When Not to Settle Probabilistically:
Sparse Settlement and Regime Selection for x402

Working draft v0.9 · PRE-RELEASE

22 September 2026

Pre-release — work in progress. This is a working draft, not a peer-reviewed paper. The measurements come from a single thirty-day snapshot; an out-of-sample re-run on a second snapshot is scheduled for late October 2026, and the verifier contract is unaudited and undeployed. Expect numbers and conclusions to be revised. Scripts and data: `github.com/tollstile/Tollstile`, `research/sparse-settlement`.

The new thing is not the ticket. It is the measurement that the region where a probabilistic ticket pays is, in today's x402 market, almost empty even after every approximation is removed; that of the three quantities the public index cannot see, two shrink that region and one would enlarge it; and that the selection rule which follows is worth more than any scheme it selects among.

Abstract

HTTP 402 commerce is priced in cents and fractions of cents and, on its default path, settled one on-chain transfer per call. We measure the largest public index of x402 resources — 14,964 endpoints, 487,261 reported calls in thirty days — and split it by payer recurrence as well as price. The split exposes a mismatch: **between 47% and 91% of calls come from repeat payer relationships** (the range reflects an unpublished definition of "call"), yet resources advertising the schemes built for those relationships — `batch-settlement`, Circle Gateway — account for at most 4.8% of calls, and none advertises them exclusively; and at least 61% of all payer–resource pairs are provably single-call (at least 43% single-call *and* sub-cent), a cell where every efficient path in x402 begins with the buyer depositing somewhere. We construct the missing regime for that cell — *sparse settlement*, a Permit2 authorization for a ticket value T that settles with probability p/T, decided by a facilitator commit–reveal inside the request and enforced on-chain so no party can settle a losing ticket — and reduce it to four invariants independent of x402. We then measure where it applies, under four bounds: merchant variance, buyer exposure, the client's per-payment cap (**$1 by default in the x402 reference client, filtered before any policy; $0.05 in agent-wallet documentation**), and a bound prior probabilistic-payment work could ignore: the buyer's one-time Permit2 approval, which the default `exact` path does not need. We use the exact binomial bound $T \le p(1 + n/k)$, not the small-q approximation earlier drafts of this work carried. The region that survives holds **25 of 11,630 low-recurrence resources**, unchanged by that correction and stable to the "call" definition (22–25); and in the target cell the median ticket reduces settlements by **four percent**, so that **3,648 of 4,716 sub-cent resources cannot beat per-call settlement at any share of returning buyers** — not because variance forces the ticket to the price, which it does not, but because it forces the reduction below the settlement's own gas premium. The useful result is therefore the selector: an ex-post oracle choosing per

resource costs $86/month in settlement gas across the index against $975 for always-`exact` — a bound that trivially dominates any fixed regime — while a **deployable two-threshold rule, evaluated retrospectively, costs $107** and stays within 1.0–1.7× of the oracle across 288 combinations of gas, channel amortisation, tolerance, client cap and contract-gas premium. That rule beats the best single regime in **195 of 288** scenarios; three quarters of the 93 it loses are one corner — channel deposits that live six months or more, where always-channel wins — which tells the runtime the one cost parameter it must measure before choosing. *Which* single regime is best flips with the assumptions throughout. A verifier contract run against the real Permit2 bytecode settles a winning ticket in 72,829 gas (a repeat buyer) to 89,917 (a one-shot buyer's first ticket) — 0.91–1.11× a USDC EIP-3009 transfer on Base once adjusted for the token — and the one-time Permit2 approval is 0.54 of one; with those ratios every resource inside the region clears the onboarding bound even if all its buyers are new, while three quarters of the sub-cent low-recurrence cell never does at any φ. Probabilistic settlement is one regime among several; a merchant runtime, not a protocol, is what has to know where its edges are; and we list the measurements that would move them.

Terminology. *Channel-free* means no pre-funded balance, escrow or payment channel exists between the buyer and anyone. The buyer's ordinary wallet balance is the only backing. The *server* is not stateless: it keeps replay state and a ledger, as for `exact` today. *Reported calls* are the index's `l30DaysTotalCalls`, whose definition the index does not publish.

1 What the index shows, and what it cannot

We took Coinbase's public discovery index (the "Bazaar", fetched in full on 20 September 2026). Restricting to resources priced in USDC below $1,000 per call with at least one reported call gives **14,719 active priced resources**.

Resources listed	14,964 (1,989 hosts)
Active priced resources	14,719
Reported calls, 30 days	487,261
Median advertised price	$0.01 (p90 $0.08)
Resources priced below $0.01	41.6%
Resources with exactly one paying address	69.3%
Offers using `upto` / `batch-settlement` / Gateway batched `exact`	2.3% / 0.15% / 0.6%
Gross at advertised prices	≈ $14,000

1.1 The recurrence split

The split that has not been reported is by **recurrence** — reported calls per paying address — crossed with price. The index gives each resource's total calls and unique payers, a resource-level *mean* recurrence, not per-payer counts. We call a resource with mean ≤ 2 calls per payer **low-recurrence** and one above it **repeat**; the cells below are cells of *resources*, and the pairs column counts payers at those resources, not payers known to be one-shot.

