KubeCon | CloudNativeCon

North America 2023

Reliable RPCs over Hybrid Clouds, and the End-to-End Argument

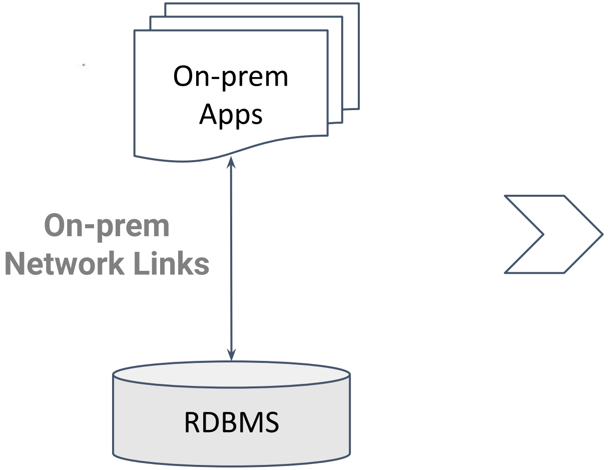
Wenbo Zhu, Vinod Lasrado

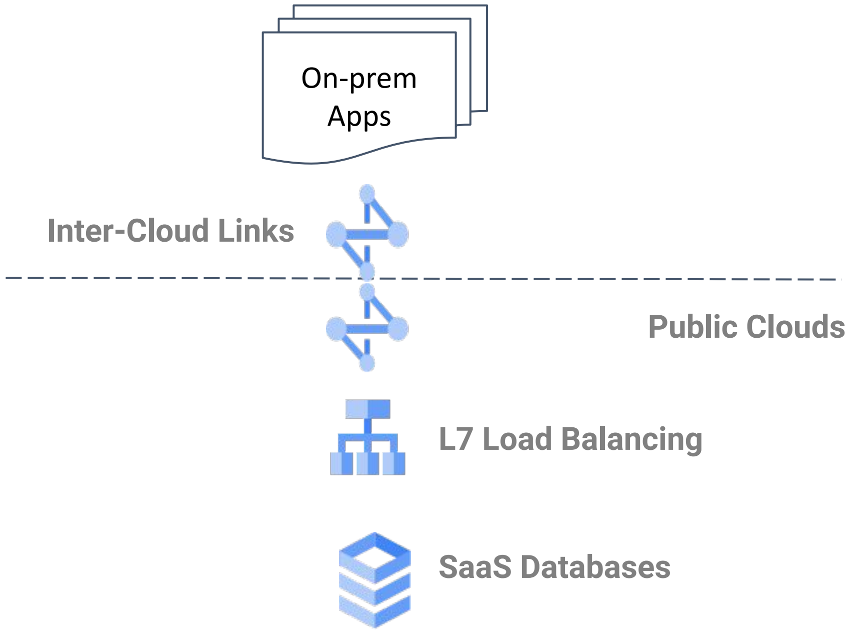
Google Inc.

Hybrid Clouds?

Background: on-prem storage/database services are moved to Clouds while applications stay on-prem.

Challenges: applications are not yet adapted to the Hybrid Cloud stack, esp. the failure modes.

