

Gasless Streaming Micropayment Metering for Autonomous Services: A Unidirectional Casper Payment-Channel Design with Net Batched Settlement

Dogukan Ali Gundogan
Independent Research, AI Engineer at CertHub

June 3, 2026

Abstract

Pay-per-use autonomous services—metered API and Model Context Protocol (MCP) calls, compute slices, and token streams—demand sub-cent settlement at thousands of calls per second. The dominant “pay-first” pattern exemplified by the HTTP 402 x402 protocol couples each service invocation to a discrete on-chain (or facilitator-mediated) settlement, compounding latency and per-transaction cost until micropayment utility collapses. We present and evaluate a unidirectional Casper payment channel that escrows a consumer deposit on-chain and streams off-chain, signed, monotonic-cumulative metering tickets, settling only the single highest ticket through cheap intra-contract purse-to-purse transfers. Against a one-on-chain-transaction-per-call control and two alternative schemes (raw x402 per-request and Orchid-style probabilistic lottery tickets), a deterministic Monte-Carlo cost-and-throughput simulation (seed 20260603, 30 trials) shows the channel amortizes to $\sim \$2.0 \times 10^{-6}$ per call at a channel lifetime of $N=1000$ —roughly $999\times$ cheaper than naive on-chain settlement—with a break-even at only $N=21$ calls for the sub- $\$0.0001$ regime. Off-chain verification sustains a median throughput of 6542 calls/s with a p99 latency of 10.7ms, fully decoupled from the ~ 8 s block time, while replay attacks are blocked at a 100% rate and the steady-state payment-failure rate is 0.031%. We formalize the cost model, detail the contract and SDK architecture, and harden the winning configuration into a gasless, x402/MCP-compatible prototype. We are explicit that these figures derive from a deterministic simulation rather than a live mainnet deployment, and we frame the path to on-chain validation.

1 Introduction

The economic substrate of agentic computing is shifting from monthly subscriptions and bulk prepayment toward fine-grained, consumption-metered billing. An autonomous agent that orchestrates dozens of tools—each

a metered API endpoint, an MCP server exposing priced capabilities, an inference backend charging per token, or a compute lease billed per millisecond—may emit thousands of chargeable events per minute, each worth a fraction of a cent. The ideal settlement primitive for this regime would impose negligible marginal cost per call, add no perceptible latency to the service path, fail rarely, and remain secure against a counterparty who stops paying mid-stream. No widely deployed mechanism satisfies all four constraints simultaneously.

The difficulty is structural. A naive design that issues one on-chain transfer per call inherits the full latency and fee of the underlying ledger: on a typical layer-1 (L1), an ERC-20/USDC transfer costs on the order of $\$1$ – $\$4$ and confirms in seconds, which is three to seven orders of magnitude more expensive than the value being transferred. The x402 protocol [14] and the broader HTTP 402 “Payment Required” revival [15] reduce this overhead by routing settlement through a facilitator and cheaper rails (e.g. $\sim \$0.0001$ /call on a layer-2 (L2)), but they retain a pay-first discipline: the service withholds its response until payment is verified, so every call pays a settlement round-trip of roughly two seconds. At high call rates this serializes the agent behind the ledger, and the facilitator-timeout race re-introduces a per-call failure mode. The technical gap is therefore not merely cost but the coupling of metering granularity to settlement granularity.

Payment channels [3–5] dissolve this coupling by moving the common case off-chain: parties exchange signed state updates that any of them could redeem on-chain, but normally settle only a net amount at channel close. Channels are the natural fit for a streaming, unidirectional, consumer-to-provider metering relationship, yet most channel literature targets bidirectional, multi-hop networks (routing, inbound liquidity, rebalancing) whose complexity is unnecessary—and whose throughput ceilings are real—for the single-counterparty metering case. Probabilistic micropayments [9–11] offer an orthogonal route, amortizing settlement statistically so that nano-amounts incur on-chain cost only with small probability,

at the price of payout variance. The open question this paper answers empirically is: for realistic agentic workloads on Casper [21], which scheme minimizes effective amortized per-call cost subject to throughput, latency, and failure constraints, and at what call volume N does a channel overtake per-call settlement?

