ServiceRouter

HYPERSCALE AND MINIMAL COST SERVICE MESH AT META

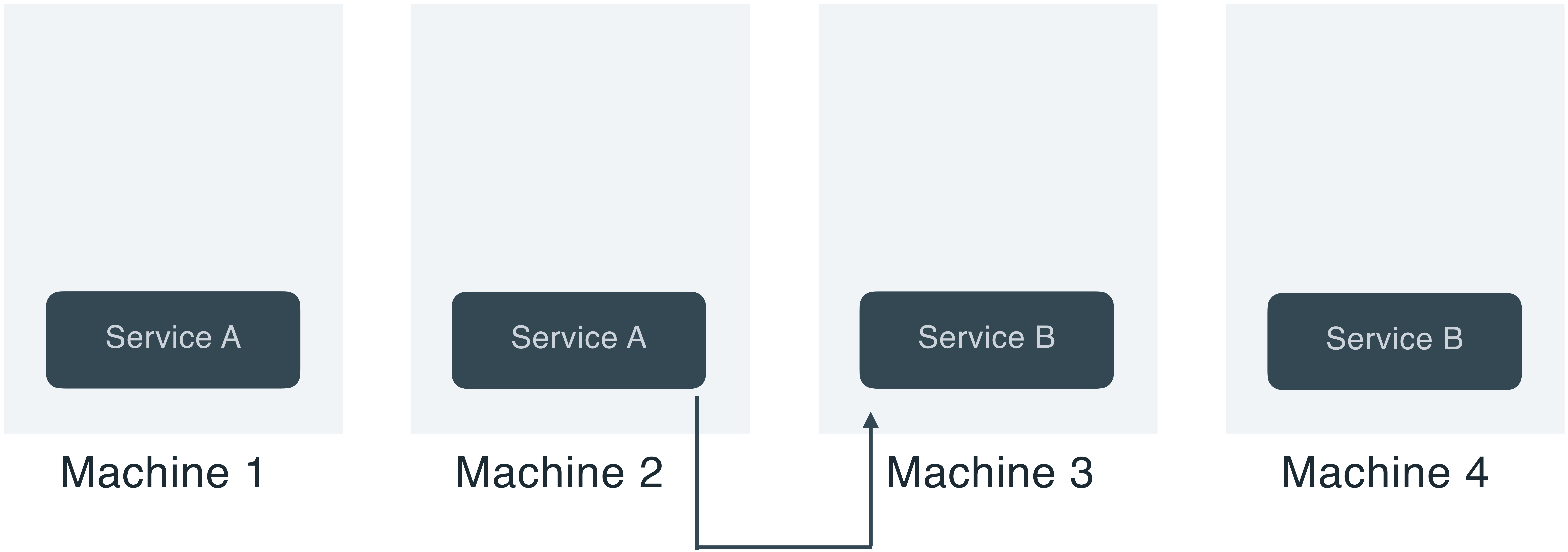
Harshit Saokar[1] | Soteris Demetriou[1,2] | Nick Magerko[1] | Max Kontorovich[1] | Josh Kirstein[1] | Margot Leibold[1] | Dimitrios Skarlatos[1,3] | Hitesh Khandelwal[1] | Chunqiang Tang[1]

[1] Meta

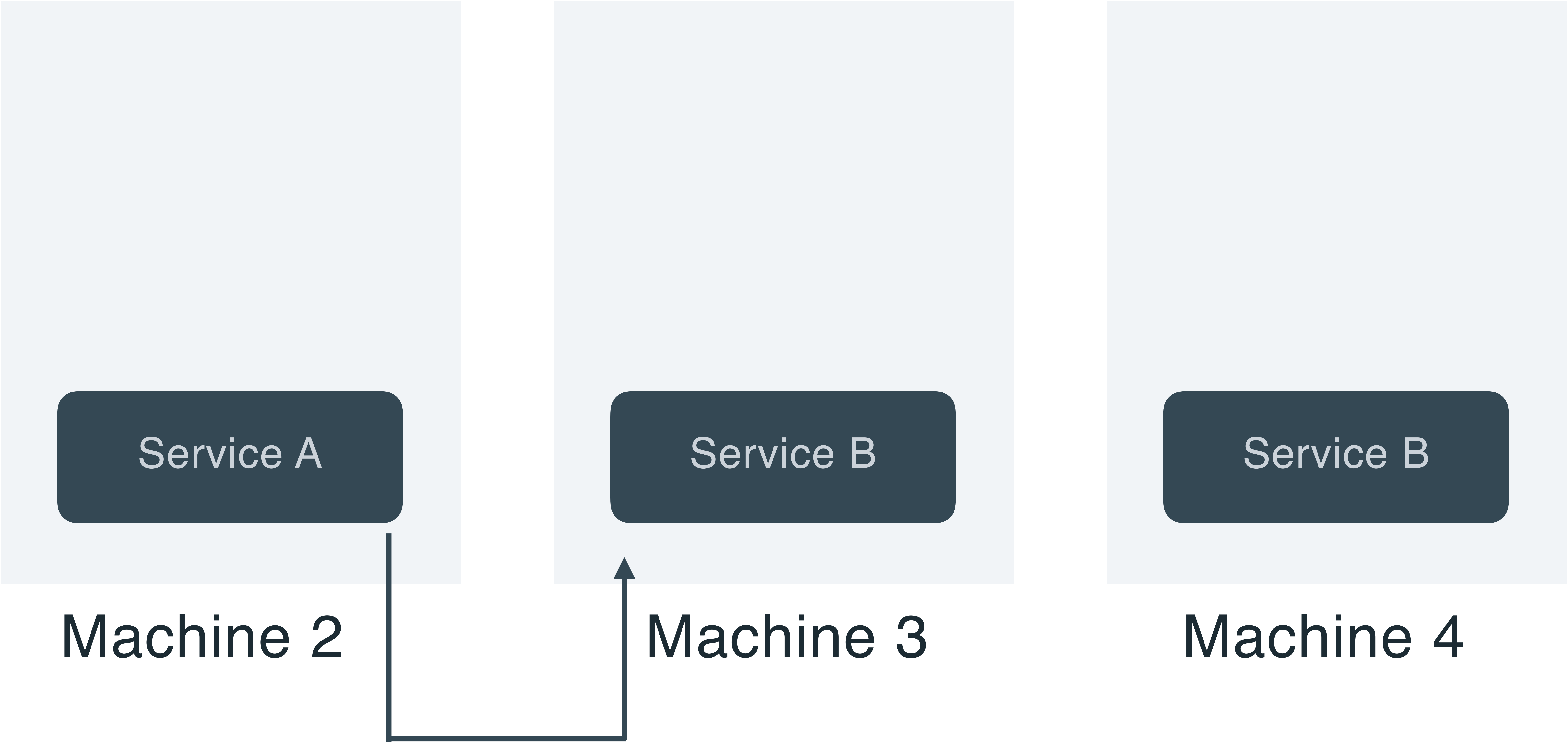
[2] Imperial College London

[3] Carnegie Mellon University Computer Science Department

01 Background & Motivation



Huye et al. Lifting the veil on Meta's microservice architecture: Analyses of topology and request workflows. USENIX ATC ‘23



RPC Frameworks

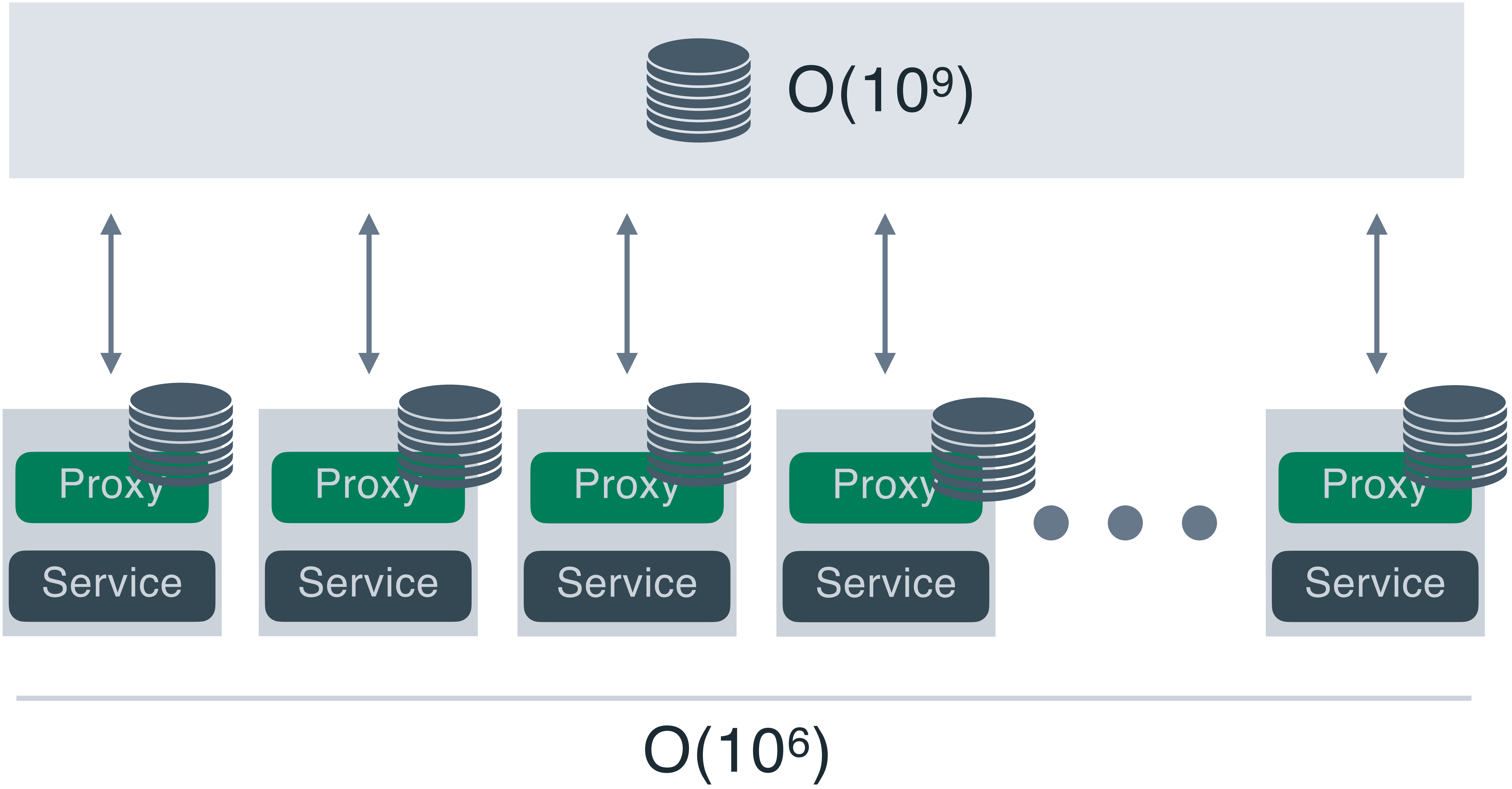
- No Advanced Load Balancing
- Need external support for service discovery
- Examples: gRPC, Thrift

01 Background & Motivation: Service Mesh

CONTROL PLANE
L7 Proxy
Service A
Machine 1
L7 Proxy
Service A
Machine 2
L7 Proxy
Service B
Machine 3
L7 Proxy
Service B
Machine 4
REGION A
L7 Proxy
Service B
Machine 5
REGION B

Service Mesh Challenges

- **[SCALABILITY]** How can we scale service discovery to O(10^6) clients and proxies?

