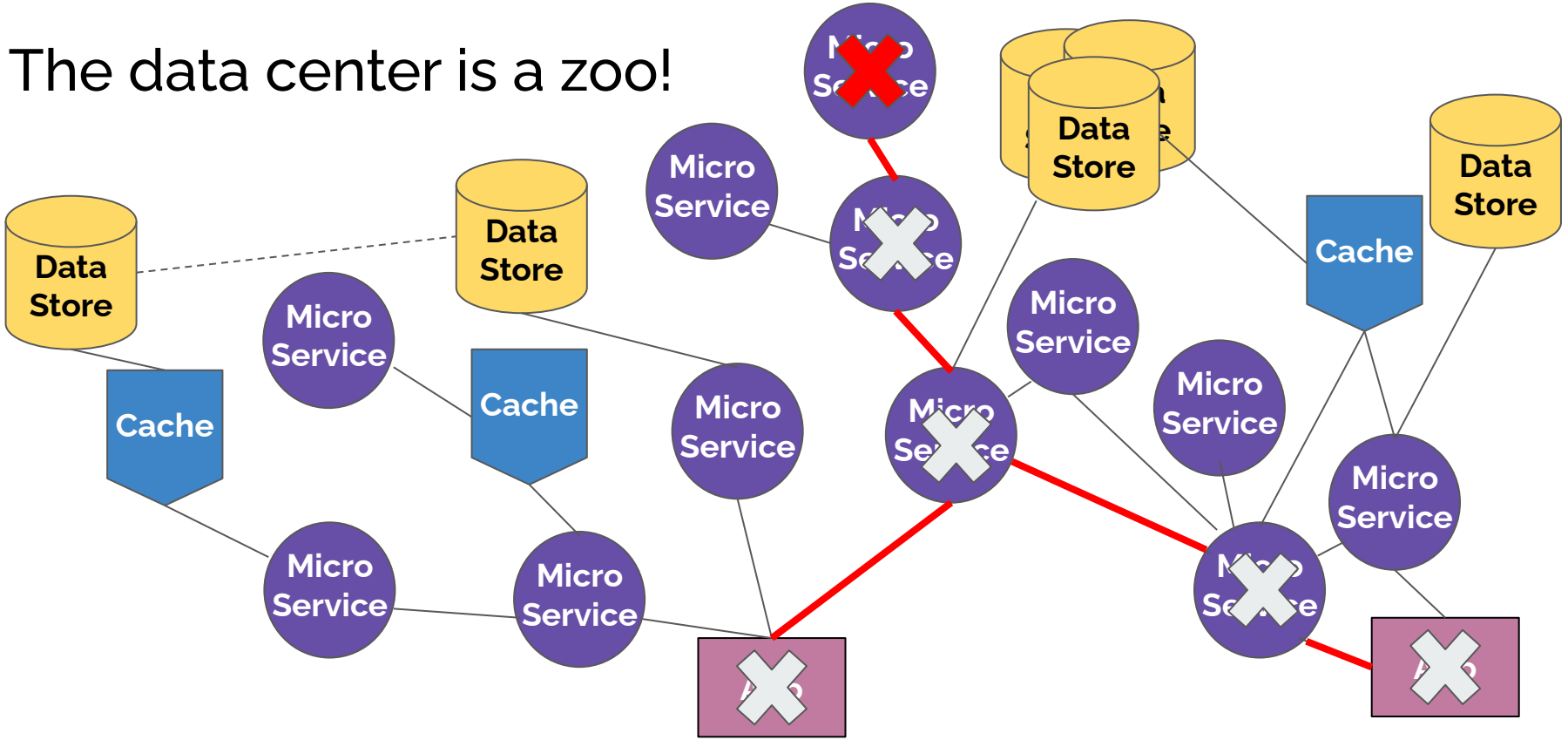
uKharon

A Membership Service for Microsecond Applications

Rachid Guerraoui , Antoine Murat , Javier Picorel,
Athanasios Xygkis, Huabing Yan, Pengfei Zuo

The data center is a zoo!