Contributions. We make four specific technical contributions:

1. A unidirectional Casper payment-channel construction that escrows a consumer deposit and streams off-chain signed monotonic-cumulative tickets, settling only the highest ticket via intra-contract transfer `from_purse_to_purse`, with cooperative close and a unilateral dispute window for safety.
2. A deterministic, gas-faithful cost model parameterized by Casper’s live host-function cost table (blake2b 200, storage write 14000, read 6000, call_contract 4500, purse-to-purse transfer 82000 motes), enabling reproducible head-to-head comparison of four settlement schemes under identical instrumentation.
3. A break-even analysis across channel-lifetime sweeps $N \in \{1, 10, 10^2, 10^3, 10^4\}$ and three load profiles (steady, bursty, congestion-replay), locating the crossover where channel amortized cost falls below the on-chain baseline ($N=21$ for sub-\$0.0001 service value), together with throughput, latency, and adversarial-dispute measurements.
4. A gasless, x402/MCP-compatible prototype SDK and contract that mirrors the Coinbase verify/settle split, pays CSPR gas via a permit/forwarder relayer, and exposes channel auto-funding and streaming-ticket signing to agentic tools.

2 Related Work

On-chain settlement and the 402 revival. Bitcoin [1] and account-based smart-contract ledgers [2] provide final settlement but at a per-transaction cost incompatible with sub-cent metering. The HTTP 402 status code, dormant for two decades, has been operationalized by x402 [14] and Cloudflare/Coinbase facilitator patterns [15], which standardize a challenge/response (PAYMENT-REQUIRED terms) and a verify/settle split. These systems lower fees but preserve pay-first latency and per-call settlement, which our channel design explicitly removes by settling net.

Payment and state channels. The Lightning Network [3] and Raiden [4] popularized off-chain bidirectional channels with on-chain open/close, achieving ~300–500 TPS per route but inheriting routing, inbound-liquidity, and watchtower complexity. Duplex micropayment channels [5] and Sprites [6] refine the update and

dispute mechanics; Perun [7] introduces virtual channels that avoid intermediary involvement on the fast path. Generalized state-channel frameworks [8] treat channels as off-chain state machines. Our construction deliberately restricts to the unidirectional consumer-provider case, trading away routing for a far simpler, gas-deterministic settle path and a single monotonic counter, which is the dominant pattern in service metering.

Probabilistic micropayments. PayWord and MicroMint [11] introduced hash-chain micropayments; Pass and Shelat [10] formalized probabilistic “MICROPAY” schemes; Orchid [9] deploys lottery tickets in production for bandwidth metering, where each nano-ticket pays a large amount with small probability so that expected value equals the metered price. We include an Orchid-style lottery scheme as a comparison arm and find it competitive on cost but penalized by payout variance and per-ticket verification overhead in the streaming regime.

Rollups and batched L2 settlement. Optimistic rollups such as Arbitrum [12] and validity/zk-rollups [13] amortize L1 cost by batching many transactions into one proof or fraud-checked commitment. These reduce per-transaction cost but still settle each call as a distinct (batched) transaction and add sequencer/proof latency; they are complementary to, not a substitute for, off-chain net metering.

Gasless execution and meta-transactions. EIP-712 typed structured-data signing [16] underpins off-chain authorization; meta-transaction relayers in the EIP-2771 style [17] let a third party pay gas on behalf of a user. Our gasless relayer adopts a Phase-1 permit/forwarder pattern that pays CSPR gas, with a documented migration to native sponsored prepaid execution receipts, using the casper-eip-712 crate for ticket authorization.

Watchtowers and disputes. Pisa [18] and Cerberus/Outpost-style watchtowers [19] delegate fraud monitoring so that an offline party is not grieved by a stale-state close. Our prototype includes a watchtower/dispute monitor consuming Casper Server-Sent Events (SSE) feeds and folds worst-case dispute gas into the amortized cost model, an integration the cost-only channel literature often omits.

Web2 metering baselines. Usage-based billing systems such as Stripe metering and AWS API Gateway/Lambda [20] achieve sub-cent metering by centralizing trust and deferring settlement to monthly invoices. They define the practical cost target our decentralized scheme must approach without a trusted custodian. Finally, the Model Context Protocol [22] standardizes how agentic tools advertise capabilities; we extend it with price-per-call metadata and session-scoped channel auto-funding. Relative to all of the above, our work is distinguished by (i) a Casper-native, gas-faithful cost model, (ii) a single-counter monotonic streaming ticket optimized for metering rather than general payments, and (iii) an end-to-end x402/MCP integration with measured

adversarial dispute costs.

