Robust and High-Performance Wide-Area Consensus Protocols

PhD Public Defense

Pasindu Tennage

Thesis director: Bryan Ford
Thesis co-director: Lefteris Kokoris-Kogias

DEDIS

EPFL


RESTAURANT
OPEN


RESTAURANT
BISTRO

14:58 Let’s meet at Cafe du flon

14:58 Let’s meet at Pizza place

15:50 Sure, I am good with that

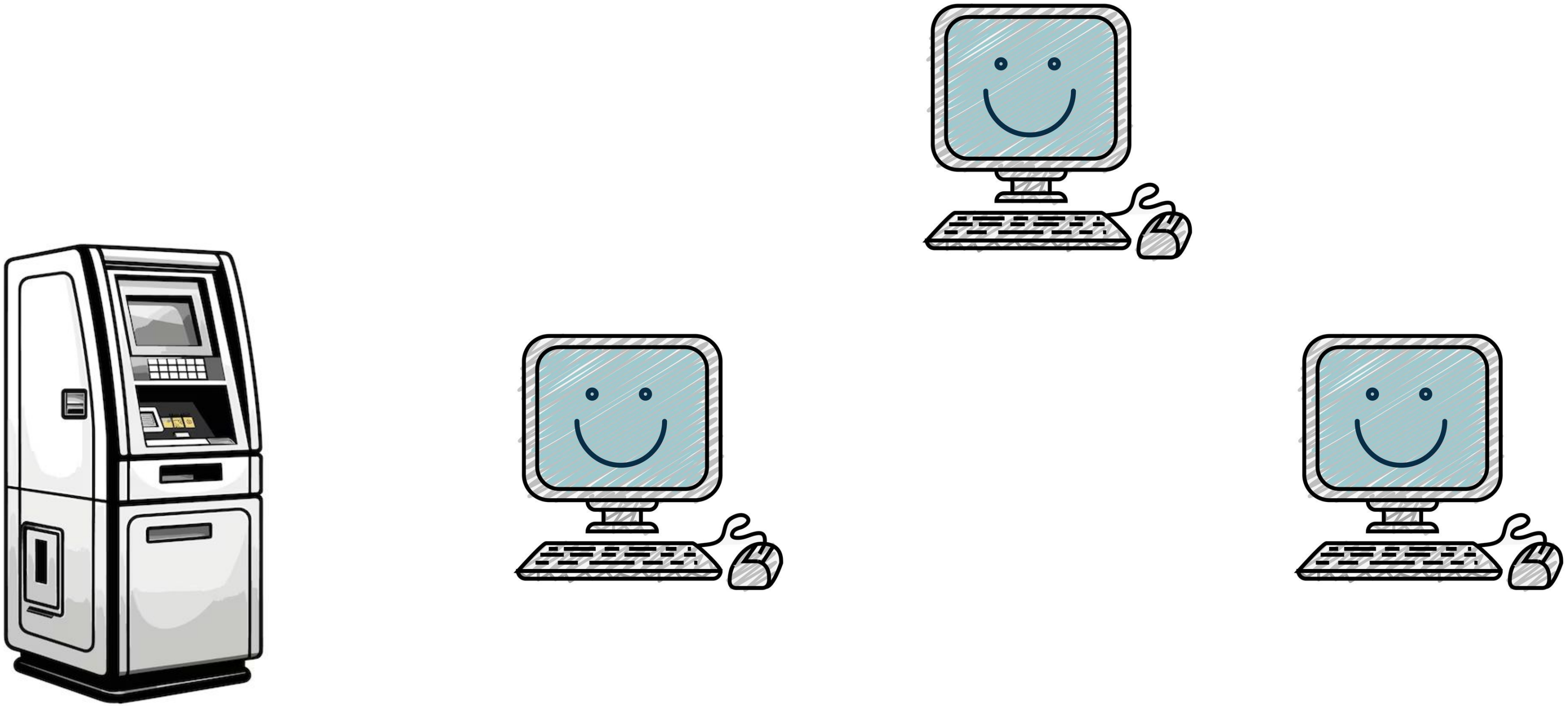
16:58 Which one you mean?

17:58 I assume it is Cafe du flon?

Distributed Agreement is Hard

Consensus: an agreement about something

Consensus

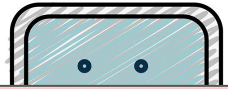


Albert	200

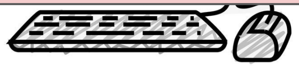
Albert	200

Albert	200

Albert	200



Inconsistent State



Albert	**100**

Albert	200

Albert	**100**

Consensus

Albert	**100**

Albert	**100**

Consensus protocols enable a distributed set of machines to agree on the same value

Robust and High-Performance Wide-Area **Consensus Protocols**

DEDIS

EPFL

Robust and High-Performance **Wide-Area** Consensus Protocols

DEDIS

EPFL

Distributed Consensus

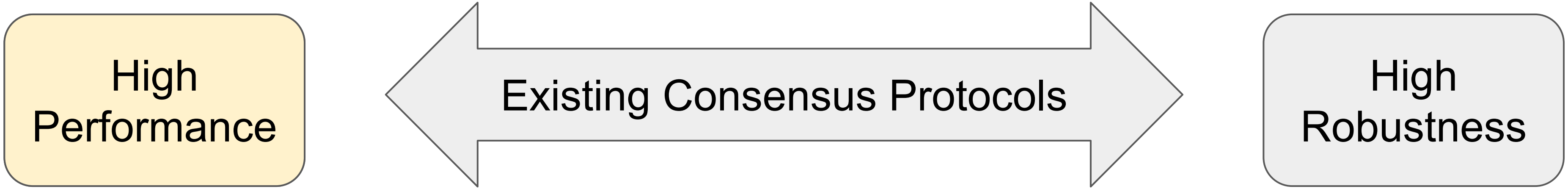
80s/90s
FLP
Paxos
View Stamp

2000
PBFT
Fast Paxos

2010
Mencius
EPaxos
Raft
WPaxos
Pig Paxos

2020
Hotstuff
Dag-Rider

Started thesis

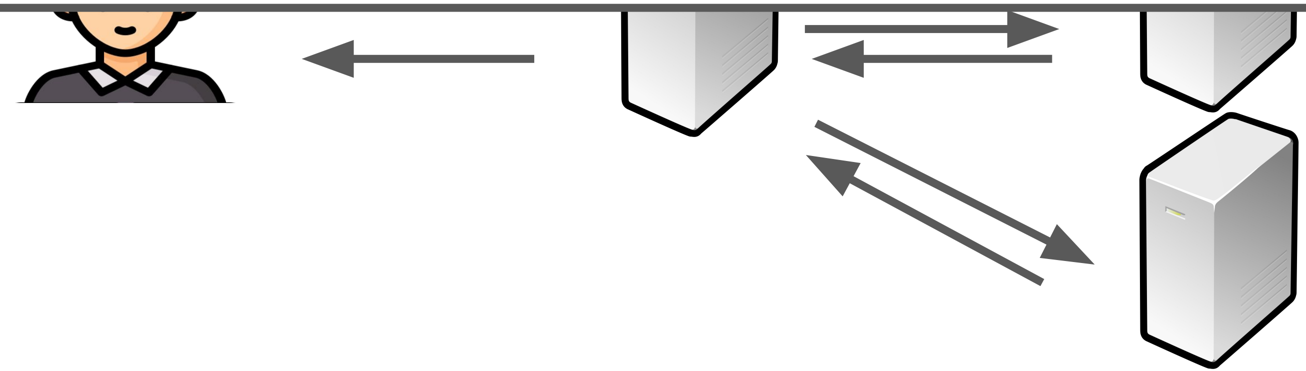

High
Performance
Existing Consensus Protocols
High
Robustness

High Performance using Leader-based Consensus

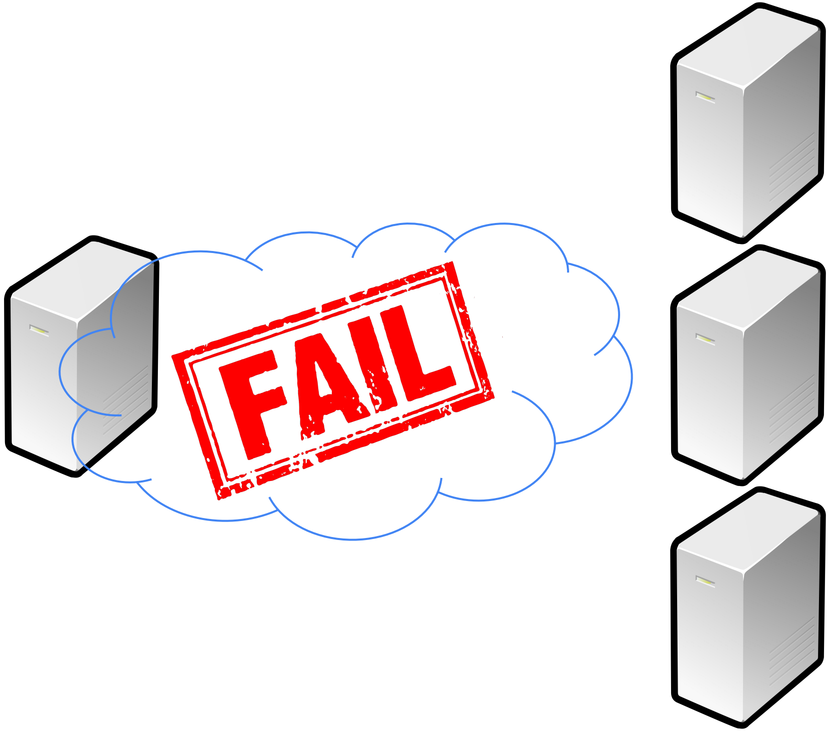
ZooKeeper: Wait-free coordination for Internet-scale systems

Patrick Hunt and Mahadev Konar
Yahoo! Grid
{phunt,mahadev}@yahoo-inc.com

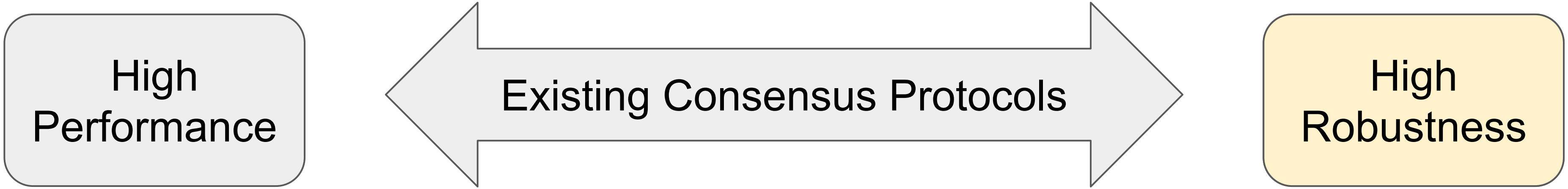
Flavio P. Junqueira and Benjamin Reed
Yahoo! Research
{fpj,breed}@yahoo-inc.com