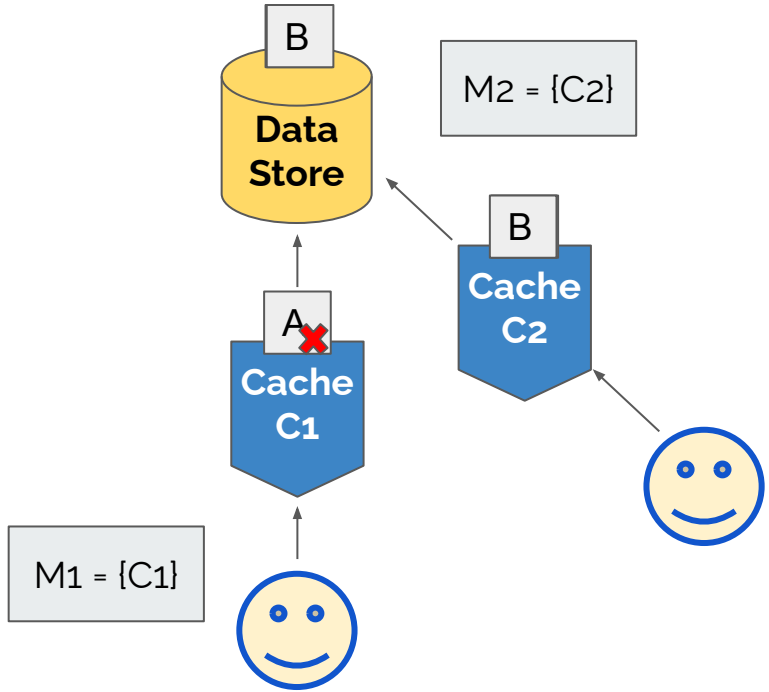
Today more than ever, failures are first class citizens!

How do we usually deal with failures?

Using etcd or ZooKeeper

Membership services:

- Are **reliable** configuration **stores**
- **Update** their configuration **sequence**
- **Invalidate** old memberships



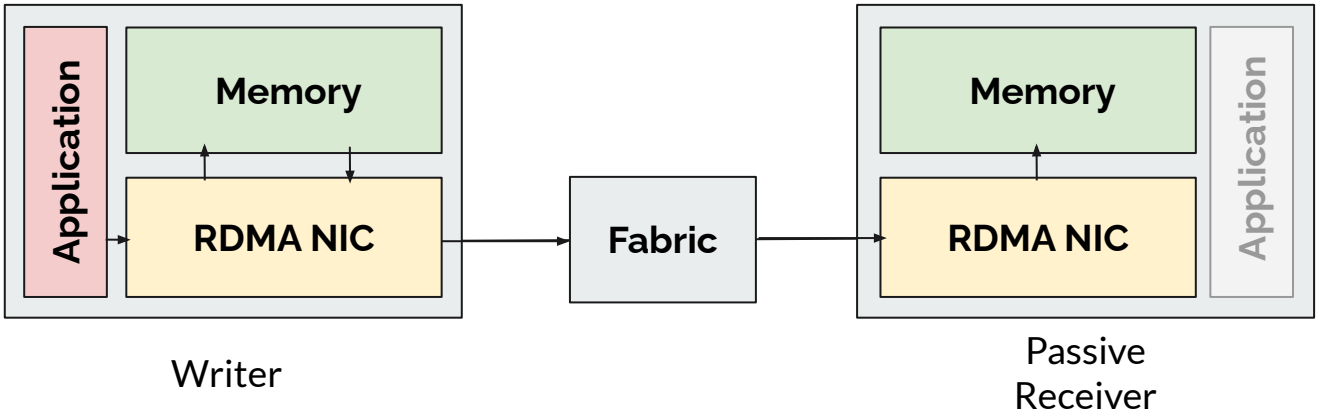
The problem is NOT solved at the microsecond scale.

Our Contribution:

- A Microsecond-scale Membership Service
- Detects failures in **15 us**
- Updates the membership in **10us**
- Invalidates old memberships in **25us**

uKharon reacts to failures in 50us by leveraging RDMA!