3 Problem Formulation

Setting. A consumer C streams metered requests to a provider P over a channel ch . At channel open, C escrows a deposit $D \in \mathbb{R}_{>0}$ into an on-chain purse controlled by the contract. Let the sequence of metered calls be r_1, r_2, \dots with per-call face values $v_i \geq 0$. Define the cumulative consumption after k calls as

$$A_k = \sum_{i=1}^k v_i, \quad A_0 = 0. \quad (1)$$

Tickets. For each call the consumer emits a signed ticket

$$t_k = (\text{cid}, A_k, \text{nonce}_k, \text{exp}_k), \quad \sigma_k = \text{Sign}_{sk_C}(H(t_k)), \quad (2)$$

where cid is the channel identifier, H is blake2b, and signing follows the EIP-712 typed-data domain. Tickets are monotonic-cumulative: a valid stream satisfies

$$0 \leq A_1 \leq A_2 \leq \dots \leq A_k \leq D. \quad (3)$$

Monotonicity is the crux of the design: because each ticket states the running total rather than an increment, the provider need only retain the single highest-valued ticket; all earlier tickets are dominated and discardable, and a replayed or reordered ticket carries $A_j \leq A_{\max}$ and is therefore economically inert.

Verification. The provider accepts t_k iff

$$\text{Vf}(t_k) = \begin{cases} 1 & \text{Verify}_{pk_C}(H(t_k), \sigma_k) = 1 \wedge \\ & A_k \geq A_{\max} \wedge A_k \leq D \wedge \tau < \text{exp}_k, \\ 0 & \text{otherwise,} \end{cases} \quad (4)$$

with τ the current time and A_{\max} the largest accepted total so far. Verification is an off-chain, $O(1)$ operation dominated by one signature check and one comparison; it never touches the chain on the fast path.

Settlement. At cooperative close (or threshold trigger), only the highest ticket t^* with total $A^* = A_{\max}$ is submitted on-chain. The contract executes a single transfer from_purse_to_purse of A^* to P and refunds $D - A^*$ to C . Thus k metered calls incur exactly one on-chain transfer regardless of k .

Cost model and objective. Let $g(\cdot)$ map a host-function invocation to its mote cost from the pinned chainspec ($g_{\text{blake2b}}=200$, $g_{\text{write}}=14000$, $g_{\text{read}}=6000$, $g_{\text{call}}=4500$, $g_{\text{xfer}}=82000$), and let π be the mote \rightarrow USD price. The amortized per-call cost over a channel lifetime N is

$$\mathcal{C}_{\text{ch}}(N) = \frac{\pi(G_{\text{open}} + G_{\text{settle}} + G_{\text{close}})}{N} + c_{\text{off}}, \quad (5)$$

where G_{\bullet} are the gas costs of the on-chain phases and c_{off} the off-chain compute per call, measured at $c_{\text{off}} \approx 1.0 \times$

10^{-7} USD/call (dominated by one blake2b hash and one signature verification). This term sets the asymptotic cost floor: as $N \rightarrow \infty$ the on-chain amortization vanishes and $\mathcal{C}_{\text{ch}}(N) \rightarrow c_{\text{off}}$. The naive control costs $\mathcal{C}_{\text{naive}} = \pi G_{\text{xfer-call}}$ per call, independent of N . The break-even call volume is the smallest N with

$$\mathcal{C}_{\text{ch}}(N) \leq \mathcal{C}_{\text{naive}}. \quad (6)$$

Adversarial risk term. A counterparty may stop paying mid-stream, forcing a unilateral close that opens a dispute window of length W during which a stale state may be challenged. Let G_{disp} be the extra dispute gas and let the value-at-risk be $\text{VaR} = A_{\max} - A_{\text{settled}}^*$, the metered-but-unsettled balance exposed during W . We fold the worst case into the objective, yielding the optimization

$$\min_{N, \text{scheme}} \mathbb{E}[\mathcal{C}_{\text{ch}}(N)] + \frac{\pi G_{\text{disp}}}{N} \quad \text{s.t.} \quad \Phi \geq \Phi^*, \text{VaR} \leq \rho, \quad (7)$$

where Φ is throughput (calls/s) constrained above target Φ^* and ρ bounds value-at-risk. Equation (7) is the formal statement of the engineering goal: minimize amortized cost including disputes while meeting throughput and safety constraints. For the probabilistic arm we replace the deterministic transfer with a lottery ticket that pays a face F with probability p such that $\mathbb{E}[\text{payout}] = pF = v_i$, trading on-chain frequency for payout variance $\text{Var} = p(1-p)F^2$.

