Zero Latency (0.lat)

Distributed AI inference network at the telecom/edge layer

Explainer · June 2, 2026

Distributed AI inference network that routes workloads to low-latency edge capacity across carriers and clouds

COMPANY	Zero Latency (0.lat)	OWNERSHIP	Private
CEO	Not publicly disclosed		
HEADQUARTERS	Not publicly disclosed (operates as a remote-first/Internet-native company)		

ABSTRACT

Zero Latency (0.lat) (0.lat) is building a distributed AI inference network that sits on top of telecom and cloud infrastructure to dispatch model inference workloads to edge locations that best satisfy latency, data-gravity, and burst capacity requirements. Rather than running all inference in a few hyperscale regions, it treats global edge capacity as a single logical fabric and routes requests based on real-time network and compute conditions. The distinctive bet is that inference, especially for latency-sensitive or data-resident applications, will fragment toward the edge and that a neutral routing layer with programmatic SLAs across carriers will be valuable. If it works, this enables developers to specify constraints (e.g., "<30 ms latency, stay in-country") and let 0.lat place workloads automatically, while operators can monetize underutilized edge GPUs. Key implications: 0.lat could become a traffic broker for AI inference at the edge, a leverage point between clouds and telcos, and a dependency for any "real-time AI" product that cannot tolerate hyperscaler-only latency.

Keywords: AI inference; edge compute; low latency; telecom; distributed systems; data gravity; AI infrastructure

1. Snapshot

Zero Latency (0.lat) (0.lat) operates a **distributed AI inference network**, routing workloads to edge compute nodes across telco and cloud infrastructure based on latency, data-gravity, and burst constraints. The website presents the company as an infrastructure provider rather than a model lab or an end-user application. Founding year, corporate domicile, funding history, valuation, headcount, and founders are not publicly disclosed or readily verifiable from open sources. There is also no public list of investors or board members. The product is positioned as a programmable network for inference placement with "sub-50 ms global inference" claims at the marketing layer, but without public benchmarks or customer logos. Key unknowns for diligence are: revenue scale and growth, concrete customer segments and number of active customers, capacity under management (nodes, regions, GPU types), and the specifics of commercial agreements with carriers and cloud providers.

2. Thesis: Why This Company, Why Now

The core bet is that **AI inference will increasingly need to run close to users and data**, and that neither hyperscalers nor telcos alone provide a developer-friendly abstraction for that. Foundation models are growing, but the latency requirement for many emerging workloads is trending downward: real-time copilots, in-game agents, AR/VR, robotics, and industrial control often need sub-100 ms end-to-end, sometimes below 50 ms, including network and application overhead. At the same time, sensitive data is either regulated to stay in country or operationally constrained to stay in specific facilities ("data gravity"). Telcos and CDNs have edge footprints, but the APIs are fragmented, GPU deployment is spotty, and AI teams do not want to negotiate per-carrier deals.

0.lat's thesis is that a neutral inference-routing layer, with programmable SLAs over a fabric of edge locations and clouds, becomes valuable as AI shifts from a batch/analytics model to an always-on, interactive service model. The reachable market is not "all AI" but the subset of inference spend that is latency-sensitive or data-resident and willing to pay a premium for low latency and locality guarantees.

3. The Core Idea in Plain English

0.lat wants to be **the "anycast router" for AI inference**. Today, you either pick a region in a hyperscale cloud or a specific edge provider and hope the latency, data residency, and capacity work out. With 0.lat you would, in principle, tell the network "run this model where the user is, keep data in jurisdiction X, and stay under Y ms", and the system chooses an appropriate edge location and hardware pool. A useful analogy is DNS anycast for web traffic: you hit one address, but your request is answered by the nearest or best-performing node. Old world: you deploy and manage N regional clusters or custom telco deals to get acceptable latency. New world: you integrate with a single inference routing layer that abstracts the heterogeneity of edge locations and underlying providers, while still giving you control over performance and data locality policies.