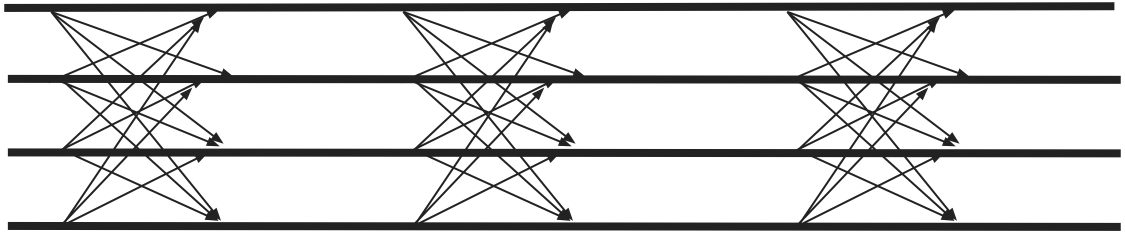
Robustness Problem of Leader Based Protocols



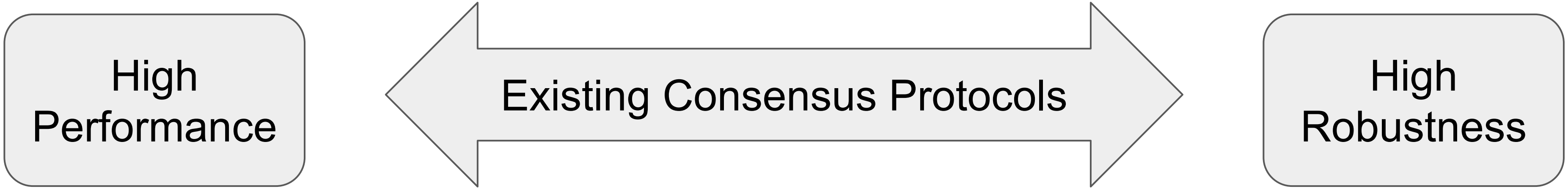
- Network partition.
- Link failures.
- DDoS attacks.
- Leader crash.


High
Performance
Existing Consensus Protocols
High
Robustness

Robust randomized consensus protocols



- Less efficient.
- Hard to understand.
- Rarely deployed.



Can we have the best of both worlds?

Robust and High-Performance Wide-Area Consensus Protocols

DEDIS

EPFL

Robust and High-Performance Wide-Area Consensus Protocols

DEDIS

EPFL

Thesis Contributions

Baxos

RACS-SADL

QuePaxa

Mahi-Mahi

Thesis Contributions

Baxos	Robustness against leader-targeted attacks	
RACS-SADL	Asynchronous liveness and high scalability	(IEEE CLOUD 2025)
QuePaxa	Mechanisms to avoid tyranny of timeout problems in consensus	(ACM SOSP 2023)
Mahi-Mahi	Scalable, asynchronous liveness in BFT	(IEEE ICDCS 2025)

Thesis Contributions

Baxos

RACS-SADL

QuePaxa

Mahi-Mahi

Outline

- **Consensus**
- **Thesis Contributions**
- QuePaxa
- Summary

Outline

- Consensus
- Thesis Contributions
- **QuePaxa**
- Summary

QuePaxa: Escaping the tyranny of timeouts in consensus

Pasindu Tennage*, Cristina Basescu*, Lefteris Kokoris-Kogias, Ewa Syta, Philipp Jovanovic, Bryan Ford

SOSP 2023

RoadMap

- **Tyranny of timeouts**
- Parallels of QuePaxa and hedging
- QuePaxa algorithm
- Evaluation

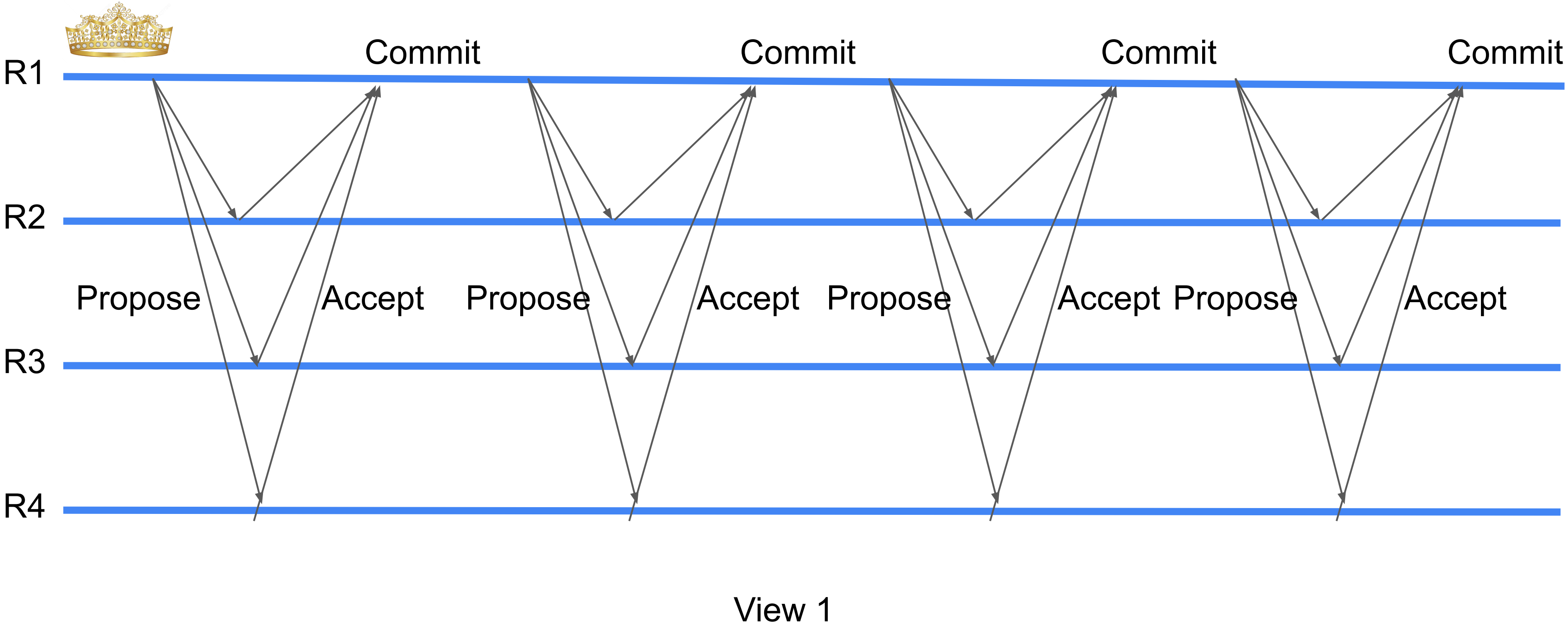
Tyranny of Timeout Problems in Consensus

Timeout based view change

Conservative timeouts

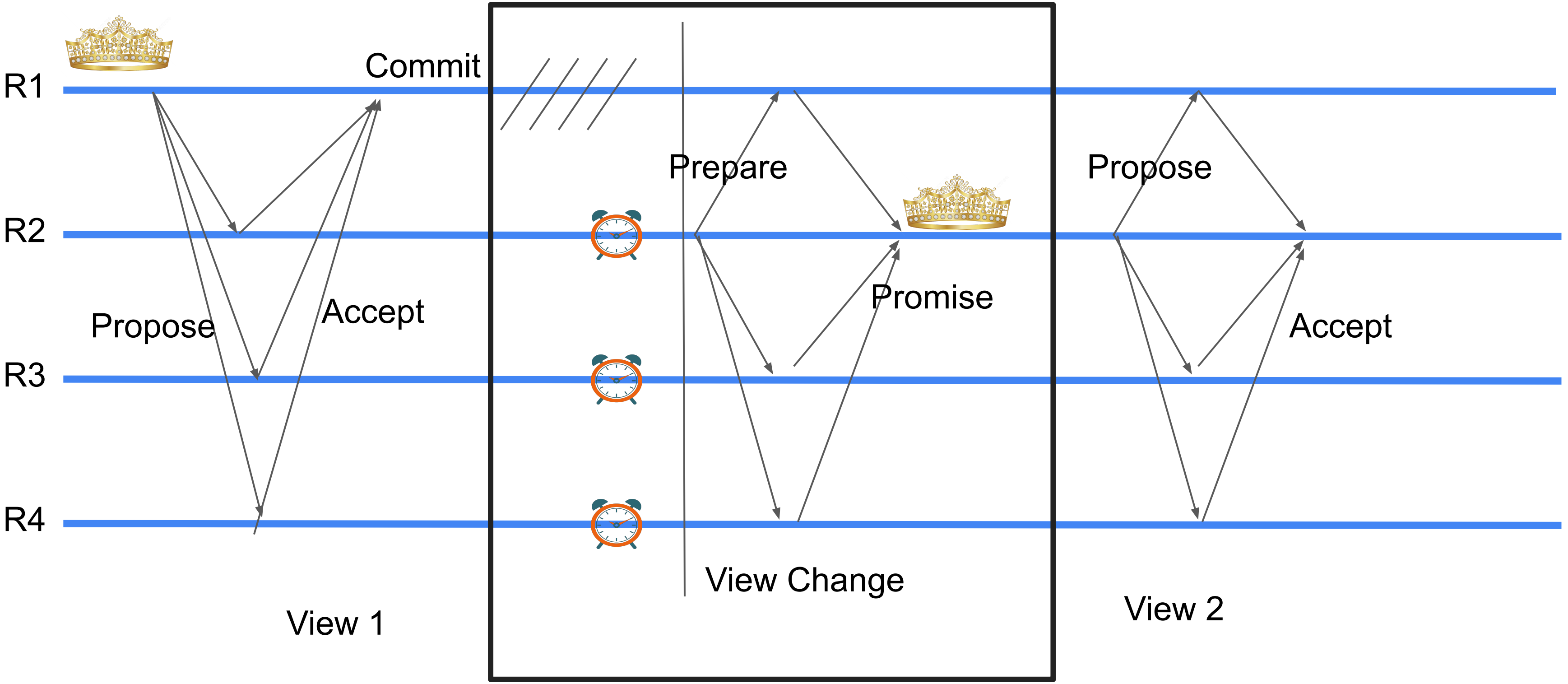
Manually configured timeouts

Timeout based view change [Multi-Paxos]



As long as the network is synchronous, the leader will keep committing new requests

Timeout based view change [Multi-Paxos]



No new commands are committed during view change

Liveness depends on partial synchronous network conditions

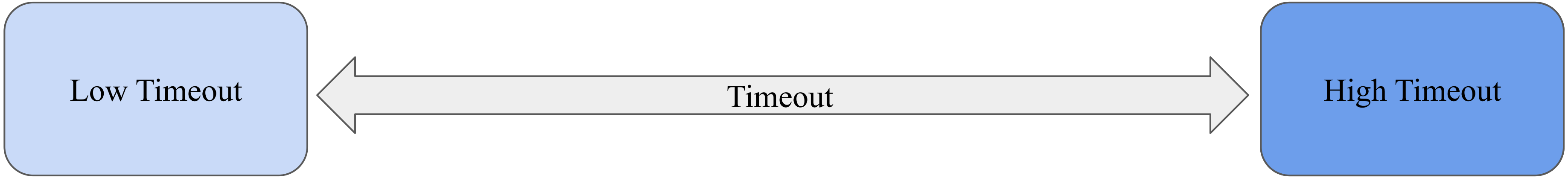
Tyranny of Timeout Problems in Consensus

