A routing layer for open inference

One OpenAI-compatible API that scores every provider on cost, latency, and reliability, routes each request to the cheapest healthy one, and fails over on its own. Inference takes the path of least resistance.

ABSTRACT

Open-model inference is sold by dozens of providers at prices that differ by multiples for the identical model, and those prices move week to week. Developers wire their code to one provider, inherit its price and its outages, and rarely re-shop. Current is a thin routing layer in front of the whole market: a single OpenAI-compatible endpoint that continuously scores providers and dispatches each request to the best one, with transparent pricing and automatic failover. This document covers the problem, the architecture, the routing score, the economics, and the path toward an open routing network.

01 A fragmented, opaque market

The same open model, say a 70B-class chat model, is served today by many providers, each with its own price, latency profile, rate limits, and reliability. Prices for the very same model routinely run 2 to 5 times higher from one provider to the next, and they shift as capacity and competition change. There is no single market price for a token. There is a spread.

Yet the typical integration is static. A developer points their SDK at one provider, hard-codes a base URL and a key, and ships. From that moment they pay that provider's price and absorb that provider's outages and rate limits, even when a cheaper, healthier provider is serving the same model a few milliseconds away. Re-shopping means code changes, new accounts, new keys, and new failure modes, so almost nobody does it. That spread is real money left on the table on every request.

02 Current in one line

Current is one OpenAI-compatible endpoint that scores every provider on cost, latency, and reliability, routes each request to the best one, and fails over automatically. One API, one bill, the cheapest route. You change a base URL and everything downstream stays the same.

The goal is to be invisible infrastructure: drop-in compatibility, transparent decisions instead of a black box, and a price that tracks the catalog floor across providers plus one small, capped routing fee.

03 Architecture

Current is a stateless gateway in front of a live registry and a routing engine.

- **OpenAI-compatible gateway.** Accepts standard /v1 chat, completions, and embeddings requests. A request names a model (or "auto") and the gateway resolves it to a concrete provider deployment. Existing OpenAI SDKs work by changing only base_url and key.
- **Capability registry.** A validated catalog of which providers serve which models, at what price, with what context window and features — overlaid with live latency and success-rate observations learned from real traffic. This is the menu the router chooses from.
- **Routing engine.** For each request it computes a score per eligible provider and dispatches to the lowest (best). The full breakdown rides on the response, so any decision can be audited.
- **Failover.** If a provider rate-limits, errors, or stalls before the stream begins, the engine retries the next-best provider. The blip never reaches the caller.
- **Metering and billing.** Usage is metered per request in micro-dollars against a prepaid balance. Cost is exact, not estimated, and the balance hard-stops at zero.

04 The routing score

Every eligible provider is scored on three normalized axes, cost, latency, and reliability, combined by tunable weights. The lowest score wins.

```
score(p) =  w_cost  · price(p)        / price_ref
          + w_lat   · latency(p)      / latency_ref
          + w_rel   · (1 − reliability(p))

route → argmin_p  score(p)        # lowest cost-of-choice wins
```

Weights default to cost-first but are tunable per API key or per request, so a latency-sensitive workload can pay for speed while a batch job optimizes purely for price. Because the breakdown comes back with each response, routing is explainable rather than magic. You can always see why a provider was chosen and what the runner-up would have cost.

05 Economics

You pay the provider's price plus a small routing fee: a percentage of provider cost, capped per million tokens, so cheap open models stay cheap and expensive ones are bounded. No subscriptions, no marked-up token prices. Because Current routes to the cheapest healthy channel and the spread between providers is wide, every response reports exactly what it saved versus the priciest eligible channel (x_current.savings) — the claim is measured per request, not asserted.

Balances are prepaid and topped up by card (stablecoin top-ups are planned). Spend is exact and capped by the balance. The fee funds the routing layer itself, so the incentive is aligned with finding you the cheapest route, not with steering volume to any particular provider.

06 Reliability and failover

Provider health is learned from live traffic — every dispatch outcome feeds a rolling latency and success-rate estimate plus a per-provider circuit breaker — and folded into the score, so a degrading provider is deprioritized before it hard-fails. When a dispatch does fail before streaming starts, a rate limit, a 5xx, a timeout, or a provider-side configuration fault, the engine reroutes to the next-best provider within the same request. Uptime is the product: the caller sees one reliable endpoint, not the churn behind it.

07 Privacy and security

API keys are hashed at rest and shown only once. Current logs identifiers, latency, and cost, the data needed to route and bill, and not prompt or completion content. Compatibility is drop-in, so your secrets and data paths stay under your control. Only the base URL changes.

08 Toward a routing network

The protocol grows in phases, from an aggregator today to an open routing layer.

- **Phase 1, one channel (shipping).** A single OpenAI-compatible endpoint over every provider, a live capability registry, and routing you can see through.
- **Phase 2, routing that learns.** Continuous cost optimization, failover that reroutes around real provider health, and per-request analytics.
- **Phase 3, open the network.** Bring your own provider, list community providers, and earn a share of the traffic you carry.
- **Phase 4, the network layer.** The default routing layer for open-model inference, with decentralized onboarding and on-chain reputation for every channel.

09 Conclusion

Inference is becoming a commodity, and commodities are won by the layer that connects demand to the cheapest reliable supply, not by owning the supply. Current is that layer: one endpoint, transparent routing, the live price floor, and automatic failover. Integrate once and let every request take the path of least resistance.