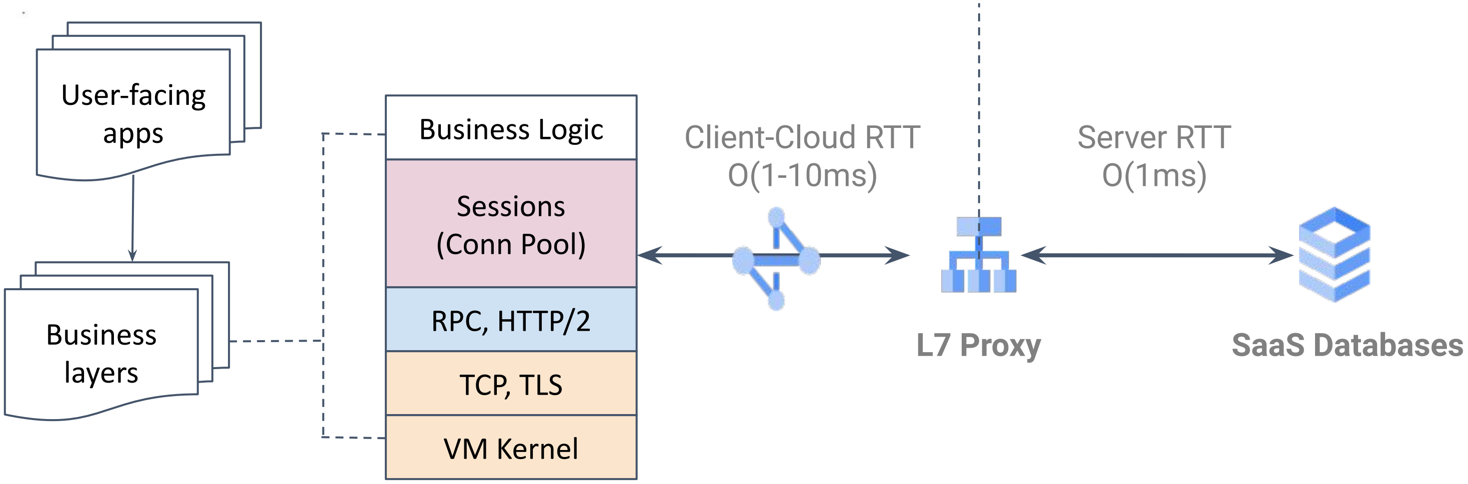




Case Study

A generalized version

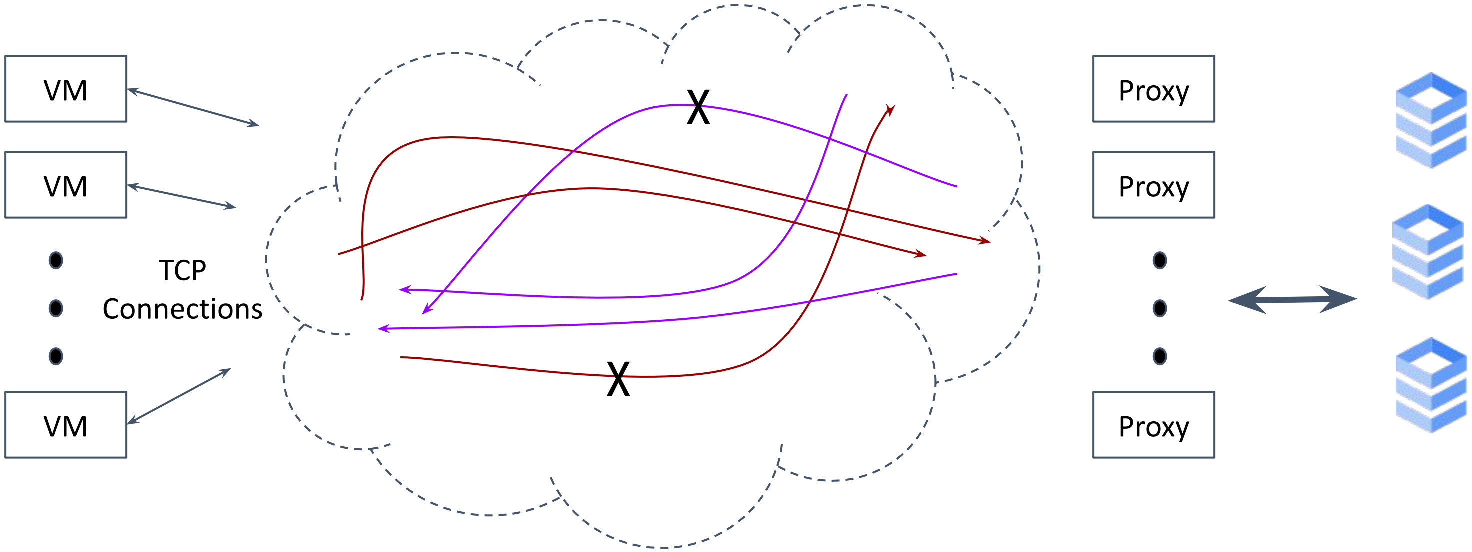
- Client-Cloud RTT >> Server RTT
- Transactional ops (writes) v.s. non-transactional ops (reads)
- Timed-out reads may cause visible application errors while timed-out writes will abort the tx
- Connection-based affinity with client-side connection pools to distribute sessions



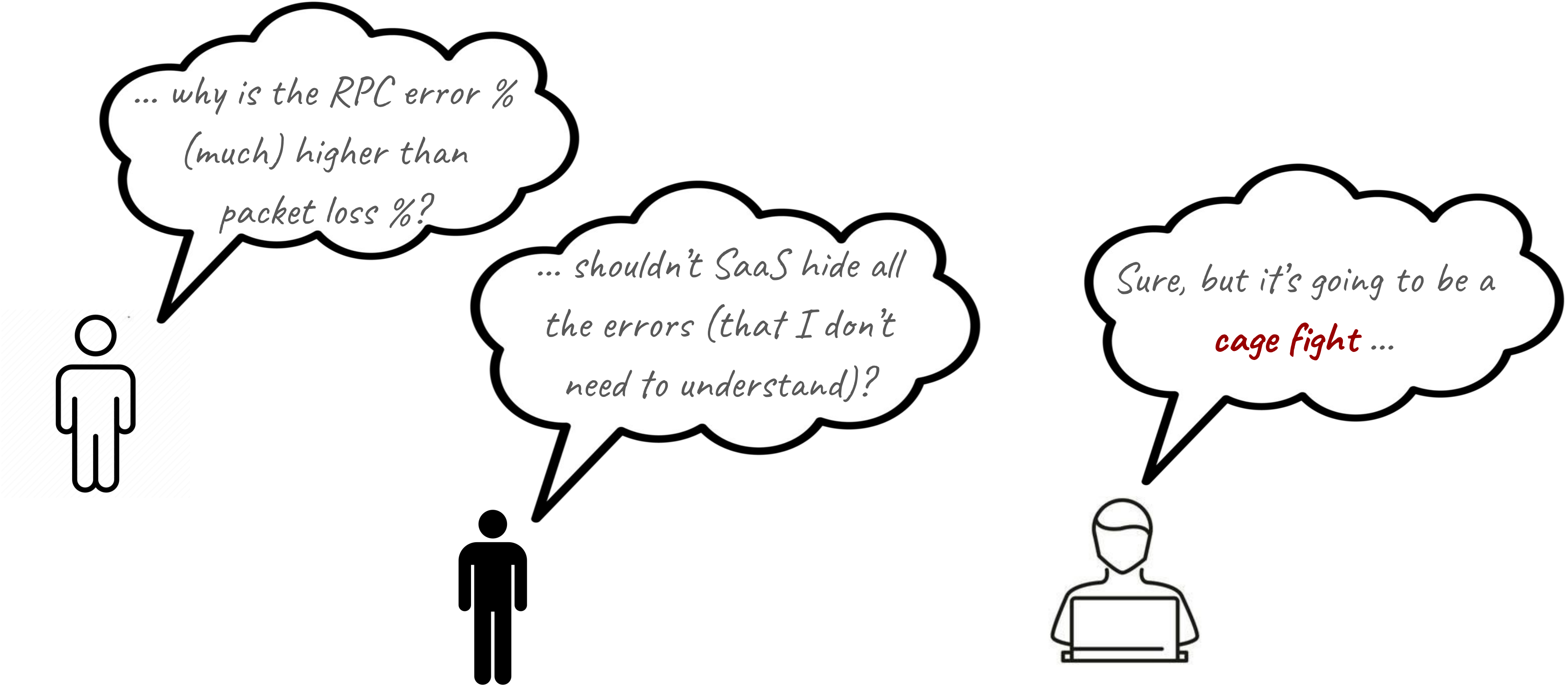
Failure Modes

L3 failures between the client and Cloud

- Failures affect only a small fraction of unidirectional “flows” (hence TCP connections)
- Failures often result in short-lived black holes but partial packet losses are also possible
- Transient server or proxy failures (overloading) may happen which are not correlated to L3 failures