Timeout based view change

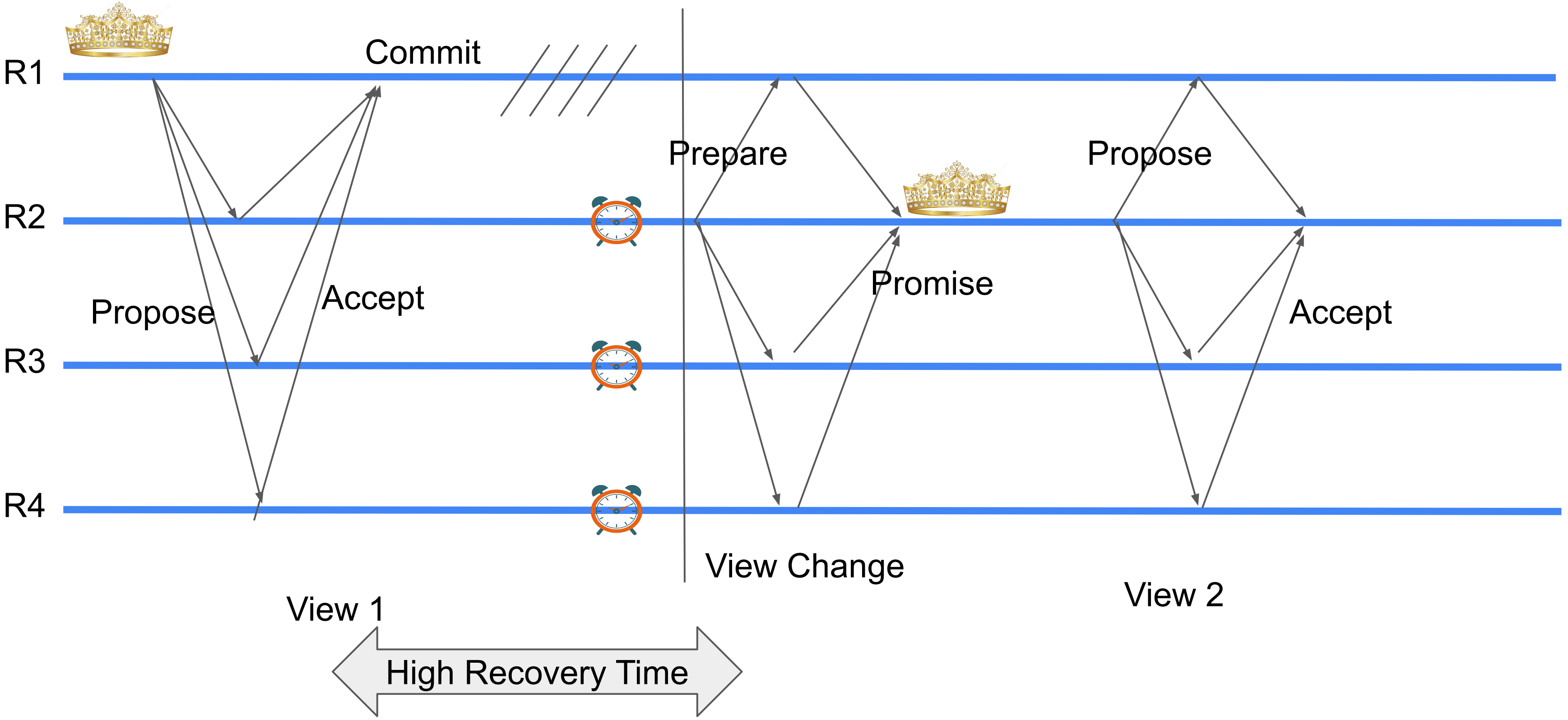
Conservative timeouts

Manually configured timeouts

Choosing Timeouts in leader based protocols

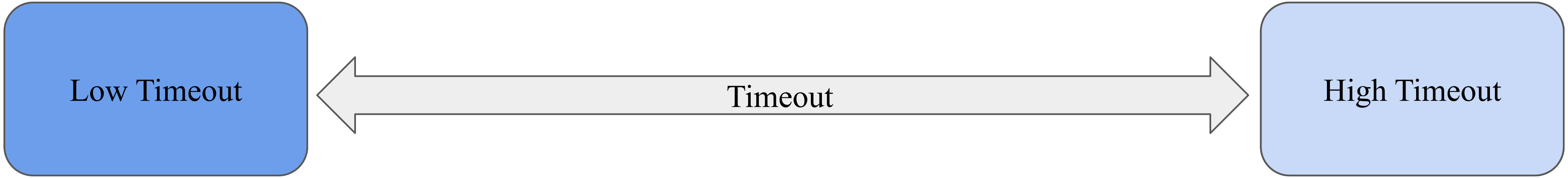


Timeout based view change [Multi-Paxos]



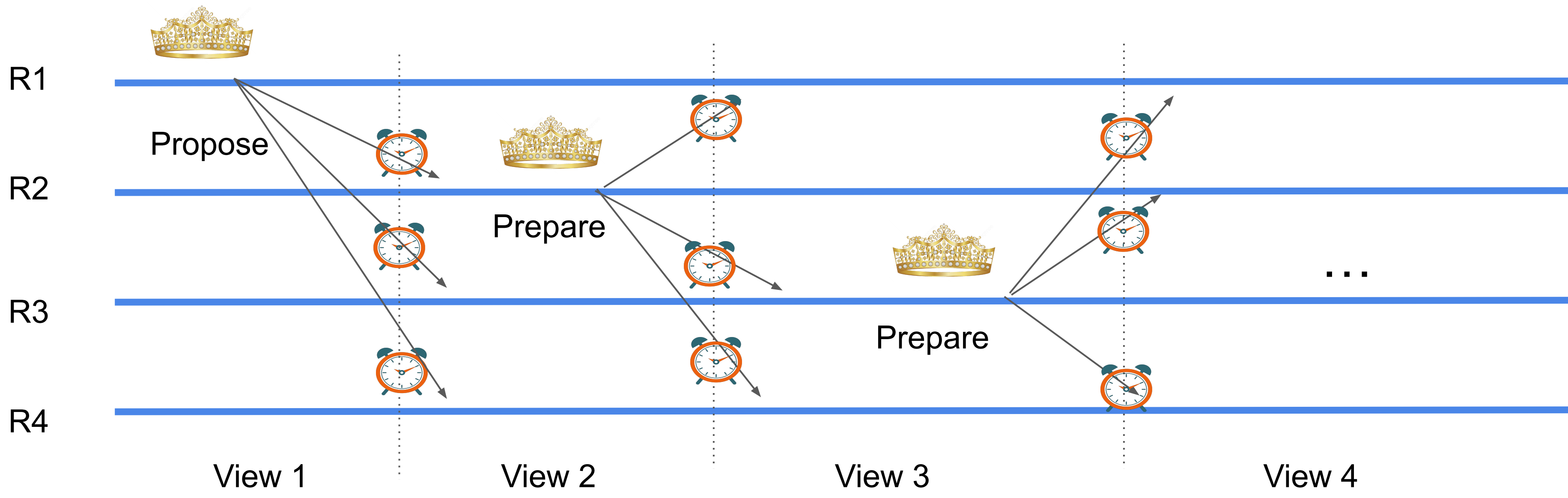
High timeouts result in high recovery time

Choosing Timeouts in leader based protocols



High Recovery Time

Liveness loss with low timeouts



No commands are committed when the timeout is low

Choosing Timeouts in leader based protocols

Low Timeout ⟷ Timeout ⟷ High Timeout

Liveness Loss

High Recovery Time

Both choices of timeouts have negative consequences

Tyranny of Timeout Problems in Consensus

Timeout based view change

Conservative timeouts

Manually configured timeouts

Manual configuration of timeouts

- Stuck with a live but slow leader replica
- Do not consider dynamic network state for leader election

Manual timeouts are sub optimal

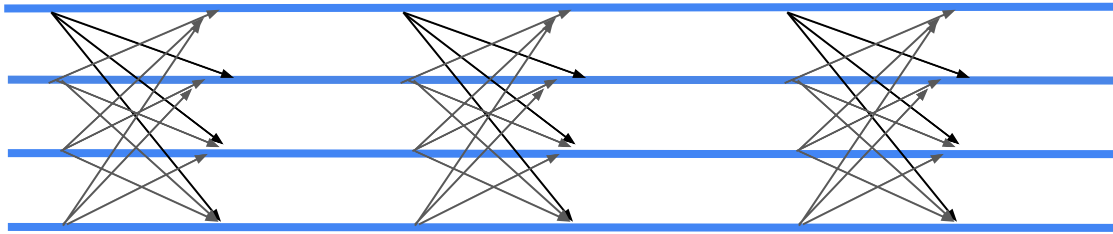
Are timeouts necessary for progress?

Can we eliminate the impact of timeout for liveness?

Do asynchronous protocols solve this problem?

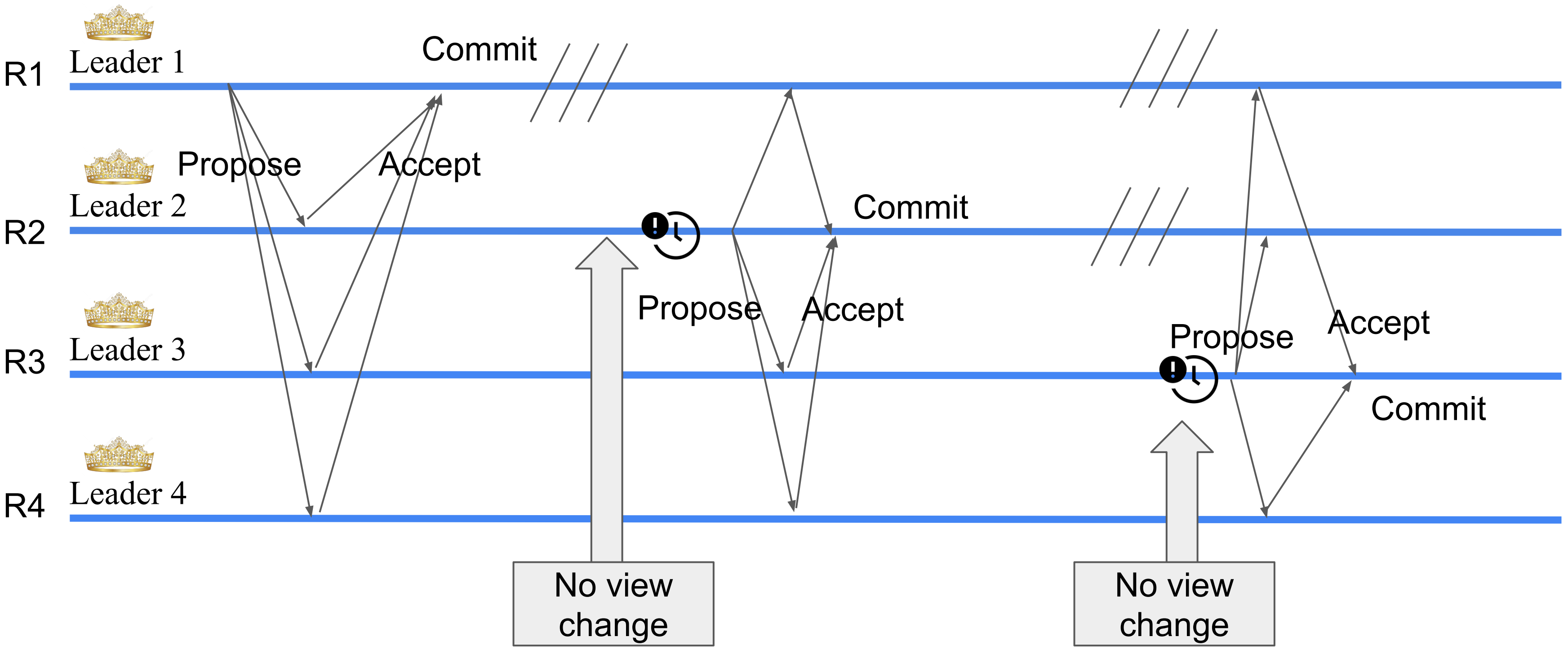
- Asynchronous protocols do not depend on timeout for progress
  - Use randomization to alleviate the FLP impossibility