Remote Direct Memory Access (RDMA)



Allows µs-scale communication

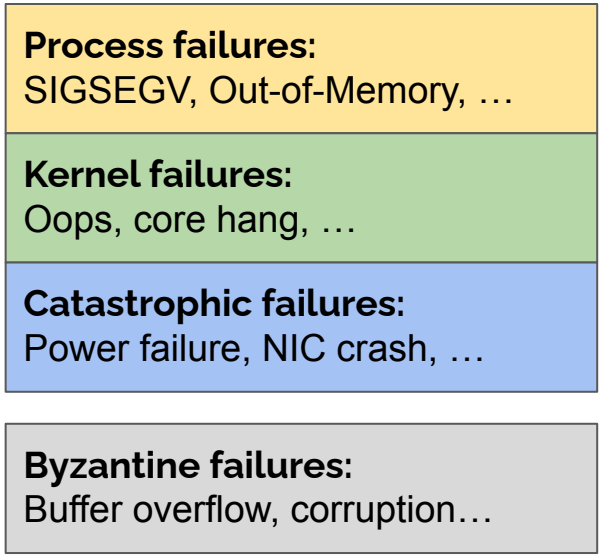
Detects failures

Updates the membership

Invalidates old memberships

Microsecond-scale failure detection

- Timeouts do not help avoiding false positives
- Not all failures are equal