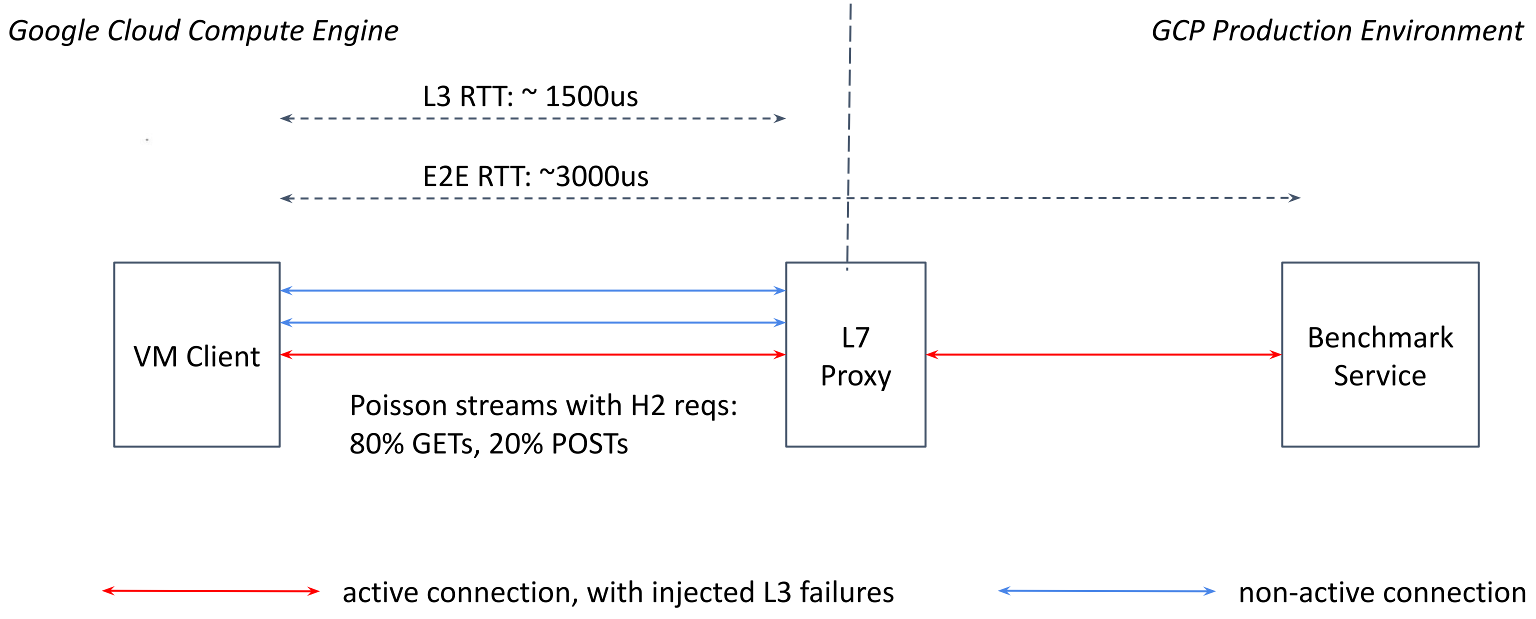
Key Questions



Evaluations

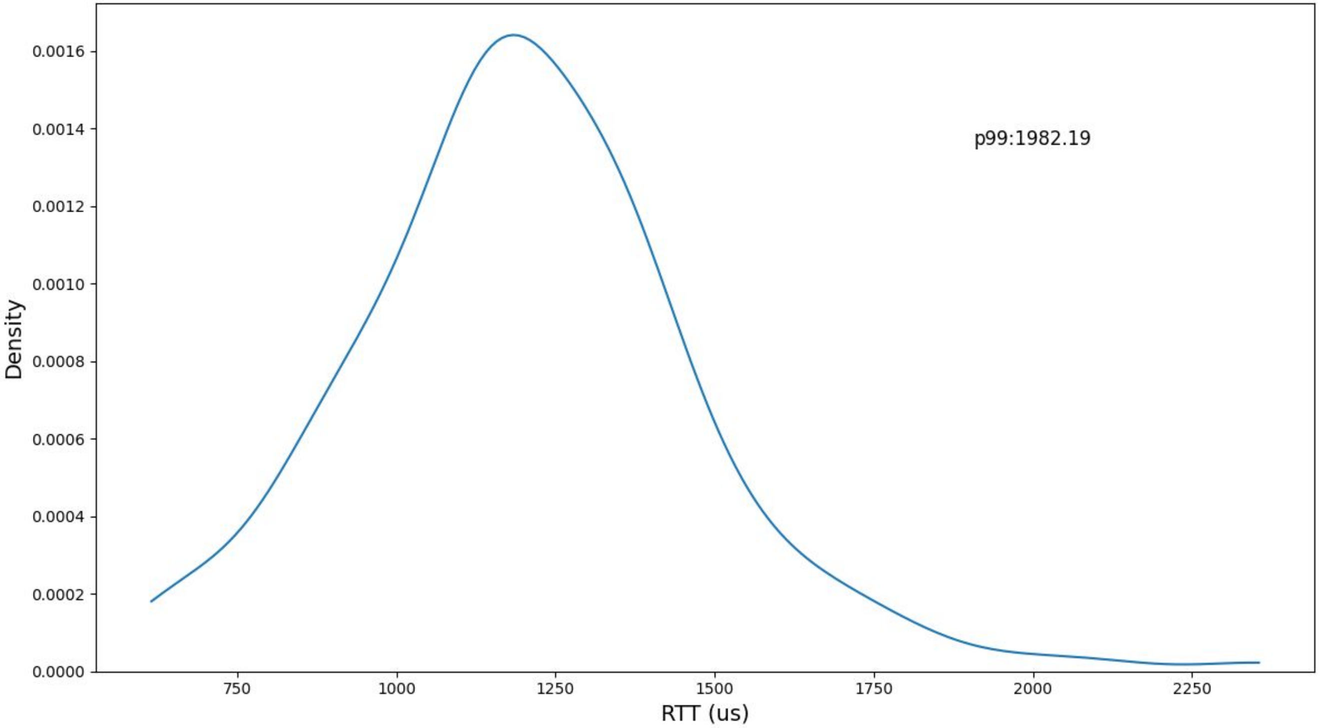
A general-purpose benchmark framework

- Evaluate impacts of different workload and environment parameters