- Message complexity
  - In general asynchronous protocols have $O(n^2)$ / $O(n^3)$ complexity in the normal case
    - In contrast, partially synchronous protocols have $O(n)$
  - Less efficient than leader-based protocols
  - Hence rarely deployed



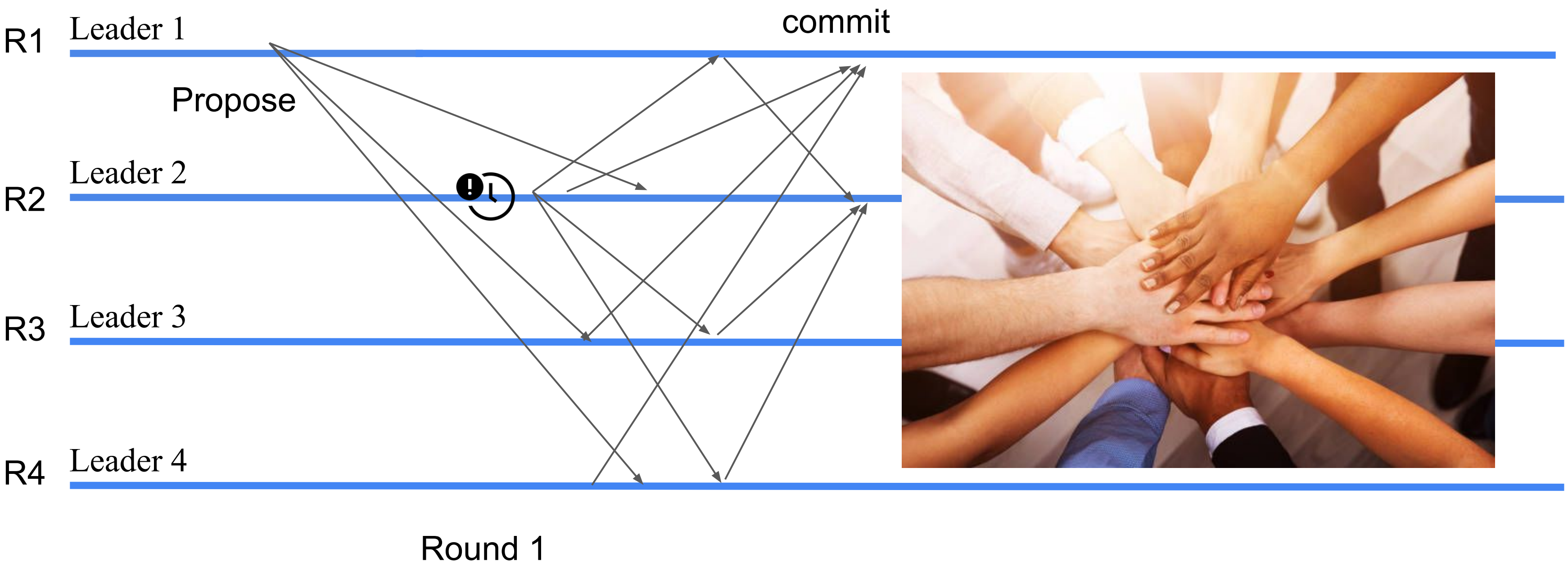
Asynchronous protocols are slow and rarely deployed

An alternative approach?



Can we change leaders **without view changes** if the current leader is sub optimal?

What if multiple leaders could **cooperate** instead of **interfere**?



Can we support multiple leaders to be non destructive?

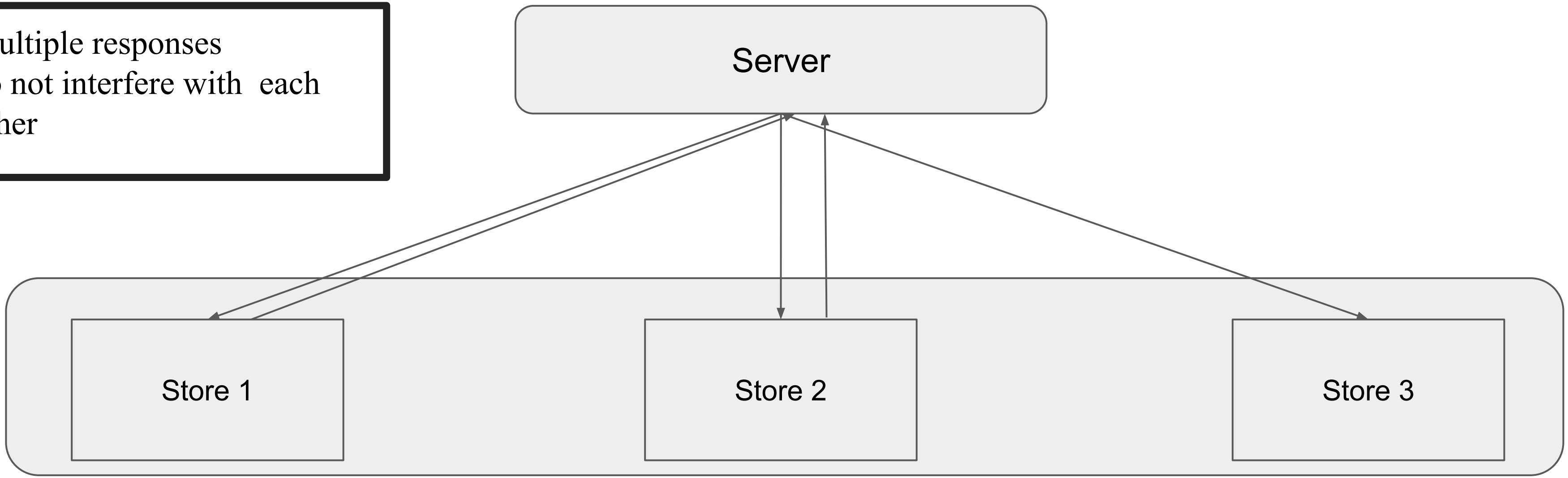
RoadMap

- Tyranny of timeouts
- **Parallels of QuePaxa and hedging**
- QuePaxa algorithm
- Evaluation

Hedging

- Hedging is a way to curb latency variability
  - Key idea: issue the same request to multiple replicas and use the results from whichever replica responds first

Multiple responses do not interfere with each other



Can we apply hedging to consensus so that multiple proposers don't interfere?

RoadMap

- Tyranny of timeouts
- Parallels of QuePaxa and hedging
- **QuePaxa algorithm**
- Evaluation

QuePaxa Contributions

- A consensus protocol that eliminates the tyranny of timeouts problems

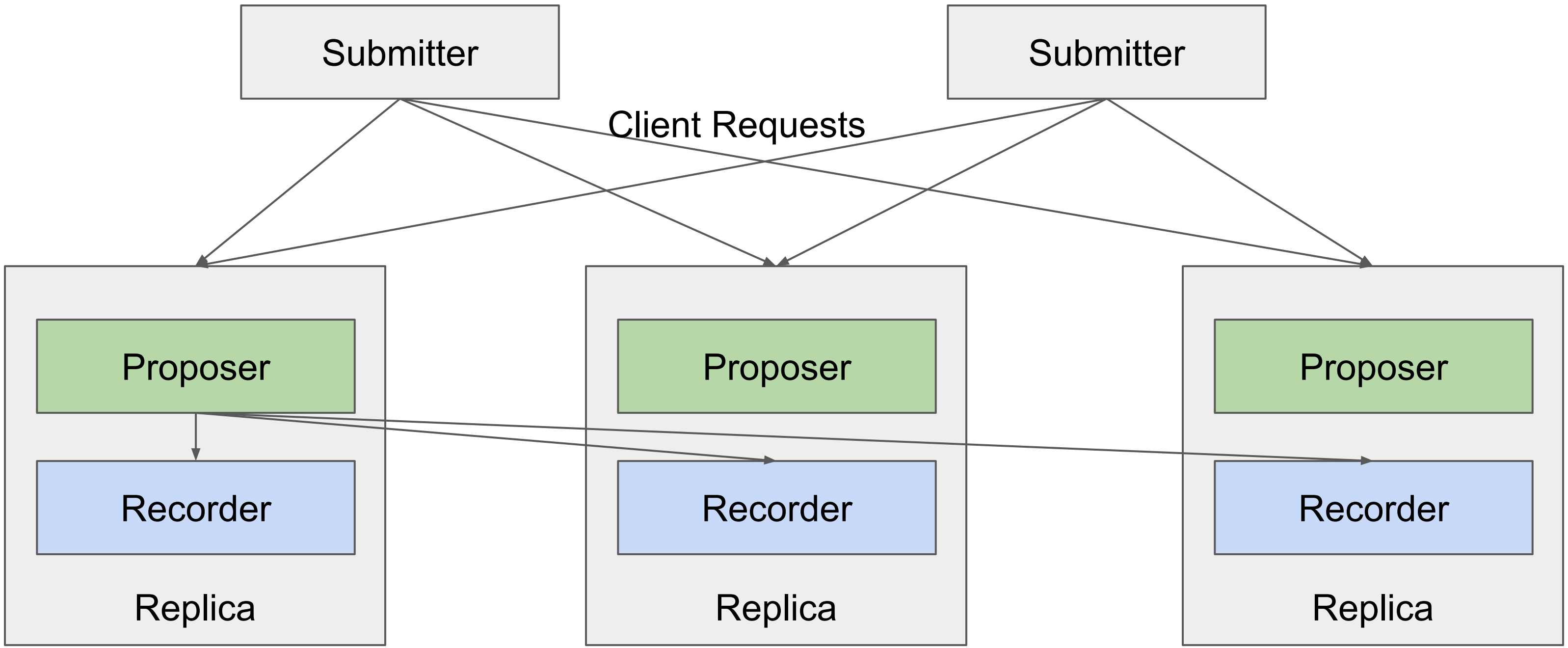
- First consensus protocol to support hedging in consensus