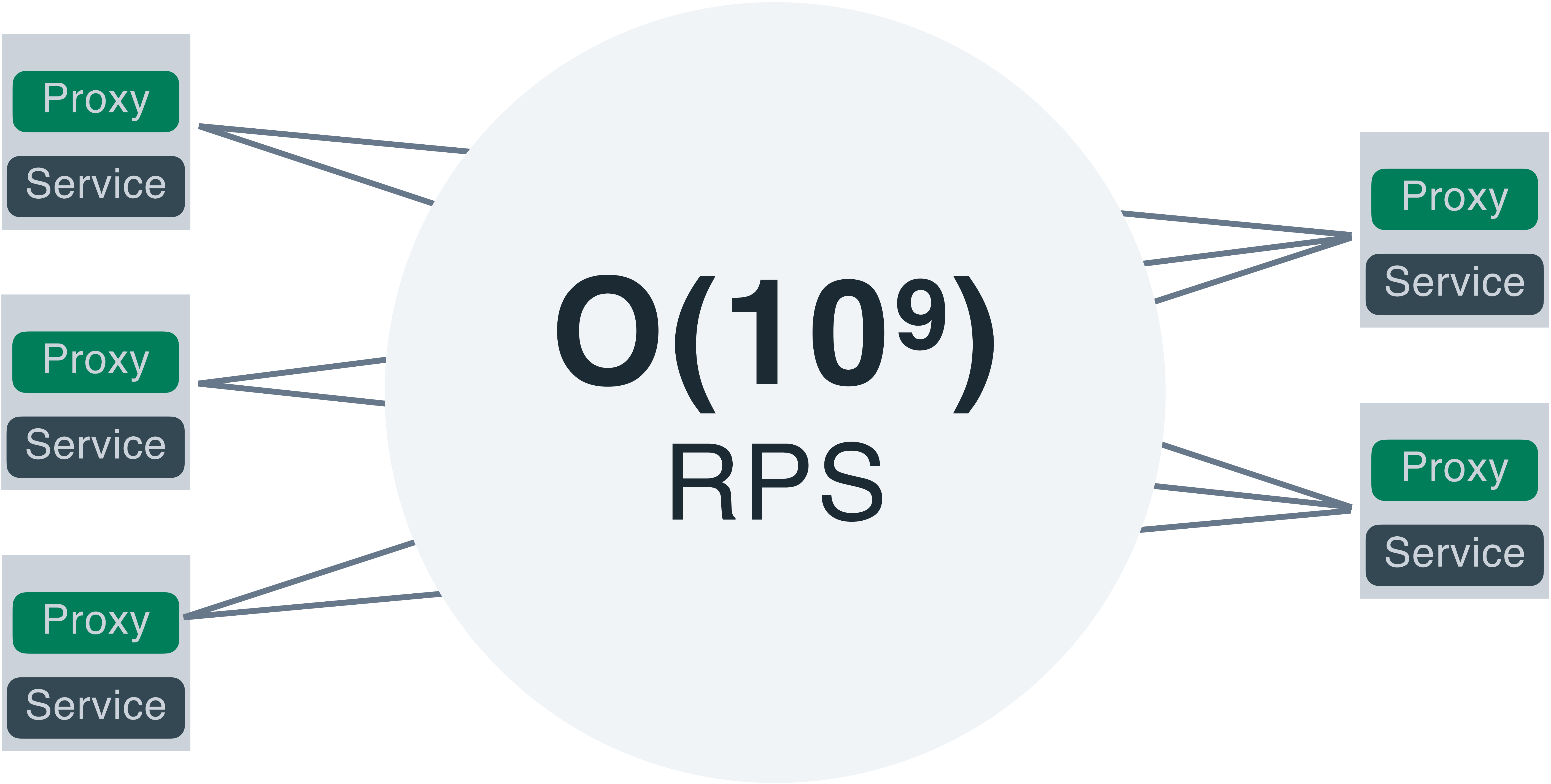


Service Mesh Challenges

- **[SCALABILITY]** How can we scale service discovery to O(10^6) clients and proxies?
- **[HW COST]** How to minimize HW cost?

Istio: 0.35vCPU for O(10^3) rps

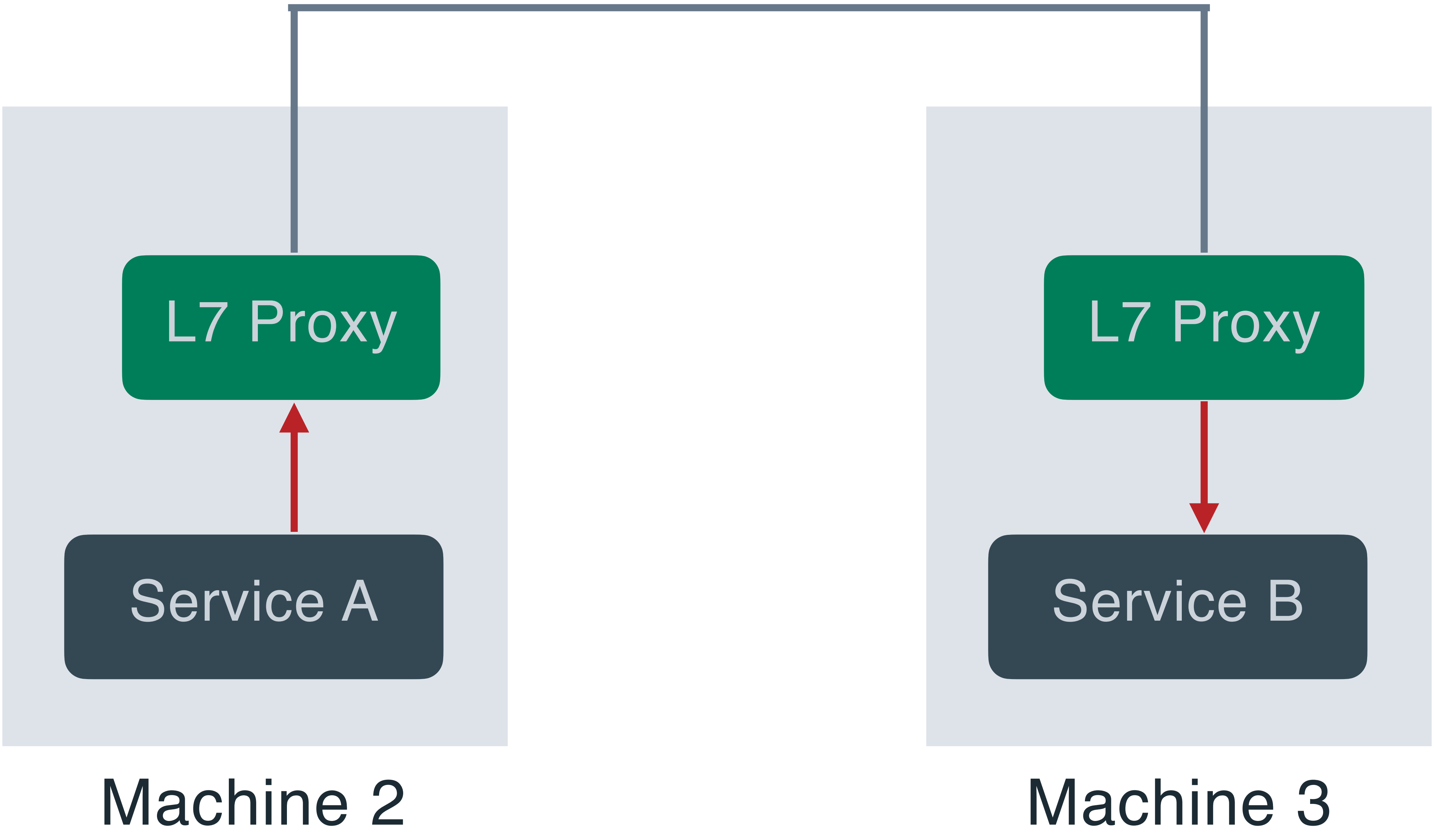
1,750,000 AWS t4g.small VMs for 10B rps



Service Mesh Challenges

- [SCALABILITY] How can we scale service discovery to O(10^6) clients and proxies?
- [HW COST] How to minimize HW cost?

- **[RPC LATENCY & LB]** How to simultaneously minimize RPC latency and load balance across geo-distributed hosts?
  - Sidecars add extra latency



Zhu et al show that Istio

- increases the latency by **185%**

Zhu et al. Dissecting Service Mesh Overheads. In *arXiv preprint arXiv:2207.00592*, 2022.