- Evaluate effectiveness of different recovery parameters



L3 RTT Distribution

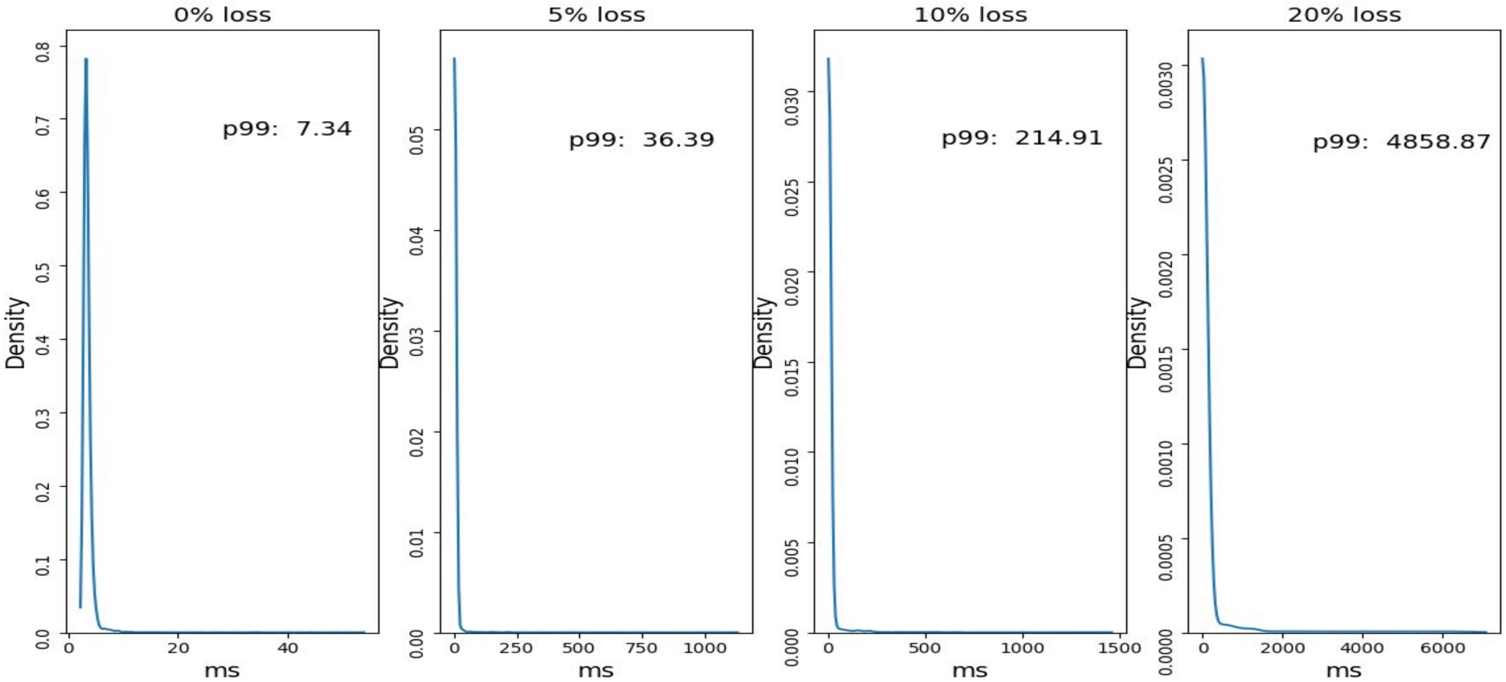
Probability Density Function. P50:1187 P99: 1982 us



