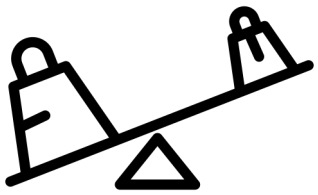
U LISBOA | UNIVERSIDADE DE LISBOA



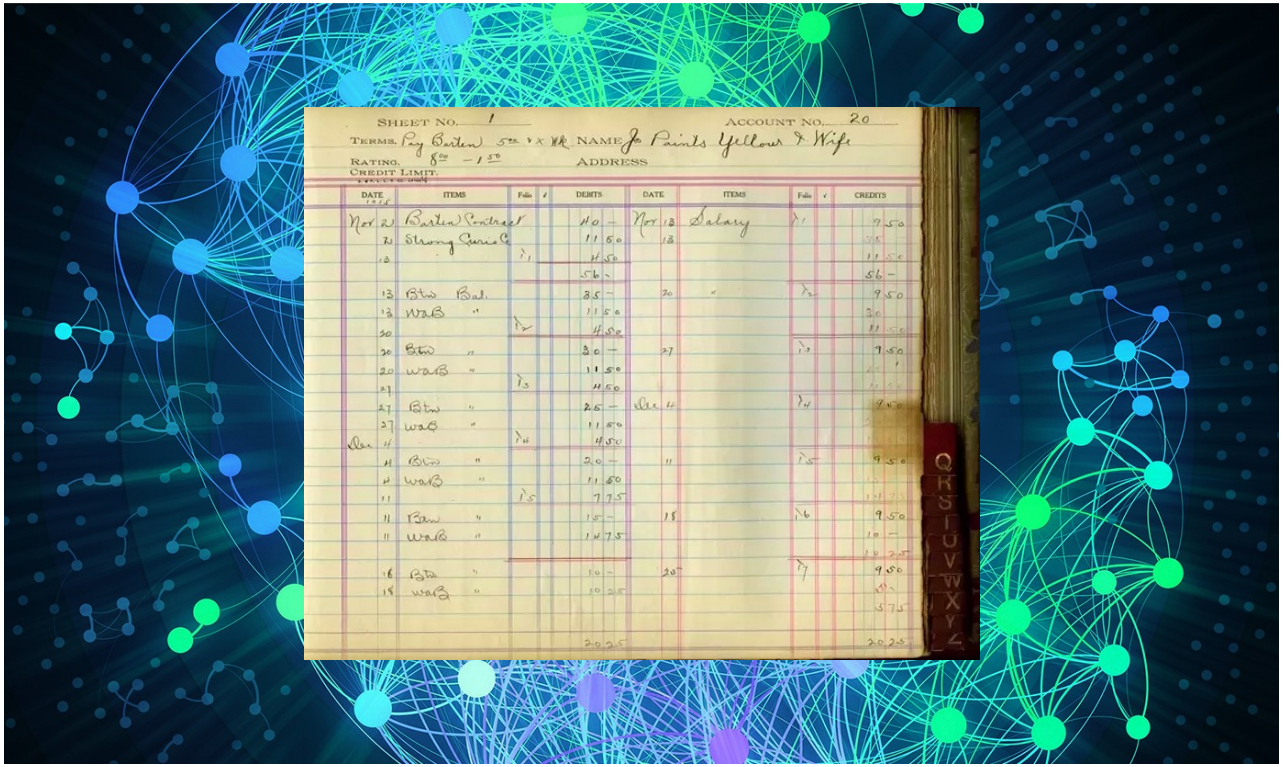
Embracing the Dice: Playing with Quorums to derive Efficient Blockchain Consensus

Alysson Bessani

Ciências ULisboa

LASIGE

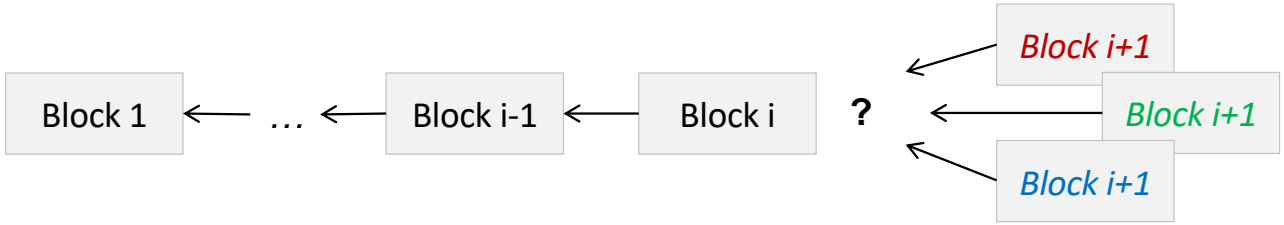
What is a Blockchain?