	Resources	Payer–resource pairs	Reported calls	Gross
low-recurrence, sub-cent	4,716 (32.0%)	23,991 (50.1%)	27,323 (5.6%)	$97 (0.7%)
low-recurrence, ≥ $0.01	6,914 (47.0%)	11,420 (23.8%)	14,114 (2.9%)	$3,614 (25.8%)
repeat, sub-cent	1,401 (9.5%)	5,073 (10.6%)	115,609 (23.7%)	$365 (2.6%)
repeat, ≥ $0.01	1,688 (11.5%)	7,417 (15.5%)	330,215 (67.8%)	$9,955 (70.9%)

A mean of two calls per payer is consistent with fifty payers at two calls each or forty-nine at one and one at fifty-one; the aggregate cannot tell. But it can bound. With n calls over u payers each making at least one, at most $n - u$ payers made two or more, so **at least** $2u - n$ **made exactly one**. Summed over the index this is a provable floor: **at least 29,388 of 47,901 payer–resource pairs (61.4%) are single-call, and at least 20,659 (43.1%) are single-call and sub-cent**. For 8,514 resources $n = u$ exactly — every payer called once — with no inference needed. We use "one-shot" below only for these bounded quantities.

The buyers and the calls are in different cells. At least 43% of buyer–resource pairs are single-call and sub-cent and carry 0.7% of the money; 91% of reported calls come from repeat resources. **The cheap paths for the big cell exist and are, at most, lightly used.** The index shows what a resource *advertises*, not which scheme each call settled on. Resources advertising `batch-settlement` or Circle Gateway account for **4.8% of calls — an upper bound** on the share that could have used them — and every one of the 164 also advertises plain `exact`, so actual usage is unknown and no higher than that. This is an advertisement gap; whether it is also an adoption gap is one of the measurements §11 asks for. **The remaining gap is structural**: every efficient path begins with the buyer depositing somewhere, and a wallet that has deposited nowhere is served only by per-call `exact`.

1.2 Three things the index cannot tell us

Each of the statements above rests on a quantity the index does not expose. We name them now because §5–6 test the results against all three, and because they are the measurement agenda this paper leaves behind.

(i) What a "call" is. If only a fraction f of reported calls were settled payments — the rest unpaid `402` challenges — the recurrence split shifts. We apply f as a **stress transform, not a reconstruction**: paid calls cannot be fewer than unique paying addresses, so per resource $n_{\text{paid}} = \max(u, fn)$.

f	repeat share of calls	provable single-call floor
1.0	91.5%	61.4%
0.5	83.7%	78.0%
0.25	70.7%	85.8%
0.1	47.3%	90.1%

The headline "91%" is an $f = 1$ number. The direction — most calls from repeat relationships, most buyers one-shot — holds at every f; the magnitude does not.

(ii) Who the buyer is across resources, and which scheme each call used. Recurrence is measured per resource from unique payer addresses, and scheme adoption only from what resources advertise. A buyer who calls fifty resources once each looks one-shot at every one of them, yet is a repeat buyer of the market and a natural user of a gateway balance. The one-shot cell therefore *overcounts* buyers with no efficient path.

(iii) Whether the buyer has ever used Permit2. The default `exact` path (EIP-3009) needs no prior transaction from the buyer. Every Permit2 path — `upto`, and the mechanism below — needs a one-time approval. The index cannot say what share of its one-shot payers have one.

2 Background: the settlement structures x402 already has

Who chooses. A `402` carries `accepts[]`; **the client selects** — the reference client filters by scheme and network, then by spend controls, then by policy. This is load-bearing (§4).

`exact` on EVM is three transfer methods. EIP-3009 `transferWithAuthorization` (the default; "simplest, truly gasless"), Permit2 via a proxy, ERC-7710 delegation. Under EIP-3009 the signature authorizes the token contract directly and anyone holding it can submit it; nothing can be interposed. Under Permit2 a proxy is sole spender and can impose conditions. **Conditional settlement is possible only in the second structure.**

`upto` is the existing conditional-Permit2 scheme (ceiling permit, `x402UptoPermit2Proxy`, witness binding `to`, `facilitator`, `validAfter`). §3 reuses it.

Permit2's prerequisite. A one-time gas-paying approval of the canonical Permit2 contract, payable by the user, sponsored by a facilitator (`erc20ApprovalGasSponsoring`), or bundled via EIP-2612 (`eip2612GasSponsoring`); otherwise `412 PERMIT2_ALLOWANCE_REQUIRED`. One approval serves every seller — but it is a transaction the EIP-3009 path never asks for.

`batch-settlement`. Its current EVM binding is a pre-funded per-receiver channel with cumulative vouchers. The network-agnostic specification is wider — capital-backed models including delegated authorization against a wallet balance, and credit-backed identities. "Deposit-first" describes the EVM deployment, not the scheme.

Circle Gateway Nanopayments. Mainnet since April 2026; one deposit into a Gateway Wallet, per-call EIP-3009-shaped authorizations against the `GatewayWalletBatched` domain, off-chain immediate credit, periodic netted on-chain settlement at Circle's gas cost; one balance pays any seller; minimum $0.000001; EOA only. Present on 116 resources in the index.

	buyer has deposited somewhere	ordinary wallet only
repeat buyer	`batch-settlement`, Circle Gateway	`exact` per call
one-shot buyer	Circle Gateway, *if the deposit came first*	`exact` per call — or nothing below a cent

3 Sparse settlement in four invariants

A *sparse ticket* is a Permit2 `PermitTransferFrom` for amount T whose spender is a verifier contract, with witness

```
SparseWitness(
  address to, address facilitator, uint256 price, uint256 threshold,
  bytes32 commitment, bytes32 challengeId, uint256 validAfter)
```

The verifier enforces:

1. **Sole spender.** Only the verifier moves the funds; only `witness.facilitator` may invoke it.

