypto routinely fuses. Usage — a paid `execute`, a settled call — does not need a token at all; it settles in stable USDC over x402 [3], and we are emphatic that it stays that way. FDRY exists for exactly one job the stablecoin cannot do: the security/accountability layer, where a maintainer's bond, a challenger's stake, and a slashing penalty must be denominated in an asset whose value is tied to the network's own integrity [2]. Payment surface and security stake are kept strictly separate, because the moment a holder is forced to spend the security asset to *use* the network, the asset becomes a tollgate and the holder a beggar-at-the-gate. This is a separation-of-concerns constraint: a single asset cannot be both the payment rail and the security stake — USDC settles usage; FDRY only bonds. An asset asked to do both jobs does neither honestly.

Why the distribution is a security parameter. The non-obvious constraint is on FDRY's *distribution*, and it falls straight out of the same corroboration logic that governs route trust. Accountability is credible only when many *independent* parties hold the bond — the same way a claim stands on two independent corroborators rather than one. A security asset concentrated in insider hands is a single party wearing many hats: it contradicts the very decentralisation it is meant to underwrite. So FDRY launched fair by necessity, not branding: no pre-sale, no team allocation, no insider round; supply placed on a public Solana pool at launch, acquirable only on the open market at the same price for the founder as for everyone else. The fair launch is therefore not a gesture of goodwill — it is the parameter that makes the bond mean anything. The fair launch and platform-fee collection are live. The vault cycle that returns the platform's USDC fee to the distributed holders — *staking by holding* rather than by lock-up — ships as a runnable, tested reference (`reference/network/vault_cycle.py`): a test confirms the fee pool is split pro rata to each holder's balance × duration held, that the split is conservative (no unit minted or lost), that holding longer or larger earns more, and that holding nothing earns nothing — no lock-up required. The on-chain deployment of that cycle is the remaining step.

Why money could not be the point. There is a deeper reason the distribution had to be fair, and it is structural rather than financial: it concerns the order in which trust and value can

be created. A network that puts money first inverts its own foundation. If the token is launched to enrich its issuer, then the issuer's incentive at the root is extraction, and every participant downstream correctly discounts the system's trust claims accordingly — the rot is at the origin, so no amount of mechanism layered above it recovers credibility. An accountability layer whose entire job is to let strangers trust each other's claims *cannot* sit on a money-first root, because a root optimised for the issuer's gain is precisely the thing strangers are right to distrust. FDRY clears this bar only because money was not its point: it was distributed with no issuer advantage and then repurposed as the security stake. The asset earns trust the same way a route does in this system — by what it does for the network, corroborated by independent holders, not by what it promised its founder. This is a sequencing constraint stated as economics: an asset held purely for its own appreciation generates no trust, and accrues real value only after it has first done useful work for the network. Trust precedes value here; the value follows from the trust, never the other way round.

Disclosed in full, including the address. None of this is a license to be coy. A trust paper that is evasive about money is not trustworthy, so we state the economics plainly — who pays, who earns, who opts in — and we print the contract address on purpose: `2ZiSPGncrkwWa6GBZB4EDtsfq7HEWwkwsPFzEXieXjNL`. What the design refuses is the *investment narrative*, the framing of the token as a bet on its own price. The "what" is fully disclosed; only the money-as-motive framing is denied, because that framing is exactly what would corrupt the trust the security layer depends on. We make no promise of profit: the mechanism is the claim, not a guarantee.

10 The maintained graph, measured

The mechanisms above are economics and cryptography; the question a maintenance paper has to answer with numbers is narrower: *does the maintained graph actually resolve and stay covered, and where does it fail?* We answer it with a re-runnable capability gate over a fixed, tiered intent set, recorded to a two-witness history ledger (`bench/capability/`) so every figure is a gated measurement rather than an assertion.