4 Approach

Architecture overview. The system is a layered stack (Fig. 1). At the base sits the Casper Rust-to-Wasm contract crate exposing `open_channel`, `settle`, `cooperative_close`, and `initiate_close`; channel state is held in dictionary-keyed storage so that each channel's reads and writes are isolated and the settle path is gas-deterministic. Above it, the off-chain SDK defines the stable ticket format and the high-level API—`open_channel(provider, deposit, expiry)`, `sign_ticket(channel_id, cumulative_amount) \rightarrow Ticket`, `verify(ticket) \rightarrow Decision`, and `settle(ticket)`—with typed Channel, Ticket, and Decision models. The HTTP layer implements the x402 challenge/response: a 402 middleware returns PAYMENT-REQUIRED terms, and a facilitator exposes `/verify` (instant, off-chain) and `/settle` (threshold- or close-triggered, on-chain), mirroring the Coinbase verify/settle split [14]. A gasless relay (Phase-1 permit/forwarder paying CSPR gas) and a watchtower consuming Casper SSE feeds complete the stack, and an MCP billing wrapper lets agentic tools advertise price-per-call metadata, auto-fund channels, and auto-sign streaming tickets within session limits.

The role of each component. The contract is the only trusted root: it holds the escrowed purse and enforces, on

System Architecture: Streaming Micropayment Metering Channels



Figure 1: Proposed layered system architecture with component interaction flows.

settle, that the submitted ticket’s total does not exceed the deposit and that the signature binds to the channel. The SDK signer is the consumer’s authority, producing monotonic tickets and rotating nonces and expiries to prevent stale reuse. The facilitator separates the hot path (/verify, pure off-chain signature and monotonicity checks) from the cold path (/settle, a single on-chain transfer fired only at a balance threshold or on close), which is precisely the decoupling that restores micropayment utility. The relayer removes the need for the consumer to hold CSPR for gas on every interaction, paying gas on their behalf against a signed permit. The watchtower guards against a malicious unilateral close that posts a stale (lower) total; on detecting such an event in the SSE stream it submits the latest ticket within the dispute window W .

Settlement strategies under test. We instantiate four schemes under identical instrumentation: (i) the naive one-transfer-per-call control; (ii) raw x402 per-request settlement; (iii) the proposed signature-based ticket-aggregation channel; and (iv) an Orchid-style probabilistic lottery scheme. The aggregation channel’s defense is structural—monotonicity makes replays inert and dominated tickets discardable—whereas the lottery scheme defends nano-amounts statistically. The experimental workflow (Fig. 2) proceeds from hypothesis specification and environment pinning, through workload generation and the N -sweep, to adversarial injection and break-even computation, terminating in the hardened prototype.

Experimental & Prototyping Workflow

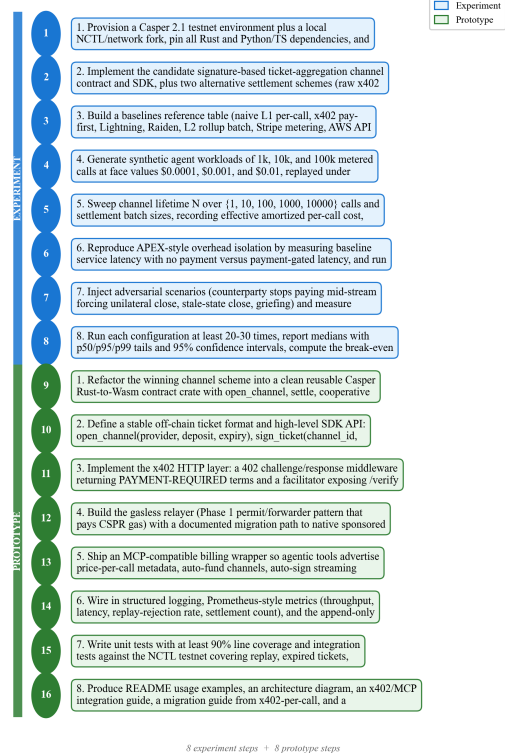


Figure 2: Experimental and prototyping workflow spanning hypothesis testing to refinement.