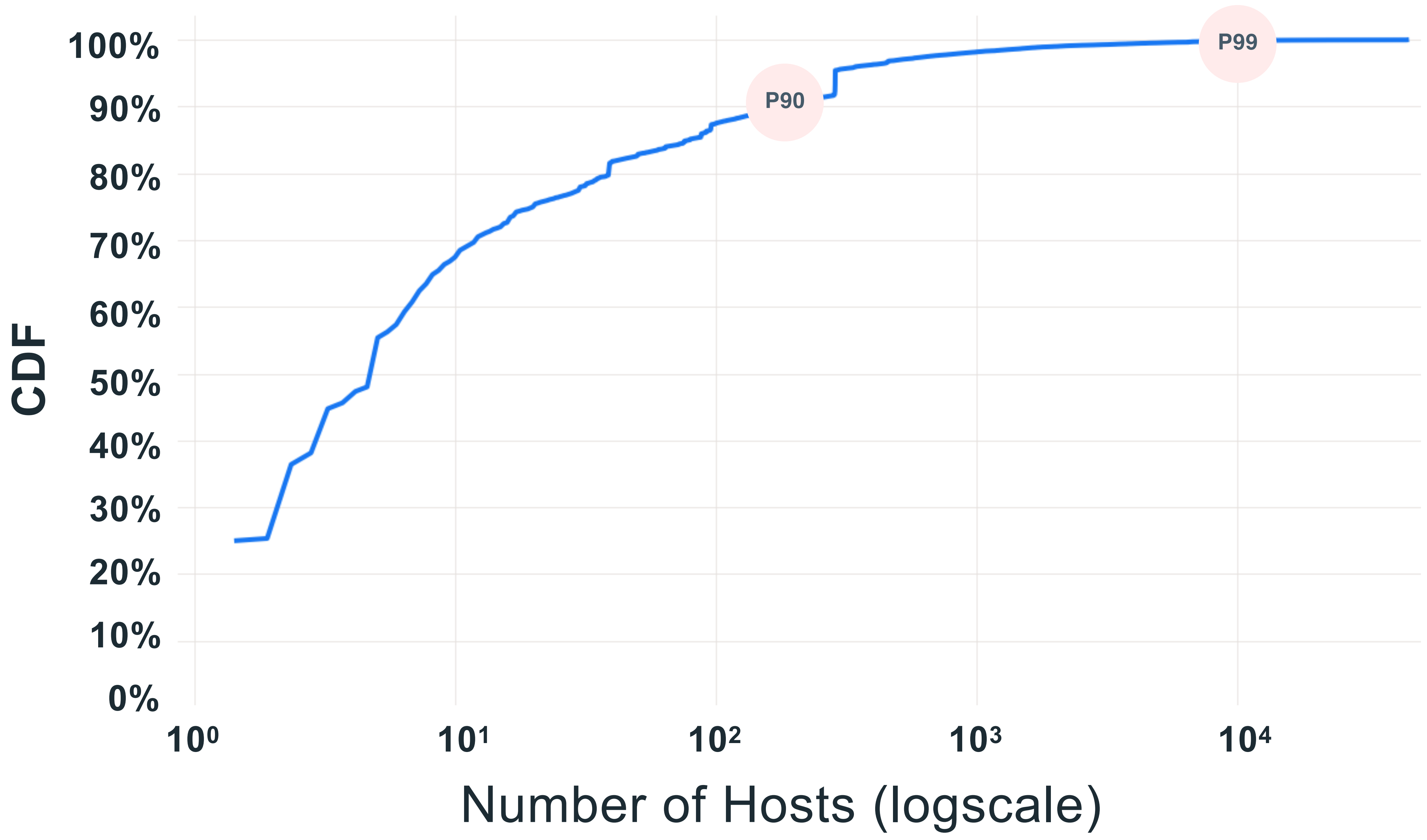
mRPC shows that a sidecar approach:

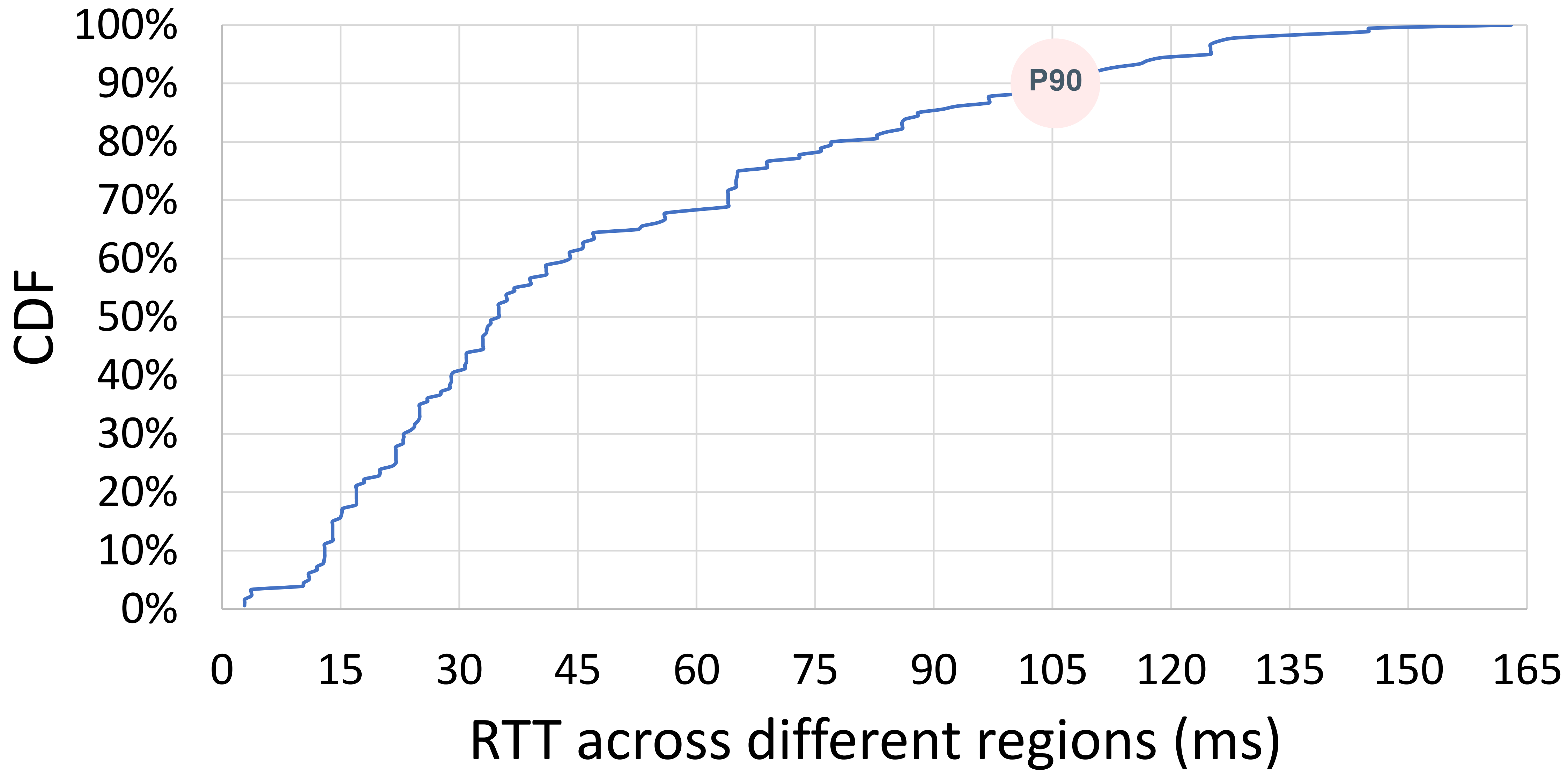
- increases P99 RPC latency by **180%**

Chen, et al. Remote procedure call as a managed system service. *NSDI '23*

Service Mesh Challenges

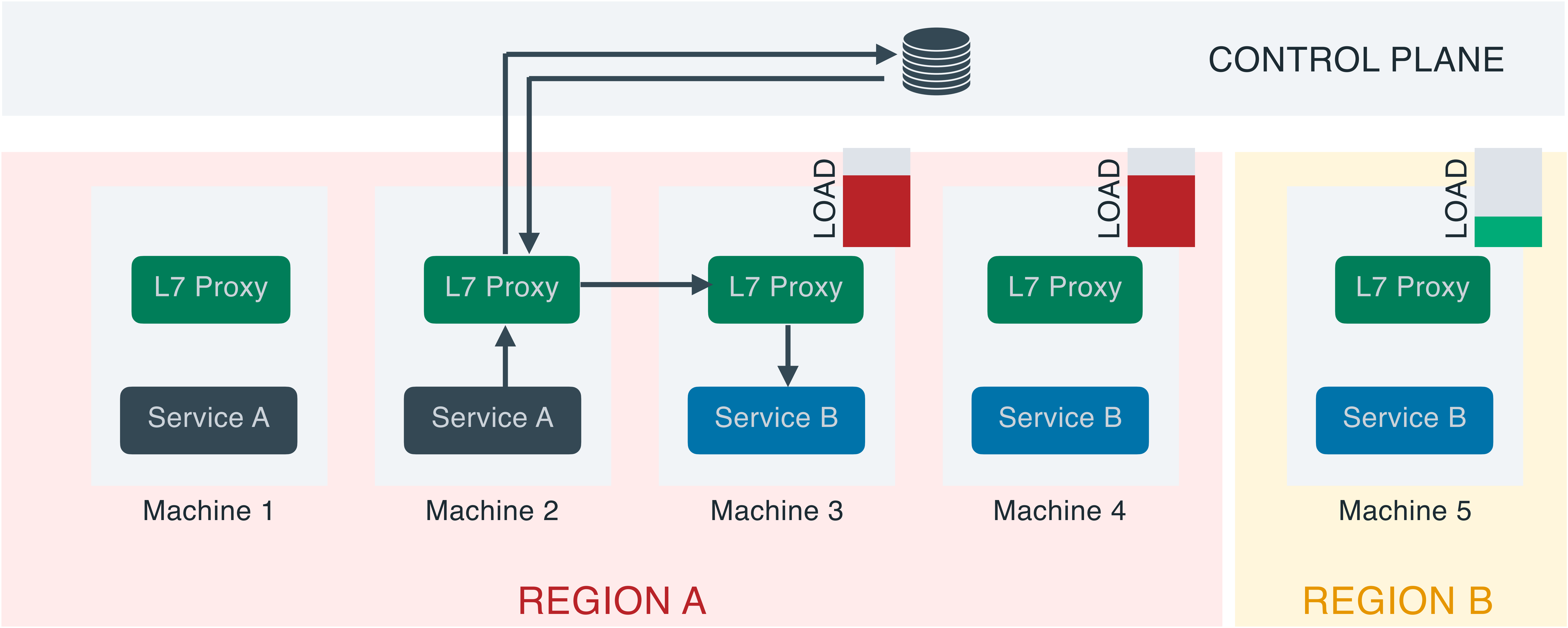
- [SCALABILITY] How can we scale service discovery to O(10^6) clients and proxies?
- [HW COST] How to minimize HW cost?
- **[RPC LATENCY & LB]** How to simultaneously minimize RPC latency and load balance across geo-distributed hosts?
  - Sidecars add extra latency
  - O(10-10^4) hosts per service
  - P90 cross-region latency: 106ms