Coverage, by tier. Action-retrieval coverage@1 is **0.78** over nine intents split across three difficulty tiers: a public-content tier resolving at **1.0**, an anti-bot tier at **0.67**, and an automation tier at **0.67**. The misses are not hidden — they are pages whose first-party endpoint was not surfaced (single-page apps, blocked captures), which is precisely the freshness/coverage deficit the maintenance economy of this paper exists to close, and the target the proof-of-indexing cadence drives down over time.

Resolution is two-witnessed, and writes leak nothing. A resolved route counts only when it is content-bound across two independent sessions, the same two-corroborator rule the trust tiers apply to a freshness claim. Separately, the pointer-only redaction invariant is scanned, not assumed: across **446** persisted session files the gate finds **0** plaintext secrets.

The honest external comparison. Against a retrieval benchmark cloned directly from a specialised search vendor (the `webcode-benchmark` RAG harness from `exa-labs/benchmarks`), our route-graph retrieval scores groundedness **0.167** against the vendor's published target of ≈0.79. We report this gap rather than bury it: long-form grounded synthesis is dominated by the reader model above retrieval, not by the route graph, and a maintenance paper that hides its weak axis is

not one a stranger should trust. Where the substrate *does* win head-to-head is adversarial retrieval past JavaScript-challenge anti-bot, reported in the companion security paper [2].

11 What is implemented, what is referenced

In the spirit of the companion papers [1, 2] — and of not selling a roadmap as a changelog — we separate what runs in the product, what is available as runnable, tested reference code, and where deployment still stops. Running in the product today:

- The three-way split across the indexer, the domain owner, and the platform: computed on every paid execution and settled — atomically in one x402 [3] transaction on devnet, and custodially on mainnet today, where the platform collects and the contributor and owner cuts are disbursed from the attribution ledger (the trustless on-chain split program is devnet-only, pending its mainnet deployment).
- Fair-launched FDRY: supply placed on a public pool at launch with no pre-sale, team allocation, or insider round.
- Platform-fee collection on settled executions.
- The signed, append-only JSONL route ledger — the local, hash-chained record of route claims.

Shipping as runnable, tested reference code:

- Proof of indexing: a content-addressed, signed, hash-chained attestation that a route was re-indexed against its live source, re-derivable by independent re-indexing [13, 14] (`reference/network/proof_of_indexing.py`). The private capture surface that produces the index stays outside the public artifact [4]; the cryptographic proof is running, tested code, and production deployment of the bonded economics is the remaining step.
- Bonding, challenge, and slashing: the collateralised-accountability layer in which a maintainer's stake is forfeit on a proof that fails to reproduce, with conservative stake arithmetic (`reference/network/bondi`
- Native mechanism-design witness: the proof-of-indexing economy now also ships as a TypeScript runtime model (`src/values/proof-indexing-economy.ts`) with deterministic tests (`tests/proof-indexing-` proving honest dominance, Sybil split-invariance, slashable stale proofs, costly false challenges, and conservative balances. The JESPA ledger records it as a reproduced mechanism win (`bench/jespa/proof-indexing-economy-gate.sh`).
- Merkle-root checkpoints of the ledger, batching many signed entries into a single transparency-style root [12] with per-entry inclusion proofs (`reference/ledger/checkpoint.py`); publishing the root on-chain is the deployment step.
- The Sybil *mitigation*: stake-weighted, split-invariant attribution (`reference/network/sybil.py`). Full formal Sybil resistance for attributed contribution remains an open problem with a cited foundation [8]; the reference is the bound, not the closed solution.
- ERC-8004 trustless-agent records — Identity, Reputation, Validation, each wallet-signed (`reference/network/` and the vault fee-return cycle, "staking by holding," split pro rata to balance × duration (`reference/network/vault_cycle.py`). Binding to the deployed registries and on-chain settlement of the cycle are integration work.

This reference code is running, tested code with cited foundations — not a research agenda, and not a feature list with a release date. What still separates a reference from the deployed product is named on each line. The gap between running code and a press release is the entire difference between a whitepaper and a pitch, and we intend to stay on the correct side of it.