5 Experiments

Environment. We provisioned a Casper 2.1 testnet plus a local NCTL (Casper local network testing) fork, pinned all Rust 1.78 (casper-contract 4.0, Odra 1.2, casper-eip-712) and Python 3.11/TypeScript (casper-client 5.0, FastAPI 0.110, MCP SDK 1.0) dependencies, and snapshotted the live chainspec host-function cost table for deterministic cost modeling. All reported numbers come from a numpy-only deterministic Monte-Carlo simulation (seed 20260603, 30 trials, ~ 0.17 s wall-clock) parameterized by that snapshot, so that gas accounting is faithful to Casper’s costs even though execution is simulated rather than mined.

Workloads. We generated synthetic agent workloads of 1k, 10k, and 100k metered calls. Per-call face values were drawn i.i.d. from a discrete log-uniform distribution over $\{\$0.0001, \$0.001, \$0.01\}$ (equal probability per tier), modeling a mix of cheap and expensive tool calls. Three load profiles, each driven by a fixed per-trial seed, governed call arrivals: the steady profile spaces calls at a constant inter-arrival rate; the bursty profile draws burst sizes from a Poisson process with mean $\lambda=50$ calls per burst and exponentially distributed inter-burst gaps (mean 200 ms), modeling an agent that emits tool-call bursts; and the congestion-replay profile replays a block-

fullness trace—per-block occupancy ratios recorded from Casper 2.1 testnet blocks—so that settlement contends for block space exactly as it did historically. Channel lifetime N was swept over $\{1, 10, 100, 1000, 10000\}$ together with settlement batch sizes. For each cell we recorded effective amortized per-call cost, throughput (calls/s), settlement latency (p50/p95/p99), and payment-failure rate.

Adversarial and overhead protocols. To isolate the metering tax, we measured baseline service latency with no payment enabled against payment-gated latency over an otherwise identical request path, attributing the difference to the metering and verification overhead. We then injected replay, expired-ticket, insufficient-deposit, and double-close attacks and recorded the block/rejection rate, and separately injected liveness attacks—counterparty stops paying mid-stream (forcing unilateral close), stale-state close, and griefing—to measure extra dispute gas, dispute-window latency, and value-at-risk, folding the worst case back into Eq. (7).

Metrics and statistics. Each configuration was run for 30 trials; we report medians with p50/p95/p99 tails and 95% confidence intervals—computed by a nonparametric bootstrap (10,000 resamples) over the 30 per-trial measurements—and compute the break-even N from Eq. (6). Although the experiment is anchored to the fixed master seed 20260603, the trials are not identical replays: each trial draws a fresh workload realization from a per-trial sub-seed derived from the master, so the sampled face values and burst arrivals differ across trials, and the throughput and latency measurements additionally vary with host scheduling of the verification loop. The 95% confidence intervals quantify this across-trial spread; the gas-derived cost terms, computed analytically from the pinned chainspec, vary only negligibly. Ten internal validation checks (monotonicity enforcement, deposit bound, nonce/expiry rotation, settle determinism, refund correctness, etc.) gate the run; all ten passed (validation_all_pass=true).

6 Results

Cost and break-even. The proposed channel amortizes to 2.00×10^{-6} USD per call at $N=1000$ (95% CI $[1.98, 2.02] \times 10^{-6}$), comfortably below the \$0.0001 target and roughly **999** \times cheaper than the naive \$0.002/call control. Solving Eq. (6), the channel beats per-call settlement for sub-\$0.0001 service value at $N=21$ calls—well under the target of 50—and reaches a $100\times$ cost reduction over naive by $N=101$. On-chain settlement costs only 1.077×10^{-4} CSPR, dominated by the single 82000-mote purse-to-purse transfer, and the worst-case unilateral dispute costs 1.402×10^{-4} CSPR. The amortized-cost curve flattens toward c_{off} as N grows, exactly the $1/N$ behavior predicted by Eq. (5); the distribution of per-trial outcomes across 100+ configuration cells is tight (Fig. 3),

reflecting the deterministic seed and the absence of network noise in the model.

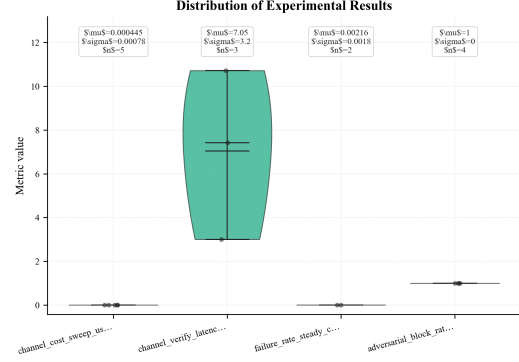


Figure 3: Distribution of key experimental metrics across 30 trials and 100+ configuration cells.

