QuePaxa: Escaping the Tyranny of Timeouts in Consensus

Venue: October 25, 2023, SOSP, Koblenz, Germany

Tennage P., B˘asescu C., Kogias K.L., Syta E., Jovanovic P., Galiñanes V.E., Ford B.
Affiliation: EPFL, Mysten Labs, Trinity College, UCL.

Presented by: Aiman, Ajinkya, Isha, Ujjawal
10-23-2023
ECS 265: Distributed Database Systems, Fall 2023

Leader Based Consensus Algorithms

- Goal of these algorithms
- Role of The Leader and its Followers
- Properties - Safety, Liveness
- Fault Tolerance

- What happens when the leader fails? - Consensus stalls till view change
- What about adverse network conditions?

- How do these algorithms handle such problems? - Timeouts

Timeouts

What?

- process or node takes longer than expected to respond or when a message is delayed.
- Timeouts are used by protocols to make sure that the system is not stuck infinitely waiting for a response.

Why?

- Timeouts ensure liveness: system progress under adverse conditions.
- Crucial for availability: prevents system stalling if a leader fails.

How long?

- Need to set timeouts conservatively large to avoid false triggers.

To understand the problem created by timeouts let's recall the communication models

Different Communication Models

Synchronous Communication	Asynchronous communication	Partial synchronous communication
A known delay or fixed upper bound (known to the consensus protocol)	arbitrary delay, might get dropped, delayed or in any order (no upper bound)	Fixed upper bound of delay, but unknown to the consensus protocol.
Guarantees liveness, however difficult to implement in real life because of rigidity in assumptions.	Doesn’t guarantee liveness	Asynchronous before some unknown point in time (GST), and synchronous after that. Guarantees safety before GST and both safety and liveness after the GST.

Partial synchronous analogy

No clue When the hour starts

Prepare as if bus could come any moment

Predict subsequent arrivals based on the first arrival

GST - Global Standardisation Time (a special time after which the network becomes fully synchronous)

Problems with Timeouts

1. We have timeout triggered view changes to ensure liveness.
2. Simultaneous *leaders interfere destructively*.
3. Timeouts incur administrative cost of manual configuration.

BIG QUESTION: How can practical consensus algorithms escape the tyranny of timeouts?

Solution to the Tyranny

QUEPAXA!

