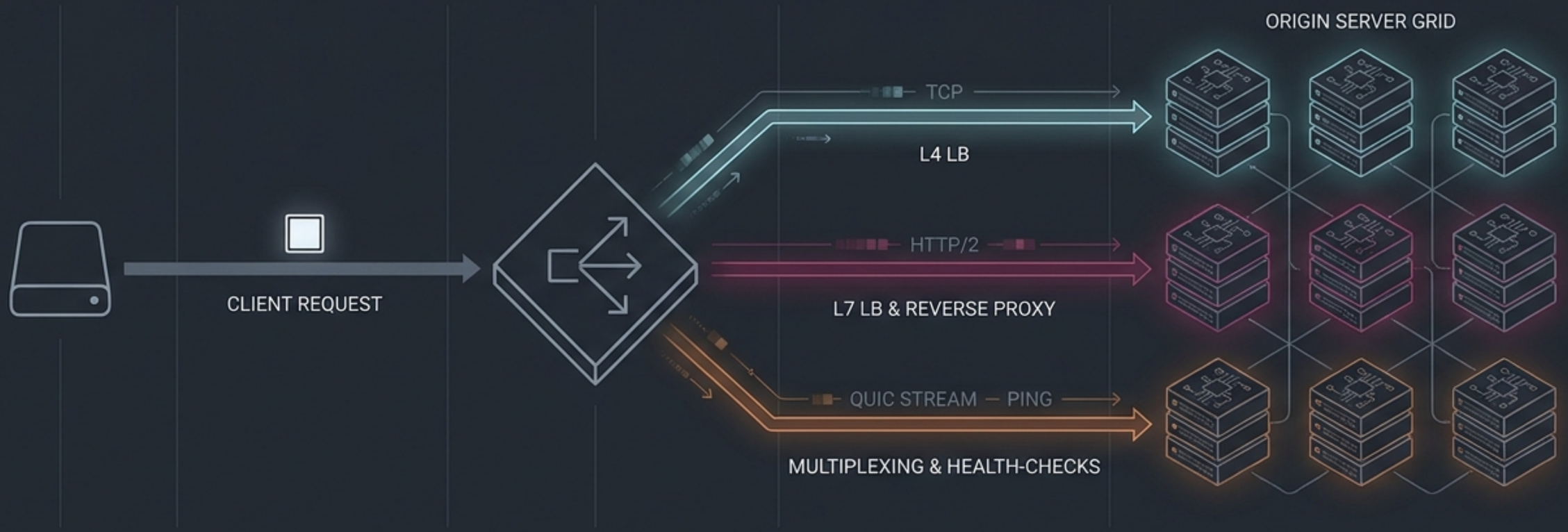
Web Architecture — S02E04

Reverse Proxies, L4 vs. L7 Load Balancing, HTTP/2 & QUIC Multiplexing, and Health-Check Engineering



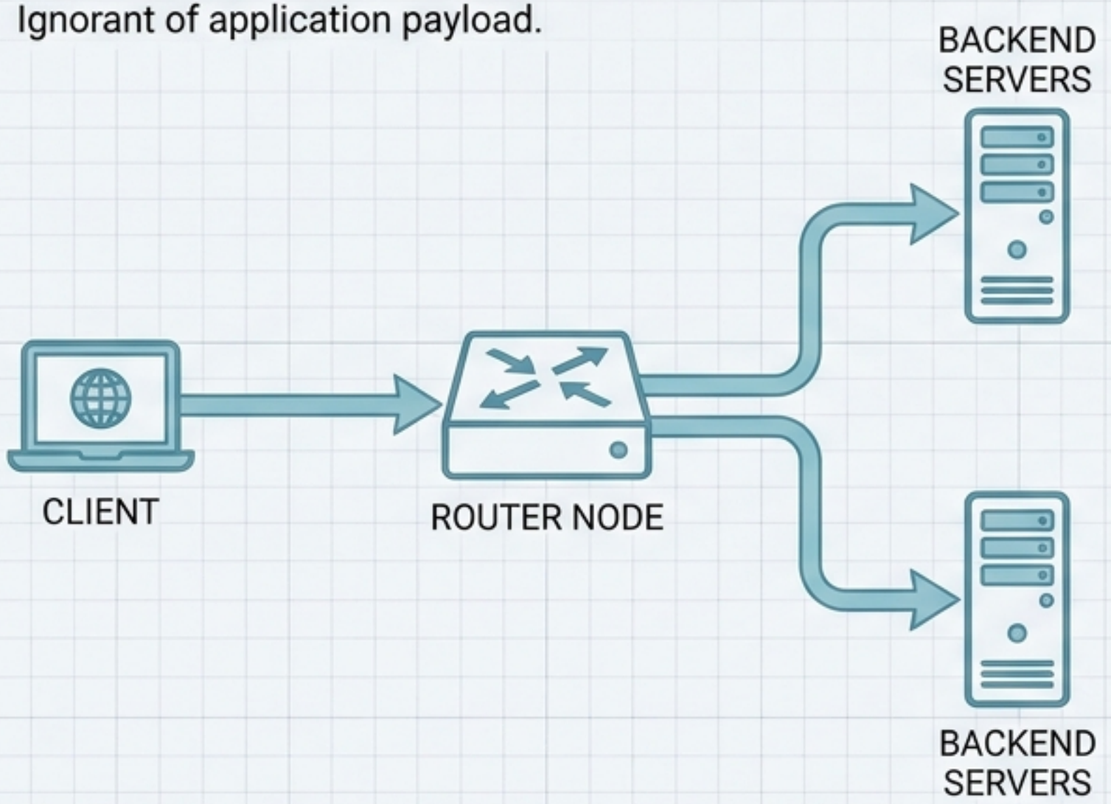
A rigorous datapath survival guide for Backend, SRE, and Platform Engineers.

Gemini Notebook

THE ARCHITECTURAL SPLIT: TRANSPORT VS. APPLICATION

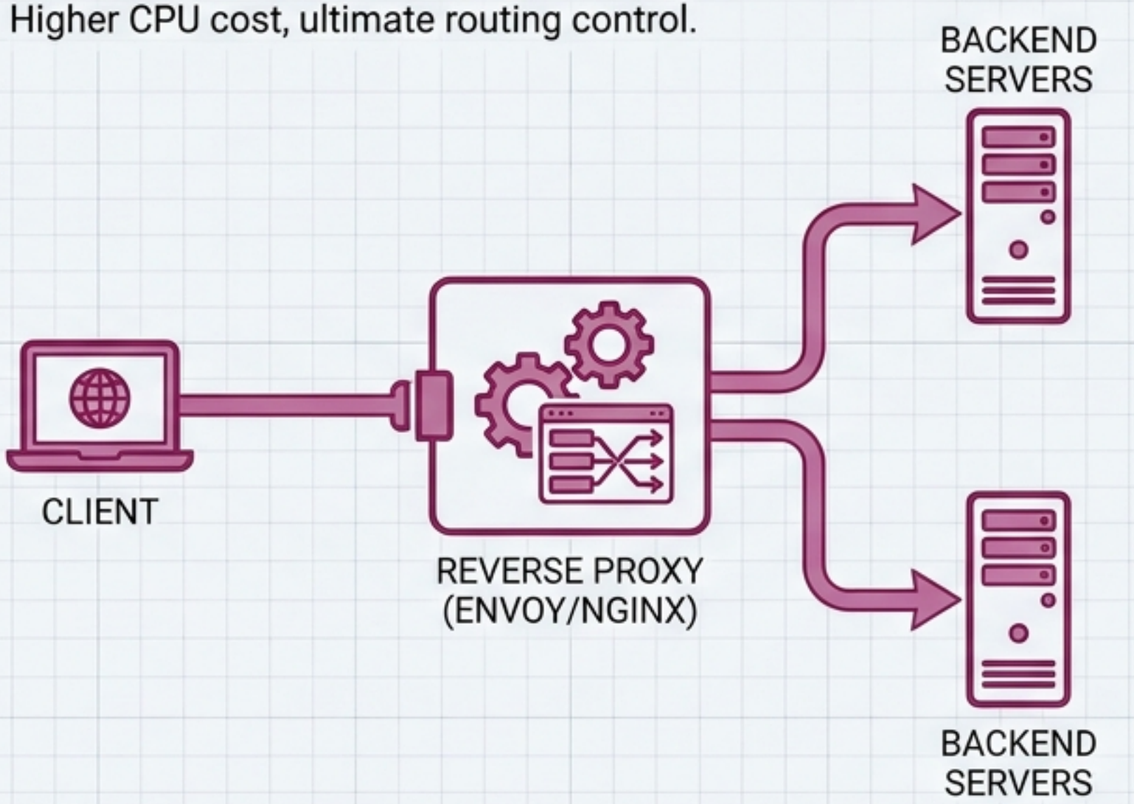
LAYER 4 (TRANSPORT)

Blind packet forwarding.
Extremely high throughput, low CPU cost.
Ignorant of application payload.