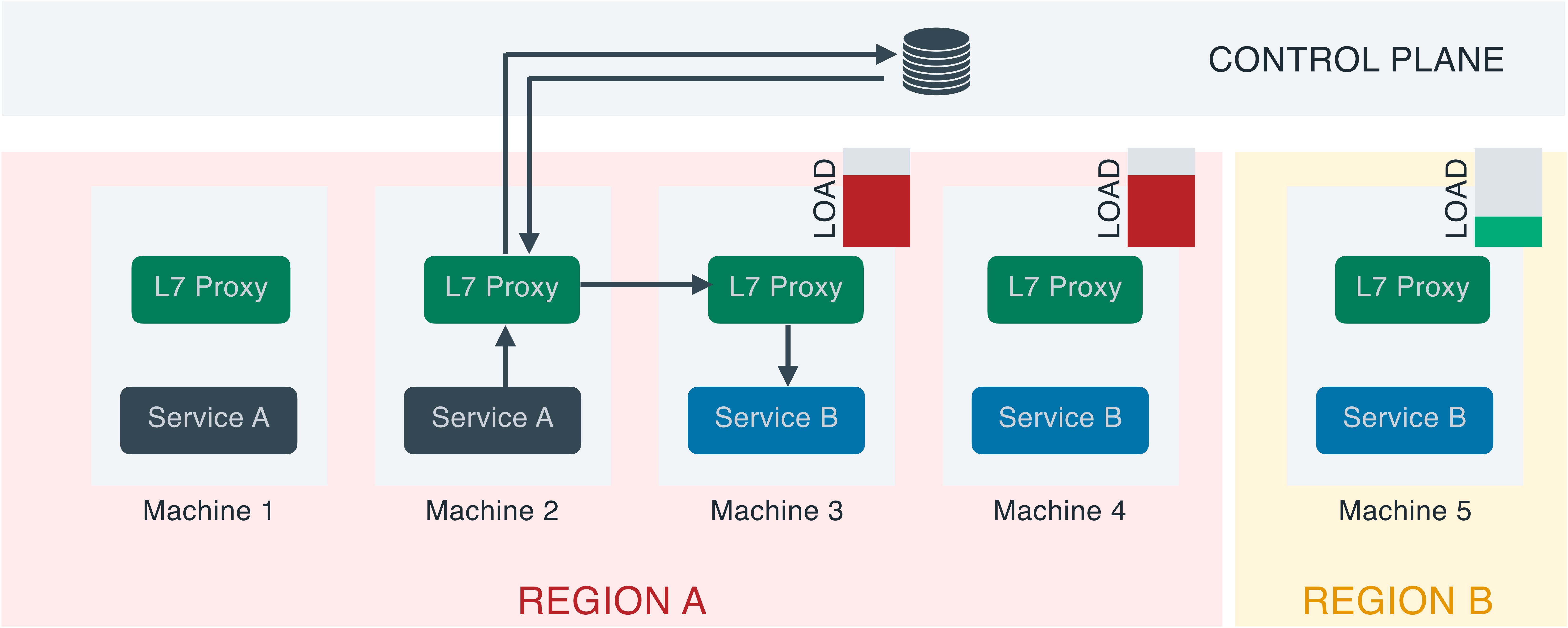
Service Mesh Challenges

- [SCALABILITY] How can we scale service discovery to O(10^6) clients and proxies?
- [HW COST] How to minimize HW cost?
- **[RPC LATENCY & LB]** How to simultaneously minimize RPC latency and load balance across geo-distributed hosts?
    - Sidecars add extra latency
    - O(10-10^4) hosts per service
    - P90 cross-region latency: 106ms



Service Mesh Challenges

- [SCALABILITY] How can we scale service discovery to O(10⁶) clients and proxies?
- [HW COST] How to minimize HW cost?
- **[RPC LATENCY & LB]** How to simultaneously minimize RPC latency and load balance across geo-distributed hosts?
    - Sidecars add extra latency
    - O(10-10⁴) hosts per service
    - P90 cross-region latency: 106ms
- [**SHARDED SERVICES**] Support for shared services **NOT COVERED**



03 ServiceRouter

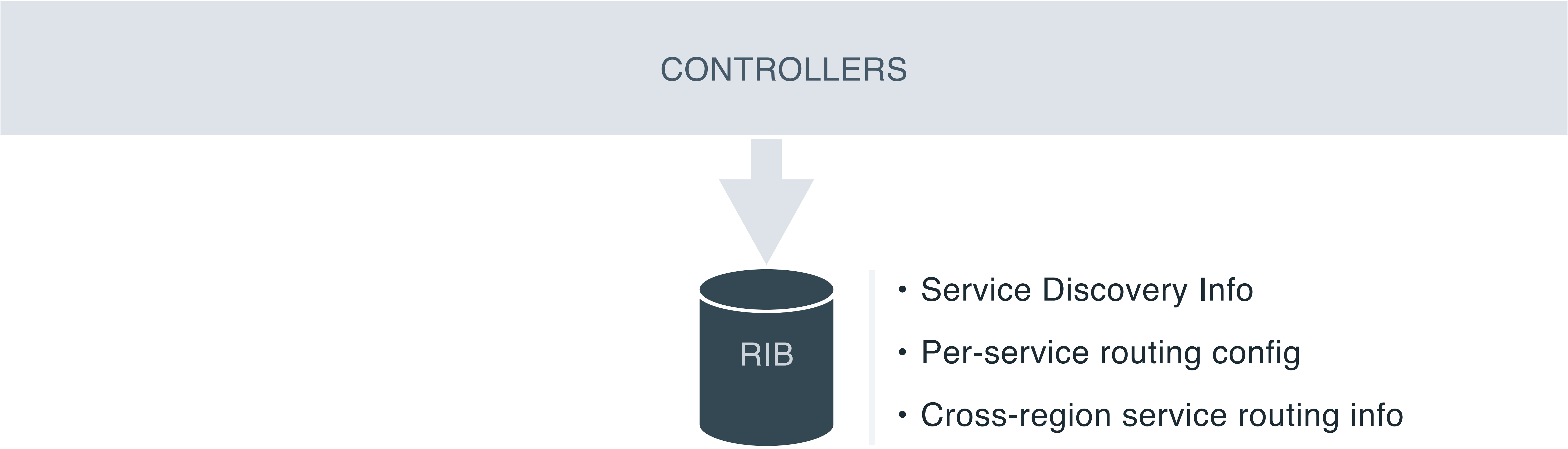
KEY DESIGN CONCEPTS

RIB

Routing Information Base

Decentralize the unscalable part of the control plane in order to scale out.

- Independent controllers execute different functions such as registering services and generating a per-service cross-region routing table.



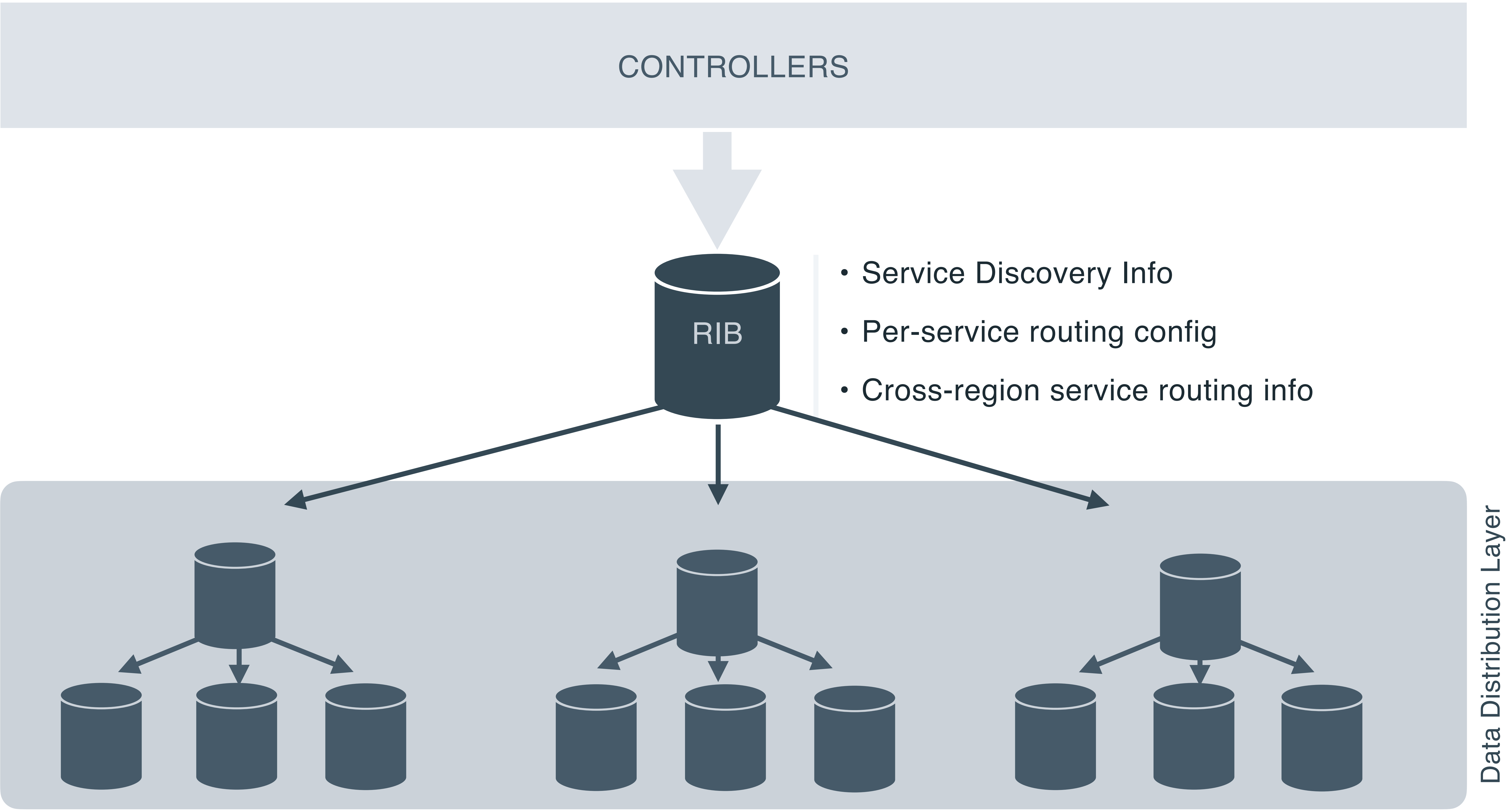
- Service Discovery Info
- Per-service routing config
- Cross-region service routing info

RIB

Routing Information Base

Decentralize the unscalable part of the control plane in order to scale out.