2. **Conditional transfer.** `permitted.amount` moves to `witness.to` **iff** $H(s) =$ `commitment` and $H(d\|s) <$ `threshold`, where d is the EIP-712 digest the buyer signed and s the facilitator's secret. Otherwise nothing moves — a losing permit is unspendable by anyone.
3. **Advertised odds are settled odds.** `threshold` is recomputed from `price` and `permitted.amount`; mismatch reverts.
4. **One purchase, one roll.** The signature covers `challengeId`, and the verifier can enforce no more than that. The rule has a client half (never re-sign for the same `challengeId`) and a merchant half (bind one logical purchase to one `challengeId`, durably) — both in §7, and the second is what keeps the odds fair rather than merely bounded.

Expected transfer per ticket is p; buyer exposure is T with probability $q = p/T$. The commitment $c = H(s)$ travels in the `402` challenge, so d covers it; s is revealed at `/verify`, and a winner settles inside the request against the balance just checked — the delay of a VRF round or a future block would reopen the gap between "has T" and "still has T". We hash the digest rather than signature bytes because the digest is canonical and signature bytes are not.

Whether this is a new scheme or a probabilistic EVM binding of `batch-settlement` (whose specification admits delegated wallet authorization) is a taxonomy question the paper does not depend on.

4 Four bounds on the ticket

Merchant variance. Wins are Binomial(n, q) with $q = p/T$, so realized revenue has coefficient of variation

$$\mathrm{CV} \;=\; \frac{\sqrt{nq(1-q)}}{nq} \;=\; \sqrt{\frac{1-q}{nq}} \;=\; \boxed{\sqrt{\frac{T/p-1}{n}}}$$

and $\mathrm{CV} \le c$ holds exactly when

$$T \;\le\; p\left(1+\frac{n}{k}\right), \qquad k = 1/c^2$$

($k = 25$ for CV $\le 20\%$, 100 for 10%). Earlier drafts of this work used the small-q approximation $\mathrm{CV} \approx 1/\sqrt{nq}$, giving $T \le pn/k$ and dropping the leading 1. The correction is one term and it matters at both ends: a $\ge 10\times$ reduction needs $n \ge k(R-1) = 225$ calls, not 250; and at $n = 1$ the bound is $T \le 1.04p$, so variance never forces T exactly to p — it forces the reduction to a few percent, which is a different statement and the one §5 now makes. Everything below uses the exact form.

Buyer exposure. A buyer making m calls pays Binomial$(m, q) \cdot T$. For $m = 1$, at $p =$ \$0.01 and $T =$ \$1: expected \$0.01, realized "\$1 with probability 1%".

Client cap — measured, not modelled. The x402 reference client applies spend controls as step 3 of requirement selection, before policies:

```
export const DEFAULT_MAX_AMOUNT_PER_PAYMENT: Money = "$1";
```

Coinbase's agent-wallet documentation: "Max per call: Max for a single payment (e.g., \$0.05)", "Agents respect these limits but can't change them." Cloudflare virtual wallets cap agent spend at an owner-set limit. So a ticket above \$1 is dropped by the default client before any policy runs; a merchant can only *propose* several `accepts[]` entries at different T with a deterministic fallback, and the client selects.

Onboarding — the bound the classical literature could ignore. Rivest, Peppercoin, Orchid and Livepeer all assume the payer has an account, escrow or deposit; onboarding is outside the model. In x402 it is inside it, and asymmetric: the default `exact` path costs the buyer **no** transaction; every Permit2 path costs a first-time buyer one approval. Measured (§6): the approval is 46,403 gas and a sparse settlement 72,829–89,917, against a median 86,242 for a USDC EIP-3009 transfer on Base — ratios $a = 0.54$ and $s = 0.91$–1.11 of one `exact` settlement. Writing φ for the share of a resource's payers who are first-time Permit2 users, sparse beats `exact` on total gas iff

$$\varphi \;<\; \frac{n}{u} \cdot \frac{1 - s\,p/T_{\text{eff}}}{a}.$$

For a low-recurrence resource $n/u \approx 1$, so the condition is roughly $\varphi < (1 - s\,p/T_{\text{eff}})/a$: sparse wins only to the extent that it actually reduces settlements, and a cheap approval ($a = 0.54$) buys it headroom. Where variance holds T_{eff} near p, the right-hand side is negative — a near-deterministic Permit2 settlement at $s = 1.11$ plus an approval never beats EIP-3009 — and it cannot win at any φ.

The combined bound.

$$T = \min\!\big(p\,(1 + \tfrac{n}{k})\,,\; T_{\text{client}},\; T_{\text{policy}}\big)\,, \qquad R = T/p, \qquad \text{subject to } \varphi < \frac{n}{u} \cdot \frac{1 - s\,p/T}{a},$$

where s and a are the gas of one sparse settlement and of one Permit2 approval, each relative to one `exact` settlement — measured in §6 as $s = 0.91$–1.11 and $a = 0.54$, not assumed to be 1. A reduction of at least R needs all three of $n \geq k(R-1)$, $T_{\text{client}} \geq Rp$ and $T_{\text{policy}} \geq Rp$. That is a region in the (n, p) plane.