General ledger on top of a peer-to-peer network

What is a Blockchain?

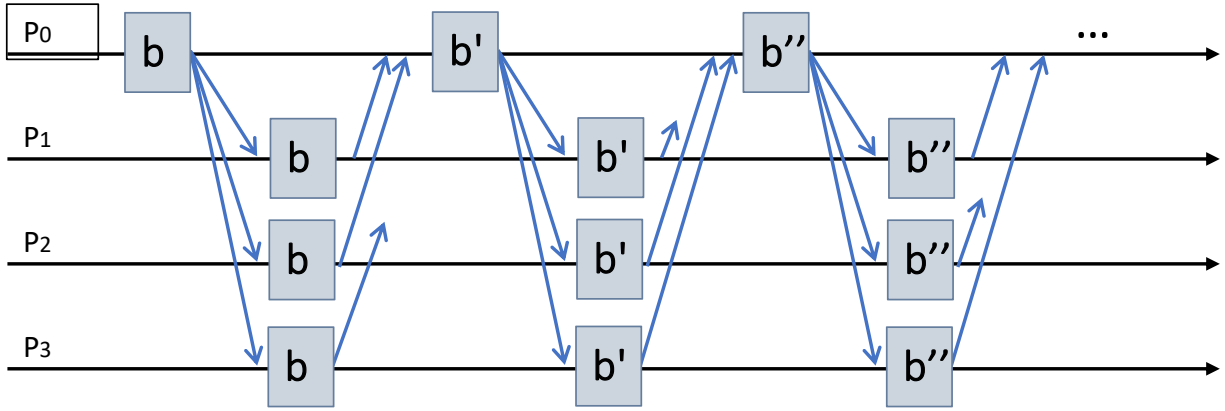
- It requires solving **distributed consensus** under **Byzantine faults**
  - Given a blockchain of *i* blocks, and several proposals for block *i+1*, how to decide (in a distributed way) which proposal to adopt?



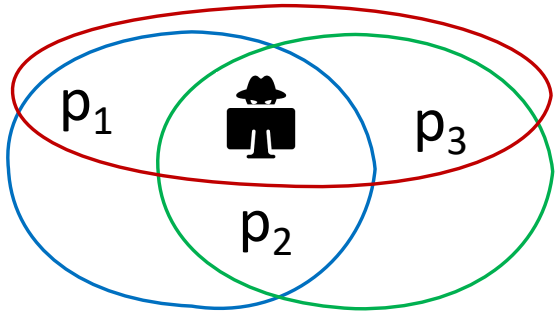
  - **Byzantine faults:** a faulty process can do anything. It can model hardware defects, software bugs, and even **intrusions.**

Byzantine Consensus Protocols

- Many consensus protocols use voting quorums to make progress without endangering safety properties (e.g., agreement, blockchain consistency)
- For example, **HotStuff** [PODC'19] uses leaders to propose blocks that need to be approved by > 2/3 of the servers (a quorum)



Why > 2/3 (3 out of 4)?

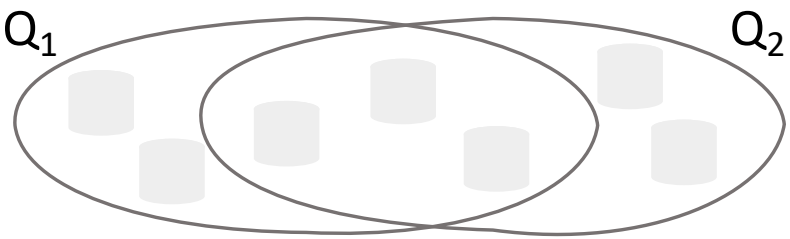


Byzantine Quorum Systems

- A set of subsets (*quorums*) of *n* servers satisfying:
  - *Consistency*: Every two quorums intersect in at least one correct server
  - *Availability*: There is at least one quorum of correct servers

- Dissemination quorum systems:
  - $n > 3t$ (e.g., $n = 3t+1$)
  - $q = \lceil (n+t+1)/2 \rceil$ (e.g., $q = 2t+1$)

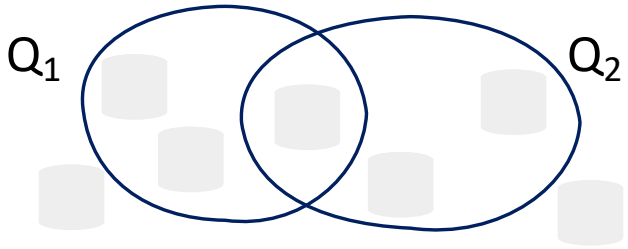
Example $n=7$, $t=2$:



Quorums with $q = 5$ servers
Intersections with at least $t+1 = 3$ servers
Always one correct!