RPC Tail Latencies

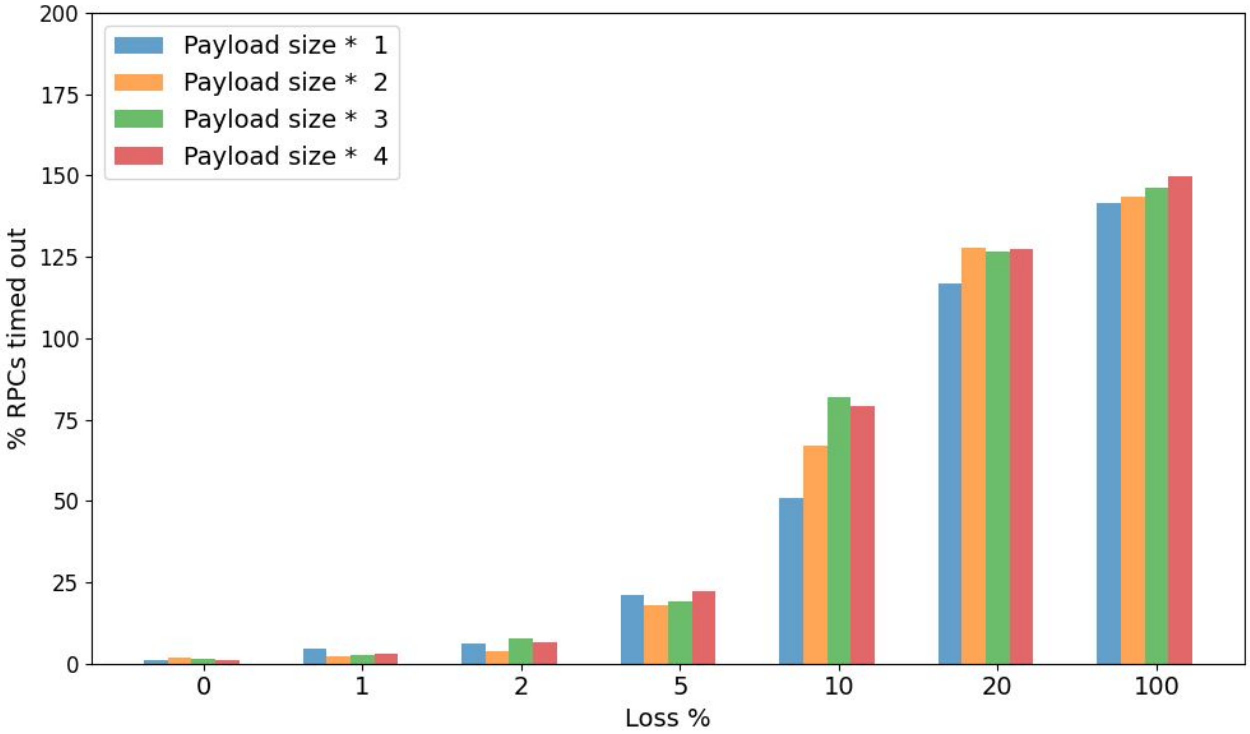
KubeCon CloudNativeCon North America 2023

E2E RPC latencies (ms) at different loss rates. 20 rpc/s



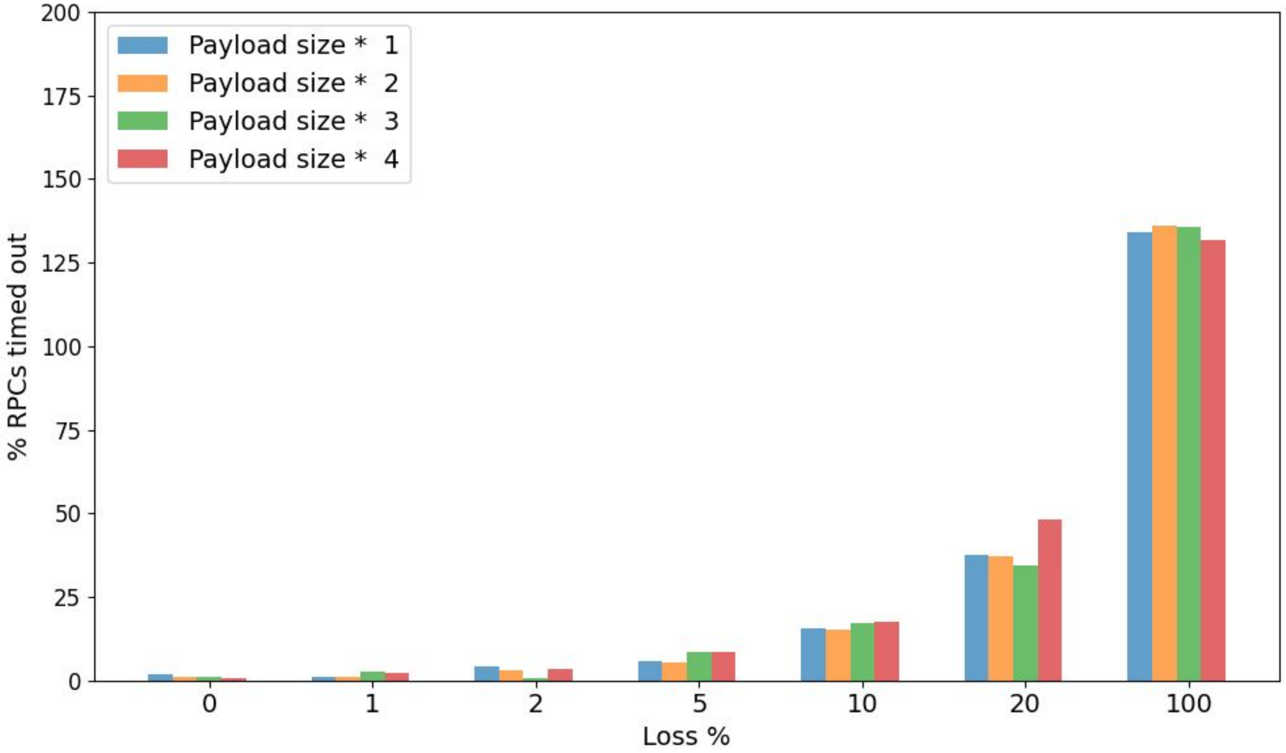
RPC Timeouts (Payload Sizes)