4. The Technical Space

The technical problem here is **distributed AI inference placement and routing**: deciding where to run each inference request across a heterogeneous, geographically distributed set of compute nodes, subject to constraints like latency, locality, cost, and capacity. Standard approaches today include:

- **Single-cloud regional deployment.** Most teams deploy models in a few cloud regions and rely on cloud load balancers and CDNs; this is simple but often yields >100 ms latency for global users and poor control over data locality.
- **Multi-region, multi-cloud hand-roll.** Larger players build their own mesh of regions and sometimes multiple clouds, with internal traffic routing and observability; this is powerful but requires significant SRE, networking, and capacity-planning investment.
- **CDN/edge function extensions.** Some use provider-specific edge runtimes (Cloudflare Workers, Fastly Compute@Edge, Lambda@Edge) and, increasingly, their AI APIs; this gives better latency but is fragmented and often constrained by provider-specific runtimes and model options.

"What good looks like" in this category: low and predictable tail latency (p95/p99), strict data residency compliance (per country or per facility), dynamic placement based on real-time network conditions and demand spikes, high availability across provider failures, and a simple developer model (few concepts, clean APIs, clear SLAs). The hard part is less the ML and more the distributed systems and networking: topology awareness, routing algorithms, failure handling, and capacity orchestration across entities you do not fully control.

5. How Their Technology Works (and What's Proprietary)

Public information is sparse, but the website implies 0.lat has built a **logical network layer** over existing edge and cloud compute, focusing on inference workloads. From that, a reasonable, clearly marked inference is that the stack likely has several components:

- **Global control plane.** A central service that maintains a view of edge nodes, their location, hardware type (GPU/CPU, memory), utilization, and health, plus network performance metrics between user populations and nodes.[inference]
- **Routing and placement engine.** Given a request annotated with latency or locality requirements, an engine selects a target node based on current metrics and policies (e.g., minimize p95 latency while staying in a given jurisdiction or on a given carrier's footprint).[inference] This is the "anycast for inference" logic.
- **Data-plane integration.** Gateways or SDKs that intercept inference calls and forward them into the network, plus agent software on edge nodes to accept and execute workloads (containers, serverless functions, or model runtimes).[inference]
- **Policy and SLA layer.** APIs that let customers express requirements (latency bounds, geofencing, preferred providers) and receive observability (latency distributions, error rates) suitable for contractual SLAs.[inference]

What might be proprietary: the specific routing algorithms tuned for AI inference, the telemetry and metrics pipeline tailored to telco-grade networks, and any proprietary partnerships/integrations that expose otherwise hard-to-access edge capacity.[inference] However, much of the base stack—control plane, routing logic, agent software—can be replicated by a competent team using standard distributed-systems tooling. Without visibility into code, patents, or published architectures, the technical moat appears more in execution and integrations than in deep algorithmic IP.

6. Business and Go-to-Market

The business model is not explicitly documented publicly, but the positioning suggests an **infrastructure/usage-based model**: charging per inference, per compute time, or via bandwidth plus SLA tiers.[inference] 0.lat sells to AI application developers, platforms, or enterprises that need low latency and locality control but do not want to build and operate their own global edge mesh. The go-to-market motion looks B2B and likely sales-led or sales-assisted, since integrating into production inference paths, negotiating SLAs, and potentially addressing compliance often requires solution engineering.[inference]

There are no public customer logos, pricing pages, or case studies. That lack of signals makes it hard to assess revenue scale or product-market fit. A plausible adoption pattern is "land" via a specific low-latency use case (e.g., real-time conversational agent, gaming, fintech personalization) and "expand" as more traffic or additional models move onto the network.[inference] Gross margins will depend heavily on how wholesale capacity is procured from carriers and clouds; if 0.lat acts as a broker on top of on-demand capacity, margins may be thinner than if it can secure discounted, long-term edge GPU capacity.

7. Competitive Landscape and Moats

Direct competitors. The closest analogue is **Fermyon-style or Cloudflare/Cloudflare Workers AI-style edge AI platforms**, plus any startup explicitly advertising "distributed inference routing" or "AI anycast." Public search does not surface a named startup with an identical positioning, so the nearest competitors are adjacent: Cloudflare's AI/Workers stack, Fastly Compute@Edge with AI integrations, Akamai's edge compute/Cloud Wrapper, and hyperscaler offerings like AWS Local Zones with their own traffic steering.[inference]

Cloudflare is the single most threatening rival: it has a vast global edge footprint, built-in developer adoption, and is already moving into AI inference at the edge. Where 0.lat might win versus Cloudflare: neutrality across carriers and clouds, more flexible model runtimes, and more fine-grained data residency across telco-owned facilities rather than only Cloudflare POPs.[inference] Where it likely loses: brand, existing developer base, security/performance add-ons, and horizontal product breadth.