LAYER 7 (APPLICATION)

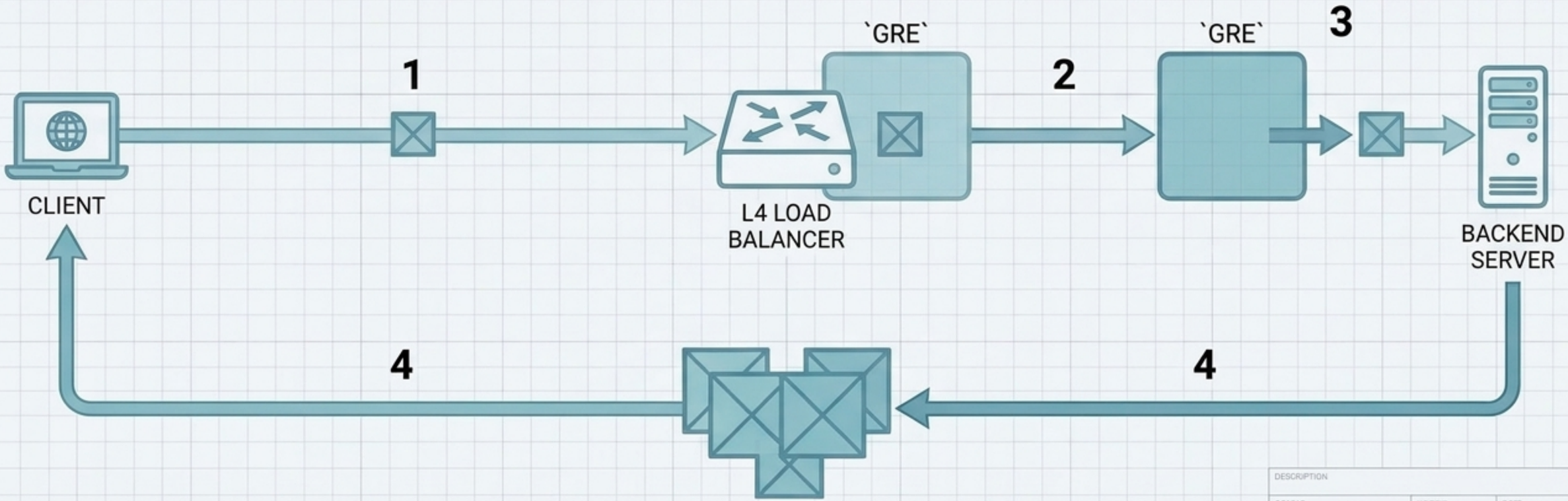
Terminating reverse proxy.
Inspects HTTP headers, manages multiplexed streams.
Higher CPU cost, ultimate routing control.



Gemini Notebook

LAYER 4 MECHANICS: GRE ENCAPSULATION & DIRECT SERVER RETURN (DSR)

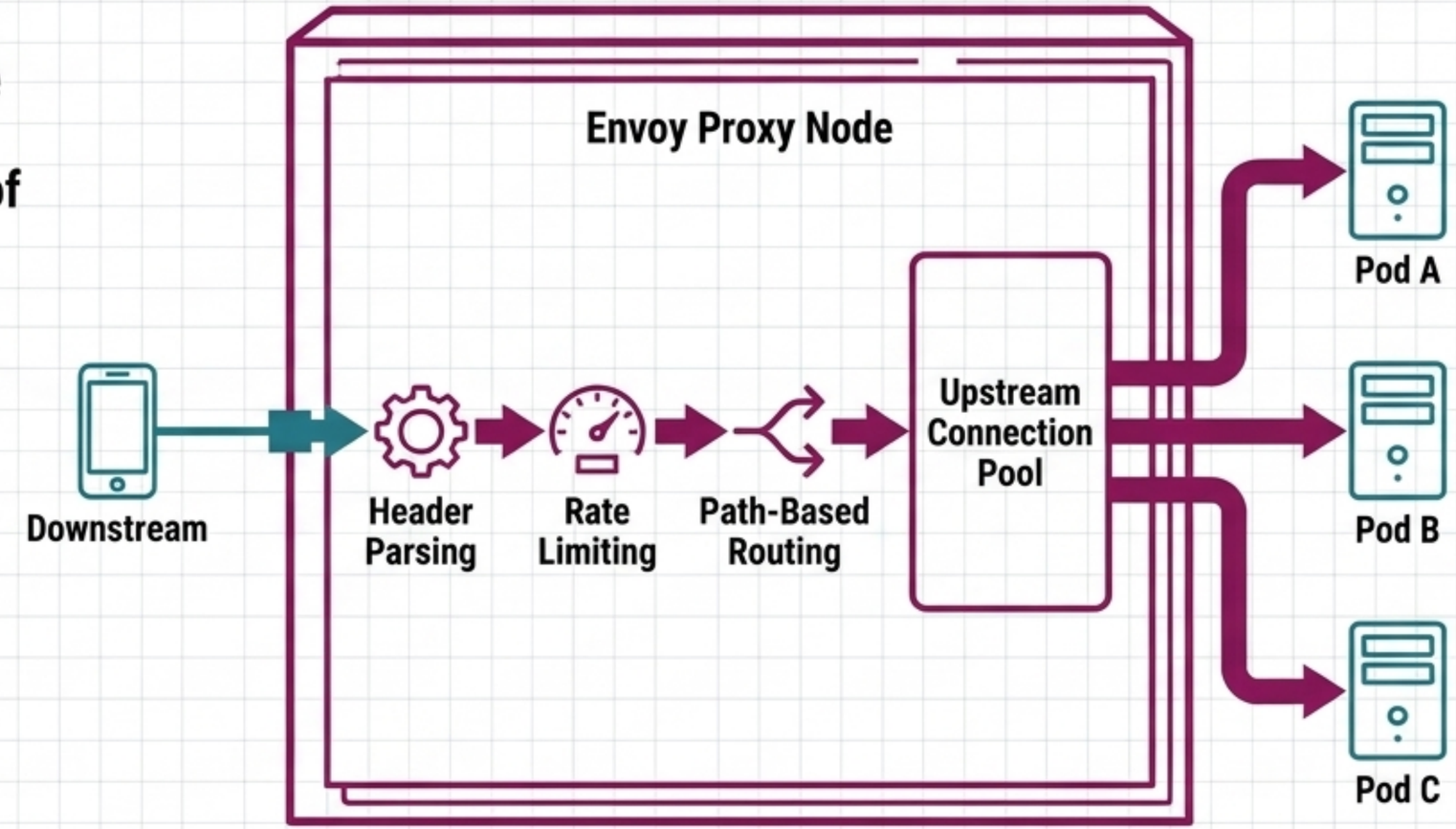
- **Mechanism**: L4 does not terminate the TCP connection. It **encapsulates the packet** (GRE) or **mutates it** (NAT).
- **The DSR Advantage**: Because response payloads are exponentially larger than requests, **backends return data directly to the client**. The **LB is bypassed on the return trip,** preventing a massive **network bottleneck**.
- **Ideal Use Case**: Stateful TCP/UDP, extreme throughput, cost-efficiency.



Gemini Notebook

LAYER 7 MECHANICS: THE TERMINATING PROXY

- **Mechanism: The proxy acts as an active middleman, terminating the downstream client socket and managing a separate, pooled set of upstream backend sockets.**
- **The Trade-off: Pays the CPU and latency price of terminating TLS and managing two separate TCP states.**
- **The Reward: Unlocks HTTP semantic inspection, enabling path-based routing, header rewriting, retries, and granular rate-limiting.**



Gemini Notebook

THE EVOLUTION OF CONNECTION MANAGEMENT

HTTP/1.1 (THE TRAFFIC JAM)

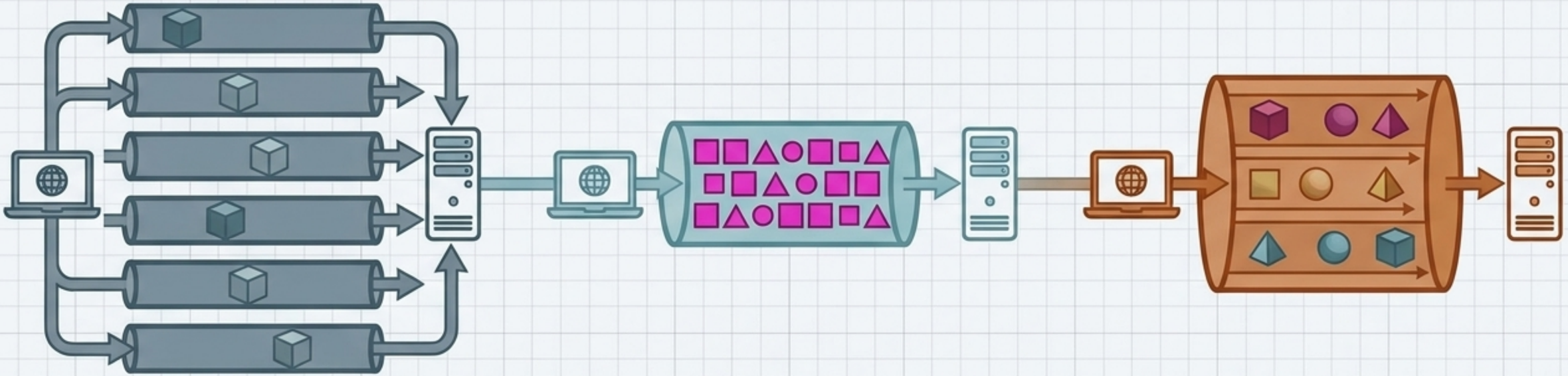
Sequential, blocking requests requiring multiple expensive TCP sockets.