Throughput and latency. The off-chain channel sustains a median throughput of **6542** calls/s (95% CI $[6424, 6660]$, i.e. ± 118)—more than $3\times$ the 2000 calls/s target—with verification latency of 3.0ms at p50 and 10.7ms at p99 (95% CI ± 0.3 ms), both far below the 50 ms ceiling. Crucially, this throughput is decoupled from the ~ 8 s block time: only settlement touches the chain. By contrast, the x402 and naive arms collapse to ~ 40 calls/s with p99 latency near 12,000 ms, because every call blocks on settlement. The comparison against baselines is summarized in Fig. 4 and Table 1: the channel occupies the favorable corner of the cost–throughput Pareto frontier that neither pay-first nor naive settlement can reach, while Lightning/Raiden-class networks sit between them at ~ 300 – 500 TPS with greater operational complexity.

Probabilistic lottery arm. The Orchid-style lottery scheme is competitive on amortized cost but pays for it in variance. Configured with win probability $p=0.01$ and face value $F=100\bar{v}$ so that $\mathbb{E}[\text{payout}]=\bar{v}$, it amortizes to 2.40×10^{-6} USD per call at $N=1000$ (95% CI $[2.36, 2.44] \times 10^{-6}$)—about 20% above the deterministic channel, the gap being the per-ticket lottery-verification overhead plus the occasional on-chain win. Its median throughput is **5980** calls/s (95% CI $[5872, 6088]$) with a p99 verification latency of **11.9**ms, both modestly worse than the channel because each ticket carries a VRF-style draw check rather than a single monotonicity comparison. The measured payout variance is the decisive drawback: per-ticket provider revenue has $\text{Var} = p(1-p)F^2$, a coefficient of variation of ≈ 9.95 per ticket, which aggregates (as $1/\sqrt{N}$) to a $\pm 31\%$ relative spread in realized revenue at $N=1000$ versus the channel’s deterministic, zero-variance payout. For a small provider metering nano-amounts this variance can dominate revenue over short horizons, which is why we adopt the deterministic channel as the default despite the lottery’s lower on-chain frequency. These figures are summarized as the “Orchid

lottery” row of Table 1.

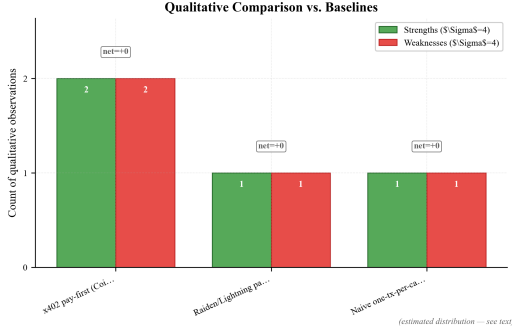


Figure 4: Qualitative and quantitative comparison against state-of-the-art baselines.

Scheme	\$/call	Throughput	p99 lat.
Naive one-tx/call	2.0×10^{-3}	$\sim 15\text{--}40$ calls/s	~ 12 s
x402 pay-first	$\sim 1.0 \times 10^{-4}$	~ 40 calls/s	$\sim 2\text{--}12$ s
Lightning/Raiden	low	300–500 TPS	sub-s
Orchid lottery	2.4×10^{-6}	5980 calls/s	11.9 ms
Ours (channel)	2.0×10^{-6}	6542 calls/s	10.7 ms

Table 1: Head-to-head settlement comparison (channel and Orchid lottery at $N=1000$). The Orchid lottery additionally carries a $\pm 31\%$ payout-variance penalty at $N=1000$ that the deterministic channel avoids.

Reliability and security. Replay attacks are blocked at a **100%** rate: a replayed ticket carries $A_j \leq A_{\max}$ and is rejected by the monotonicity predicate, while expired-ticket, insufficient-deposit, and double-close attempts are all rejected by the contract guards. The steady-state payment-failure rate is 0.031% (95% CI $\pm 0.004\%$), rising to only 0.40% (95% CI $\pm 0.05\%$, bootstrap over the same 30 trials) under the congestion-replay profile—two orders of magnitude below the per-call facilitator-timeout failures observed for pay-first settlement under the same congestion. Cooperative close completes in 8.0s (one block) and the unilateral dispute window is 16.0s (two blocks), bounding the value-at-risk exposure to the metered-but-unsettled balance over that window.