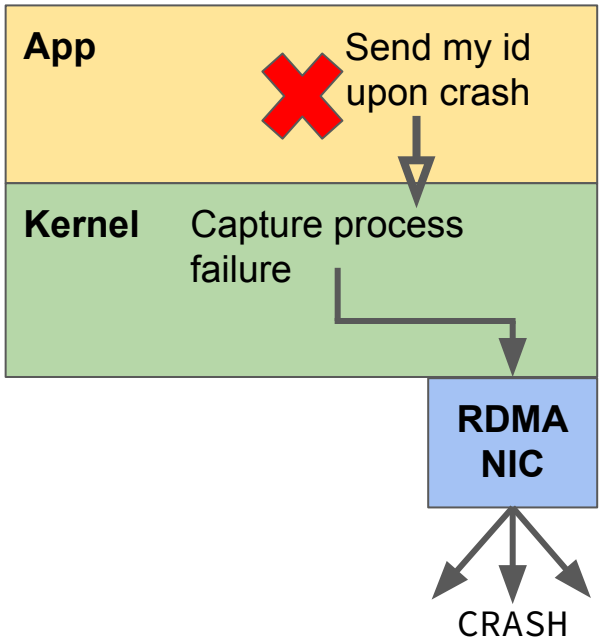
Help from kernel → No timeouts

Help from NIC → No timeouts

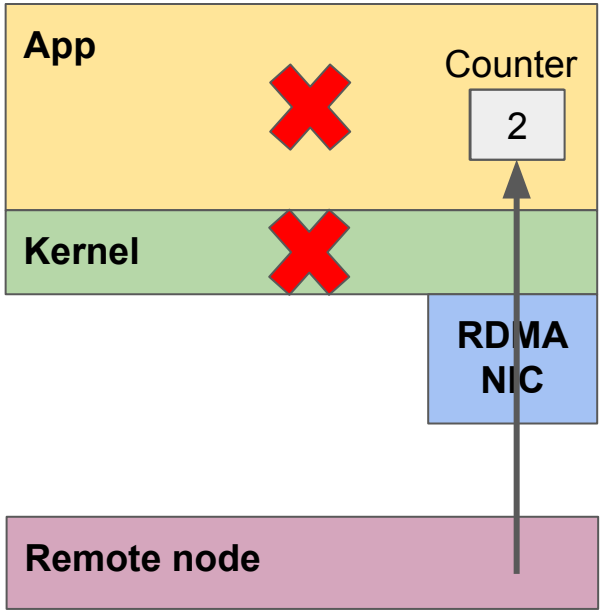
Catch-all → Timeout-based

Microsecond-scale failure detectors

Process failures



Kernel failures



Take network synchrony out of the equation ⟶ fast and accurate failure detection