- A novel consensus protocol that
  - Under normal network conditions as good as Multi-Paxos /Raft
  - Under adversarial network conditions, provides liveness

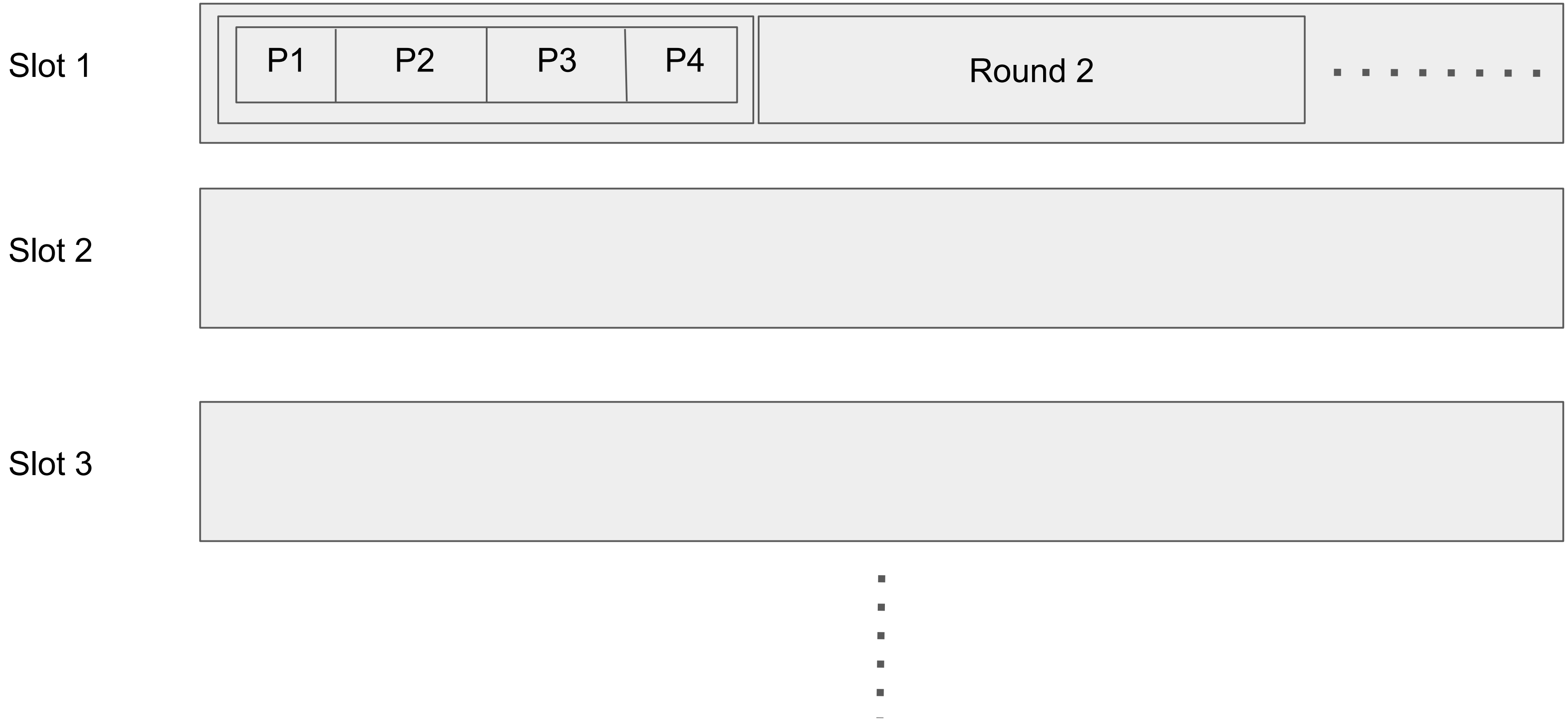
QuePaxa RoadMap

- Operation Overview
- Abstract QuePaxa – *a simplified version*
- Concrete QuePaxa overview

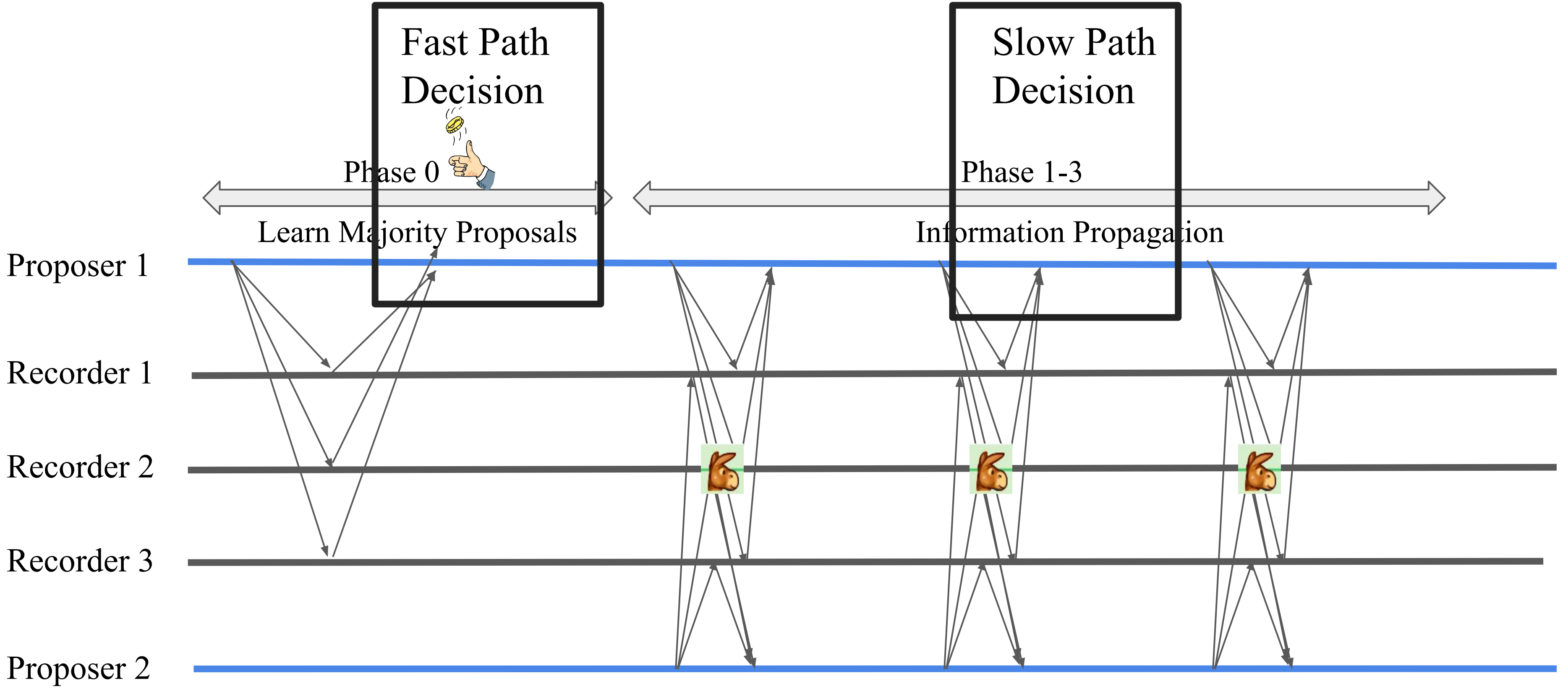
QuePaxa Architecture



QuePaxa Log Structure



QuePaxa Protocol Diagram



QuePaxa has a fast path decision and a slow path decision

QuePaxa RoadMap

- Operation Overview
- **Abstract QuePaxa - *a simplified version***
- Concrete QuePaxa overview

Abstract QuePaxa is a simplified version of QuePaxa

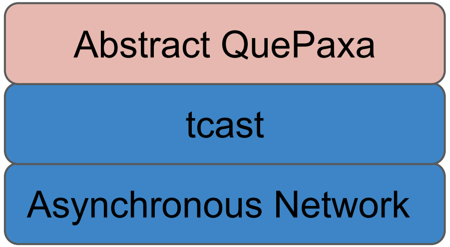
Introducing threshold broadcast (**tcast**)

- Divide the problem in to two parts
  - Handling replica failures
  - Handling asynchrony

- **First ignore asynchrony** and focus on replica failures

- Using **tcast** let us assume a synchronous lock step network

- **tcast** (threshold synchronous broadcast): an abstraction which provides lock step synchrony to the consensus layer



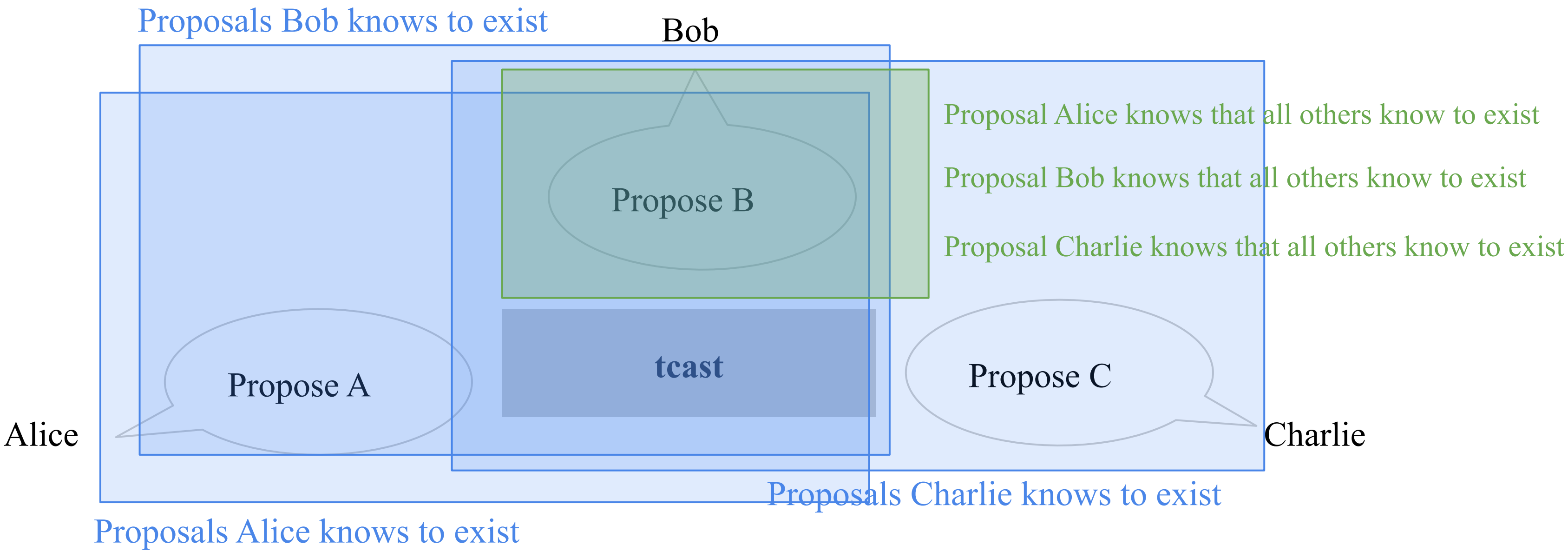
Abstract QuePaxa assumes synchrony and solves the replica failure challenge

Abstract QuePaxa Algorithm

Algorithm 1: Abstract QuePaxa consensus algorithm

```
Input: v ← value preferred by this replica
repeat                                      // iterate through rounds
    p ← ⟨v, random()⟩                       // prioritized proposal
    (P, _) ← tcast({p})                     // propagate our proposal
    (E, P′) ← tcast(P)                      // propagate existent sets
    (C, U) ← tcast(P′)                      // propagate common sets
    v ← best(C).value                       // next candidate value
    if best(E) = best(U) then               // detect consensus
        deliver(v)                          // deliver decision
```

Abstract QuePaxa is just a few lines of pseudocode!