HTTP/2 (RFC 9113) (THE SINGLE PIPE)

Binary framing multiplexes concurrent streams over a single TCP connection.

HTTP/3 (RFC 9114) (THE UNIFIED HIGHWAY)

QUIC over UDP natively integrates TLS 1.3 and eliminates kernel-level dependencies.



Multiple TCP Connections

Multiplexed Streams (Binary Frames) over One TCP

QUIC over UDP, Native TLS 1.3, Independent Lanes

Gemini Notebook

HTTP/1.1 VS. HTTP/2: EXHAUSTION VS. HOL BLOCKING

HTTP/1.1 PATHOLOGY

Bypasses application head-of-line (HOL) blocking by opening multiple TCP connections, leading directly to socket exhaustion at the edge.



HTTP/2 PATHOLOGY

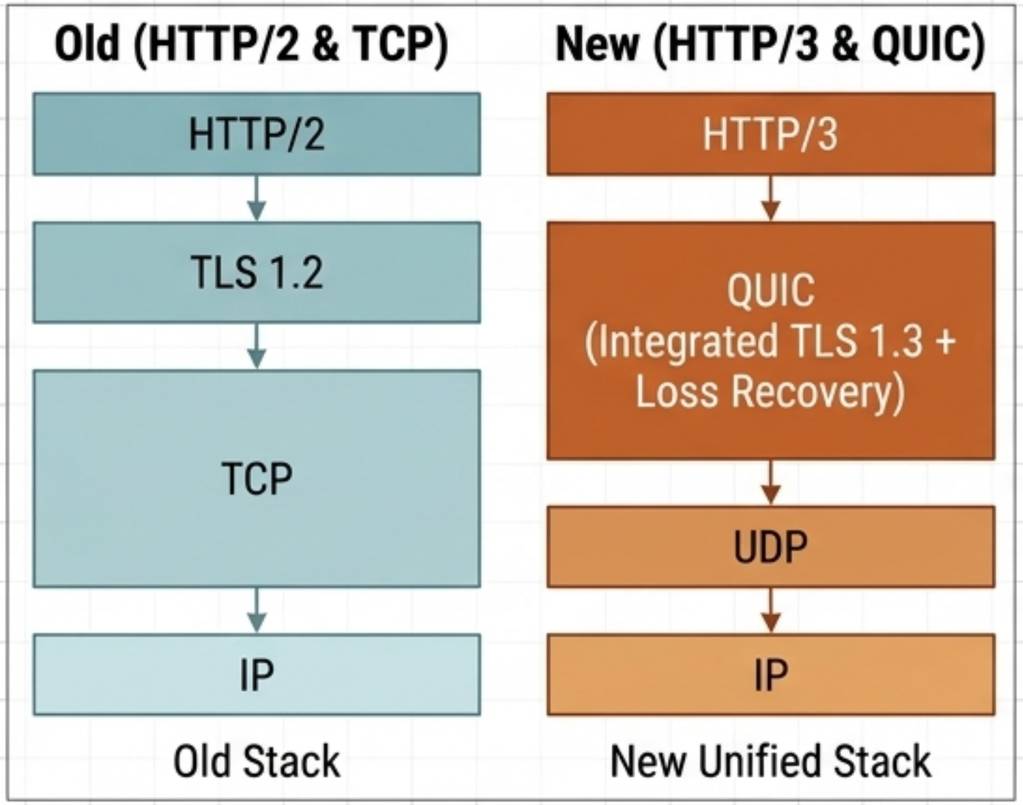
Solves connection exhaustion via stream multiplexing, but introduces OS-level Kernel TCP Head-of-Line Blocking—a single dropped packet on the wire forces the OS to stall all multiplexed streams until retransmission.

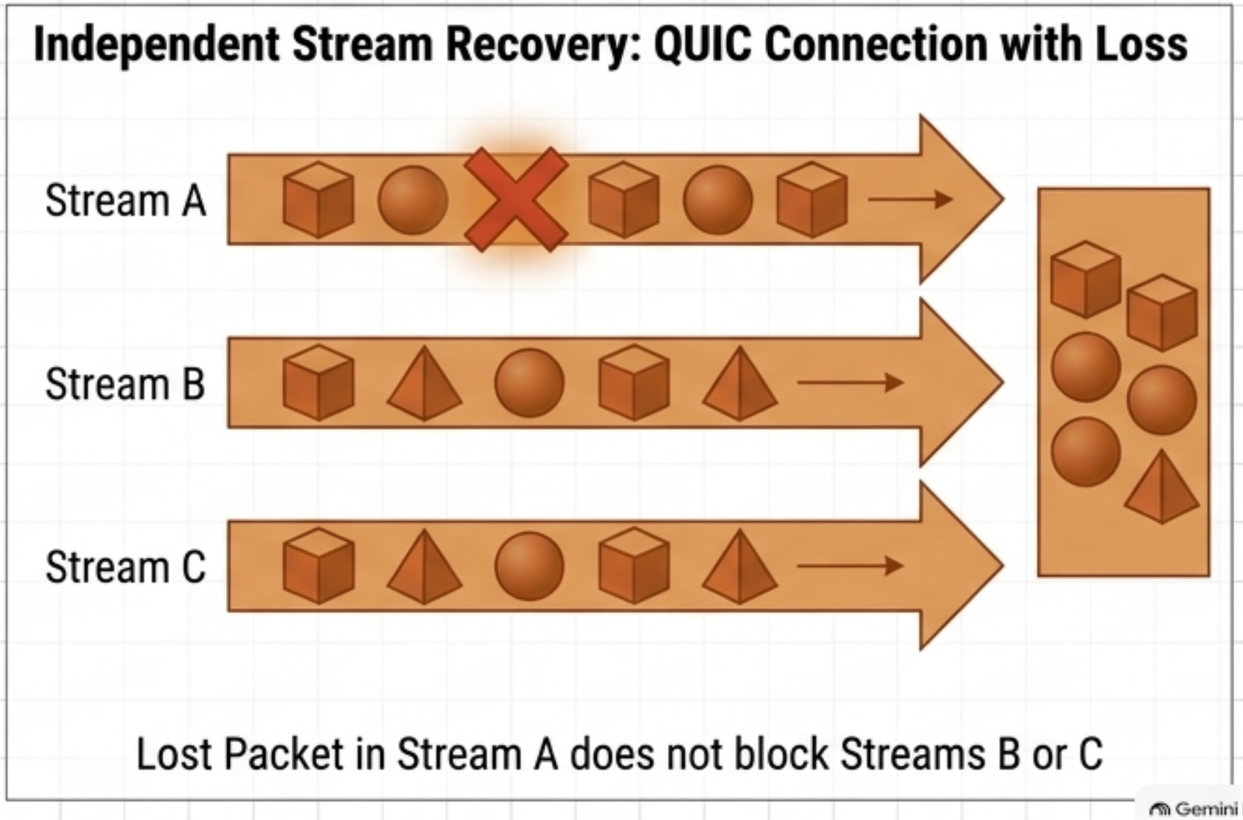


Gemini Notebook

HTTP/3 & QUIC (RFC 9000/9114)

- **UDP-Based Transport:** Bypasses rigid kernel TCP limitations.
- **Independent Stream Loss Recovery:** Eliminates head-of-line blocking natively; a lost packet only delays the specific stream it belongs to.
- **0-RTT Handshakes:** Integrated TLS 1.3 connection setup reduces latency to zero for returning clients.