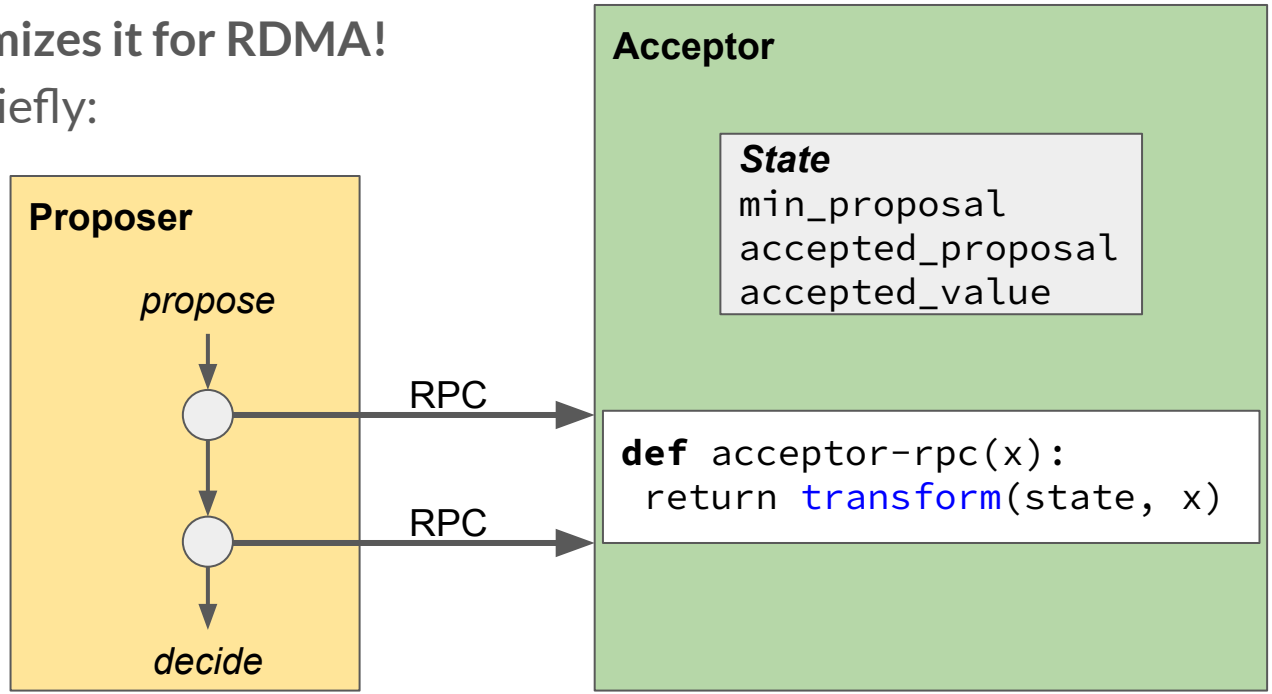
Detects failures

Updates the membership

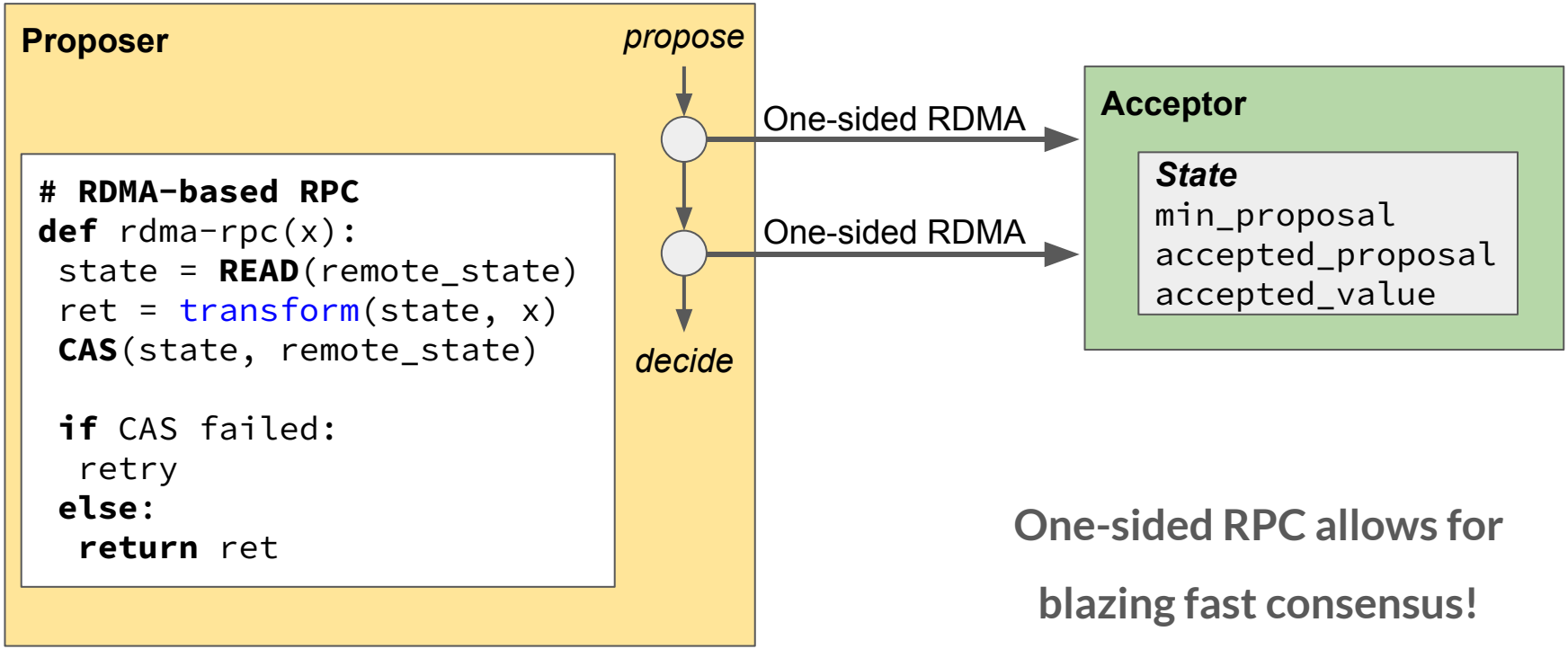
Invalidates old memberships

Microsecond-scale replication

- Uses Paxos
- **But optimizes it for RDMA!**
- Paxos, briefly:



One-sided Paxos



One-sided RPC allows for blazing fast consensus!