5 The region

Proposition 1. *Take a resource priced p with n calls and u distinct payers in the period, a merchant tolerance* CV $\leq c$ *($k = 1/c^2$), a client cap T_{client} and a merchant cap T_{policy}, and let $T = \min(p(1 + n/k), T_{\text{client}}, T_{\text{policy}})$. Then a sparse ticket reduces settlements by a factor of exactly T/p; it reaches a factor R iff*

$$n \geq k(R-1) \quad \textit{and} \quad T_{\text{client}} \geq Rp \quad \textit{and} \quad T_{\text{policy}} \geq Rp;$$

and it reduces total gas relative to per-call `exact` *settlement iff the resource's first-time-Permit2 share φ satisfies*

$$\varphi < \frac{n}{u} \cdot \frac{1 - s\,p/T}{a},$$

with s, a the sparse-settlement and Permit2-approval gas, each relative to one `exact` *settlement. In particular the gas condition is unsatisfiable at any φ when $T/p \leq s$: a ticket that does not reduce settlements by more than the settlement's own gas premium can never pay for itself.*

T_{client}	region ($k = 25$)	resources inside	their calls	share of index calls
\$1 (x402 reference default)	$n \geq 225$, $p \leq$ \$0.10	**25** of 11,630	14,839	3.0%
\$0.05 (agent-wallet docs)	$n \geq 225$, $p \leq$ \$0.005	**22**	12,930	2.7%

Robustness. The exact bound moves the volume threshold from 250 to 225 calls and admits **the same 25 resources** — the index has none between the two, so the correction changes the arithmetic and not the answer. Relaxing the tolerance to $k = 10$ (CV 32%) admits 39;

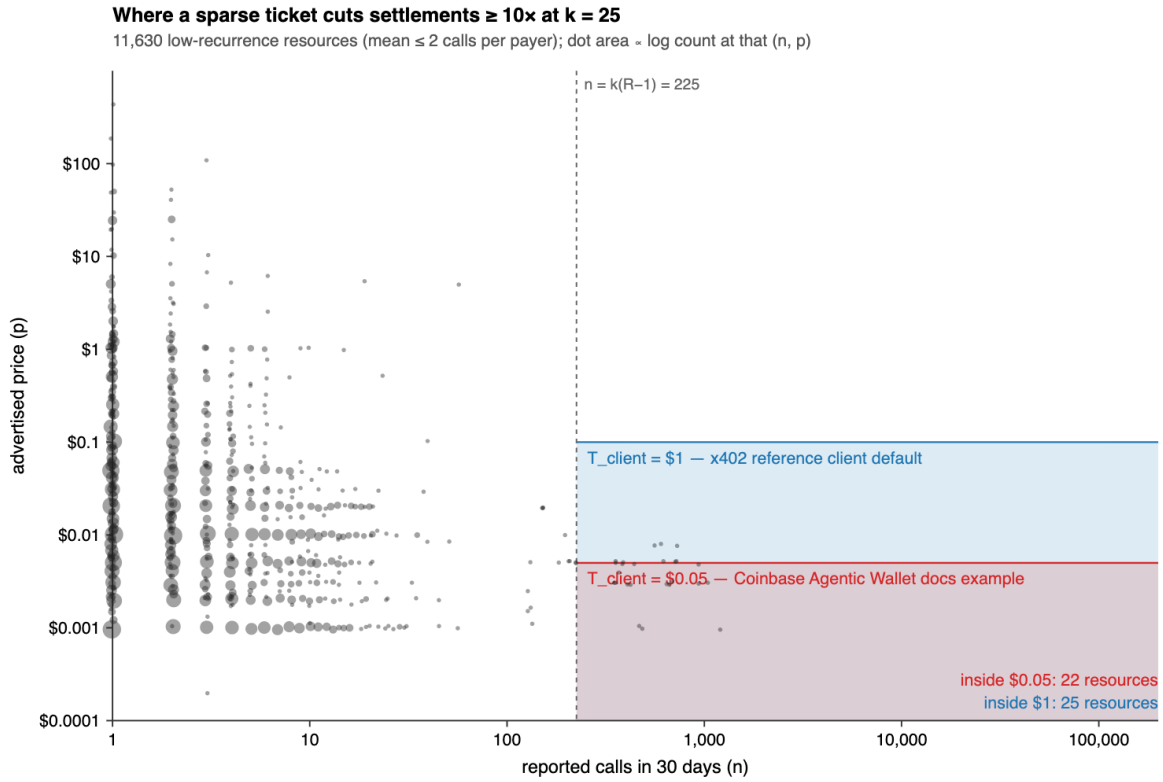

Figure 1: Every low-recurrence resource in the index against the two boundaries the measured client caps imply. The vertical line is the merchant-variance bound $n \geq k(R-1)$; the horizontal lines are the client caps.

tightening to $k = 100$ (CV 10%) admits 4; demanding a 50× reduction admits none. Against the call definition (§1.2 i), the region is stable — $25 \to 23 \to 22 \to 22$ at $f = 1, 0.5, 0.25, 0.1$ — because its members are resources with hundreds of distinct payers, whose counts are not inflated by repeated unpaid probes.

Onboarding inside the region. For the 25 resources, at the measured ratios ($s = 1.11$ for a one-shot buyer's first ticket, $a = 0.54$) the break-even first-time share φ^* has median 1.79: **all 25 beat per-call settlement even if every buyer is new.** Under the earlier one-transaction-unit assumption for the approval, six did.