- Independent controllers execute different functions such as registering services and generating a per-service cross-region routing table.
- The data distribution layer massively replicates the RIB so that there are sufficient RIB replicas to handle read traffic from millions of proxies.



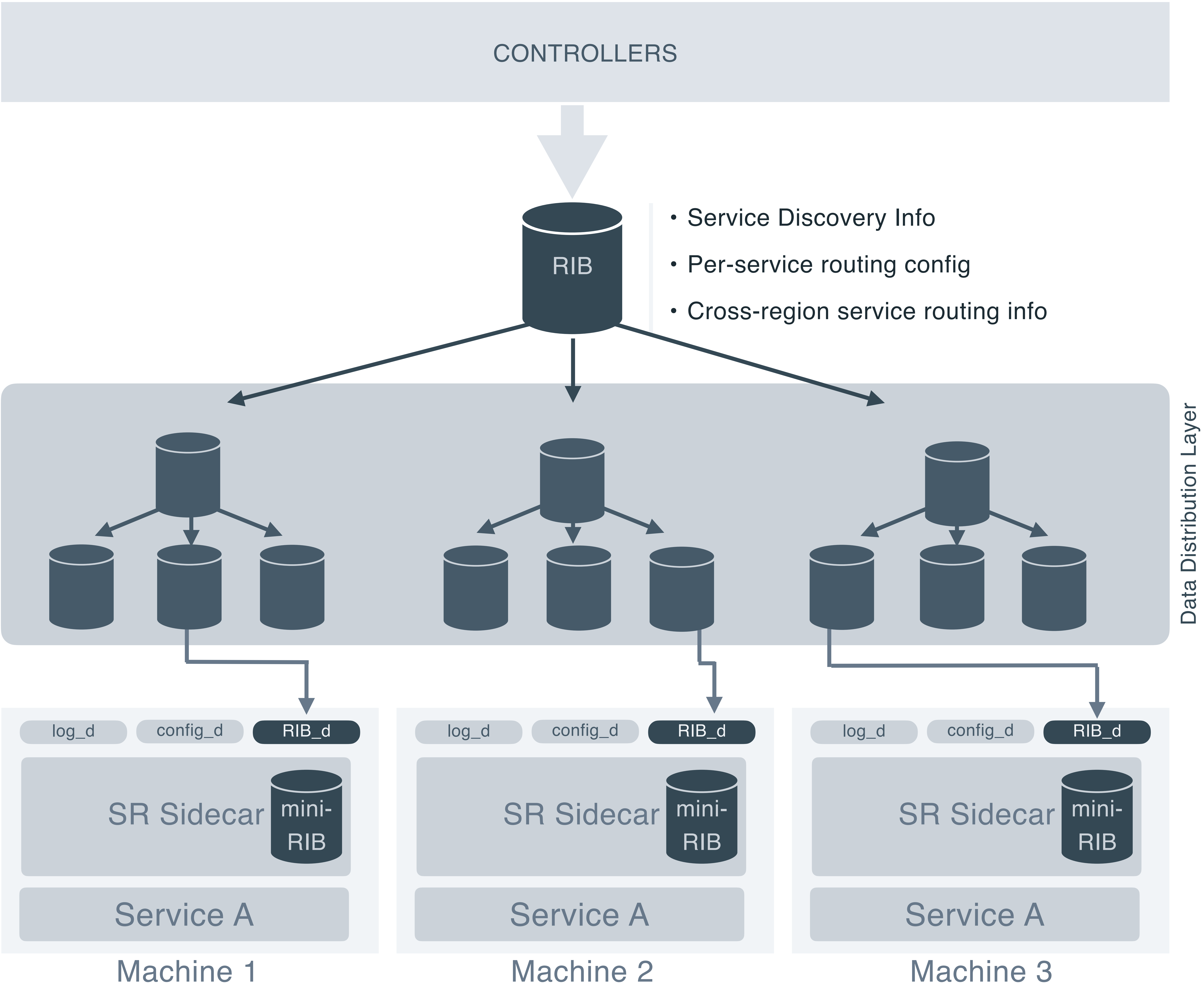
- Service Discovery Info
- Per-service routing config
- Cross-region service routing info

RIB

Routing Information Base

Decentralize the unscalable part of the control plane in order to scale out.

- Independent controllers execute different functions such as registering services and generating a per-service cross-region routing table.
- The data distribution layer massively replicates the RIB so that there are sufficient RIB replicas to handle read traffic from millions of proxies.
- Each proxy self-configures and self-manages without the control plane's direct involvement.



- Service Discovery Info
- Per-service routing config
- Cross-region service routing info

Versatility

Controllers are agnostic to the L7 architecture.

Routing metadata traffic
Application RPC traffic
Routing Info Base (RIB)
Machine
miniRIB
RIBDaemon
Client
SRLib
Server
Server
Server
Routing Info Base (RIB)
miniRIB
SRLookasideService
Machine
Client
SRLib
Client
SRLib
Server
Server
Server
Routing Info Base (RIB)
Machine
miniRIB
SRProxy
Client
Server
Server
Server
Routing Info Base (RIB)
Machine
Client
Client
miniRIB
SRProxy
Server
Server
Server

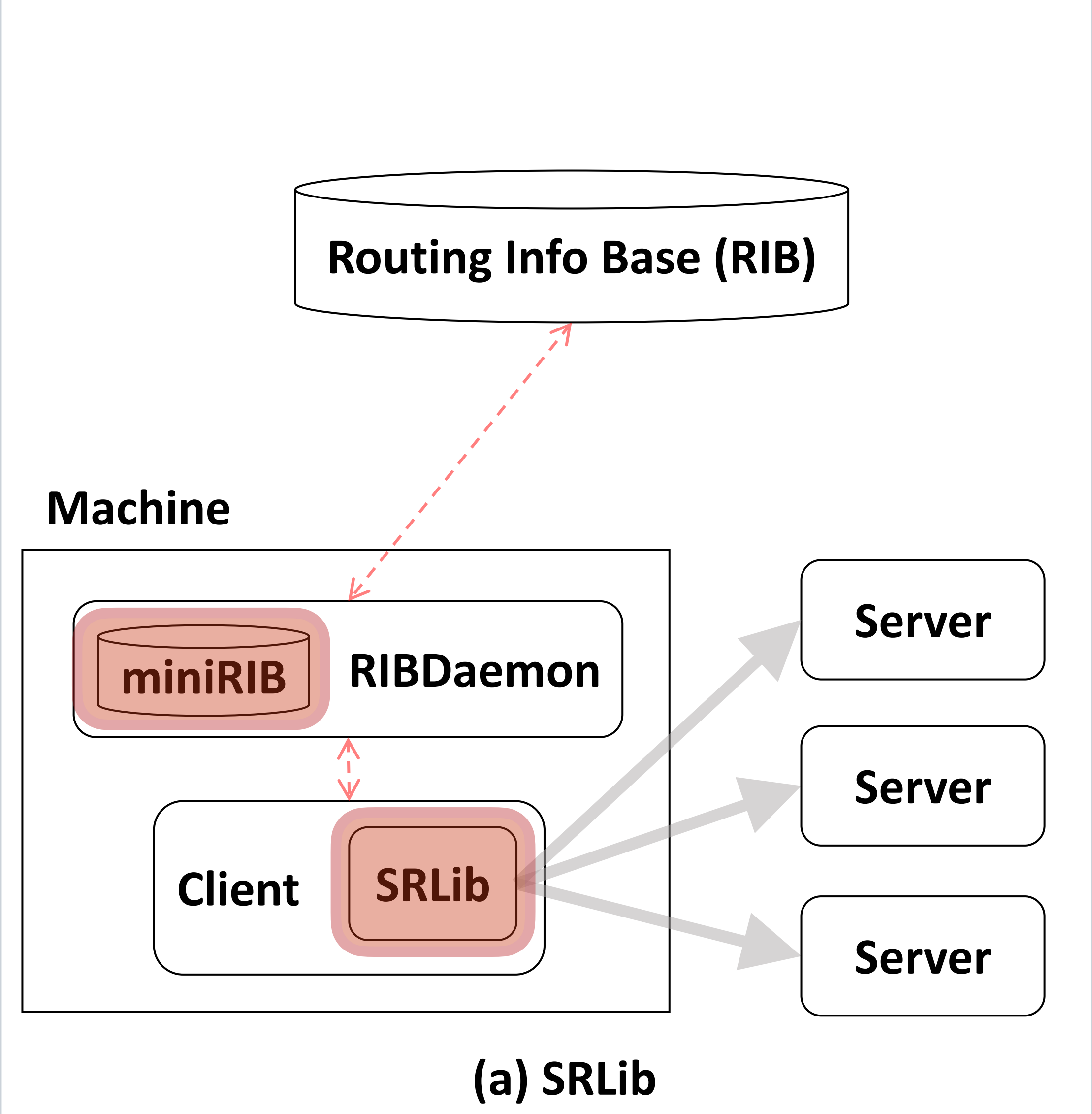
(a) SRLib

(b) SRLookaside

(c) SRSidecarProxy

(d) SRRemoteProxy

99% RPC traffic routed through SRLib.