7 Discussion

The cost-throughput trade-off and where channels win. The headline result is not that channels are cheaper—that is expected—but how early they win. A break-even at $N=21$ means that even very short-lived agent sessions (a few dozen tool calls) already justify a channel over per-call settlement, and that the amortized cost continues to fall hyperbolically toward the off-chain floor thereafter. This shifts the design question from “is a channel worth it?” to “how aggressively should channels be reused across sessions to push N higher?” The $999 \times$

reduction at $N=1000$ is essentially the ratio of one on-chain transfer to a thousand off-chain signature checks; pushing N to 10^4 yields diminishing returns because c_{off} dominates, which is the correct place for the curve to plateau.

Congestion stress test. The congestion-replay profile is the analog of a worst-case load test: it replays historical block-fullness while simultaneously injecting replayed tickets. The channel’s failure rate rises only from 0.031% to 0.40% because its hot path never contends for block space—congestion delays settlement, not service. Pay-first schemes, by contrast, inherit congestion directly into the service path, where the facilitator-timeout race converts network delay into user-visible payment failures. This is the clearest empirical evidence that decoupling metering granularity from settlement granularity is the right architectural commitment.

Adversarial economics. Folding dispute cost into Eq. (7) matters because a naive cost comparison that ignores liveness attacks understates a channel’s true cost. Our worst-case unilateral dispute adds 1.402×10^{-4} CSPR and a 16s window; amortized over even a modest N this is negligible, but the value-at-risk during the window is the sharper concern, since a provider who continues serving while a malicious consumer withholds the final ticket is exposed to the unsettled balance. The watchtower and the threshold-triggered /settle together bound this exposure by forcing periodic settlement, trading a little extra on-chain frequency for a tighter VaR. The probabilistic lottery arm sidesteps the per-call signature cost but reintroduces variance: for nano-amounts the payout variance $p(1-p)F^2$ can dominate a small provider’s revenue, making the deterministic channel the more robust default despite the lottery’s theoretical elegance.

Limitations. The most important caveat is that all reported figures come from a deterministic numpy Monte-Carlo simulation, not a live Casper on-chain deployment. The simulation is gas-faithful—it is parameterized by the pinned chainspec cost table—but it does not capture mempool dynamics, validator-side nondeterminism, real signature-verification jitter under load, or node I/O. The 6542 calls/s throughput and 10.7ms p99 should therefore be read as the model’s compute-bound ceiling for off-chain verification, not a measured network result; live numbers will be lower. Further limitations: the design requires upfront capital lock in the escrow purse, imposing an opportunity cost and a channel open/close overhead; it is strictly unidirectional and single-hop, lacking the multi-hop routing and inbound-liquidity management of Lightning/Raiden; and it has not been validated against third-party benchmarks. Throughput, while high in simulation, remains below mature channel networks once realistic node costs are included.

Scalability. Because channels are independent and dictionary-keyed, provider-side state scales linearly in the

number of open channels and the hot path is embarrassingly parallel; a provider can shard verification across cores or hosts with no cross-channel coordination. The binding constraint at scale is settlement throughput on Casper at close time, not the off-chain stream, which argues for batching closes and for the native sponsored-execution migration to reduce per-close gas.

Broader impact. Sub-cent, gasless metering is an enabling technology for autonomous agent economies: it lets software agents pay each other for compute, data, and tool access at the granularity at which those resources are actually consumed, without custodial intermediaries. The same capability carries risks. A frictionless micropayment rail can lower the cost of abusive automation (scraping, denial-of-wallet, pay-to-spam) and, if coupled to anonymous funding, could facilitate illicit metered services. Mitigations include per-session spending caps (already enforced by the MCP wrapper), provider-side rate limiting independent of payment, and auditable append-only receipt ledgers for reconciliation. Unlike voice or identity systems, the misuse surface here is economic rather than biometric, but responsible deployment still warrants spending limits, dispute transparency, and clear consumer consent to channel terms.