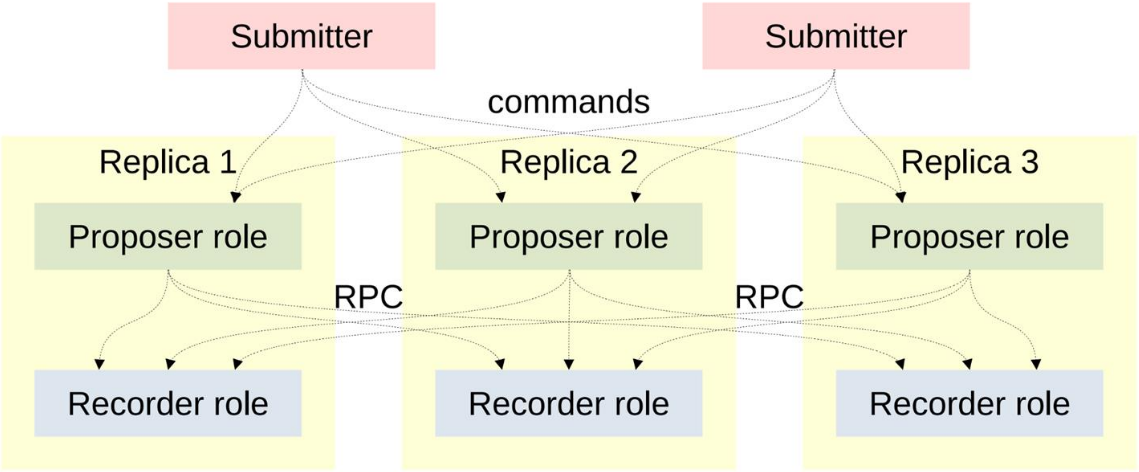


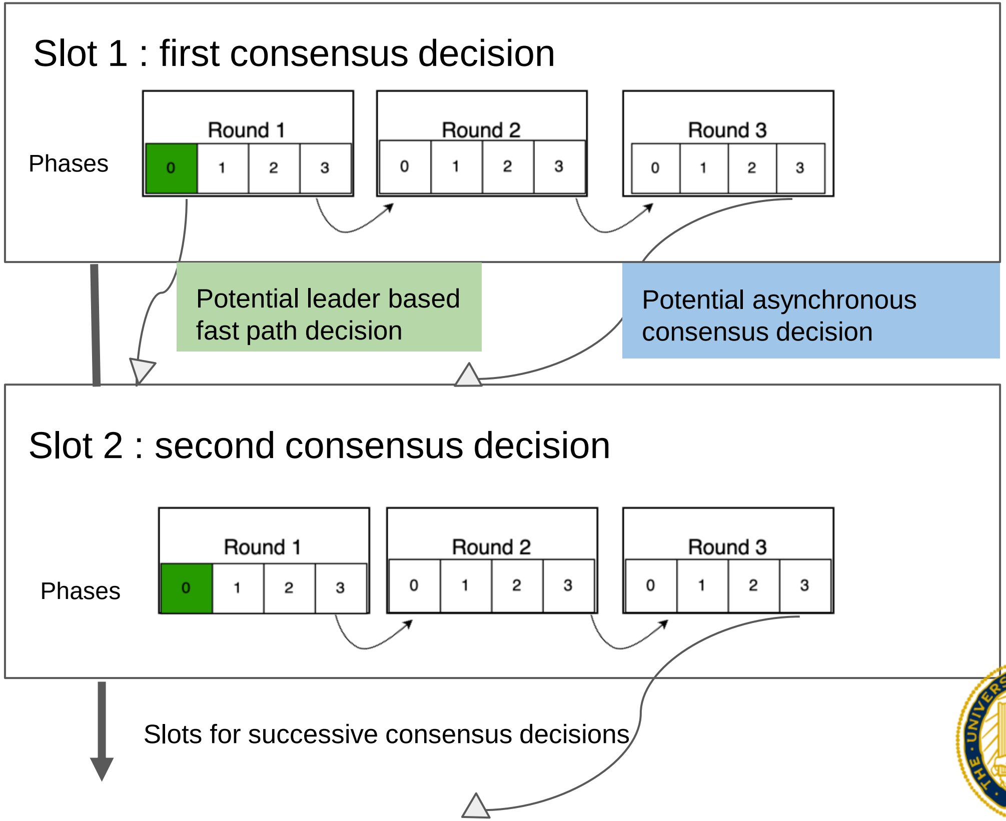
*Figure taken from the original paper

Workflow

Slot : represents a unique state transition (where unique consensus decision is made)

Round: represents multiple attempts to reach a consensus if the previous attempt fails.

Phases (0-3): represents steps the system goes through during each attempt to make a decision.



How to Achieve Consensus?

t-cast: A Special Broadcast

$$t_{cast} : \mathbb{K} \rightarrow \tilde{\mathbb{K}}; \quad \exists \tilde{\mathbb{K}} \subseteq \mathbb{K}$$

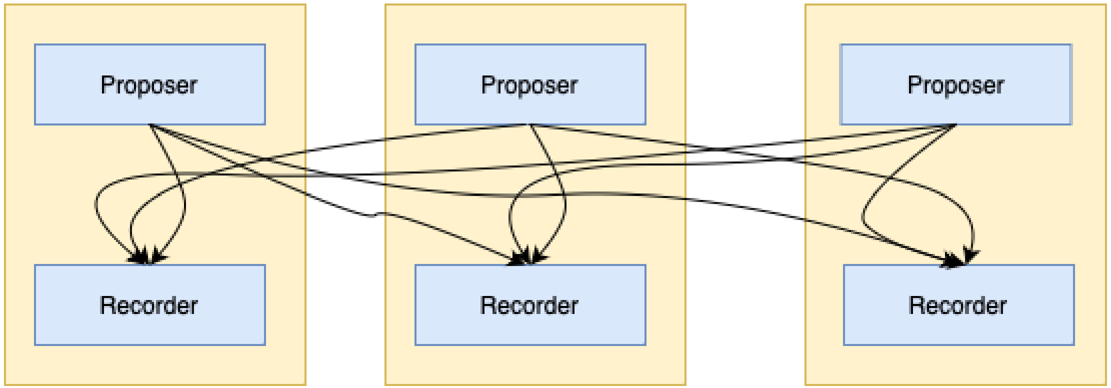
1. All live replicas receive proposals from `*a majority*' of replicas.
2. Atleast 1 replica's proposal is seen by all replicas.