Detects failures

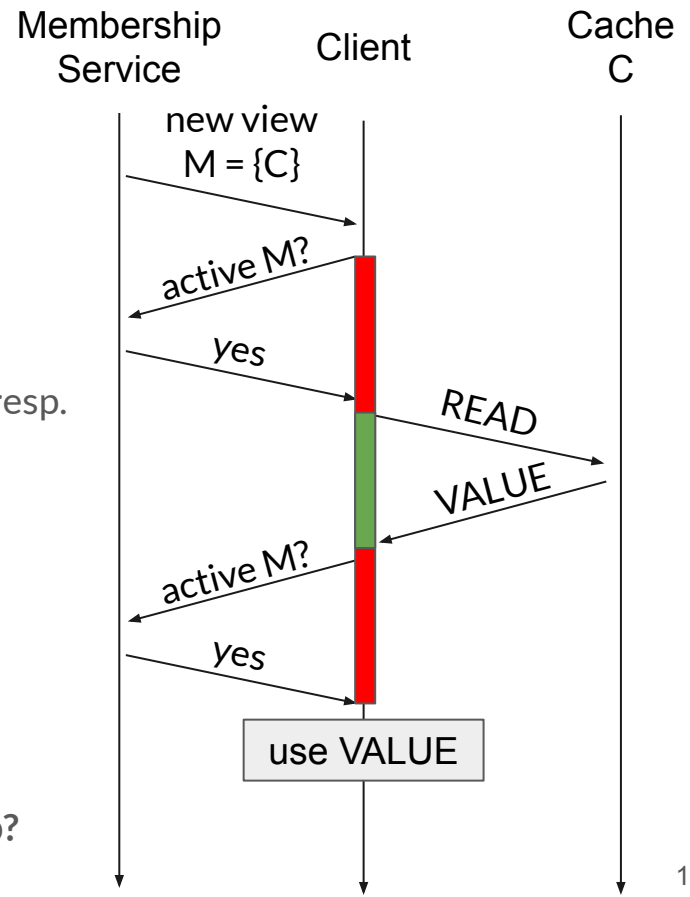
Updates the membership

Invalidates old memberships

Membership invalidation

- What is *the* active membership?
- Learn via `Active`(*Membership*) → *bool*
- `Active(M) == true` → M was active between invoc. and resp.
- High latency

How to make `Active` cheap?

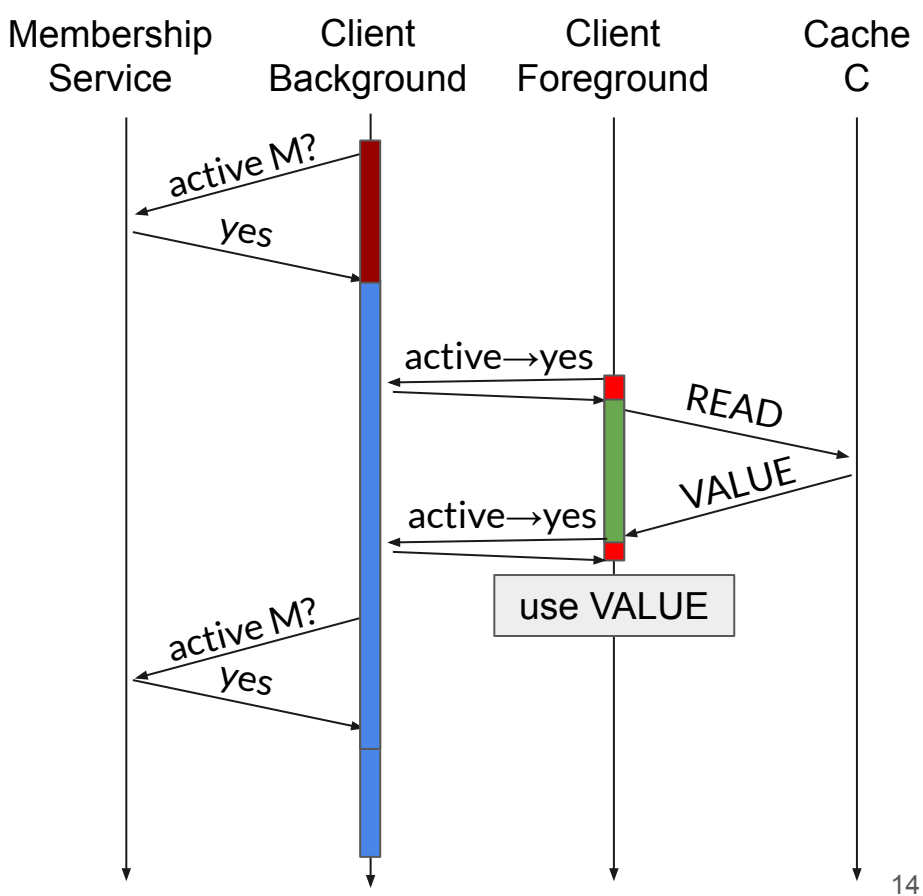


Microsecond-scale leases

- Clients lease `Active`'s response for ~20µs
- Renew their lease in the background

- NO synchronized clocks required
  - Only bounded clock drift for safety
- Delays view changes by no more than ~20µs