- **tcast property 1**: each node learns a majority of proposals
- **tcast property 2**: each node learns a proposal that all nodes know to exist

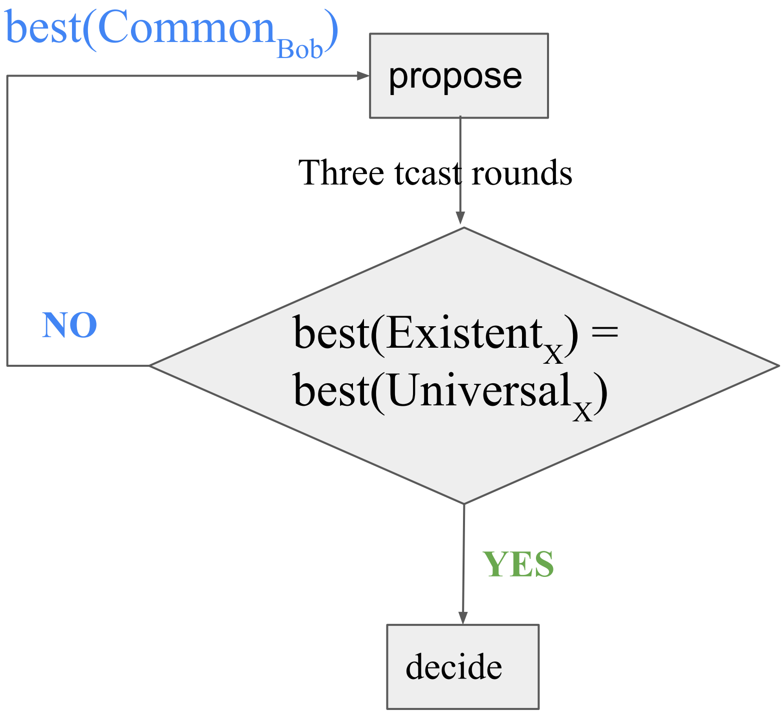
Nodes have no guarantee to learn the same sets! (no consensus yet)

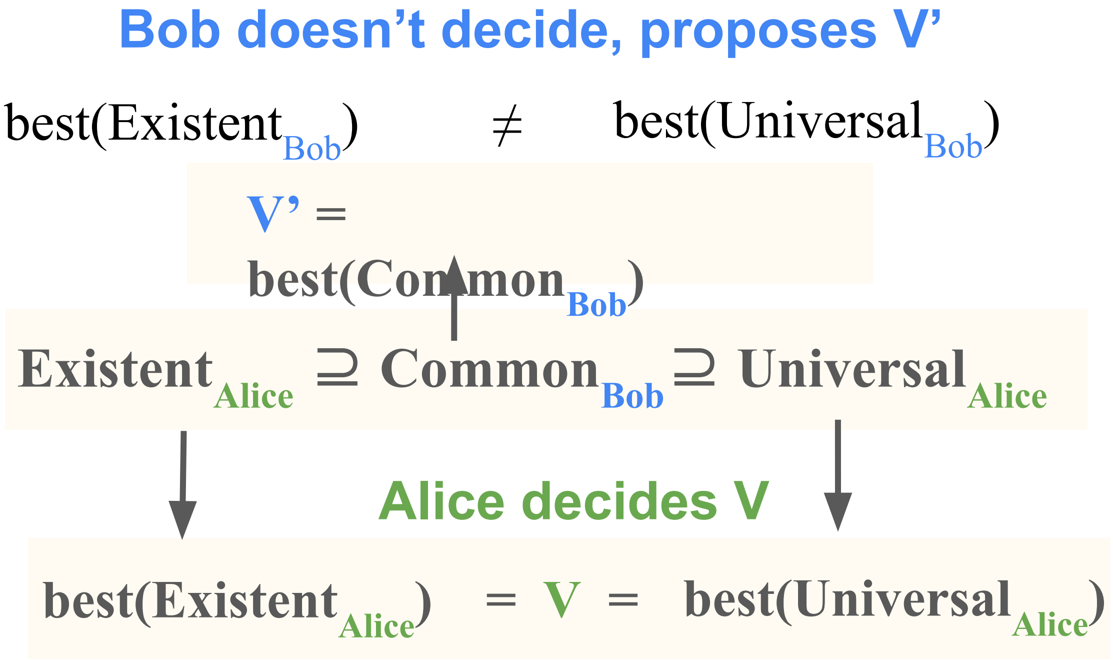
Towards consensus: approximating what others know

- Sets from one tcast invocation are **insufficient for consensus**
- **Repeat**: **three tcast invocations**, giving each node i sets with increasing guarantees
    - **An existent set**
    - **A common set**
    - **A universal set**

Key relationship for consensus: $\textbf{Existent}_i \supseteq \textbf{Common}_j \supseteq \textbf{Universal}_k$

Consensus: reaching a safe decision





Only possible decision in future is V' = best($Common_{Bob}$) = best($Existent_{Alice}$) = V

Abstract QuePaxa

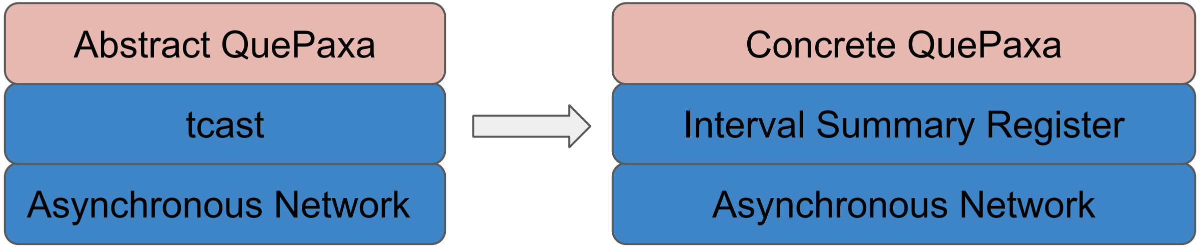
- Liveness does not depend on timeout because the protocol is randomized
- Robust against adversarial networks
- $O(n^2)$ message complexity hence slow
- Does not support hedging

Abstract QuePaxa is robust but inefficient

QuePaxa RoadMap

- Operation Overview
- Abstract QuePaxa
- **Concrete QuePaxa overview**

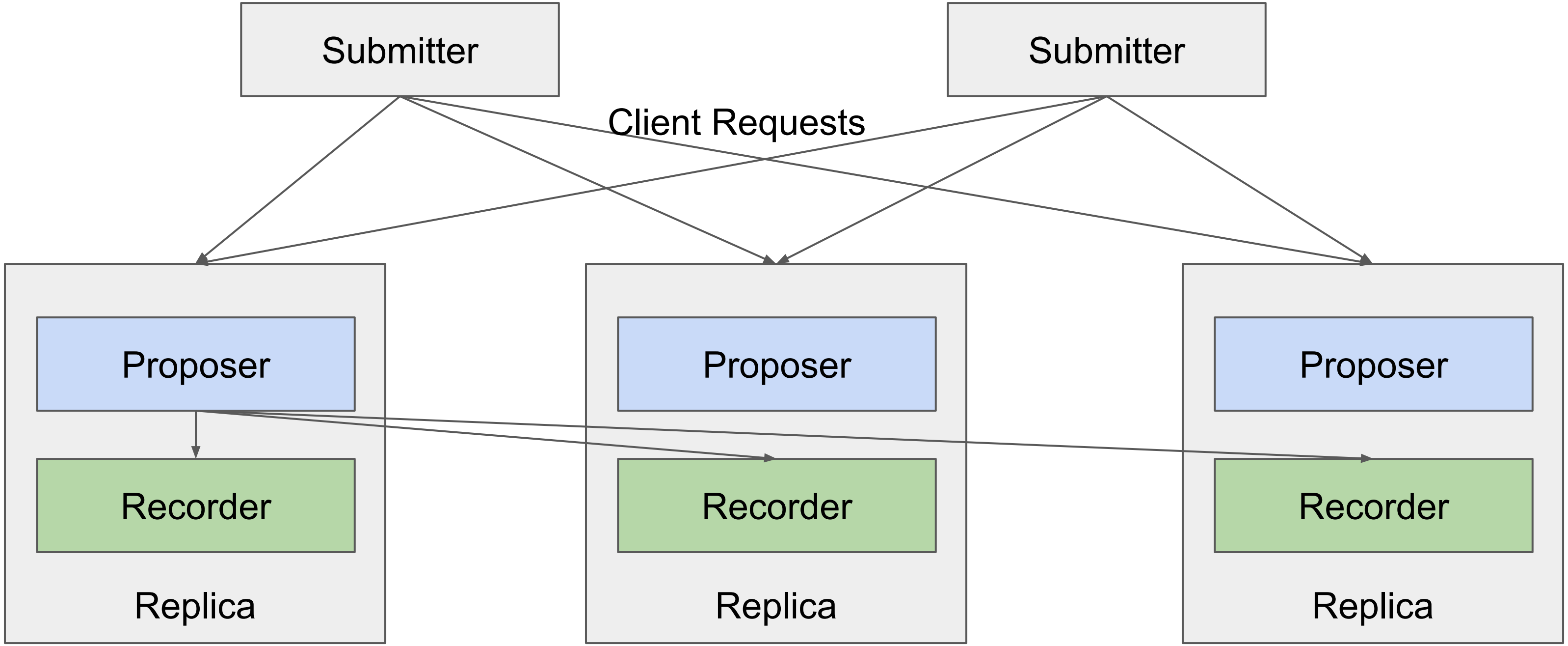
From abstract to concrete QuePaxa



- O(n) complexity in the normal case
- Robust against asynchrony
- Support hedging
- Implementation ready (4368 LOC)

Concrete QuePaxa has all we need!