$\mathbb{K}$ - Set of proposals

$\tilde{\mathbb{K}}$ - Subset of proposals

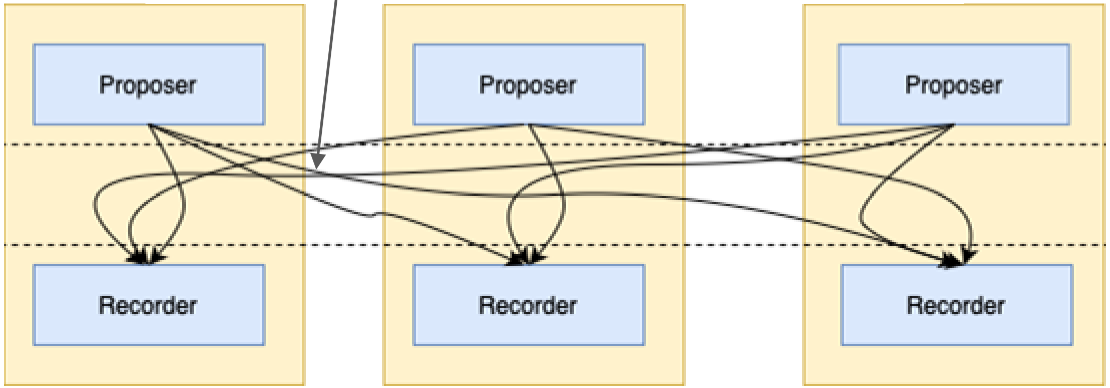
t-cast: A Special Broadcast

$$t_{cast} : \mathbb{K} \rightarrow \tilde{\mathbb{K}}; \quad \exists \tilde{\mathbb{K}} \subseteq \mathbb{K}$$



t-cast: A Special Broadcast

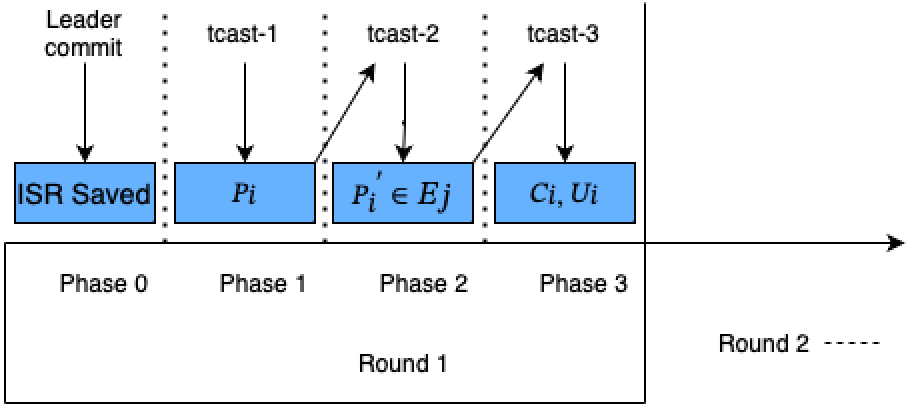
$$t_{cast} : \mathbb{K} \rightarrow \tilde{\mathbb{K}}; \quad \exists \tilde{\mathbb{K}} \subseteq \mathbb{K}$$



Consensus, How?