(a) SRLib

SRLib

Provide the service-mesh functions out of a library that is directly linked into the RPC client's executable

- Eliminates side car latency overhead

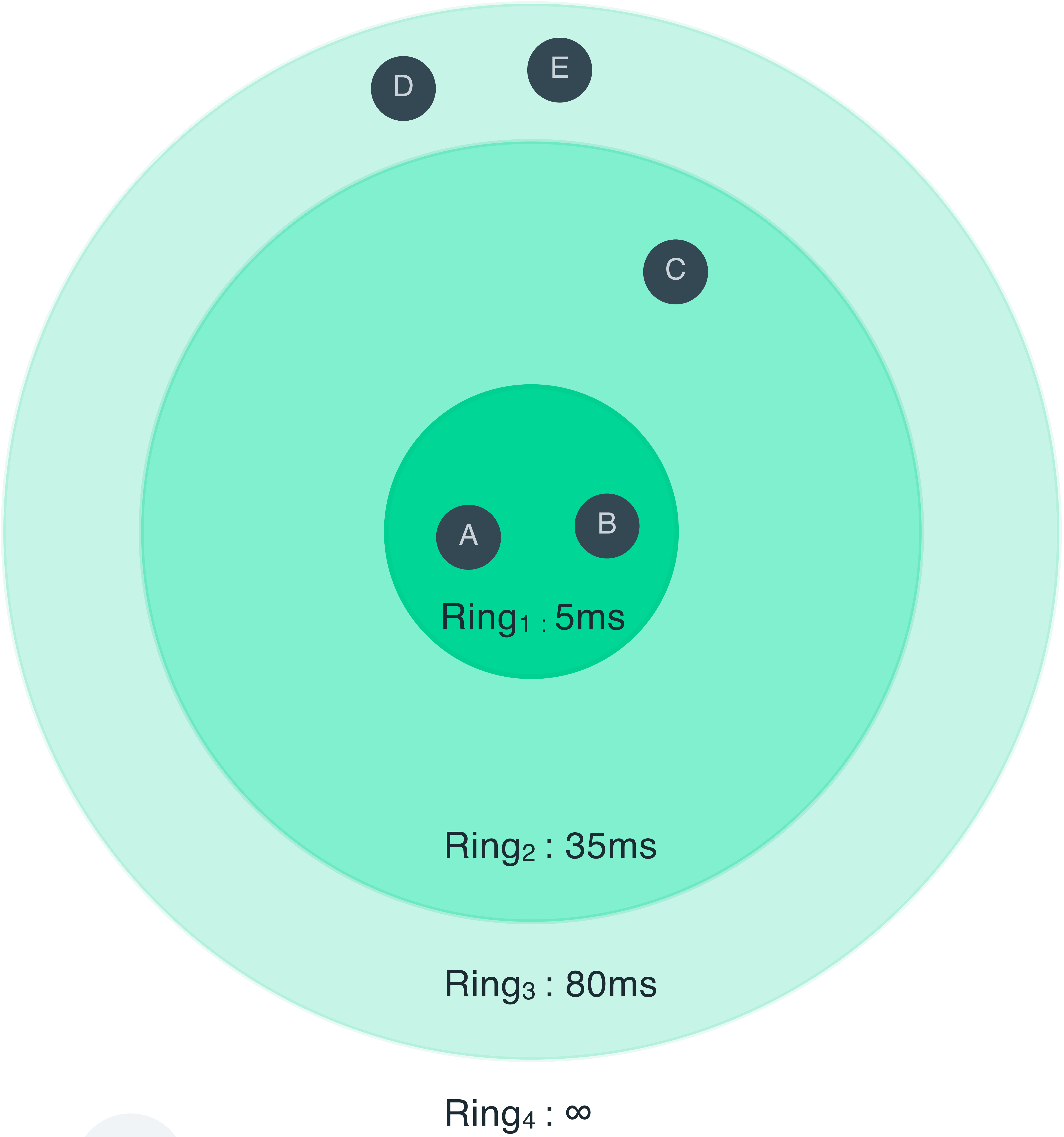
Run a separate RIBDaemon on the client machine to cache **miniRIB**.

LATENCY RINGS AND CROSS-REGION ROUTING

SR strives to simultaneously minimize RPC latency and balance load across global regions.

- SR introduces the concept of latency rings to minimize latency.
- SR collects per-service global traffic and load information, computes a per-service cross-region routing table, and disseminate it to L7 routers to guide their local routing decisions.

Ring$_1$: 5ms I Ring$_2$: 35ms I Ring$_3$: 80ms I Ring$_4$: ∞

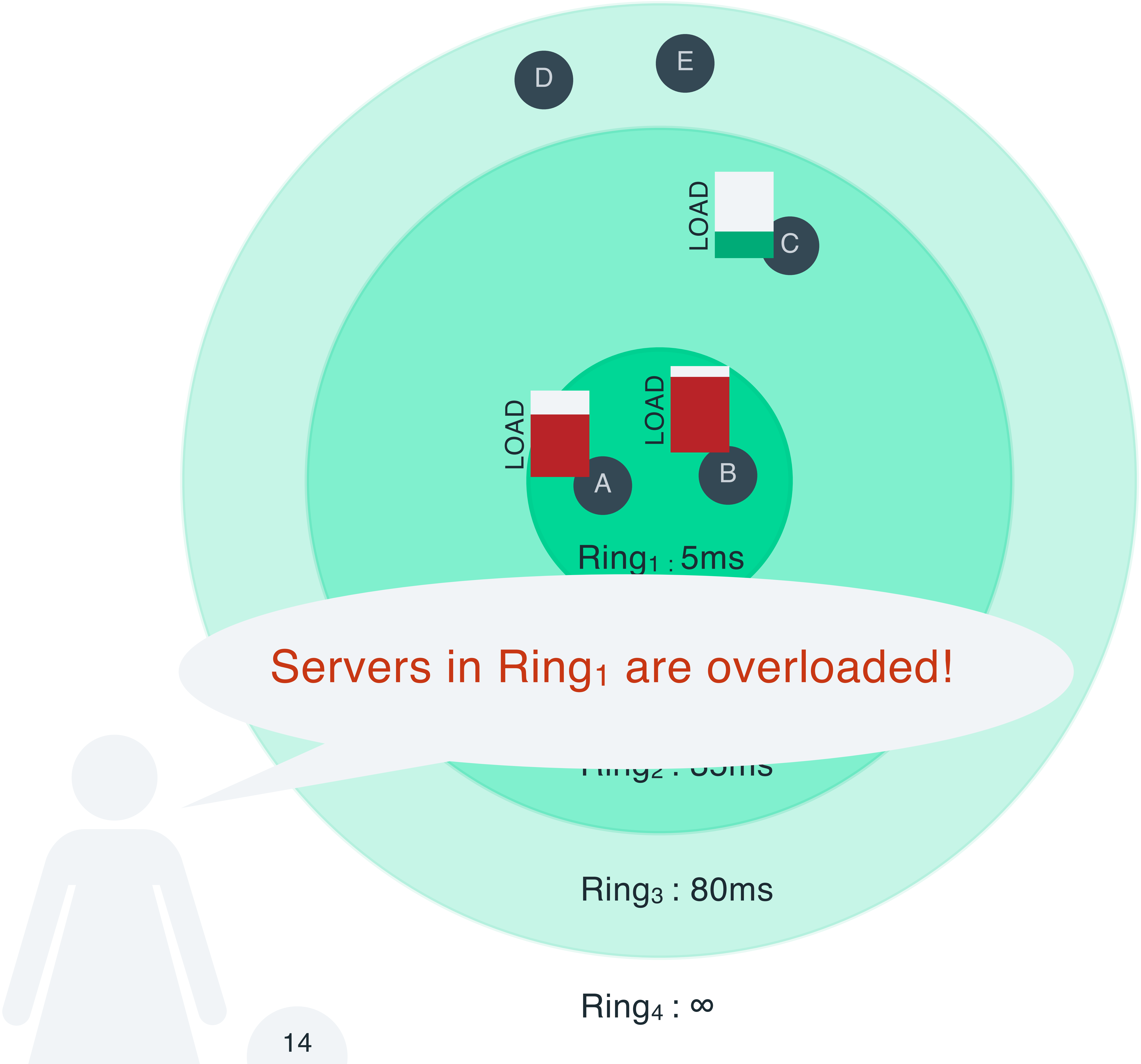


LATENCY RINGS AND CROSS-REGION ROUTING

SR strives to simultaneously minimize RPC latency and balance load across global regions.

- SR introduces the concept of latency rings to minimize latency.
- SR collects per-service global traffic and load information, computes a per-service cross-region routing table, and disseminate it to L7 routers to guide their local routing decisions.