Timeout is set to 4 *RTT (median E2E). RPCs are not cancelled, 20 rpc/s



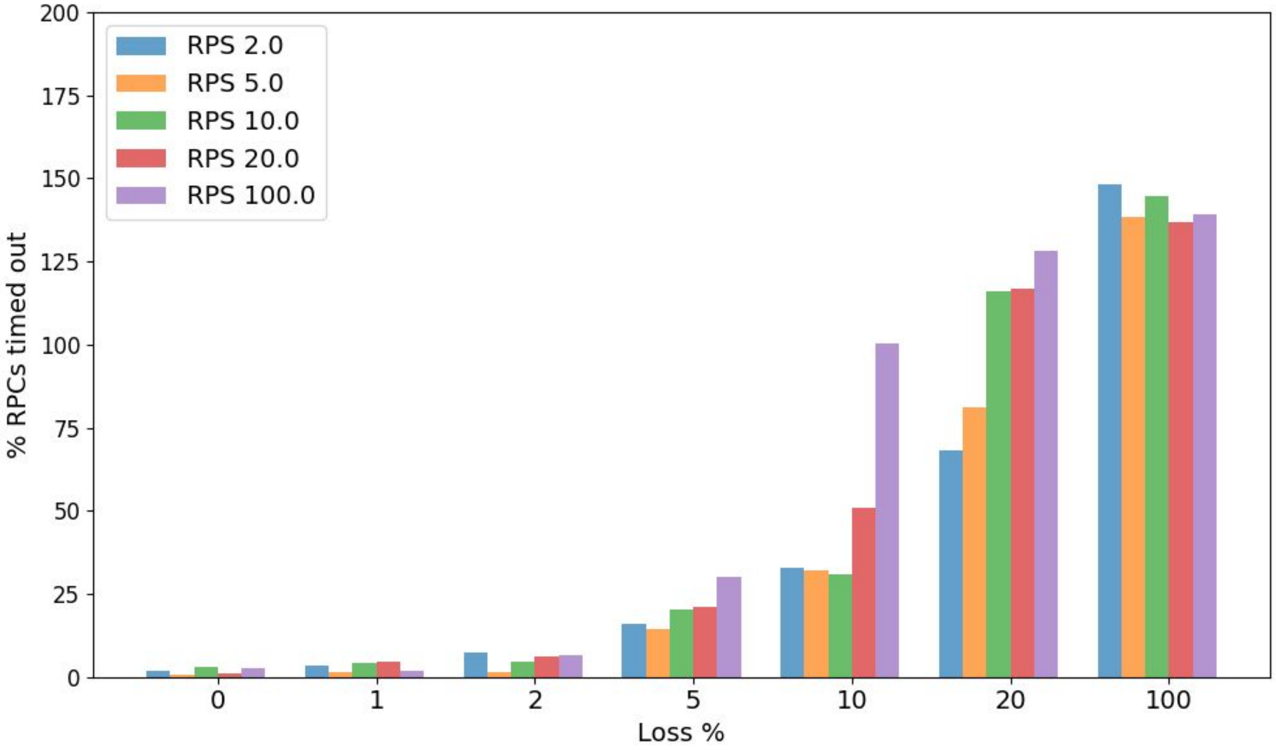
H3/QUIC (Payload sizes)

Timeout is set to 4 *RTT (median E2E). RPCs are not cancelled. 20 rpc/s



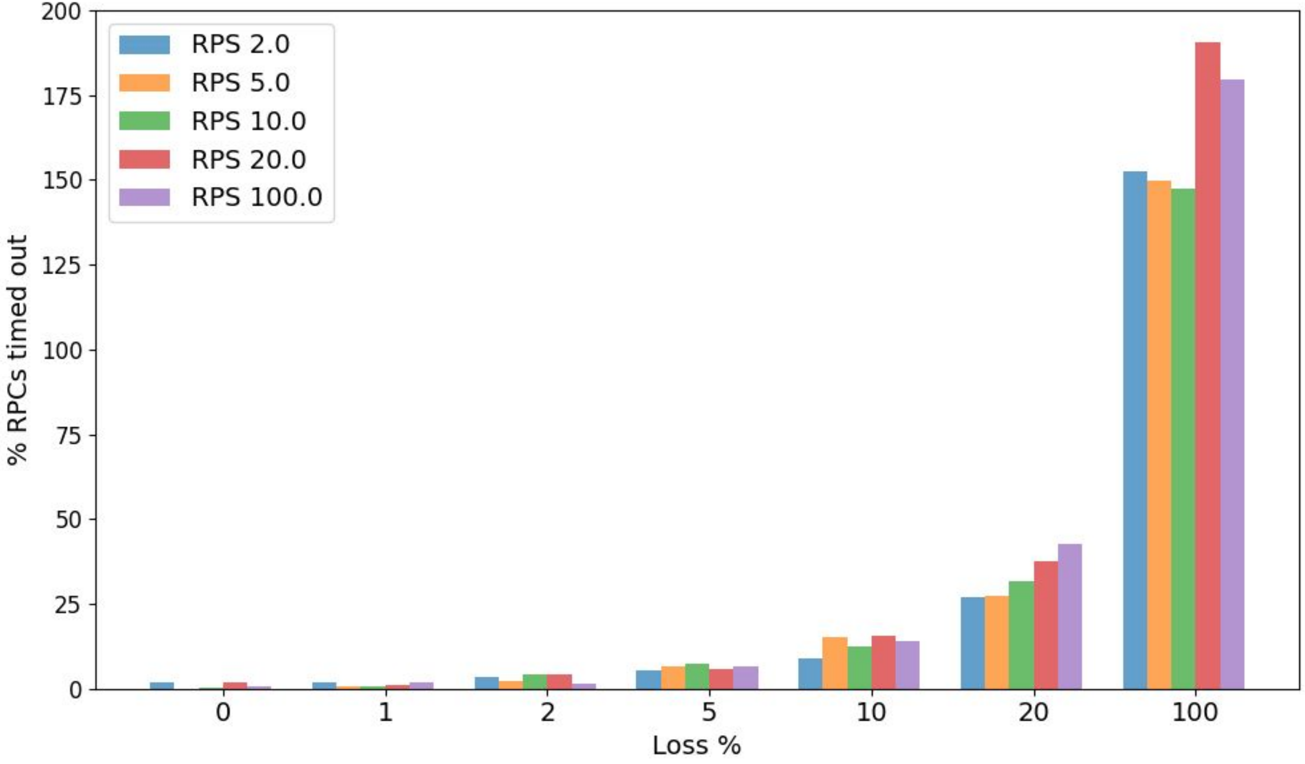
RPC Timeouts (Request Rates)

Timeout is set to 4 *RTT (median E2E). RPCs are not cancelled.