Gemini Notebook

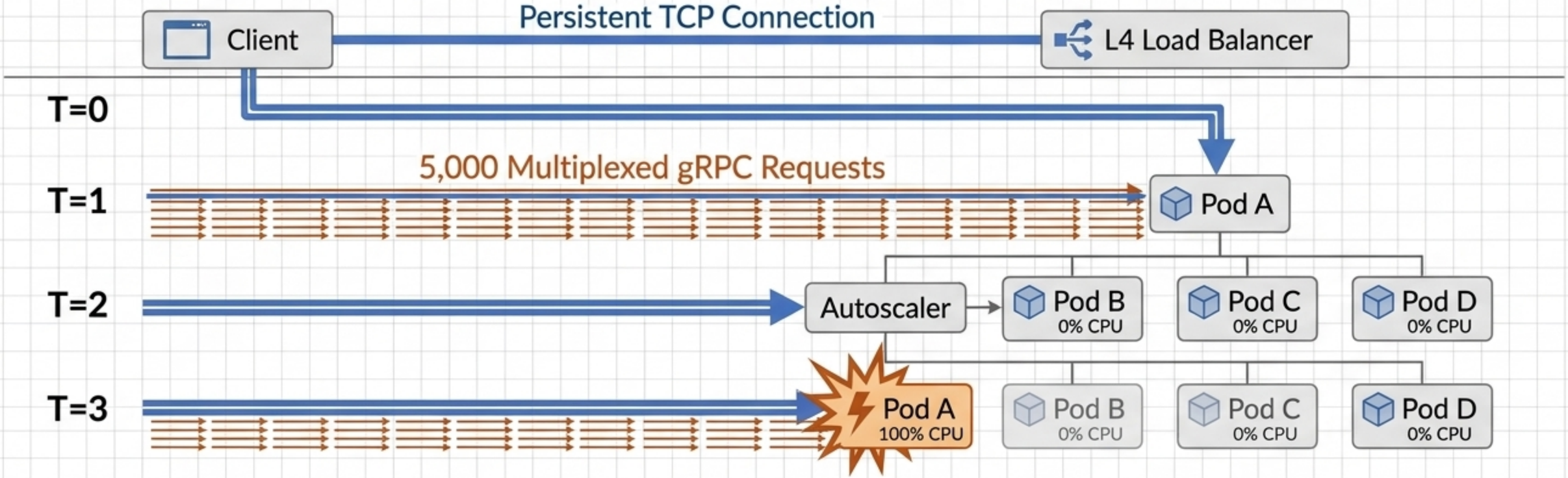
Editorial Datapath Blueprint

DANGER

The gRPC / HTTP/2 L4 Trap (The Pathology)

The Trap: Layer 4 load balancers distribute connections, not requests.

The Result: Because HTTP/2 and gRPC heavily multiplex over persistent, long-lived sockets, the **original connection remains permanently pinned** to Pod A. Autoscaled pods remain idle while the original pod is crushed.



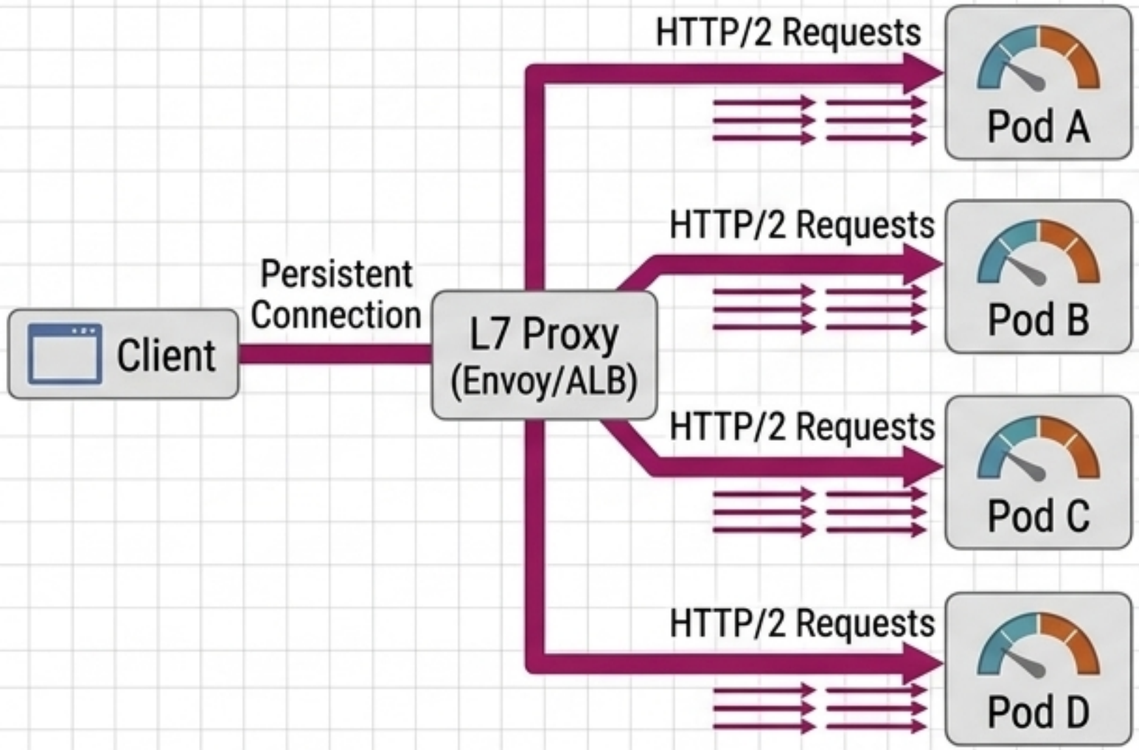
Gemini Notebook

Editorial Datapath Blueprint

Escaping the Connection-Pinning Trap

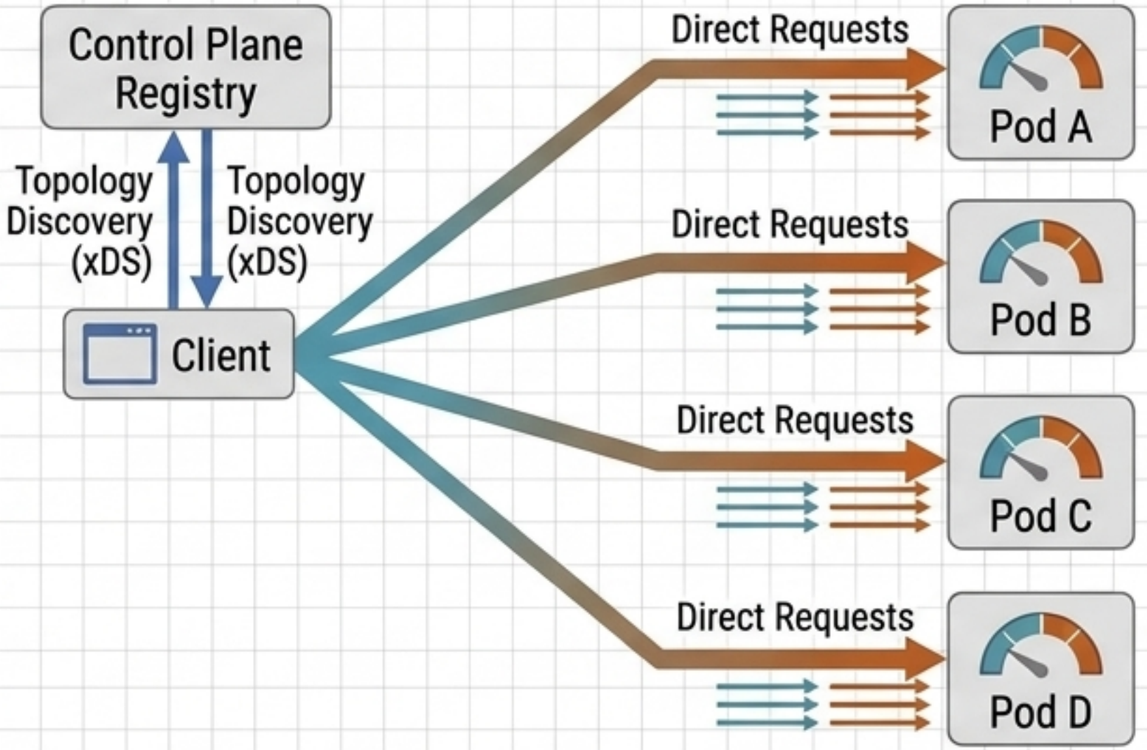
Fix 1: L7 Proxy (Envoy/ALB)

The proxy terminates the persistent TCP connection and balances the individual HTTP/2 streams across the upstream pool.



Fix 2: Client-Side Load Balancing (xDS)

The client uses a control plane to natively discover topology changes and balances requests client-side without an intermediary proxy hop.