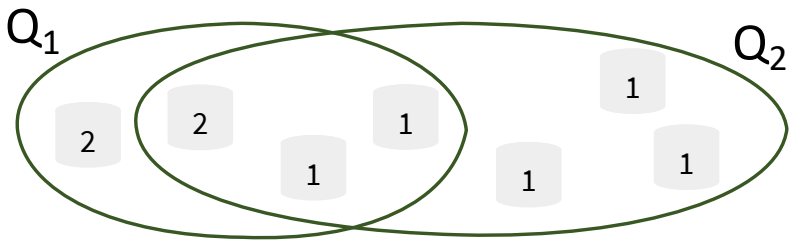
This talk:
Use of non-standard quorums for high-performance consensus

- **Probabilistic Quorums**
  - Smaller quorums of size $O(\sqrt{n})$ (instead of $O(n)$)
  - Quorum intersections contain one correct server with high probability



Less messages

- **Weighted Quorums**
  - Servers have different weights based on their "connectivity"
  - Some quorums are much smaller than others



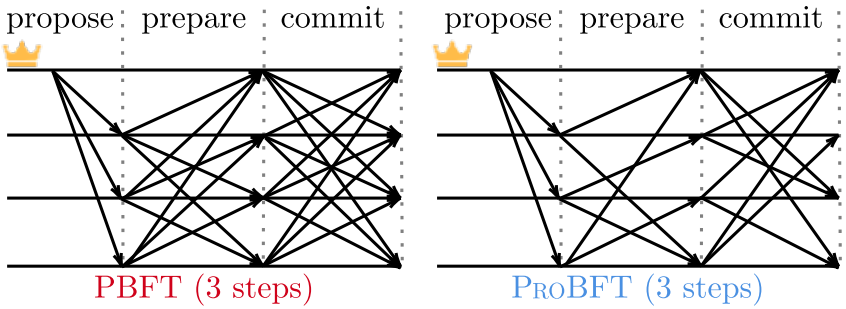
Lower latency

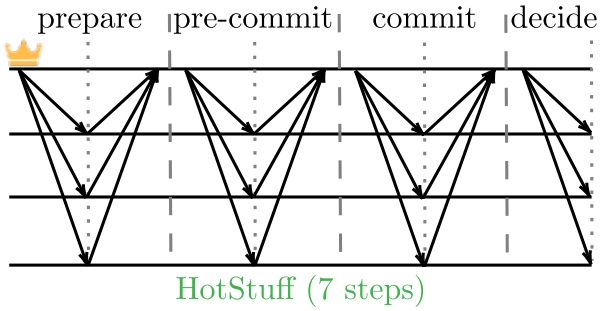
Probabilistic Quorums

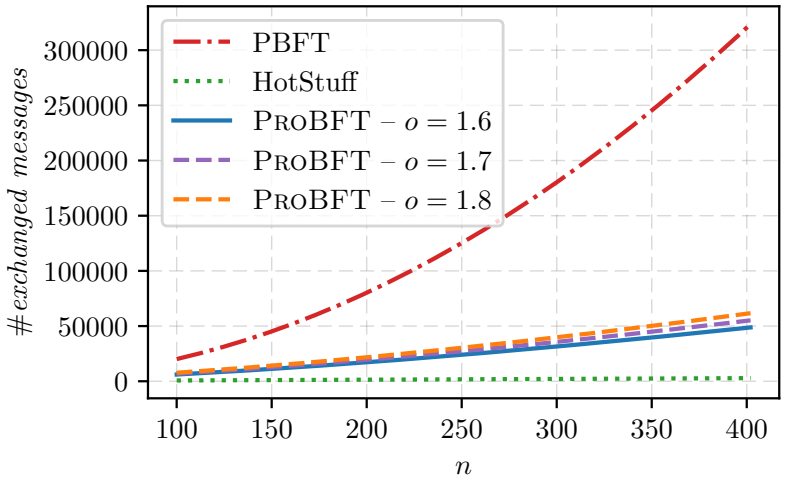
ProBFT and QScale

Probabilistic Byzantine Fault Tolerance

- ProBFT protocol [PODC'24]
  - PBFT with probabilistic quorums of size $O(\sqrt{n})$ (instead of $O(n)$)
  - Liveness (Termination) guaranteed with probability 1
  - Safety (Agreement) guaranteed with probability 1 - exp(-$\Theta(\sqrt{n})$)
- Requires $O(n\sqrt{n})$ messages (instead of $O(n^2)$ of PBFT)
- Same latency: 3 comm. steps