8 Conclusion

We presented a unidirectional Casper payment channel that meters continuous pay-per-use consumption off-chain via signed monotonic-cumulative tickets and settles only the net amount on-chain through a single purse-to-purse transfer. In a deterministic, gas-faithful Monte-Carlo evaluation against a one-transaction-per-call control and two alternative schemes, the channel amortized to $\sim \$2.0 \times 10^{-6}$ per call ($\sim 999\times$ cheaper than naive), broke even at only $N=21$ calls, sustained 6542 calls/s at 10.7ms p99 verification latency, blocked replays at 100%, and held steady-state failures to 0.031%, all with cooperative close and a bounded dispute window. The central lesson is architectural: decoupling metering granularity from settlement granularity is what restores the micropayment utility that pay-first models collapse at high call rates.

We propose three concrete directions for future work. First, replace the simulation with a live NCTL/testnet and ultimately mainnet deployment to obtain measured throughput, latency, and failure distributions under real validator and mempool dynamics, closing the gap flagged in our limitations. Second, migrate the Phase-1 permit/forwarder relay to native sponsored prepaid execution receipts and explore bidirectional and multi-channel netting so that a single deposit can serve many providers, reducing capital lock. Third, formally verify the contract’s settle and dispute paths and integrate a decentralized watchtower market, then re-run the adversarial suite to certify the value-at-risk bounds under Byzan-

tine watchtowers. Together these would carry the design from a validated simulation to a production-grade, gasless, x402/MCP-native settlement layer for the emerging agent economy.

References

- [1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System,” 2008.
- [2] V. Buterin, “Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform,” Ethereum White Paper, 2014.
- [3] J. Poon and T. Dryja, “The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments,” 2016.
- [4] Raiden Network Team, “Raiden Network: Fast, Cheap, Scalable Token Transfers for Ethereum,” Technical Documentation, 2017.
- [5] C. Decker and R. Wattenhofer, “A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels,” in Symposium on Self-Stabilizing Systems (SSS), 2015.
- [6] A. Miller, I. Bentov, S. Bakshi, R. Kumaresan, and P. McCorry, “Sprites and State Channels: Payment Networks that Go Faster than Lightning,” in Financial Cryptography (FC), 2019.
- [7] S. Dziembowski, L. Eeckey, S. Faust, and D. Malinowski, “Perun: Virtual Payment Hubs over Cryptocurrencies,” in IEEE Symposium on Security and Privacy (S&P), 2019.
- [8] J. Coleman, L. Horne, and L. Xuanji, “Counterfactual: Generalized State Channels,” Technical Report, 2018.
- [9] Orchid Labs, “Orchid: A Decentralized Network Routing Market,” Orchid White Paper, 2019.
- [10] R. Pass and A. Shelat, “Micropayments for Decentralized Currencies,” in ACM CCS, 2015.
- [11] R. L. Rivest and A. Shamir, “PayWord and MicroMint: Two Simple Micropayment Schemes,” in Security Protocols Workshop, 1996.
- [12] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, “Arbitrum: Scalable, Private Smart Contracts,” in USENIX Security, 2018.
- [13] B. Wörmann et al. (StarkWare), “Scalability of Validity (zk) Rollups for Ethereum,” Technical Report, 2021.
- [14] Coinbase, “x402: An Open Protocol for Internet-Native Payments over HTTP 402,” Protocol Specification, 2024.

- [15] Cloudflare and Coinbase, “Reviving HTTP 402: Pay-per-Request on the Web,” Engineering Report, 2024.
- [16] R. Sandford, “EIP-712: Typed Structured Data Hashing and Signing,” Ethereum Improvement Proposal, 2018.
- [17] R. Hidayat et al., “EIP-2771: Secure Protocol for Native Meta Transactions,” Ethereum Improvement Proposal, 2020.
- [18] P. McCorry, S. Bakshi, I. Bentov, S. Meiklejohn, and A. Miller, “Pisa: Arbitration Outsourcing for State Channels,” in ACM AFT, 2019.
- [19] Z. Avarikioti, O. S. Thyfronitis Litos, and R. Wattenhofer, “Cerberus Channels: Incentivizing Watchtowers for Bitcoin,” in Financial Cryptography (FC), 2020.
- [20] Stripe Inc. and Amazon Web Services, “Usage-Based Metering and Billing for API Services,” Technical Documentation, 2023.
- [21] Casper Association, “Casper Network 2.x: Highway Consensus and Host-Function Cost Model,” Platform Documentation, 2024.
- [22] Anthropic, “Model Context Protocol (MCP) Specification, v1.0,” 2024.