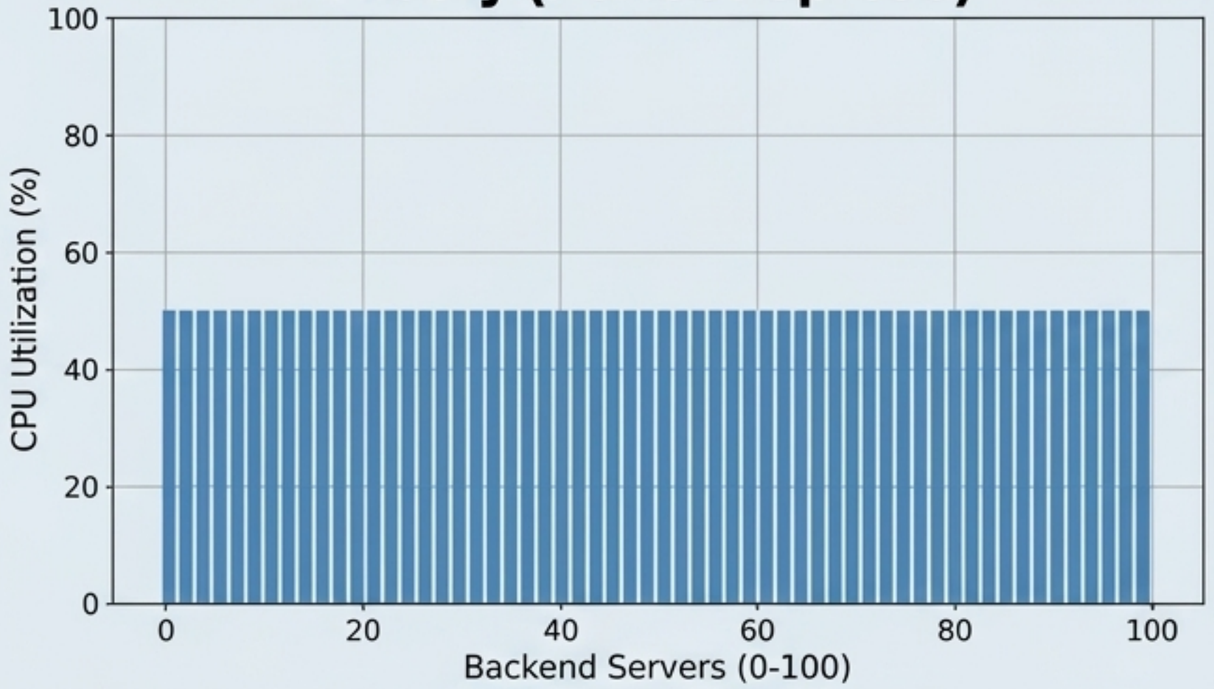


Gemini Notebook

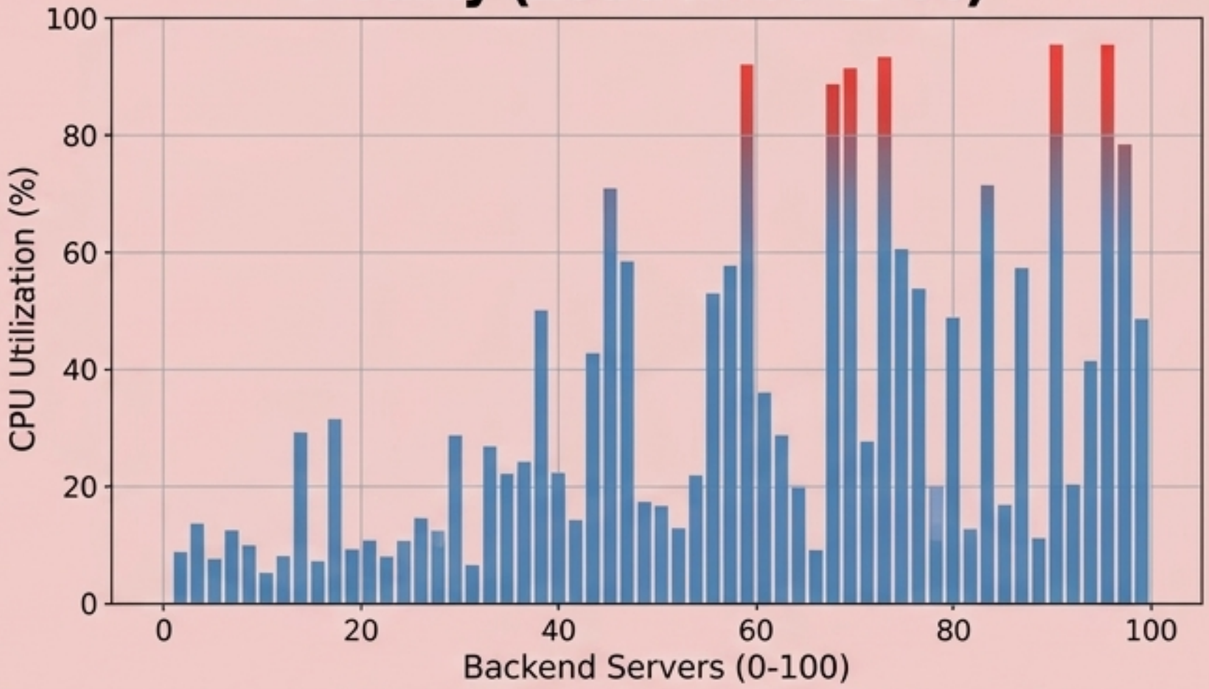
Editorial Datapath Blueprint

The Baseline: Round Robin & Its Pitfalls

Theory (Perfect Spread)



Reality (Load Imbalance)



- **The Assumption**: Round Robin assumes all requests cost the same and all servers have identical capacity.
- **The Reality**: Varying query costs (e.g., a 15ms cache hit vs. a 10-second DB scan) and antagonistic noisy neighbors on shared VMs lead to massive load imbalances.
- **The Cost**: Up to a 2x spread in CPU utilization between the most and least loaded tasks, resulting in vastly over-provisioned, wasted capacity.

Gemini Notebook

Editorial Datapath Blueprint

Least Request, P2C, & The Sinkhole Hazard

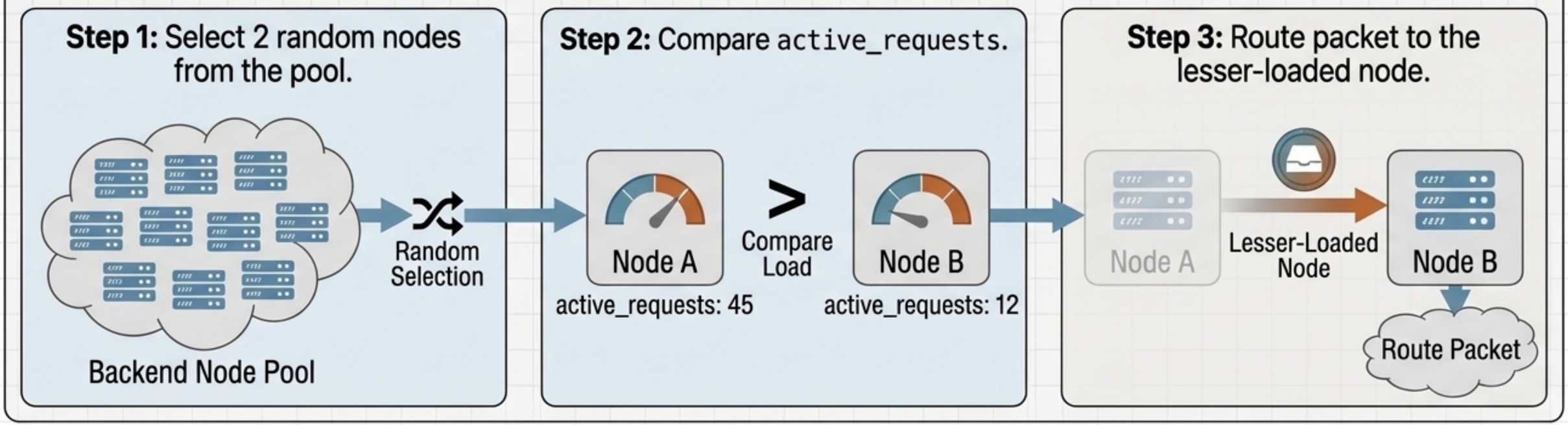
The Sinkhole Hazard (Least Request)

If a severely unhealthy node instantly returns 500 Internal Error, its `active_requests` count drops to zero. Global 'Least Request' algorithms will aggressively funnel all traffic to this failing node, creating a sinkhole.

The P2C Solution

Power of Two Random Choices operates at O(1) efficiency. It avoids the 'thundering herd' behavior of global state scans, mathematically approaching optimal distribution while gracefully handling stale telemetry.