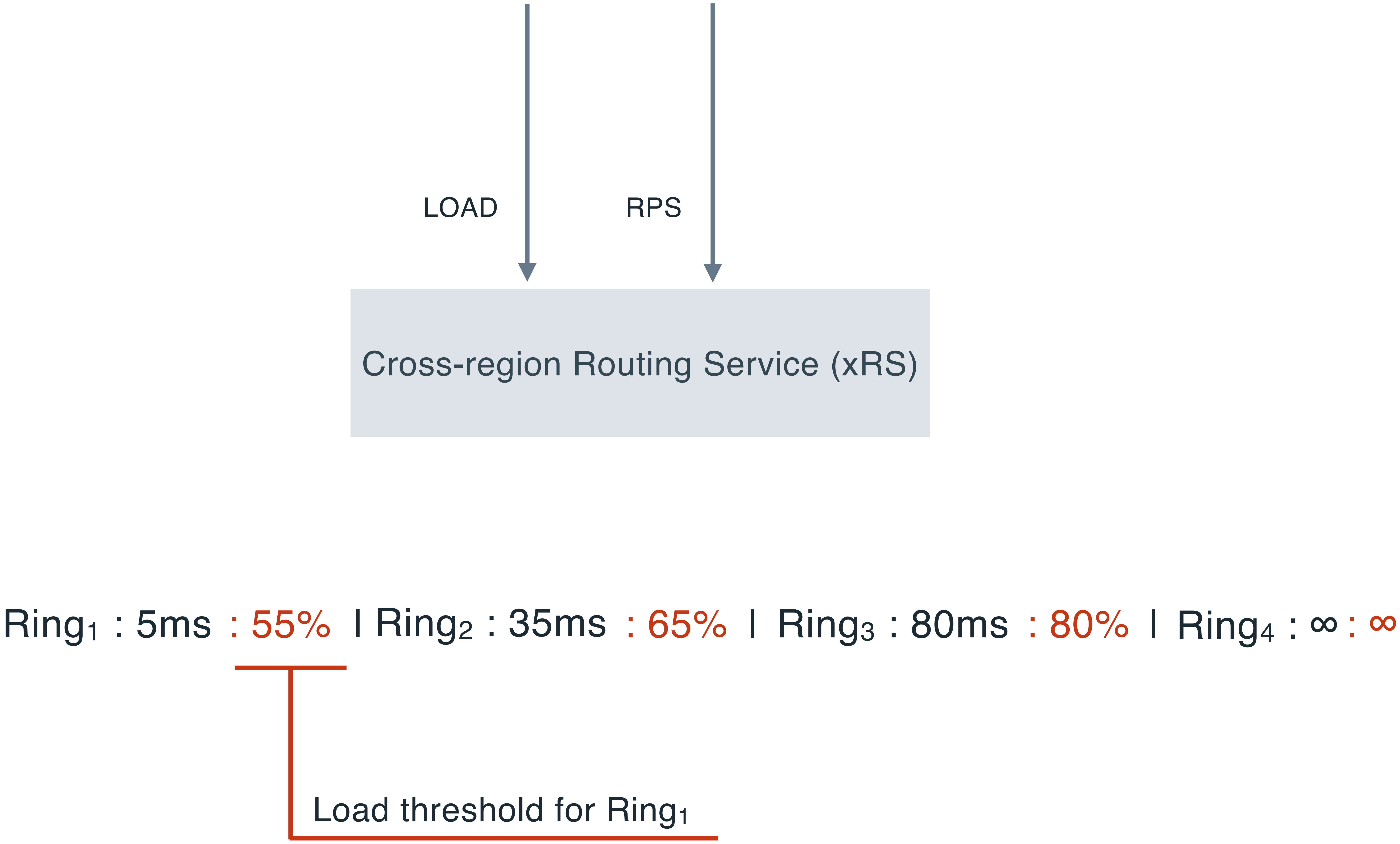
Ring$_1$: 5ms | Ring$_2$: 35ms | Ring$_3$: 80ms | Ring$_4$: ∞



LATENCY RINGS AND CROSS-REGION ROUTING

SR strives to simultaneously minimize RPC latency and balance load across global regions.

- SR introduces the concept of latency rings to minimize latency.
- SR collects per-service global traffic and load information, computes a per-service cross-region routing table, and disseminate it to L7 routers to guide their local routing decisions.


LOAD
RPS
Cross-region Routing Service (xRS)
Ring$_1$: 5ms : 55% | Ring$_2$: 35ms : 65% | Ring$_3$: 80ms : 80% | Ring$_4$: ∞ : ∞
Load threshold for Ring$_1$

04 ServiceRouter

OVERALL ARCHITECTURE

04 ServiceRouter Architecture

CONTROL PLANE
Latency Monitoring Service
Routing Config
Cross-region Latency Config
OSDI '20
Twine
SOSP '21
Shard Manager
cross-region Routing Service (xRS)
Logging
Configerator
SOSP '15
Global Registry Service (GRS)
RIB
DATA PLANE
log_d
config_d
RIB_d
SRLib
Service A
Machine 1
log_d
config_d
RIB_d
SRLib
Service B
Machine 2
log_d
config_d
RIB_d
SRProxy
Machine 3
perf_d
Service C
Machine 4