Leases make `Active` take ~40ns



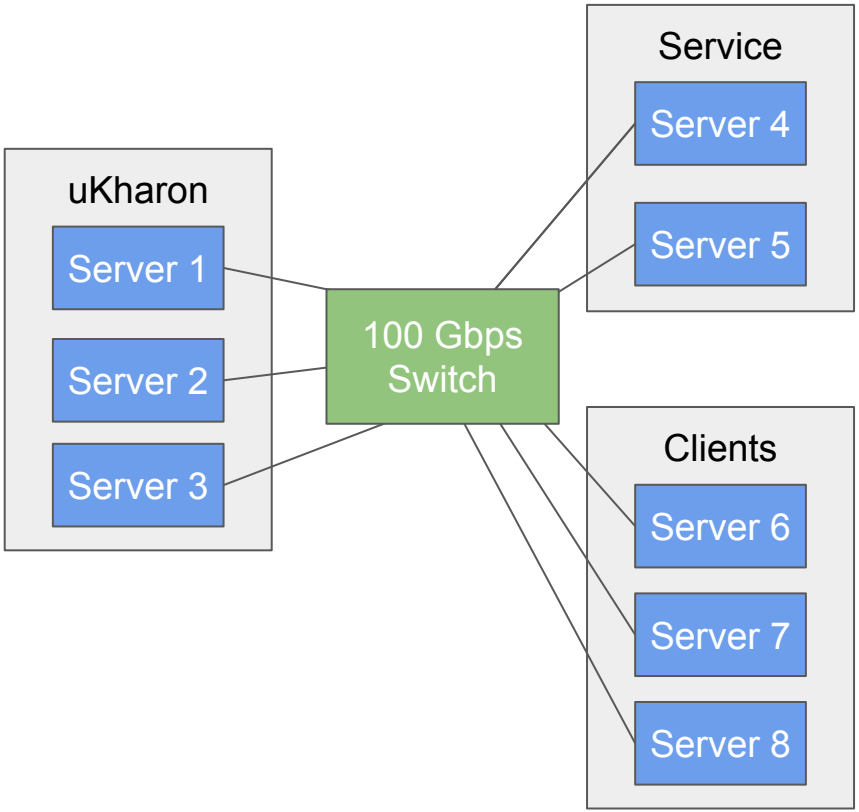
Detects failures

Updates the membership

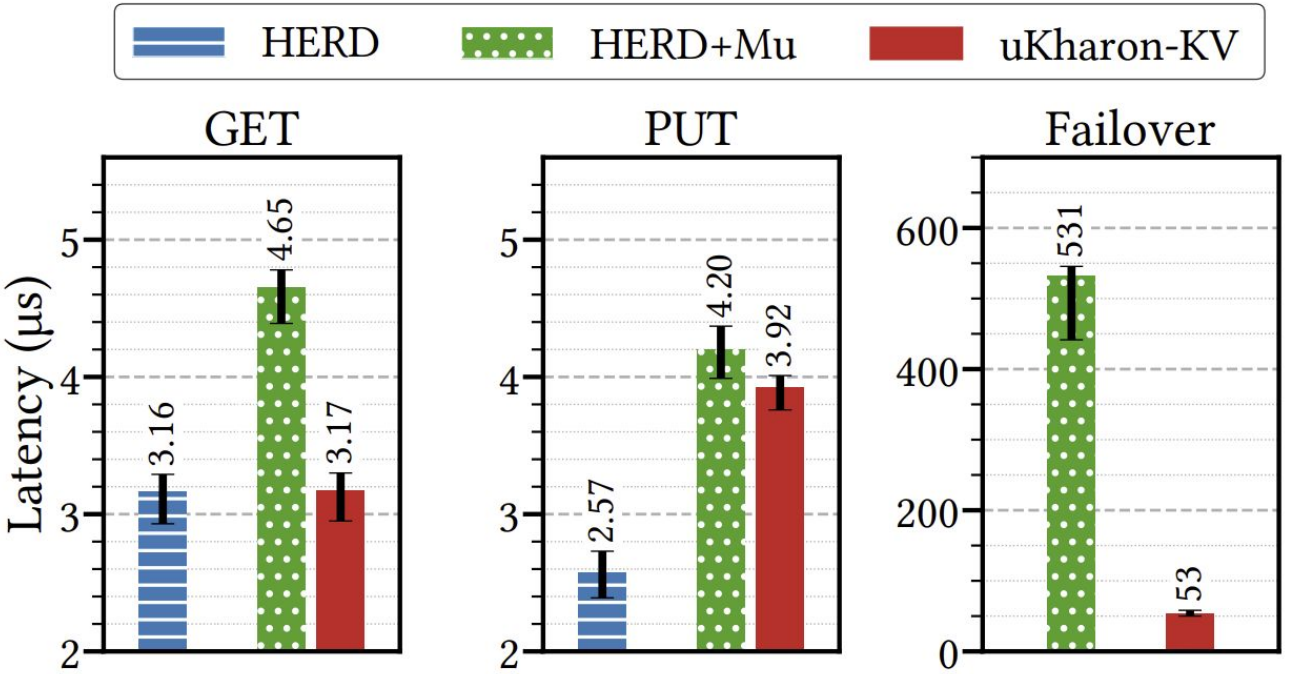
Invalidates old memberships

How does it perform?

Evaluation: setup

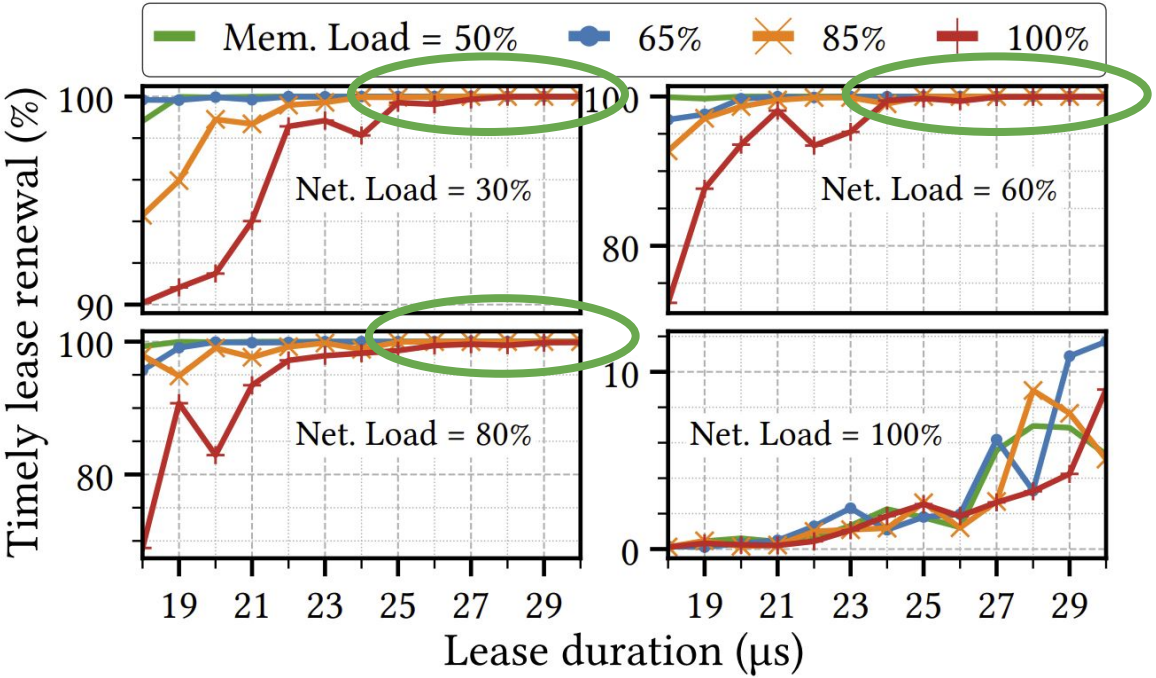


Evaluation: Replicating a KV-Store



uKharon helps beating the state of the art!

Evaluation: Are leases renewed in time?



Microsecond leases are stable!

Conclusion

- uKharon:
  - A membership service for µs-fast failover (down to 50µs)
  - Easy to integrate
  - Only 40ns latency overhead

github.com/LPD-EPFL/ukharon

Check out our paper for more details!