P2C Algorithm Flowchart

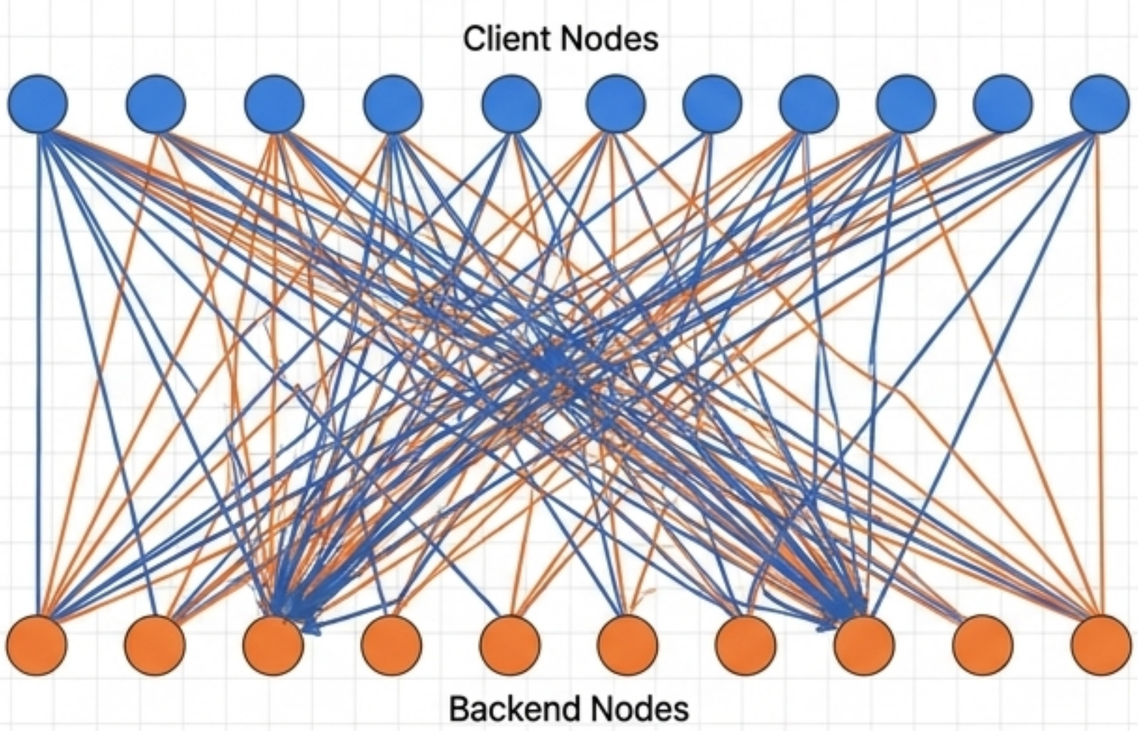


Gemini Notebook

Editorial Datapath Blueprint

Subsetting: Limiting the Connection Pool

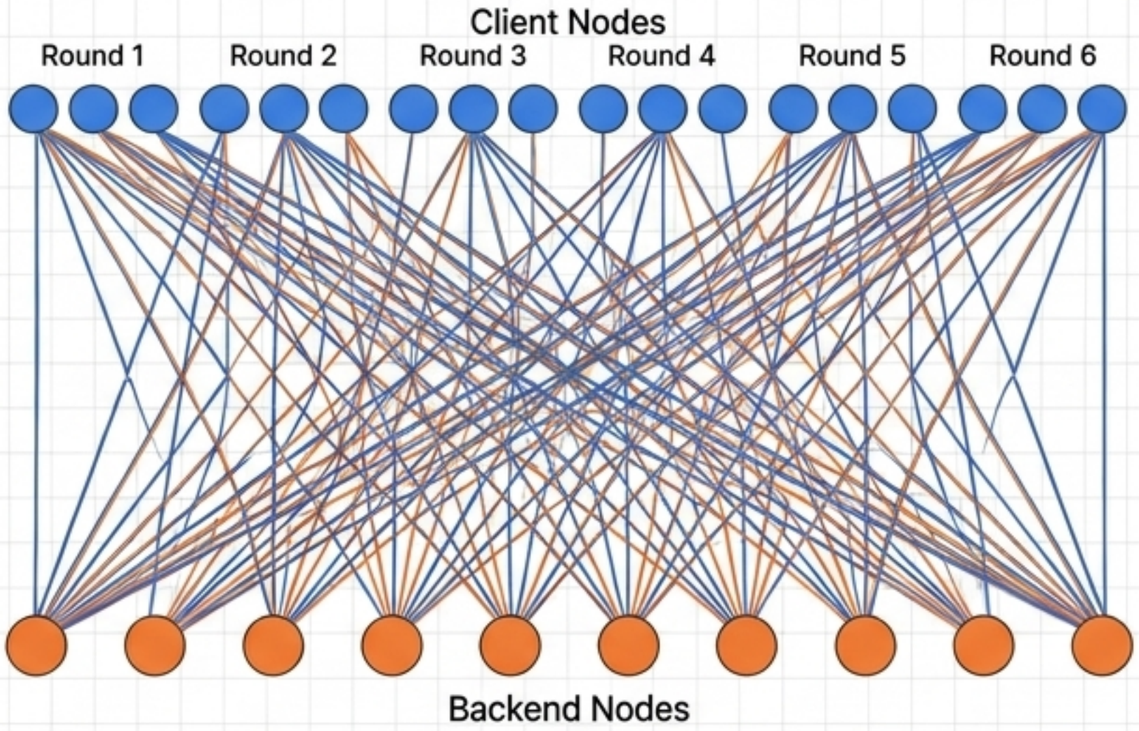
Random Subsetting



The Problem: N clients connecting to M backends creates N×M connections. At scale, the overhead of health checks and idle TCP sockets wastes massive CPU/Memory.

The Fix: Clients connect to a limited "subset" of backends (e.g., 20-100).

Deterministic Subsetting



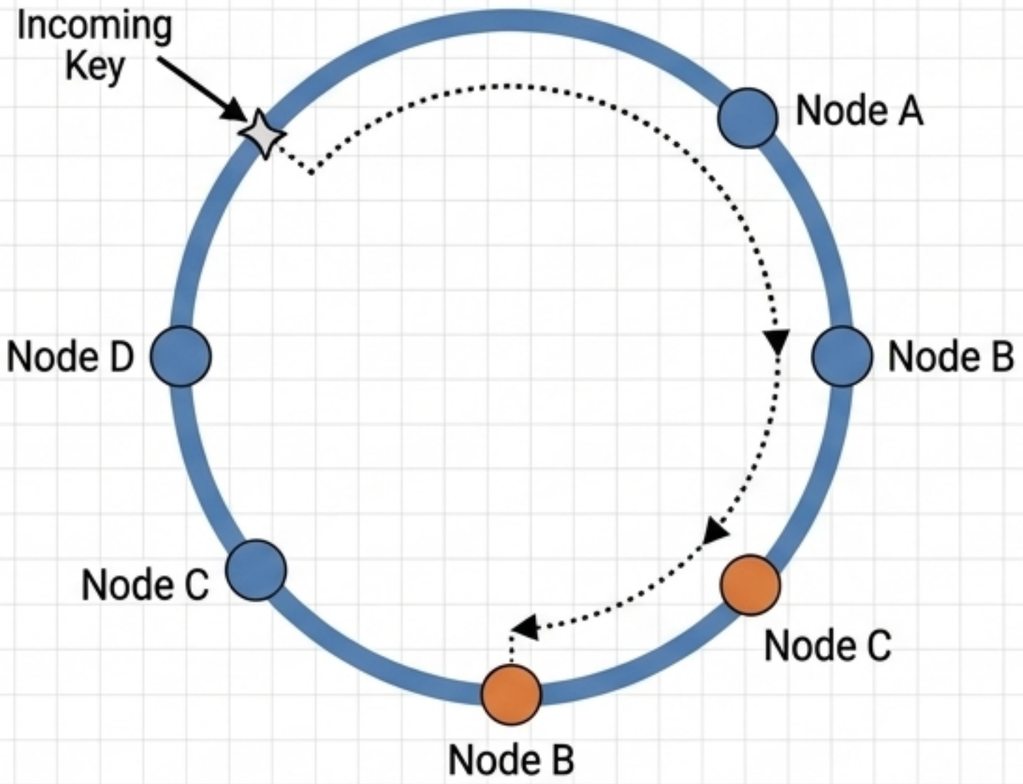
Deterministic Over Random: Random shuffling creates severe load imbalances. Deterministic subsetting mathematically guarantees uniform connection distribution across the fleet, allowing seamless rolling restarts.