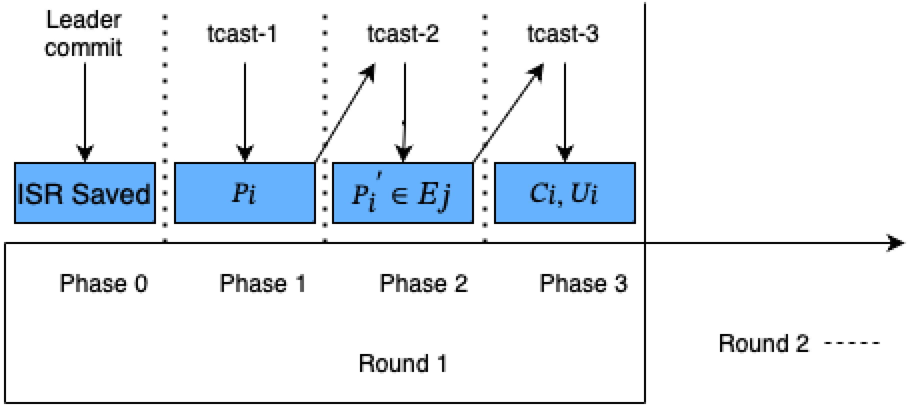
- node i's proposal = <transaction,rand_prob()>.
- ISR = Internal State Register inside of Recorder role (blue box).

- Pi = props from any majority of reps.
- Pi' = props of any other rep's majority received.
- Ci = every rep knows these props exist.
- Ui = every rep knows these props are common.



QuePaxa Hedging -

- QuePaxa uses a Hedging schedule.
- Each node gets priority based on delay interval δ. Eg:
  - Leader proposes within $\delta = 0.1$ seconds.
  - Next node in sequence proposes within $2\delta = 0.2$ seconds.
  - Next node proposes within $3\delta = 0.3$ seconds.



ESCAPING THE TYRANNY!

Liveness - solution to problem 1

- Problem:- Reliability on timeouts for liveness.
- Fast Path - Round 1, phase 0
- Single Round Trip

- But what happens when leader fails or network unstable? - subsequent leaderless rounds
- Thus ensures liveness

Hedging - solution to problem 2

- Hedging involves launching redundant operations on different nodes simultaneously.
- Purpose: Proactively mitigate risks of unexpectedly long delays in consensus protocols.

Timeout vs. Hedging

Timeout	Hedging
Reactive	Proactive
Detects failure retroactively	Initiates non-interfering parallel efforts

Auto tuning- solution to problem 3

- Choice of leader & Hedging schedules are optimisation parameters hence Multi-armed Bandit theory is used in QuePaxa, and a similar explore-exploit process is used to auto-tune these parameters
- Eliminates -
  - Administrative burden of configuring timeouts
  - Risks of misconfiguring them

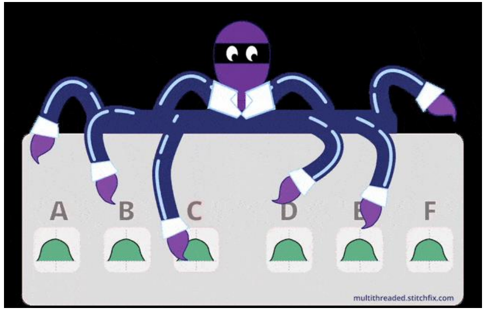
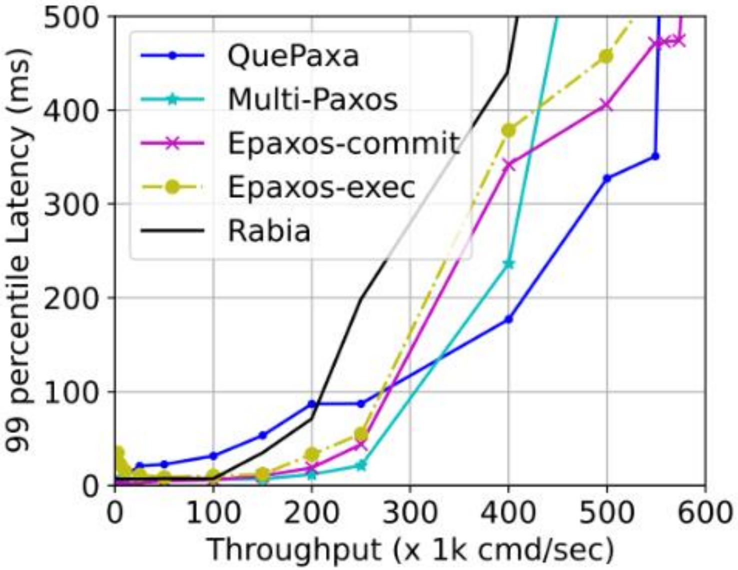
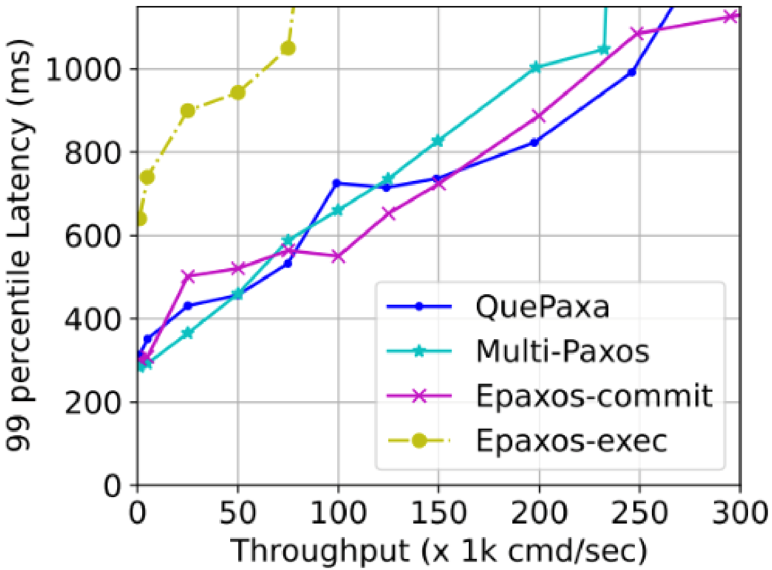