What the ticket achieves in the cell the mechanism is for. The exact bound lets us say this precisely rather than by approximation. Across all 4,716 low-recurrence sub-cent resources, the reduction T/p has **median 1.04** — four percent — and the distribution is:

reduction reached	resources
$\geq 1.1\times$	1,068 of 4,716
$\geq 1.5\times$	102
$\geq 2\times$	49
$\geq 10\times$	25

So the earlier draft's "variance forces $T = p$" was false as stated and true in effect: the median resource can carry a ticket, and that ticket saves it four percent of its settlements. Against the measured gas premium $s = 1.11$ this is worse than nothing — by the Proposition the gas condition is unsatisfiable whenever $T/p \leq s$, and **3,648 of 4,716 resources are below that line at any φ.** Sixty-three beat `exact` even with every buyer new. In the cell it was built

for, sparse settlement is `exact` with extra steps for three quarters of resources and a rounding error for most of the rest.

This is the paper's central empirical result. The mechanism is correct; today it is almost inapplicable; and of the three things the index cannot see (§1.2), two would make it more so and one — a high existing Permit2-approval rate — would make it less (§11).

6 Regime selection

Since no single regime serves the index, the merchant runtime chooses per resource. It has the inputs — its own ledger gives n, u and recurrence — and `accepts[]` lets it propose more than one.

Rule. Recurring payers → a pre-funded scheme. Many one-shot payers with $n \geq k(R-1)$ and φ below break-even → offer sparse tickets alongside a deterministic entry, client selects. Otherwise → deterministic, sponsored, or credit-backed.

Cost of misselection. Gas at \$0.002 per transaction; `exact` = one per call; channel = one deposit and one withdrawal per distinct payer per amortisation period plus one sweep per resource-month; sparse = np/T transfers at the measured ratios.

	gross	always-`exact`	always-channel	always-sparse	per-resource best
low-recurrence, sub-cent	\$97	\$55	\$105	\$22	\$21
low-recurrence, ≥ \$0.01	\$3,614	\$28	\$60	\$26	\$26
repeat, sub-cent	\$365	\$231	\$23	\$19	\$15
repeat, ≥ \$0.01	\$9,955	\$660	\$33	\$36	\$24
index	**\$14,032**	**\$975**	**\$221**	**\$103**	**\$86** (oracle)

\$86 is an oracle, not a selector — and an oracle cannot lose. It picks, for each resource, the regime that was cheapest over the same thirty days it is scored on. Under any additive cost model $\sum_i \min_r C_{ir} \leq \min_r \sum_i C_{ir}$ holds identically, so "the oracle beats every fixed regime" is arithmetic, not evidence. We report it only as the bound. The evidence is what a **deployable rule** does — one with thresholds a merchant can set in advance and hold fixed: *mean recurrence* > 2 → *channel; else* ≥ 100 *payers and* $p \leq$ \$0.10 → *sparse; else* `exact`. Evaluated retrospectively on the same window, it costs **\$107**, 1.28× the oracle's \$84 at a uniform gas ratio. Fed *forecasts* of n and u perturbed by lognormal noise instead of realised values, it costs \$90 / \$99 / \$107 at typical errors of ±65% / ×÷ 2.7 / ×÷ 4.5 — it degrades to its own fixed-threshold cost as the forecast degrades, which is the right failure mode. A true out-of-sample test needs a second snapshot (§11).

Does the deployable rule survive the cost model? Its thresholds were fixed at the central assumptions and *not* re-tuned. Sweeping gas (\$0.0005 / \$0.002 / \$0.01), channel amortisation (1 / 3 / 6 / 12 months), k (10 / 25 / 100), client cap (\$0.05 / \$0.10 / \$1 / \$5) and a 1× / 2× contract-gas premium — 288 scenarios — at three call fractions:

f	rule beats best single	best-single / rule (min / med / max)	rule / oracle	best single regime
1.0	**195 / 288**	0.63 / 1.24 / 2.63	1.01–1.68	channel 222, sparse 66
0.25	180 / 288	0.60 / 1.29 / 2.37	1.00–1.70	channel 150, sparse 138
0.1	165 / 288	0.59 / 1.15 / 2.10	1.00–1.72	sparse 180, channel 102, exact 6

The rule is never worse than 1.72× the unreachable oracle, and when it loses to the best single regime it loses by at most 1.6×. **Where it loses is the finding.** Of the 93 losses at $f = 1$, 72

are scenarios where channel deposits amortise over six or twelve months — there a deposit is so cheap that always-channel beats a rule that still routes 12,453 low-volume resources to `exact`. The rule's recurrence threshold is right for month-lived deposits and wrong for year-lived ones. That is not an argument against selection; it says the selector has one cost parameter it must measure rather than assume — **how long a buyer's deposit actually lives** — and the merchant's own ledger is where that number accumulates. Meanwhile the identity of the best *single* regime moves across the sweep, from channel 222 / sparse 66 at $f = 1$ to sparse 180 / channel 102 at $f = 0.1$: a merchant who commits to one scheme on principle is committing to a parameter regime it has not measured.