Gemini Notebook

Editorial Datapath Blueprint

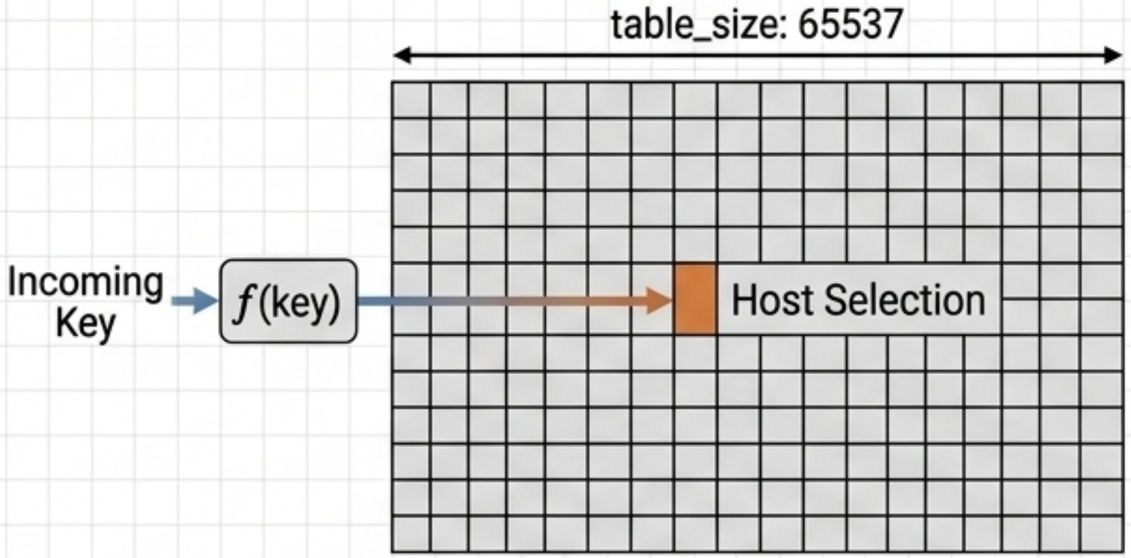
Consistent Hashing: Ring Hash vs. Maglev

Ring Hash (Ketama)



Ring Hash: Maps hosts to a continuum ring. Highly effective for cache locality, but lookups are slow and rolling deployments heavily disrupt key mappings.

Maglev Table



Maglev (Google): Utilizes a fixed 65537-entry lookup table.

Maglev Advantages:

- Yields 10x faster lookups and 5x faster host selection than Ring Hash.
- Drastically reduces key disruption when pods are added or removed.
- The definitive choice for distributed caching (e.g., Redis clusters).

Gemini Notebook

Editorial Datapath Blueprint

Synthesis: The Load-Balancing Algorithm Matrix

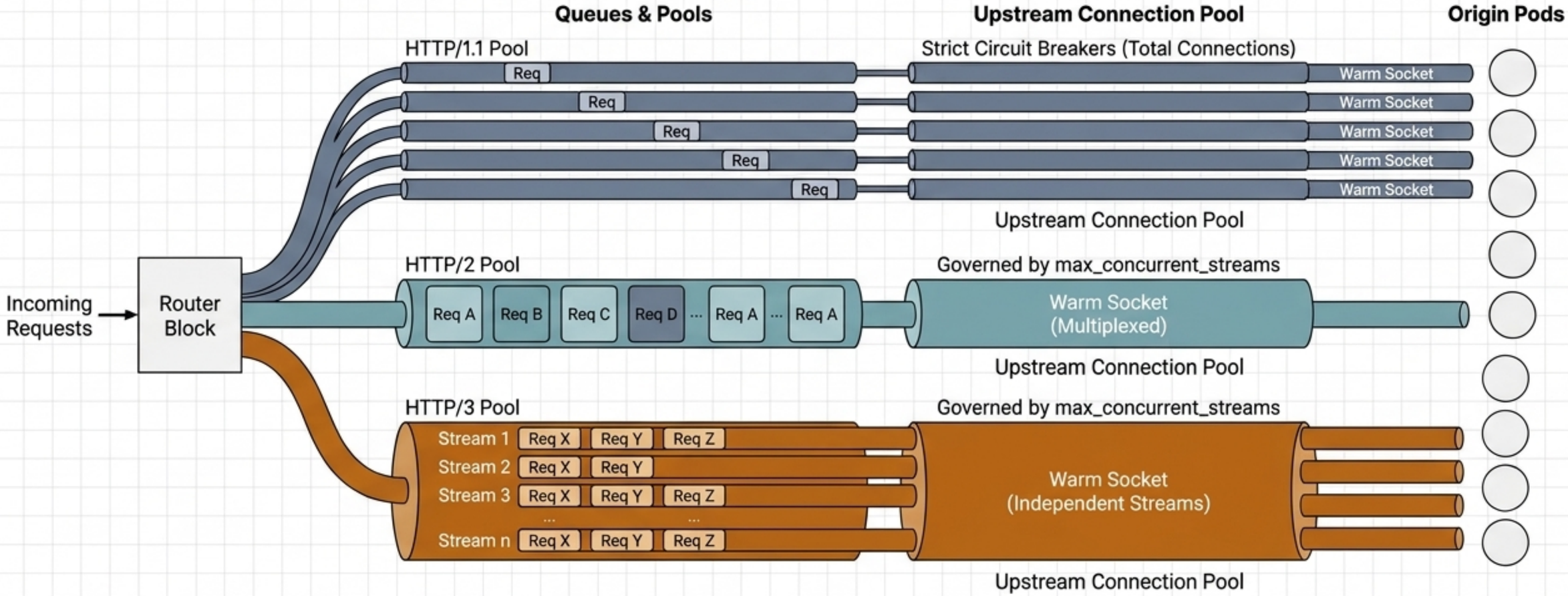
Algorithm	Best For	Worst For	CPU Overhead	Herd Resistance
Round Robin	Identical workloads	Variable query costs	Low	Low
P2C / Least Request	Heavy, variable REST/gRPC	Stateful caching	Medium	High
Ring Hash	Small stateful clusters	High-churn autoscaling	High	N/A
Maglev	Massive distributed caches	Small clusters (<10 nodes)	Medium	N/A

Gemini Notebook

Editorial Datapath Blueprint

Upstream Connection Pooling & Amortization

- **Goal:** Amorize the extreme CPU latency of TCP/TLS handshakes (Session Resumption, 0-RTT) by keeping warm sockets to origin pods.
- **HTTP/1.1 Limits:** Strictly 1 request per connection. Limited by strict circuit breakers on total active connections.
- **HTTP/2 & 3 Limits:** Deeply multiplexed. Governed by the `max_concurrent_streams` parameter rather than raw connection counts. Preferred for reverse proxies to prevent severed sockets during load spikes.

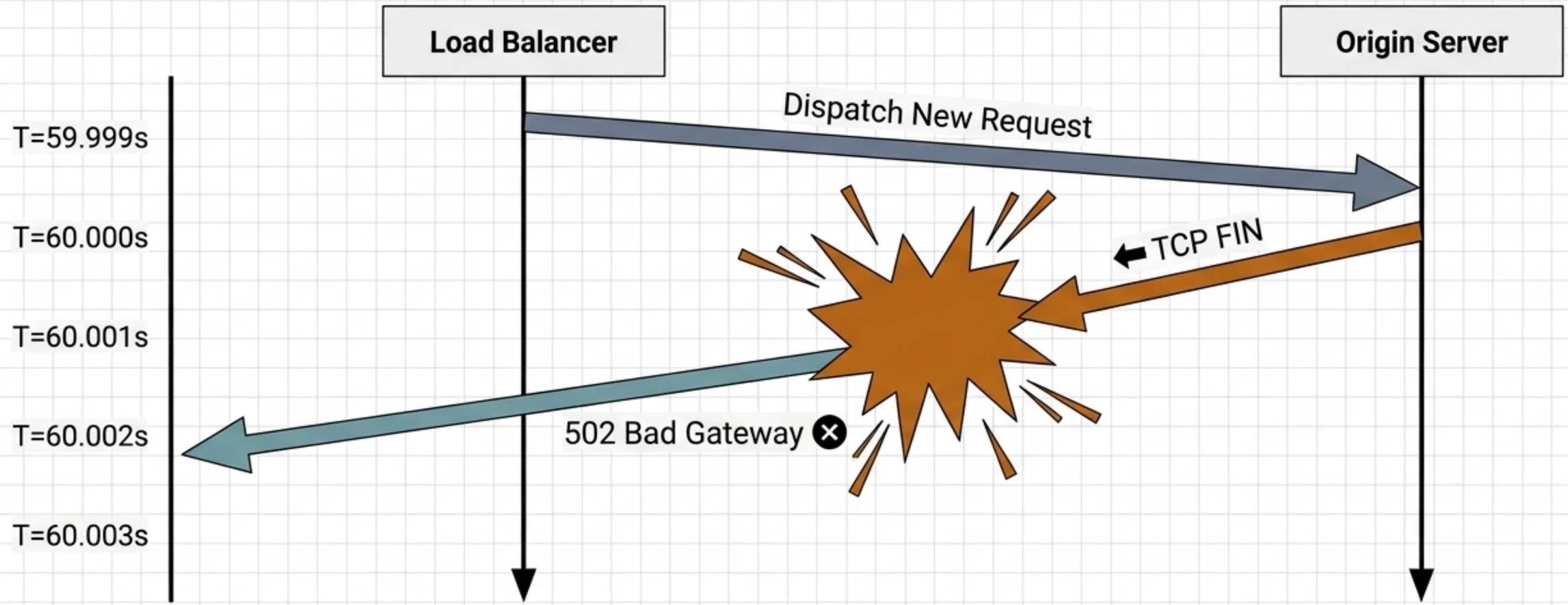


Gemini Notebook

Editorial Datapath Blueprint

The 502 Bad Gateway Keep-Alive Race

The Pathology: When the Load Balancer and the Origin Server possess identical or misaligned idle connection dion timeouts, a race condition occurs on the network wire. The LB attempts to reuse a socket in the exact millisecond the backend tears it down.