QuePaxa Architecture



Concrete Recorder Protocol (ISR)

Algorithm 2: Interval summary register (ISR)

State : S current logical clock step, initially 0
State : $F[s]$ first value recorded at each step, default **nil**
State : $A[s]$ aggregate of values in each step, default **nil**

record $(s, v) \rightarrow (s', f', a')$**:** // handle an invocation
  if $s > S$ **then** // advance to a higher step
    $S \leftarrow s$ // update current step number
    $F[s] \leftarrow v$ // record first value in this step
  if $s = S$ **then** // aggregate all values
    $A[s] \leftarrow$ **aggregate**$(A[s], v)$ // seen in this step
  return $(S, F[S], A[S-1])$ // return a summary

- Simulates lock step synchrony using a threshold logical clock
- For each step, records the the first value and the aggregate of the values submitted in the previous step
- Constant space

QuePaxa Recorder is a constant space interval summary register

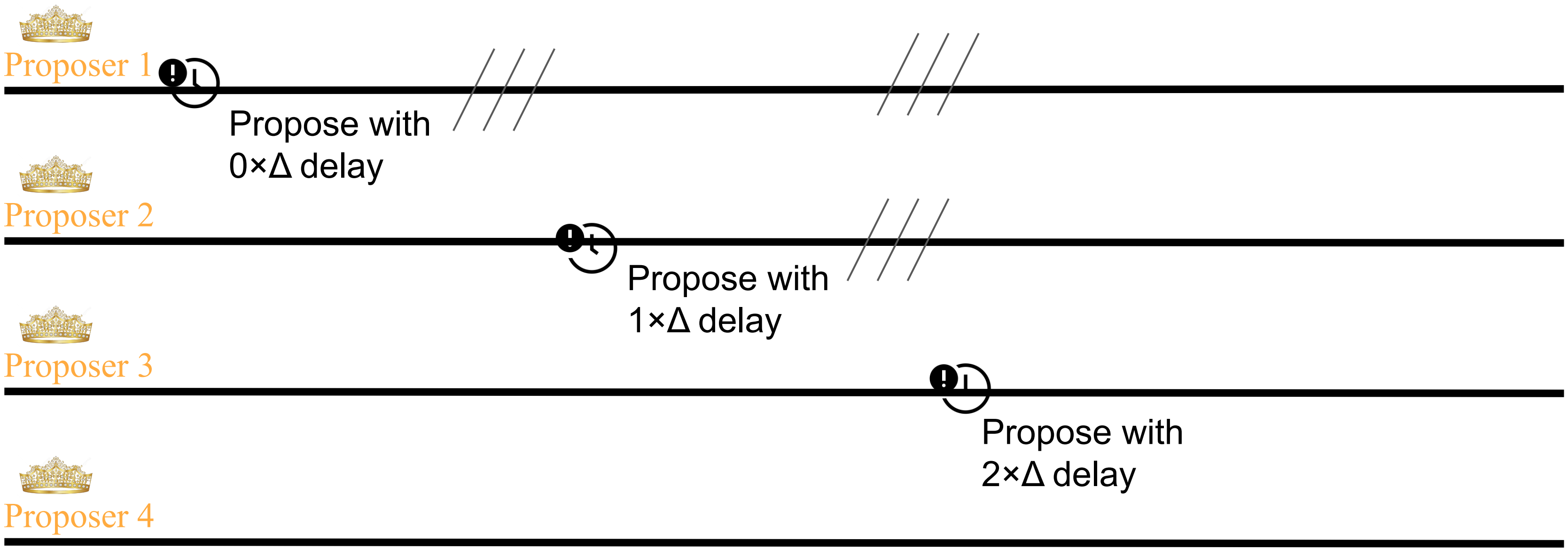
Proposer Code

Algorithm 4: Protocol for QuePaxa proposer i

```
Input: v preferred value of this proposer i
s ← 4 × 1 + 0                                  // start at round 1, phase 0
p ← ⟨H, i, v⟩                                  // initial proposal template
repeat
    p_j ← p for all recorders j                // prepare proposals
    if s mod 4 = 0 and (s > 4 or i is not leader) then
        p_j.priority ← random(1..H − 1) for all j
    Send record(s, p_i) in parallel to each recorder j
    Await R ← quorum of replies (s'_j, f'_j, a'_j)
    if s'_j = s in all replies received in R then
        if s mod 4 = 0 then                    // phase 0: propose
            if f'_j.priority = H in all replies then
                return f'_j.value from any reply in R
            p ← best_j of f'_j from all replies in R
        if s mod 4 = 1 then                    // phase 1: spread E
                                               // no action required
        if s mod 4 = 2 then                    // phase 2: gather E, spread C
            if p = best_j of a'_j from all replies in R then
                return p.value                 // report decision
        if s mod 4 = 3 then                    // phase 3: gather C
            p ← best_j of a'_j from all replies in R
        s ← s + 1                              // advance to next step
    else if any reply in R has s'_j > s then
        s, p ← s'_j, f'_j                      // catch up to step s'_j
```

QuePaxa proposer uses RPC in 4 phases to contact Recorders

Hedging in QuePaxa



QuePaxa supports hedging because multiple proposers do not cancel each other

RoadMap

- Tyranny of timeouts
- Parallels of QuePaxa and hedging
- QuePaxa algorithm
- **Evaluation**

Evaluation

- Can QuePaxa guarantee liveness under any timeout?
- Under normal case executions, how does QuePaxa compare with leader-based protocols?
- Under adversarial conditions, does QuePaxa provide liveness?

Setup

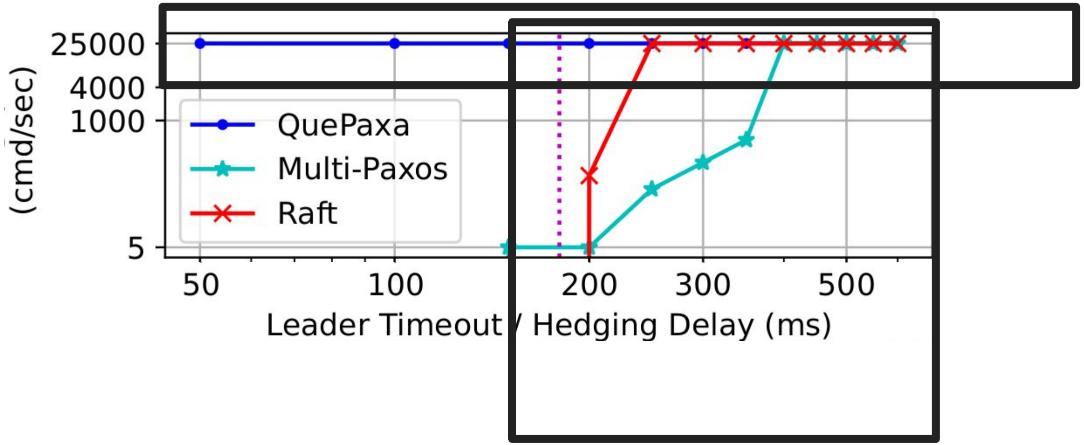
- LAN (N. Virginia)

- WAN (Tokyo, Mumbai, Singapore, Ireland, and São Paulo)

- Replicas: c4.4xlarge
  - 16 virtual CPUs, 30 GB memory
- Submitters: c4.2xlarge
  - 8 virtual CPUs, 15 GB memory

Effect of Hedging in Quepaxa

Throughput

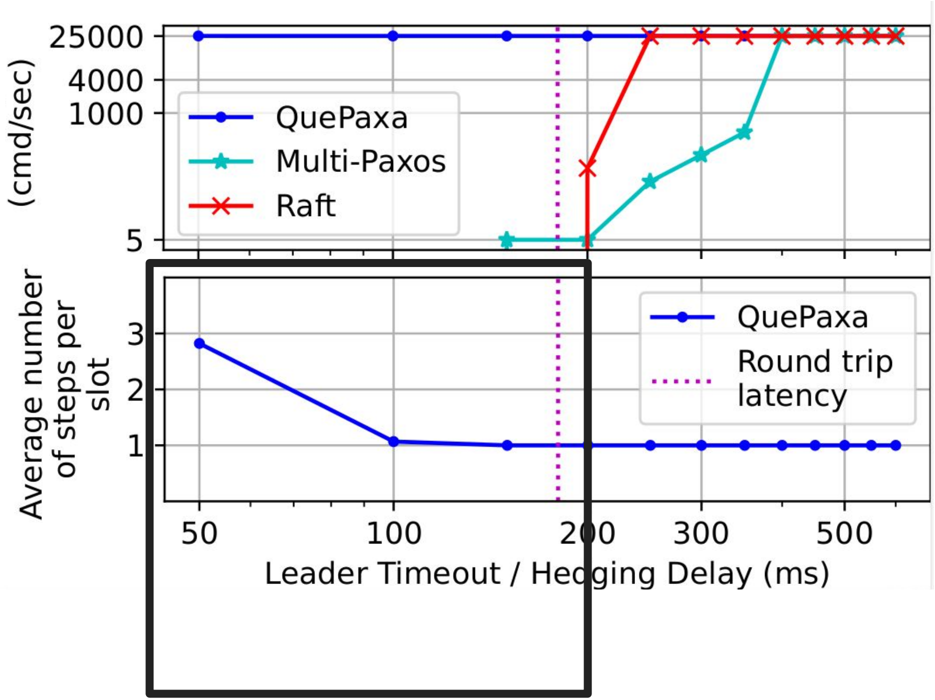


QuePaxa is live for any hedging delay

Effect of Hedging in Quepaxa

Throughput

Bandwidth Cost



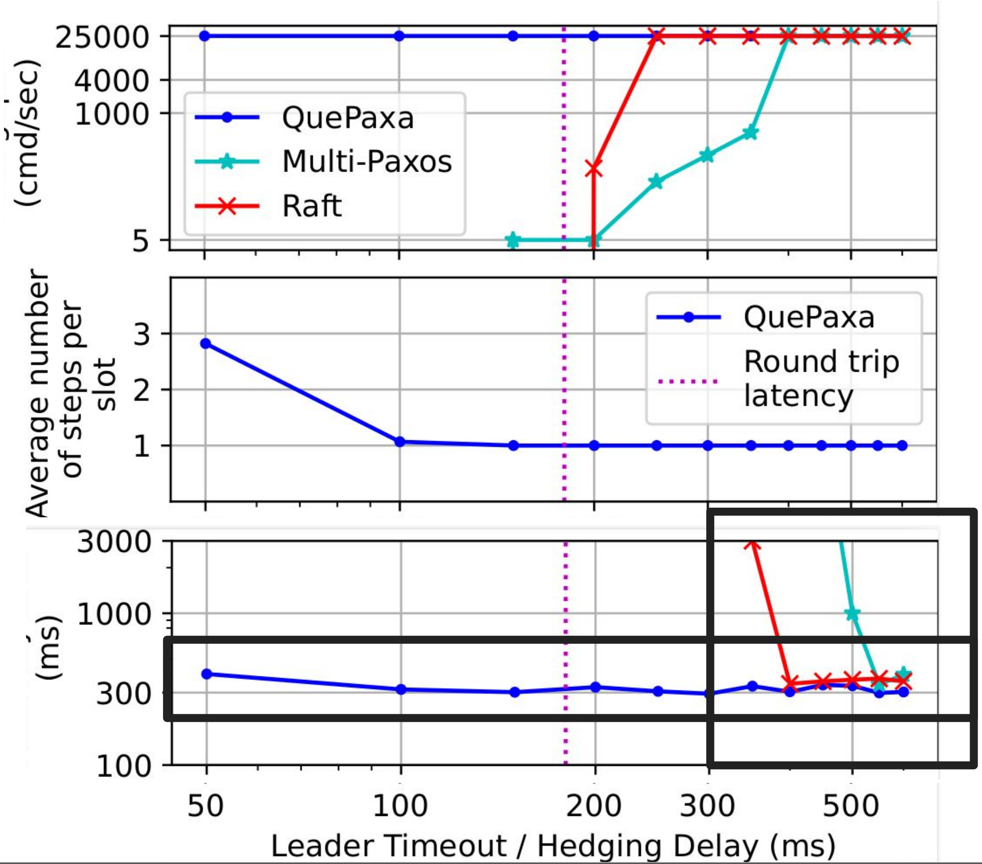
QuePaxa has an additional overhead only when hedging delay < RTT