*figure extracted from google images

EXPERIMENTS

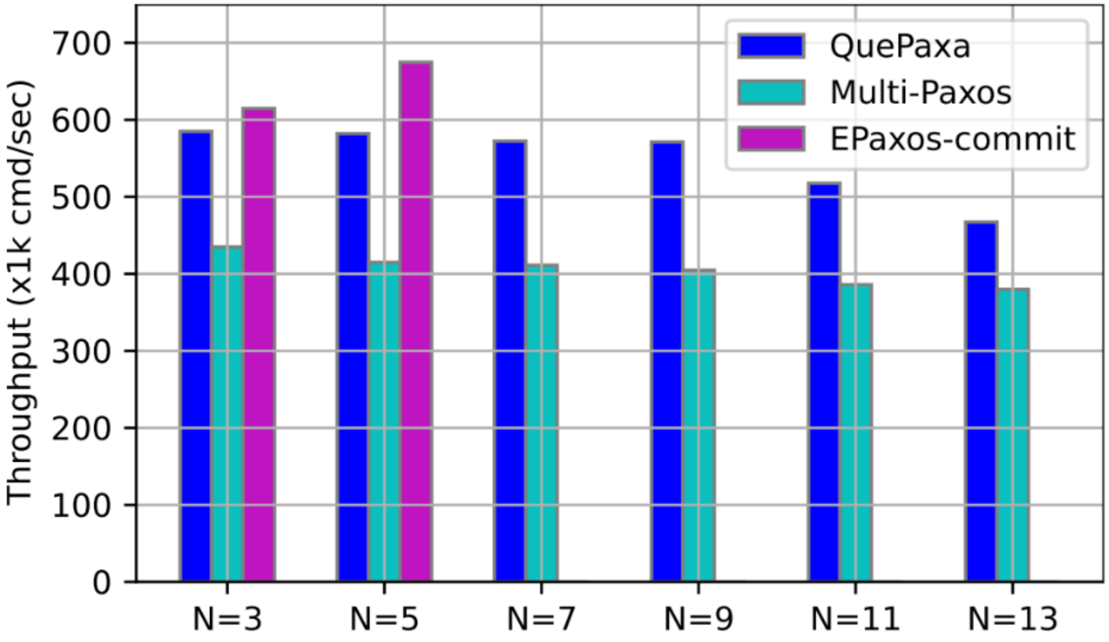
Normal case Comparison





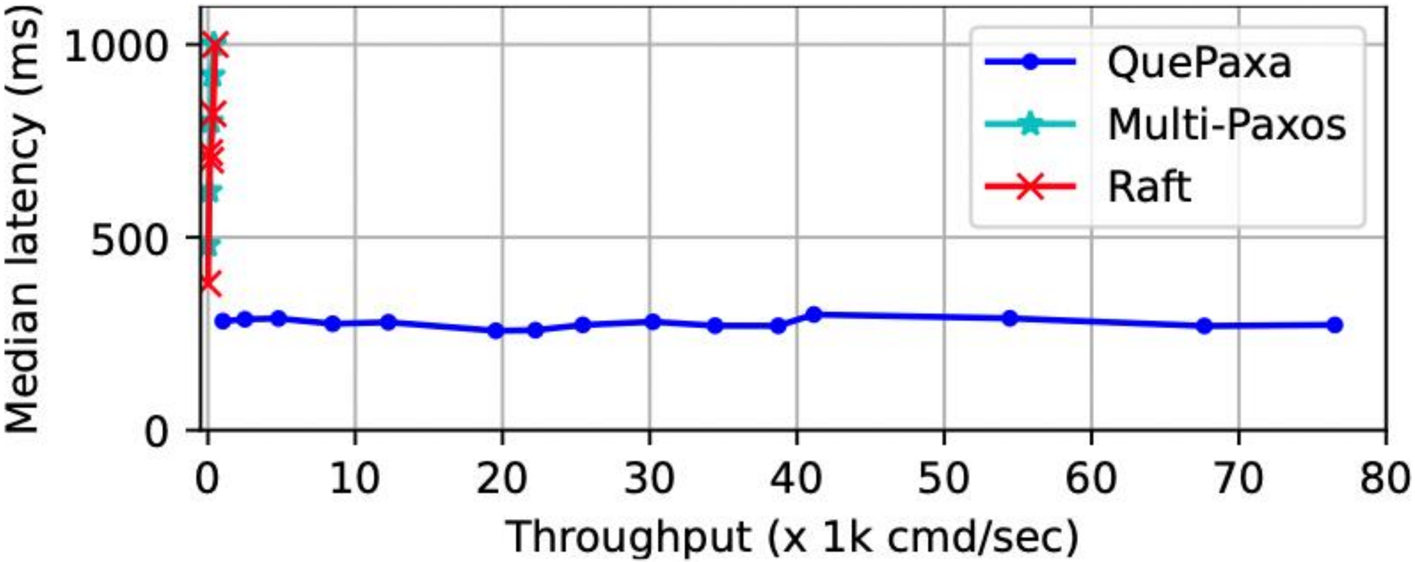
*extracted from Fig. 6 of QuePaxa.

Scalability



*extracted from Fig. 7 of QuePaxa.

Performance under Adversarial Conditions



*extracted from Fig. 7 of QuePaxa.

Conclusion

1. Observed issues with timeouts
2. Solution: QuePaxa!
3. Fast path & async consensus decision (guarantees liveness)
4. Hedging over timeouts
5. Adaptively choosing leaders
6. Experimental results

References

[1] Tennage, Pasindu, et al. "QuePaxa: Escaping the tyranny of timeouts in consensus." *Proceedings of the 29th Symposium on Operating Systems Principles*. 2023.

[2] Du, Hao, and David J. St Hilaire. "Multi-Paxos: An implementation and evaluation." *Department of Computer Science and Engineering, University of Washington, Tech. Rep. UW-CSE-09-09-02* (2009).

[3] Moraru, Iulian, David G. Andersen, and Michael Kaminsky. "Egalitarian paxos." *ACM Symposium on Operating Systems Principles*. 2012.

[4] Ongaro, Diego, and John Ousterhout. "In search of an understandable consensus algorithm." *2014 USENIX annual technical conference (USENIX ATC 14)*. 2014.

[5] Lamport, Leslie. "The part-time parliament." *Concurrency: the Works of Leslie Lamport*. 2019. 277-317.

[6] Haochen Pan, Jesse Tuglu, Neo Zhou, Tianshu Wang, Yicheng Shen, Xiong Zheng, Joseph Tassarotti, Lewis Tseng, and Roberto Palmieri. Rabia: Simplifying state-machine replication through randomization. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pages 472–487, October 2021.

Thank You! :)