12 Conclusion

The thesis of this paper compresses to one sentence: the graph is the asset. Every site an agent automates today is re-discovered from scratch, per agent, per call — the browser tax paid over and over. Unbrowse turns that one-off robotic-process work into a shared, maintained route graph: discover once, reuse forever, route the cost through one layer instead of re-paying it everywhere [1]. The crucial asymmetry is that *any single route is copyable, but the maintained graph is not.* A route can be lifted; what cannot be lifted is the standing work of keeping thousands of routes fresh as the web shifts under them, attributed to who captured them, and safe enough that another agent will stake its task on them.

That is why maintenance, not discovery, is the economic engine. The value compounds in the upkeep: realised payment over x402 [3] funds the substrate that keeps the graph trustworthy, and the network pays whoever actually created the value of a reused route — the indexer who captured it, the domain owner whose service is called, and the platform that secures the settlement. Each share is the dominant strategy for the role it pays, so the graph's coverage grows, cooperation out-competes blocking, and the platform earns only when a settlement it secured actually happens. FDRY sits underneath all of it as the bonded, fairly-distributed security stake — the asset that makes "trust this route" a claim with collateral behind it, not a hope.

We have told this as the *what* throughout, and stopped before the private capture *how*. That boundary is not secrecy for its own sake; it is the difference between giving agents checkable route contracts and publishing a method that could be turned into an unattributed scraping fleet. Internal APIs are a useful first layer. The maintained, attributed, bonded graph above them — kept fresh and paid for in proportion to the value it creates — is the next layer we are building.

References

[1] L. Tham, N. Mac Gregor Garcia, J. Hahn. *Internal APIs Are All You Need: Shadow APIs, Shared Discovery, and the Case Against Browser-First Agent Architectures*. arXiv:2604.00694, 2026.

[2] L. Tham, C. Chik. *Crypto Was All You Needed: A Signed, Layer-Descending Stack for Agent Computation*. Unbrowse AI, 2026.

[3] Coinbase. *x402: An Open Protocol for Internet-Native Payments over HTTP 402*. x402.org whitepaper, 2025. https://github.com/coinbase/x402.

[4] Unbrowse AI. *Open Source Notice — the closed/open boundary*. docs/OPEN-SOURCE-NOTICE.md, 2026.

[5] E. Ostrom. *Governing the Commons: The Evolution of Institutions for Collective Action*. Cambridge University Press, 1990. ISBN 0-521-40599-8.

[6] P. A. Samuelson. *The Pure Theory of Public Expenditure*. Review of Economics and Statistics 36(4):387–389, 1954. DOI: 10.2307/1925895.

[7] A. C. Pigou. *The Economics of Welfare*. Macmillan, London, 1920.

[8] J. R. Douceur. *The Sybil Attack*. In Peer-to-Peer Systems (IPTPS 2002), LNCS 2429, pp. 251–260. Springer, 2002. DOI: 10.1007/3-540-45748-8_24.

[9] V. Buterin, V. Griffith. *Casper the Friendly Finality Gadget*. arXiv:1710.09437, 2017.

[10] C. Komlo, I. Goldberg. *FROST: Flexible Round-Optimized Schnorr Threshold Signatures*. SAC 2020, LNCS 12804. IACR ePrint 2020/852; RFC 9591.

[11] *ERC-8004: Trustless Agents*. Ethereum Improvement Proposals (Draft). https://eips.ethereum.org/EIPS/eip-8004.

[12] B. Laurie, A. Langley, E. Kasper. *Certificate Transparency*. RFC 6962, IETF, 2013.

[13] The Graph. *Proof of Indexing (POI)*; *Indexing Overview*. The Graph Protocol Documentation, 2024. https://thegraph.com/docs/en/resources/glossary/; https://thegraph.com/docs/en/indexing/overview/.

[14] Protocol Labs. *Filecoin: A Decentralized Storage Network*. White paper, July 2017. https://filecoin.io/filecoin.pdf.