Effect of Hedging in Quepaxa

Throughput

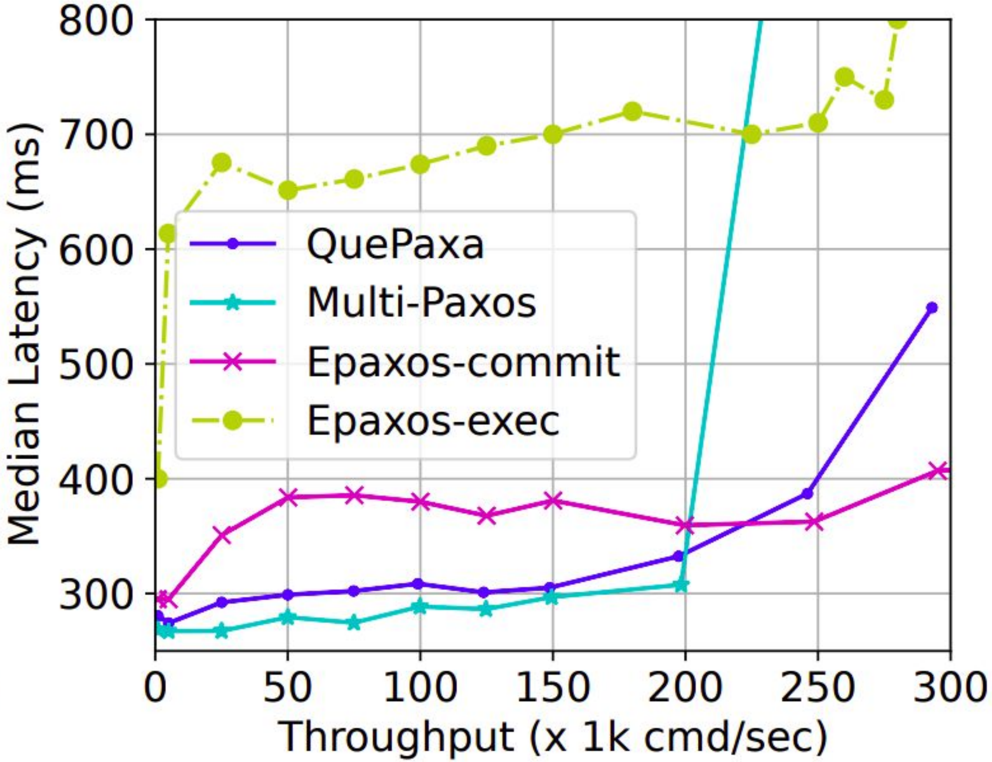
Bandwidth Cost

Recovery Time



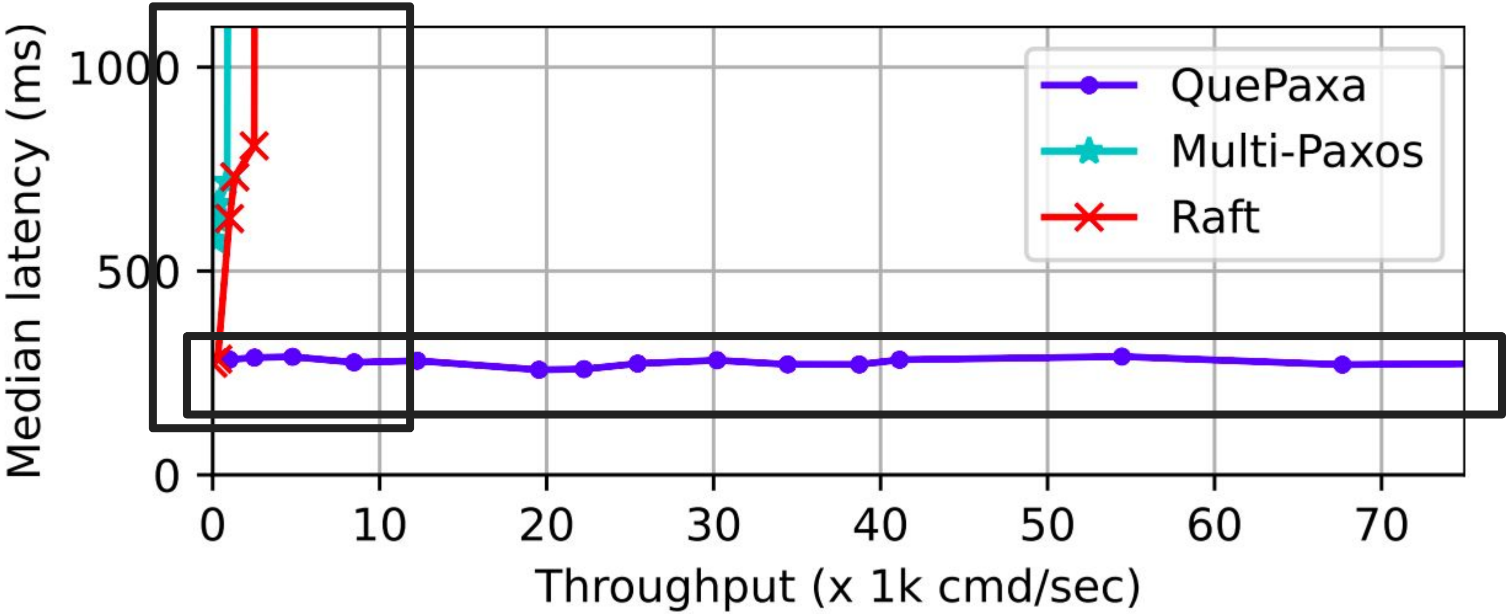
QuePaxa has low recovery time

Normal case execution in a WAN



QuePaxa performs comparable to Multi Paxos

Performance under adversarial networks



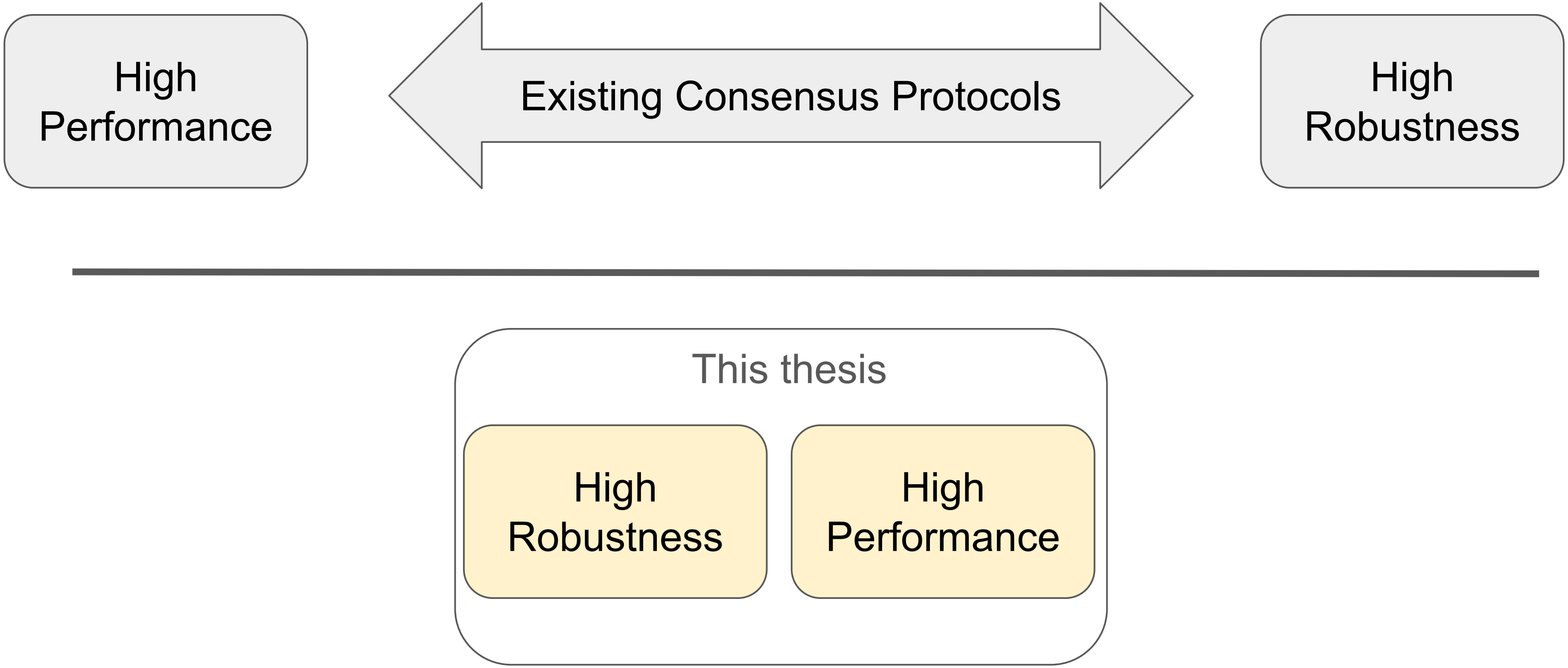
QuePaxa is live under asynchrony

QuePaxa Contributions

- A consensus protocol that eliminates the tyranny of timeouts problems
- First consensus protocol to support hedging in consensus
- A novel consensus protocol that
  - Under normal network conditions as good as Multi-Paxos /Raft
  - Under adversarial network conditions, provides liveness

Thesis Contributions

Baxos	Robustness against leader-targeted attacks	
RACS-SADL	Asynchronous Liveness and high scalability	(IEEE CLOUD 2025)
QuePaxa	Mechanisms to avoid the tyranny of timeout problems in consensus	(ACM SOSP 2023)
Mahi-Mahi	Scalable, asynchronous byzantine fault tolerance	(IEEE ICDCS 2025)


High Performance
Existing Consensus Protocols
High Robustness
This thesis
High Robustness
High Performance

Distributed Consensus Timeline

80s/90s
FLP
Paxos
View Stamp

2000
PBFT
Fast Paxos

2010
Mencius
EPaxos
Raft
WPaxos
Pig Paxos

2020
Hotstuff
Dag-Rider

2021
Baxos

2022
SADL-RACS

2023
QuePaxa

2024
Mahi-Mahi

Started thesis

Robust and High-Performance Wide-Area Consensus Protocols

PhD Public Defense

Pasindu Tennage

Thesis director: Bryan Ford
Thesis co-director: Lefteris Kokoris-Kogias

DEDIS

EPFL