Contract-path gas, measured. A sparse settlement is a Permit2 proxy call with a witness, not a bare EIP-3009 transfer. Rather than assume a premium, we wrote the verifier (a fork of `x402UptoPermit2Proxy`'s structure enforcing invariants 1–3) and ran it in an in-process EVM (Cancun) with the **real Permit2 runtime bytecode** from Base at its canonical address, tickets signed as a wallet signs Permit2 typed data:

	gas	
USDC `transferWithAuthorization` on Base (EIP-3009), median of 12 txs	**86,242**	the reference
sparse settle, winner, repeat buyer (warm nonce word)	72,829	≈ 78,771 USDC-adjusted, 0.91×
sparse settle, winner, one-shot buyer's first ticket (cold nonce word)	89,917	≈ 95,859, 1.11×
sparse settle, `upTo` route fulfilled at 1/10 of the signed price	72,777	odds fall; no extra cost
sparse settle, loser	39,186	reverts `NotAWinner`; never submitted
Permit2 approval, one-time per buyer	46,403	0.54× — the onboarding bound
wrong caller / price above signed / wrong secret / inflated threshold	—	all revert

The USDC adjustment adds 5,942 gas, the gap between USDC's own `transfer()` median (40,271) and the mock's (34,329). The cost table above already uses these ratios; at a uniform 1× it would read always-sparse $103 and oracle $84. Under a hypothetical 2× / 3× premium on every contract-path transaction, always-sparse would be $207 / $310, because `exact` is the only regime a contract premium does not touch; calls routed to sparse stay at 45–46% throughout.

The channel column is a worst case. It models x402's per-receiver EVM channel. A gateway-style deposit — one per buyer, shared across every seller — costs (distinct buyers ×2× gas) / amortisation. The index cannot count distinct buyers across resources (§1.2 ii); bounding by payer pairs, if buyers use on average 1 / 5 / 20 resources, gateway-style channel gas is $221 / $68 / $39 index-wide, against always-sparse's $103. Beyond about five resources per buyer, a gateway beats sparse everywhere.

7 Threat model

Biasing s. The facilitator commits before seeing d, the buyer signs before seeing s; neither biases alone. Collusion can only decline to pay (invariant 1).

Grinding. d is fixed before the reveal and covers c.

Selective abort. Having d, the facilitator knows a loss before the buyer does and could answer "verification failed" with a fresh commitment; an auto-retrying client would re-roll until it loses, turning q into 1 — no funds taken wrongly, but expected cost per *attempt* rises to p. The defence has two halves. **Client side, normative: a client MUST NOT**

sign a second ticket for the same `challengeId` after a verification that did not reveal s against the commitment it signed; non-reveal is a facilitator fault, not a retriable error. **Merchant side: one logical purchase must map to one `challengeId`, durably.** Otherwise a buyer colluding with the facilitator abandons the request, opens a new one, is issued a fresh `challengeId`, and re-rolls at the cost of an HTTP round trip — the client-side rule binds a signature to a challenge but nothing binds a challenge to a purchase. The runtime already has the key for this: the client's idempotency key, or the merchant's own order identifier, keyed to the issued `challengeId` in the ledger and reissued rather than regenerated on a retry. Where a merchant cannot supply such a key — an anonymous one-shot GET with no idempotency header — the merchant is trusting the facilitator not to collude with the payer, and that assumption should be stated in the deployment rather than hidden in the protocol. The underlying reason is structural: the buyer's exposure is bounded on-chain (invariant 1), but *fairness of the odds* is bounded only by the commitment, and a commitment can be discarded by abandoning the purchase it belongs to.

Safety versus liveness. The rule forbids a *new signature* for the same `challengeId`; it does not forbid *resubmitting the same ticket*. A client that cannot tell a malicious non-reveal from a dropped response resubmits the identical signed payload: it is idempotent, the facilitator must reveal s against the same commitment, and the same digest gives the same outcome — no re-roll is possible. Network faults therefore cost nothing. A facilitator that never reveals blocks the purchase, which is the same liveness a facilitator that never verifies gives `exact` today.

Replay, and what it demands of the ledger. A losing permit's Permit2 nonce is never consumed on-chain, so the *only* record that it was spent is the merchant's. That record must be durable and atomic with the verification that produced it — written before the response is served, surviving a restart, and unique across every process serving the resource. An in-memory set is adequate for a reference implementation and inadequate for a deployment: a merchant running two instances behind a load balancer, or restarting between a loss and a replay, accepts the same losing ticket twice and rolls it again. This is the one place where a rail that never touches the chain on the common path puts more weight on the merchant's own storage than `exact` does, and a `sparse` scheme proposal should say so in its MUSTs.

Settling a loser is prevented by invariants 1–3. **Insolvency at a win** is addressed by resolution inside the request; the residual race is `exact`'s. **Denial of service**: an unacceptable ticket is filtered client-side and costs nothing. **Leakage**: wins are public at resolution T.

8 Characterisation, accounting, legal

Economic. Both parties' expected value equals the price; no house, no edge; a settlement mechanism, not a product.

Accounting. Expected value and realized cash are reported separately. The difference is not a receivable — there is no claim against anyone for a losing ticket. Financial-statement recognition is out of scope.

Legal. Whether a payment with a random settlement outcome is a game of chance is a question of jurisdiction, and the absence of a house edge does not settle it. In a deployment where agents sign such authorizations automatically, this is plausibly the largest pre-launch risk, and terminology does not address it. We make no legal claim; a jurisdictional analysis by someone qualified is a prerequisite to charging real money. What the *design* can do is narrow the surface, and the reference implementation does all of it: a sparse entry is never offered alone — a deterministic `accepts[]` entry is always present; the ticket has no prize, no edge and no variable payout — it is T or nothing at exactly p/T; the merchant can disable the regime per resource; and the client's own cap filters it before the buyer sees it.