H3/QUIC (Request Rates)

Timeout is set to 4 *RTT (median E2E). RPCs are not cancelled.

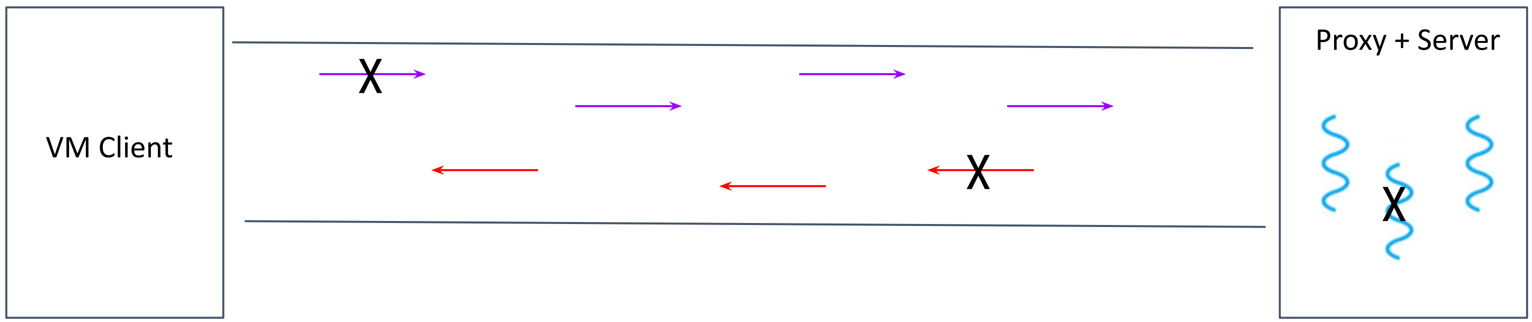


Failure Detection

Fail-fast: detect (suspect) a failed connection (or server) timely

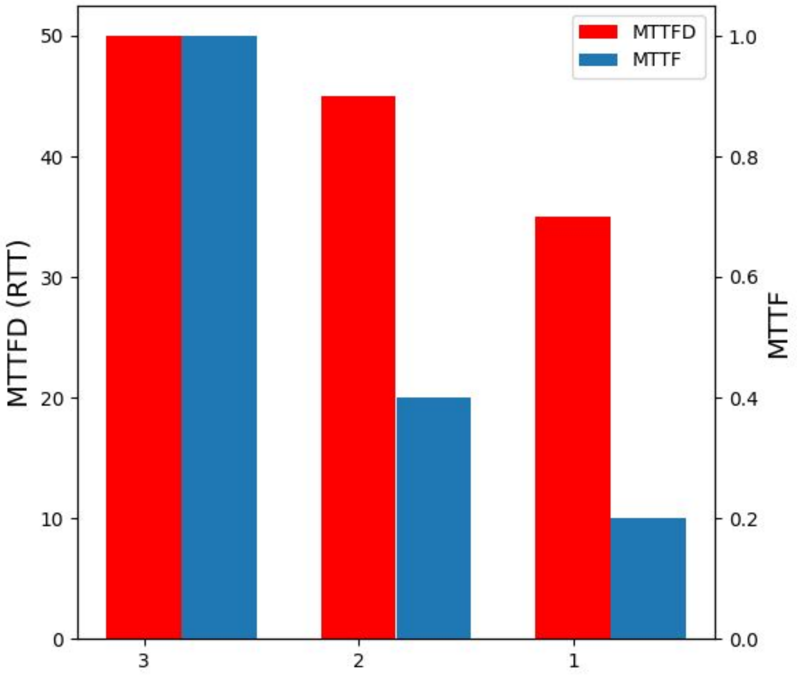
Detection params

- A: a factor of mean RTT
- B: minimum number of timed-out acks
- ack_num: number of pending L7 acks in the runtime
- timeout = 12 * A * RTT - 2 * A * RTT * min(4, ack_num)



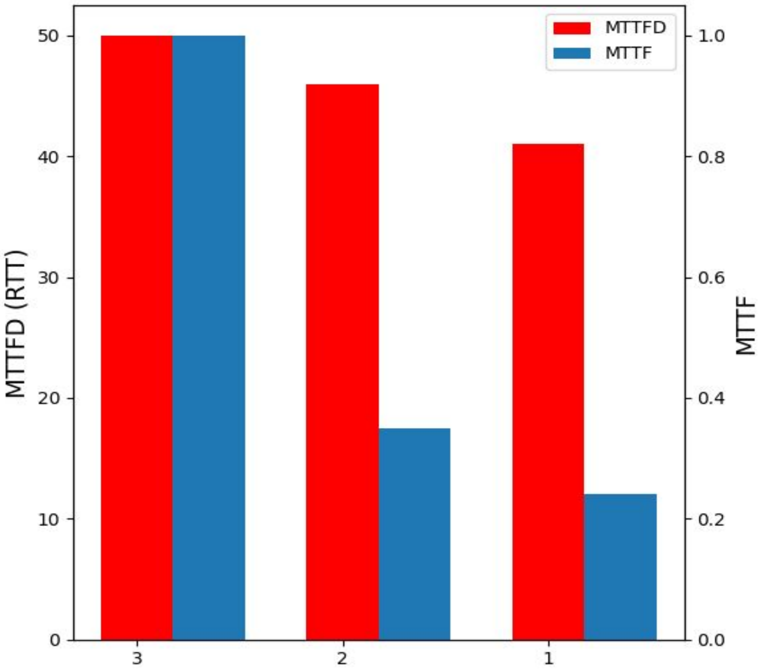
Evaluation Results

Timeliness: MTTFD with injected failures



A (B = 3)

False-positive: MTTF with failure-free runs



B (A = 3)

Failure Recovery

<u>Decoupled from failure detection</u>

- Shorter timeout for reads, longer timeout for writes, as decided by application requirements

Upon a failed connection

- New requests will use a different connection
- Pending reads will be auto-retried immediately on a different connection but only once
- Pending writes will run to completion or will be cancelled after a timeout

Conditions for minimum visible errors

- read timeout > MTTFD, e.g. 100-200ms
- write timeout > the failure window, e.g. 5s

False-positive detection => duplicated reads, misrouted writes

End-to-End Arguments

<u>E2E Principle</u>: functions such as failure detection and mitigation may "completely and correctly be implemented only" at an application level, and implementing those functions completely and correctly from a lower level is not possible.