Akamai and Fastly offer edge compute and could add more AI-specific routing; they win on existing traffic volumes and enterprise sales presence, but may be slower to build AI-specific control planes. Hyperscalers could extend their global load-balancing and Local Zones/Wavelength/Outposts, bundling inference placement into their ecosystems.

Moats.

› **Integration and distribution moat.** To the extent 0.lat embeds into inference SDKs, frameworks, or managed AI platforms, switching costs could be moderate; removing it requires re-wiring routing and observability.[inference]

› **Partnership/footprint moat.** Exclusive or hard-won integrations with telcos and edge operators, especially where they expose private capacity or QoS features not available via public APIs, could be defensible if they exist.[inference]

› **Operational data moat (limited for now).** Over time, telemetry data on network performance, failure patterns, and workload behavior across a large edge fabric could improve routing algorithms and SLAs, but this requires significant scale before it becomes meaningful.[inference]

Overall, moats look nascent and execution-dependent; a well-funded competitor with existing edge footprint could replicate the high-level concept.

8. Risks and Open Questions

The main risks are around **commercial traction, platform competition, and actual defensibility**.

Technical/defensibility:

- How sophisticated is the routing engine in practice (e.g., topology-aware, congestion-aware, model-specific), and how hard would it be for a Cloudflare or hyperscaler to replicate?
- Does the system handle multi-tenant security, noisy neighbors, and cross-provider failures to a level suitable for regulated workloads?

Commercial/distribution:

- Are there production customers at scale, and what concrete latency or compliance benefits did they achieve versus a hyperscaler baseline?
- What is the pricing model and unit economics when reselling or arbitraging third-party capacity?

Platform/dependency:

- How reliant is 0.lat on a small number of telco or cloud partners, and what happens if a key provider changes terms or launches a competing routing layer?
- If hyperscalers push AI inference deeper into their own edge footprints, does 0.lat get disintermediated or can it still add value as a cross-provider policy layer?

Additional open question:

- How does 0.lat address privacy and regulatory concerns when routing across jurisdictions, and what certifications or audits back those claims?

9. Bottom Line

0.lat is making a credible bet that **latency-sensitive AI inference needs a neutral, programmable network layer that spans telcos and clouds**, not just better single-cloud regions. The biggest swing factor is whether it can secure real production workloads and partnerships before hyperscalers and CDNs fully occupy the “AI at the edge” routing space. The key thing to watch is concrete evidence of traction: named customers, published latency/availability improvements over baselines, and any disclosed telco or cloud integrations that suggest 0.lat is becoming part of the default path for low-latency AI deployments.

10. For the Nerds

The interesting technical question is how deeply 0.lat models **network topology and data gravity** in its placement decisions. A naïve anycast strategy that just picks “closest POP” based on static geography is insufficient once you factor in dynamic congestion, BGP quirks, cross-cloud links, and application-level behavior. A production-grade engine would need continuous active probing, passive telemetry ingestion, and possibly ML-based prediction of path performance, then combine that with constraints like “must stay in EU” or “must run on GPU class ≥ X” at per-request granularity.[inference]

A second dimension is **stateful workloads and caching**. Many AI applications benefit from warm model weights, KV caches, or user-specific context. Routing every request independently to the globally “best” node can explode cache miss rates and hurt effective latency. A sophisticated system will cluster users and workloads, trade off instantaneous latency vs. cache locality, and perhaps pre-warm or migrate model instances based on predicted demand.[inference] Finally, observability and debugging in such a network are non-trivial: developers need request-level traces that cross providers, plus clear attribution when SLAs are violated. The quality of 0.lat’s telemetry, tracing, and policy debugging interfaces will heavily influence developer adoption, even if the underlying routing algorithms are strong.