9 Related work

System	Buyer pre-funds?	Enforcement	Randomness	Resolved
Wheeler 1996; Rivest 1997 [1,2]	no; bank credential	trusted bank	vendor commit, or beacon	at request / later
Micali–Rivest 2002; Peppercoin [3,4]	no; bank or card account	bank / PSP	deterministic merchant signature	at request
Pass–shelat 2015 [5]	escrow	contract	VRF / commit	at request
Orchid 2019 [8]	deposit + balance, global	Ethereum contract	keccak(reveal, nonce) $\leq$ ratio	at request
Livepeer PM [9]	deposit + per-round reserve	`TicketBroker`	keccak(senderSig, recipientRand)	at receipt
MicroCash 2020 [7]	payment + penalty escrow	miners; penalty burn	future block + VDF	later
`emc-randpay-x402` 2026 [10]	no; per-attempt L1 tx	none on-chain	server entropy	at request
Circle Gateway 2026 [12]	Gateway balance, global	Circle (TEE batches)	none	off-chain immediate
x402 `batch-settlement` [11]	per-receiver channel	contract	none	claim then sweep
Sparse (this paper)	**no — wallet + Permit2 approval**	**Permit2 verifier**	**facilitator commit, buyer digest**	**inside the request**

Probabilistic micropayments are thirty years old; bringing them to x402 has been proposed more than once during 2026, including a published scheme draft whose demo decides wins off-chain with local randomness [10], and at least one further technical note describing Rivest-style tickets as an x402 scheme with commit–reveal randomness, which we could not retrieve from a stable location and therefore do not cite. We do not claim the idea of probabilistic tickets over x402, and nothing in this paper's contribution requires it.

Among the systems and specifications we reviewed we did not find a prior construction combining no pre-funded account, ordinary wallet funds, on-chain-enforced probabilistic outcome, in-request resolution and an x402 flow — and we rest nothing on that. What the paper rests on is independent of the mechanism's adoption:

(1) the recurrence split and its f-range (§1);
(2) the client cap and the onboarding cost as bounds the classical literature could omit (§4);
(3) the measured region and its stability (§5);
(4) selection over invention (§6).

10 Cheap settlement does not create demand

If sub-cent resources were unsold because settlement costs too much, cutting that cost should unlock them. The index says otherwise. Sub-cent and cent-plus resources have the same median traffic (2 calls a month), the same median payer count (1) and nearly the same share of single-payer resources (67.5% vs 70.5%). And the one-shot buyer's dominant cost was never the per-call settlement — under EIP-3009 the buyer pays no gas at all. It is the onboarding the Permit2 paths add: at the median sub-cent price of \$0.002, one approval equals one purchase, amortising below 10% of spend only after about ten purchases, and the median buyer in that cell makes one. Whatever suppresses one-shot sub-cent traffic — discovery, trust, wallet onboarding, spend caps — it is not priced in settlement gas, and a mechanism that lowers settlement gas should not be expected to raise it.

11 What has to be measured next

The quantities of §1.2 are not limitations to apologise for; they are the measurements that decide whether any of §5's boundaries move.

(i) The index operator can say what a "call" is.
(ii) Cross-resource payer identity is recoverable from on-chain settlement transactions for `exact`, since the payer address is public.
(iii) The Permit2-approval rate among x402 payers is likewise on-chain.
(iv) Which scheme each call actually settled on, for resources advertising more than one, is visible to facilitators and partially on-chain.
(v) A second index snapshot with a disjoint thirty-day window turns the retrospective rule of §6 into an out-of-sample test — thresholds fixed on September, cost scored on October; we have scheduled one.
(vi) The lifetime of a buyer's channel or gateway deposit, the single cost parameter that decides whether a recurrence-threshold rule or always-channel is right; observable from `x402BatchSettlement` and Gateway contract events.

Until they are measured, the honest reading of every number here is the direction, not the magnitude — and the directions are **not** all the same, which an earlier draft of this work claimed. Two of them shrink the region: a call count inflated by unpaid probes means less real volume per resource (measured: the region falls from 25 to 22 as f goes from 1 to 0.1), and cross-resource payer identity would reveal buyers who are one-shot per resource but recurring in the market, and therefore gateway-shaped. The third moves the other way: **a high existing Permit2-approval rate among x402 payers would enlarge the region**, because the onboarding term is the bound doing most of the work in the target cell. At $\varphi = 0$ — every buyer already approved — the sub-cent low-recurrence resources that beat per-call settlement rise from 63 to 1,068 of 4,716. That measurement is the one most likely to make sparse settlement look better than this paper does, and it is on-chain and unowned by us.

Other limitations. The gas figures come from an in-process EVM on a minimal ERC-20 with the real Permit2 code, adjusted for USDC's transfer overhead, not from a mainnet deployment; the verifier is unaudited and undeployed; the $0.002-per-transaction unit is still an assumption; the channel model is the per-receiver worst case; client caps are three data points, not a survey; recurrence from unique addresses misreads rotated and shared addresses; sparse excludes EIP-3009 structurally and ERC-1271 unless added; the legal question is open.

12 Next steps

1. **Reference rail, off-chain — done.** A rail on the runtime's `createRail()` with an in-memory facilitator doing commit–reveal, signature and threshold checks, balance checks and idempotent settlement. It passes the runtime's nine-case rail conformance suite and eighteen mechanism tests, including the selective-abort case and its retry rule, `upTo` settlement at the charged price's odds, and a lost settle response resolved by lookup once. Its replay set and facilitator are in memory, which §7 calls adequate for an experiment and inadequate for a deployment.
2. **x402 binding.** `accepts[].extra` carries `ticket`, `price`, `commitment`, `challengeId`, `facilitatorAddress`; the payload is the Permit2 permit over the witness; `/verify` reveals s and returns the outcome; `/settle` submits winners and returns `amount: 0` with an empty transaction for losers, as `upto` already permits; the client rule of §7 goes in the MUSTs.