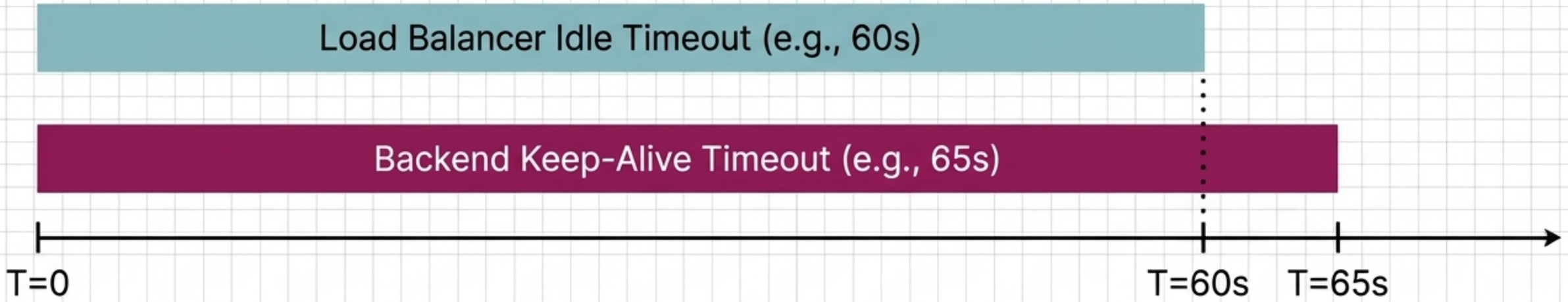
Gemini Notebook

Editorial Datapath Blueprint

Winning the Race (Keep-Alive Engineering)

The Golden Rule:
The backend origin's **keepAliveTimeout** must be **strictly longer** than the Load Balancer's connection idle timeout.

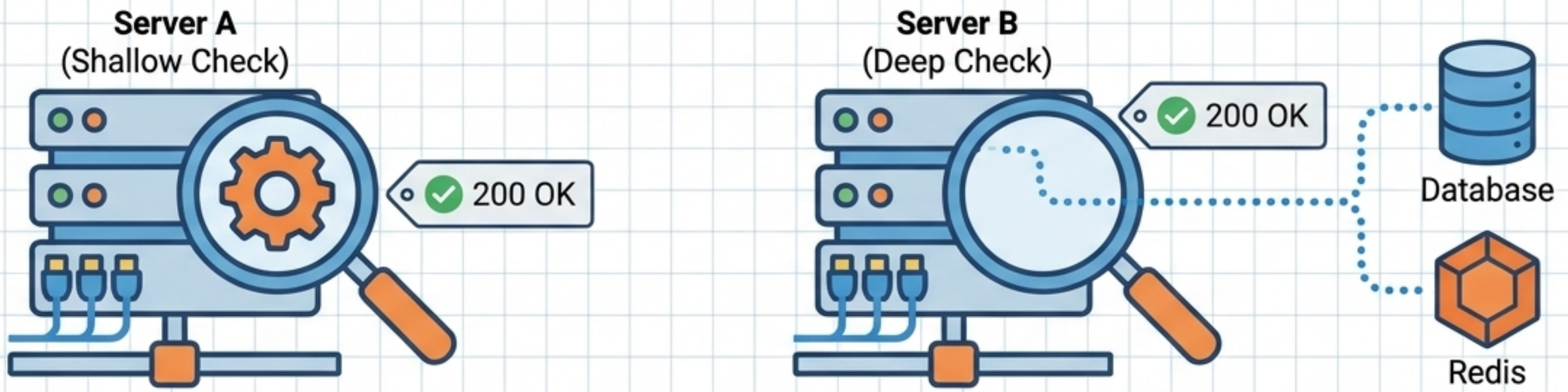
The Logic:
The load balancer must always be the authority that closes idle connections. By ensuring the backend outlives the LB's timeout window, the TCP FIN race condition is mathematically eliminated.



Gemini Notebook

Health-Check Design: Shallow vs. Deep

Shallow Readiness: Responds **200 OK** if the local application process is bound to the port and accepting TCP connections.



Responds **200 OK** if local application process is bound and accepting connections.

Responds **200 OK** only if application can successfully query downstream databases and external cache dependencies.

The Trap: Deep health checks appear logically safer, but they introduce massive, catastrophic systemic risk to the platform.

Gemini Notebook

The Cascading Failure of Deep Checks

The Pathology: Tying local pod readiness to global dependency health transforms minor, localized latency spikes into total fleet evictions. You have inadvertently weaponized your load balancer against your own application.

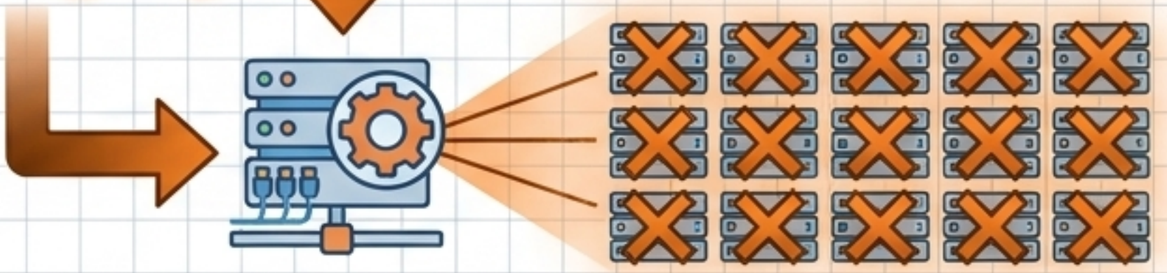


Backend Database experiences a brief **500ms** latency brownout.



Deep health checks across **ALL 100** Application Pods time out simultaneously.

The Load Balancer marks **100%** of the fleet as unhealthy.





The LB evicts all pods, resulting in a total 503 outage—even though the app pods were perfectly capable of serving cached data.

Gemini Notebook

Fail-Open Safety & Panic Thresholds

The Mechanism: Modern proxies (like Envoy) implement a configurable "Panic Threshold" (typically 50%).

The Logic: If the load balancer sees >50% of the fleet fail health checks simultaneously, it enters Panic Mode. It assumes the health-checking mechanism itself is broken, ignores the failure data, and routes traffic to all nodes (Fail-Open).

The Rationale: It is statistically more likely that the network partition or health-check endpoint failed than that 80% of your fleet died in the same second.



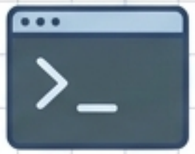
Gemini Notebook

Graceful Degradation: The Lame Duck State

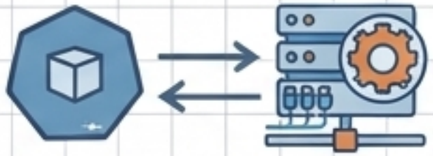
The Problem: Hard-killing backend tasks during rolling deployments results in dropped connections and user-facing 502s.

Lame Duck State: The backend task keeps its port open but explicitly asks the LB to cease new routing.

Connection Draining: Existing long-lived requests are given a generous grace period to finalize before the process cleanly exits.



Orchestrator (K8s) sends SIGTERM.



Pod enters Lame Duck State –> Broadcasts explicitly to LB: “Stop sending new requests.”



Connection Draining Phase –> In-flight requests are allowed 10s-150s to finish.



Safe Exit –> 0 dropped packets.

Gemini Notebook

Synthesis: The Definitive Routing Decision Matrix

The Takeaway: Load balancing is not one-size-fits-all. The architectural layer you choose dictates your scalability limits, TLS overhead, and incident blast radius. Use this matrix to match the workload to the wire.

Architectures	L4 NLB	L7 ALB/Envoy	API Gateway	Client-Side xDS
High-TPS TCP/UDP (Gaming, DBs)	⬢ Optimal ⬢	⬡ Possible ⬡	● Anti-Pattern	⬡ Possible ⬡
Standard REST/Web	⬡ Possible ⬡	⬢ Optimal ⬢	⬢ Optimal ⬢	⬡ Possible ⬡
WebSockets/SSE	⬡ Possible ⬡	⬢ Optimal ⬢	⬢ Optimal ⬢	⬡ Possible ⬡
Internal gRPC Meshes	● Anti-Pattern (Connection Pinning)	⬢ Optimal ⬢	⬡ Possible ⬡	⬢ Optimal ⬢

Gemini Notebook