05 ServiceRouter

EVALUATION

Scalability

Overall scale

- Regions
- Routers/Clients/Servers
- Throughput

$O(10)$

Regions

$O(10^6)$

L7 Routers

Clients

$O(10^9)$

RPS

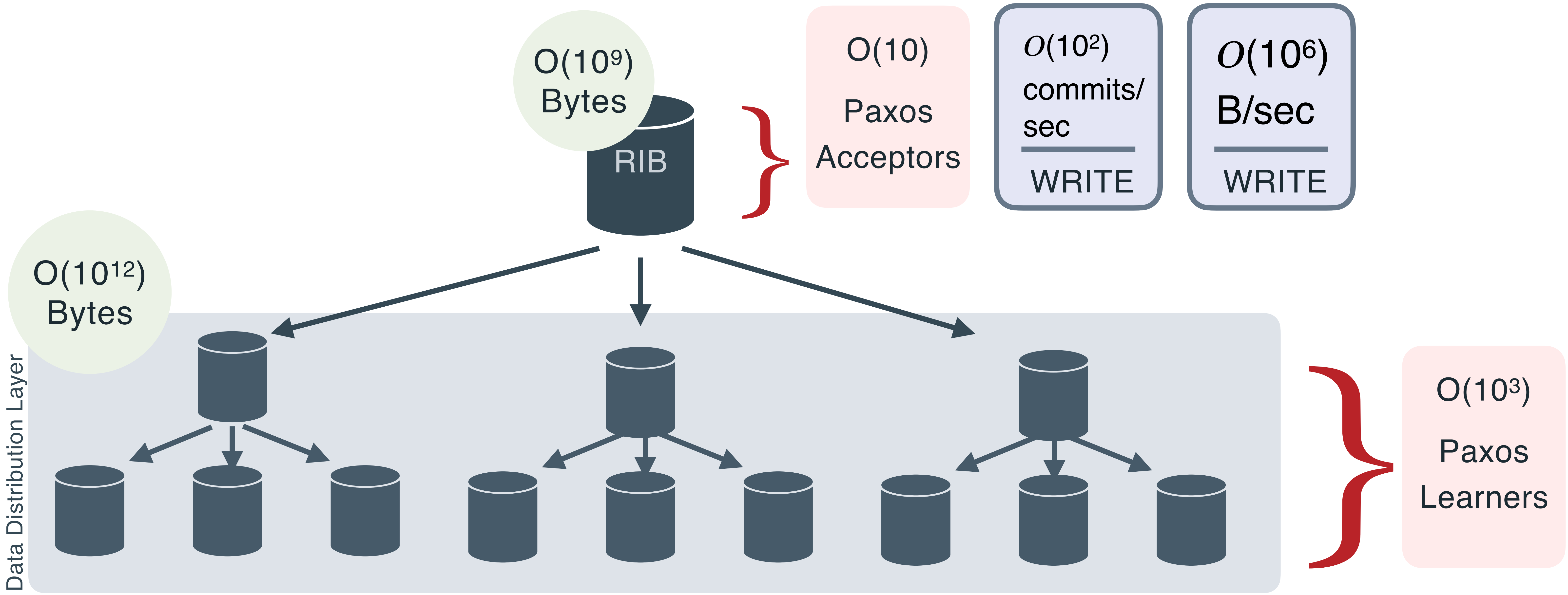
$O(10^{14})$

B/sec

Scalability

RIB - Routing Information Base

- RIB Replicas
- RIB Write bandwidth
- RIB Write throughput

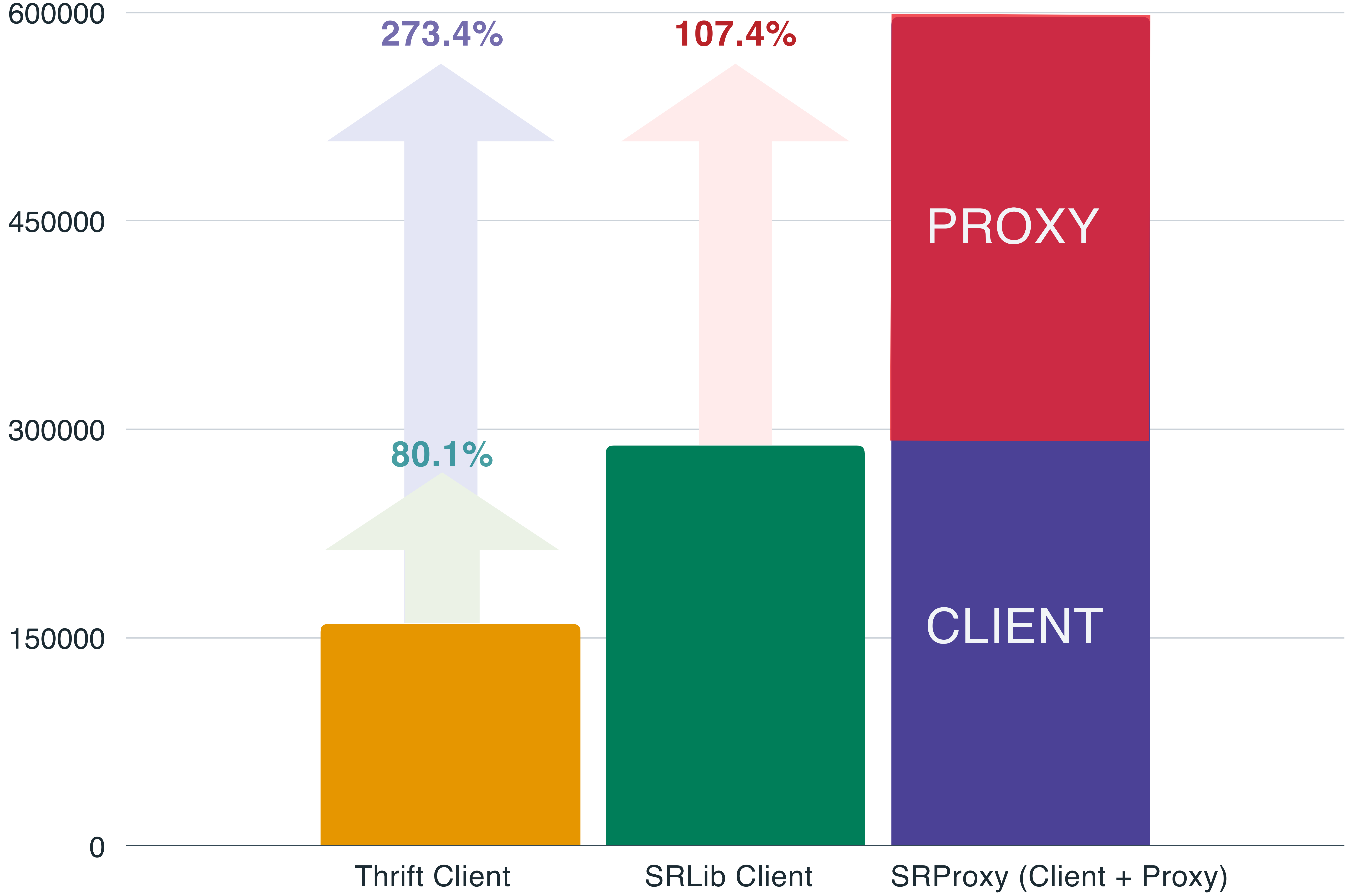


Cost

METHODOLOGY

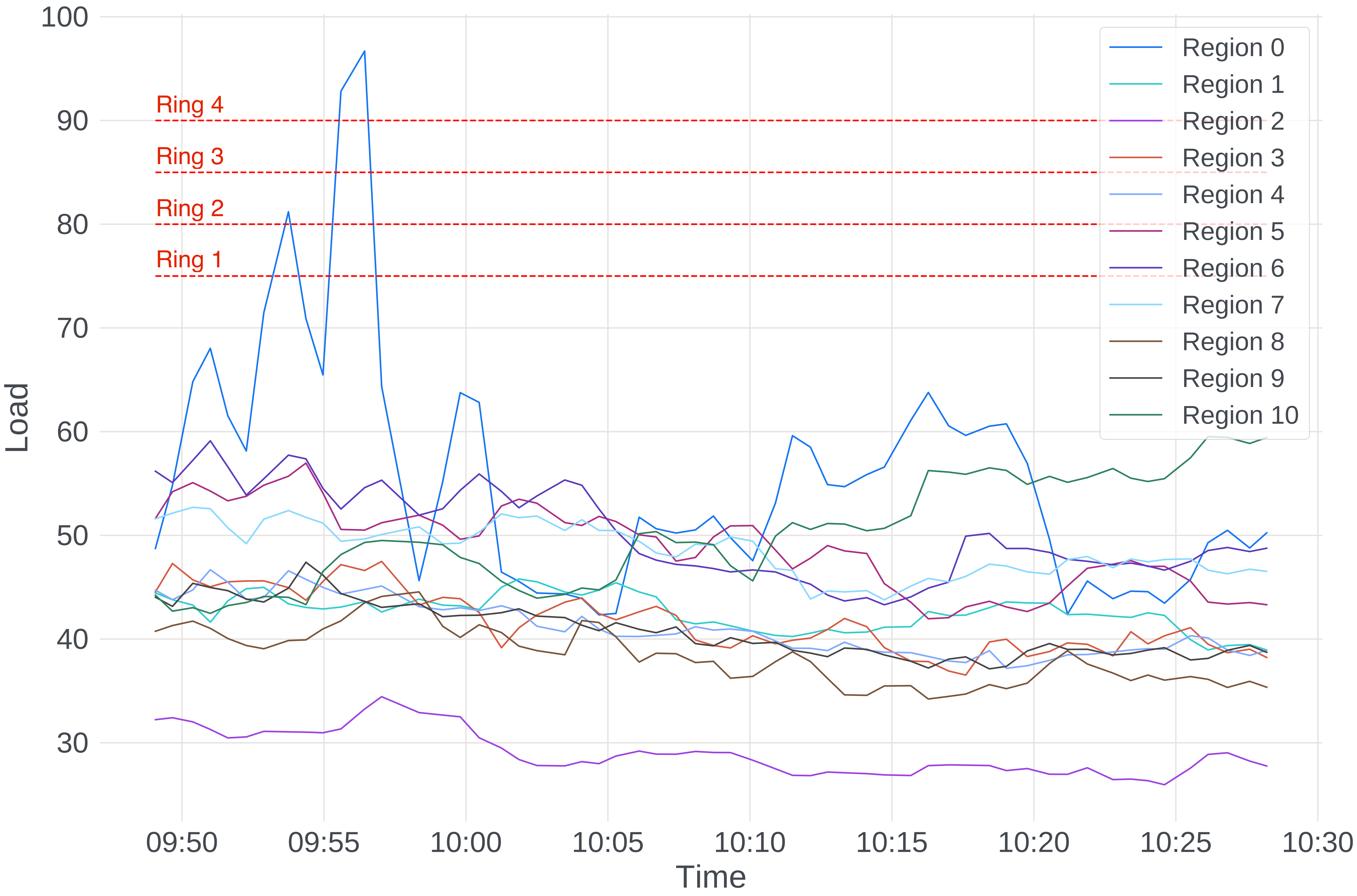
- Metrics: P50 avg request latency; CPU Instructions per request
- Designs
  - Baseline: Thrift RPC
  - SRLib
  - Remote SRProxy
- Simulated Payload:
  - Production avg request and avg response size
    - O(10^3) B
- 100K requests
- 3 trials per design

CPU Instructions/Request



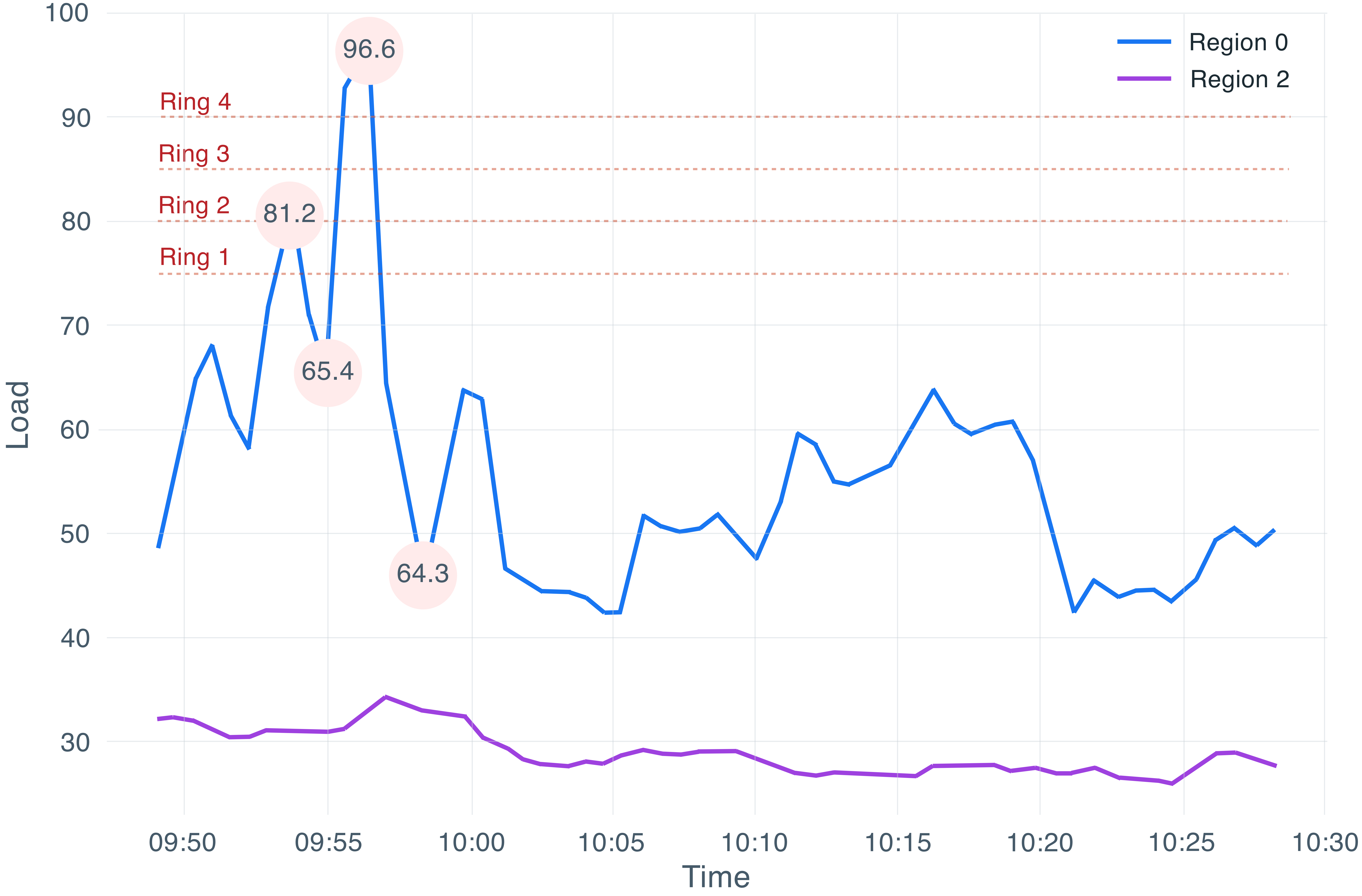
Cross-Region Load Shift

- Real-world Example



Cross-Region Load Shift

- Real-world Example
- 9:53 —> Region 0 Load = 81.2%
  - xRS Traffic Shifts
    - R0 -5.35%
    - R0 to R2 +5.35%
- 9:54 —> Region 0 Load = 65.47%
- 9:56 —> Region 0 Load = 96.69%
  - xRS Traffic Shifts
    - R0 -25%
    - R0 to R2 +25%
- 9:57 —> Region 0 Load = 64.34%



ServiceRouter

HYPERSCALE AND MINIMAL COST SERVICE MESH AT META

06 Summary

ServiceRouter’s massive RIB replication allows decentralizing L7 router management and to scale to millions of routers and proxies.

ServiceRouter routes 99% of the traffic with an optimized embedded library approach with astounding HW savings.

ServiceRouter’s source-based locality rings and xRS strike a balance between latency wins and load balancing.

Built-in **support for sharded services** which account for 68% of our RPCs **[not covered in this talk]**.

Soteris Demetriou I s.demetriou@imperial.ac.uk

Meta