3. **Verifier contract — written and measured, not audited or deployed.** A benchmark compiles it and runs it against the real Permit2 bytecode, printing the table in §6. Next: audit, and the live measurement on Base Sepolia with the buyer-side distribution.
4. **The measurements of §11**, then re-run §5. The region is a function of the market, not the mechanism.

13 Conclusion

x402 settles isolated payments well, and repeat relationships well for buyers who deposit first. Its index is mostly buyers who have not deposited and mostly calls from those who have, and the schemes built for the second group are almost unused. Sparse settlement closes the smaller gap with a channel-free, on-chain-enforced probabilistic ticket resolved inside the request; we did not find a prior construction of that shape and rest nothing on it. Its benefit is bounded four ways — merchant variance, buyer exposure, a per-payment cap already shipped in the reference client, and an onboarding transaction the default path never asks for — and crossing those bounds with real traffic leaves 25 resources of 11,630, while in the cell it was built for the median ticket saves four percent of settlements and three quarters of resources cannot pay for the mechanism at any share of returning buyers. The output is therefore the boundary, not the ticket: a merchant runtime that measures recurrence, volume and the client's cap, chooses per resource, and — because the best single scheme flips with assumptions the market has not yet measured — refuses to commit to one.

References

[1] D. Wheeler. Transactions Using Bets. *Security Protocols Workshop 1996*, LNCS 1189, pp. 89–92.

[2] R. L. Rivest. Electronic Lottery Tickets as Micropayments. *Financial Cryptography 1997*, LNCS 1318, pp. 307–314.

[3] S. Micali and R. L. Rivest. Micropayments Revisited. *CT-RSA 2002*, LNCS 2271, pp. 149–163.

[4] R. L. Rivest. Peppercoin Micropayments. *Financial Cryptography 2004*, LNCS 3110, pp. 2–8.

[5] R. Pass and a. shelat. Micropayments for Decentralized Currencies. *ACM CCS 2015*, pp. 207–218.

[6] A. Chiesa, M. Green, J. Liu, P. Miao, I. Miers and P. Mishra. Decentralized Anonymous Micropayments. *EUROCRYPT 2017*, LNCS 10211, pp. 609–642.

[7] G. Almashaqbeh, A. Bishop and J. Cappos. MicroCash: Practical Concurrent Processing of Micropayments. *Financial Cryptography 2020*. arXiv:1911.08520.

[8] Orchid Labs. *Orchid: A Decentralized Network Routing Market* (whitepaper), 2019. Nanopayment construction §5.2–5.5; `lottery0.sol` in `OrchidTechnologies/orchid`.

[9] Livepeer. *Probabilistic Micropayments* (Streamflow specification), `livepeer/wiki`, `spec/streamflow/pm.md`.

[10] `Jeff-Bouchard/emc-randpay-x402`, an x402 scheme draft over Emercoin's `randpay`, May 2026. Underlying protocol: O. Konashevych and O. Khovayko, *Randpay*, arXiv:1912.00007 (withdrawn by its authors).

[11] x402 Foundation. *x402 specification and schemes* (`exact`, `upto`, `batch-settlement`), `x402-foundation/x402`, main branch, read 2026-09-20. Reference-client spend controls: `typescript/packages/core/src/client/x402Client.ts`.

[12] Circle. *Gateway Nanopayments* documentation and mainnet announcement, developers.circle.com and circle.com, 29 April 2026; read 2026-09-21.
[13] Coinbase Developer Platform. *Agentic Wallets* FAQ (per-call and per-session spending limits), docs.cdp.coinbase.com, read 2026-09-21.
[14] Uniswap Labs. *Permit2*, canonical deployment `0x000000000022D473030F116dDEE9F6B43aC78BA3`.
[15] Coinbase Developer Platform. *x402 Bazaar discovery API*, `api.cdp.coinbase.com/platform/v2/x402/discovery/resources`, fetched in full 2026-09-20.

Reproducibility

Index fetched 2026-09-20 from the CDP Bazaar public discovery API (150 pages of 100). Population: 14,719 resources whose first offer is priced in a known USDC contract below $1,000 with ≥ 1 reported call; recurrence = `l30DaysTotalCalls / l30DaysUniquePayers`. Gas: `gas-benchmark.ts` (solc 0.8.37, `@ethereumjs/vm`, Cancun, chainId 8453, Permit2 runtime bytecode read from Base on 2026-09-21), output in `gas-benchmark.json`; EIP-3009 and USDC `transfer()` references from Base mainnet receipts on 2026-09-21. Scripts: `exact-variance.mjs` (every §4–6 number under the exact binomial bound), `sparse-analytic.mjs`, `quadrant.mjs`, `buyer-side.mjs`, `region2.mjs` (figure), `misselect.mjs`, `sensitivity.mjs`, `fixed-selector.mjs`, `callfrac.mjs`, `oos.mjs` (the out-of-sample test, awaiting the second snapshot). Gas $0.002 per transaction. Specification quotations from `x402-foundation/x402` at main, 2026-09-20; Circle from developers.circle.com and the Coinbase agent-wallet FAQ from docs.cdp.coinbase.com, 2026-09-21. A second snapshot is planned for late October 2026.