Key takeaways

- Leverage service-level RPC semantics in failure detection and recovery
- Passive detections to adapt to runtime conditions
- Graceful recovery to handle short failure windows

Low-level recovery

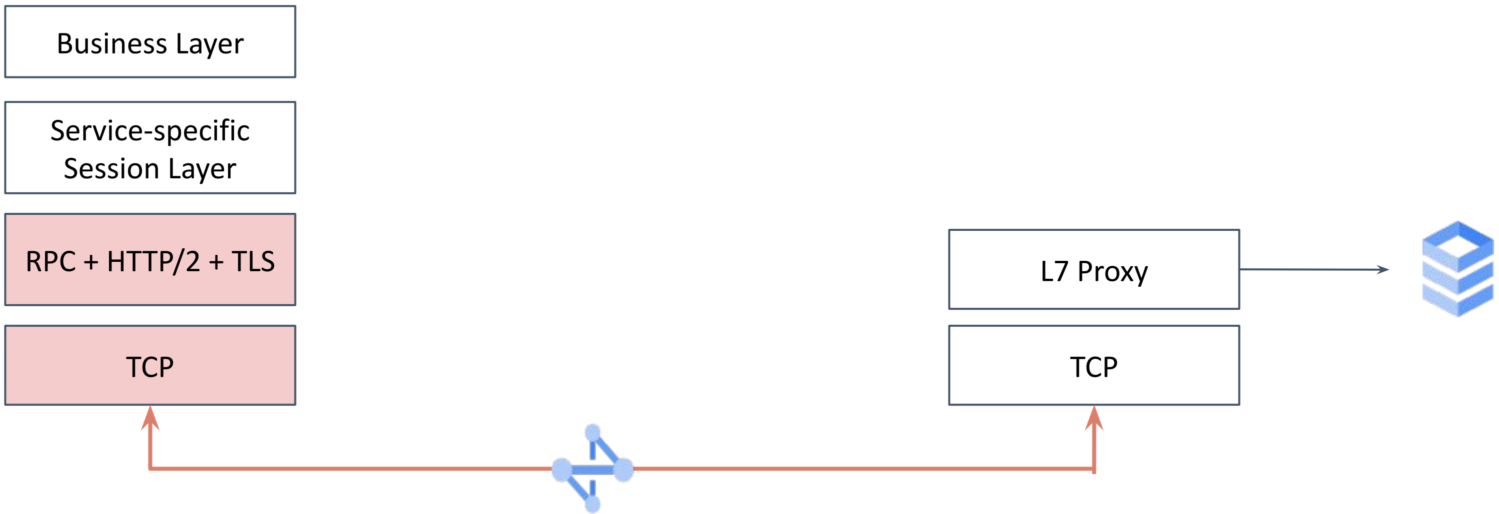
- Desired if it can be done quickly and transparently (without tearing down TCP connections)
- An optimization aligned with the E2E principle

Production Environments?

No silver bullet

Monitoring data is critical to identify root causes and to evaluate solutions

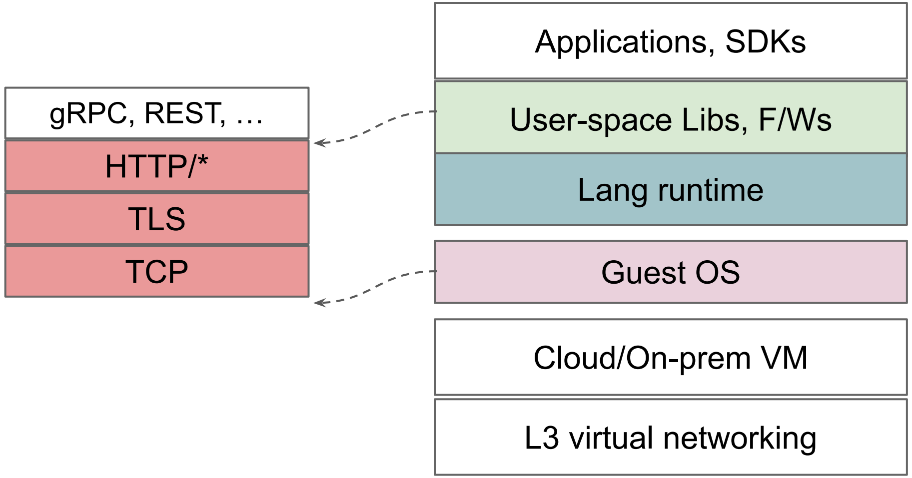
- Client-side transport layer
- Correlation between userspace/guest transports and L3 networking



eBPF Transport Monitoring

github.com/google/ebpf-transport-monitoring

- A standalone toolkit
- Correlation between kernel events (TCP) and user-space events (TLS, HTTP/2)
- Future work: more languages, compact event encoding, QUIC/H3 …



Contact Us

Feedback and contributions

- eBPF transport monitoring
- Benchmark framework
- Example data sets, analytical methods

Discussion: *github.com/google/ebpf-transport-monitoring/issues*

Email: vlm@google.com, web@google.com

Please scan the QR Code above to leave feedback on this session