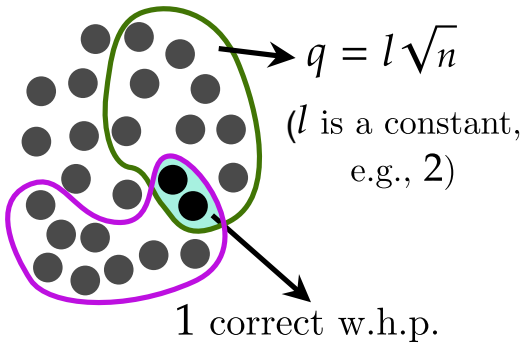




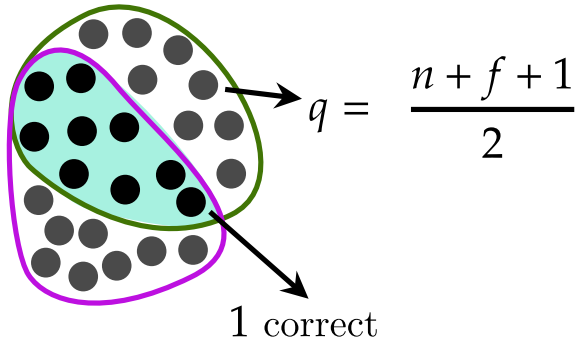


ProBFT main techniques

Probabilistic BQS

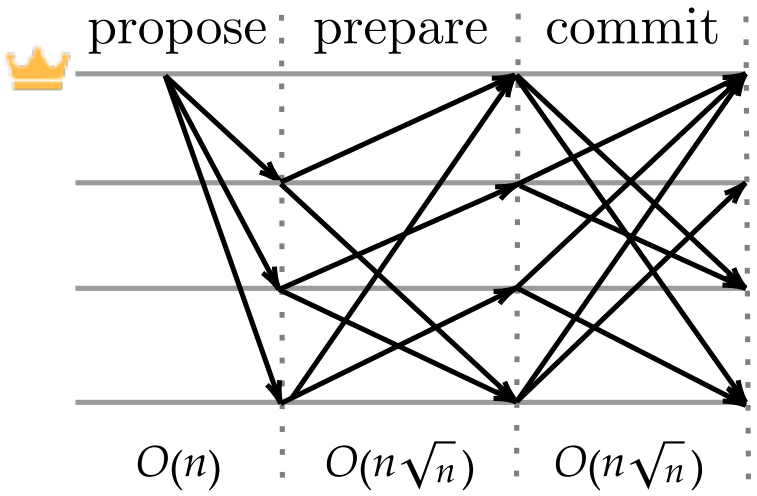


Dissemination BQS



- **Replica sampling**
  - Problem: Can't send messages to all nodes and wait for small quorums
  - Solution: Send messages to a random sample of $o \times q$ servers
- **Verifiable Random Functions (VRFs)**
  - Problem: Faulty nodes can handpick samples to include faulty nodes
  - Solution: Make nodes prove their samples were randomly generated

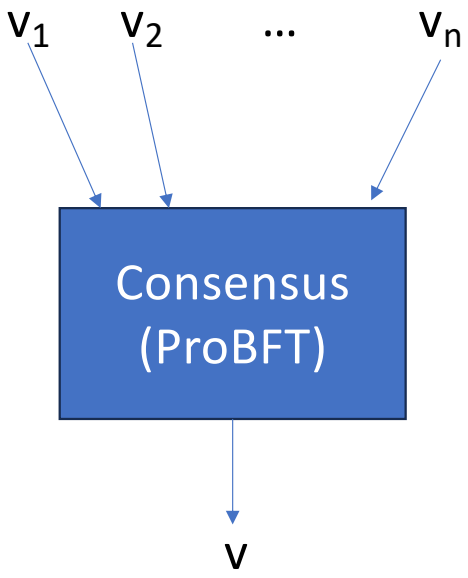
ProBFT in Action

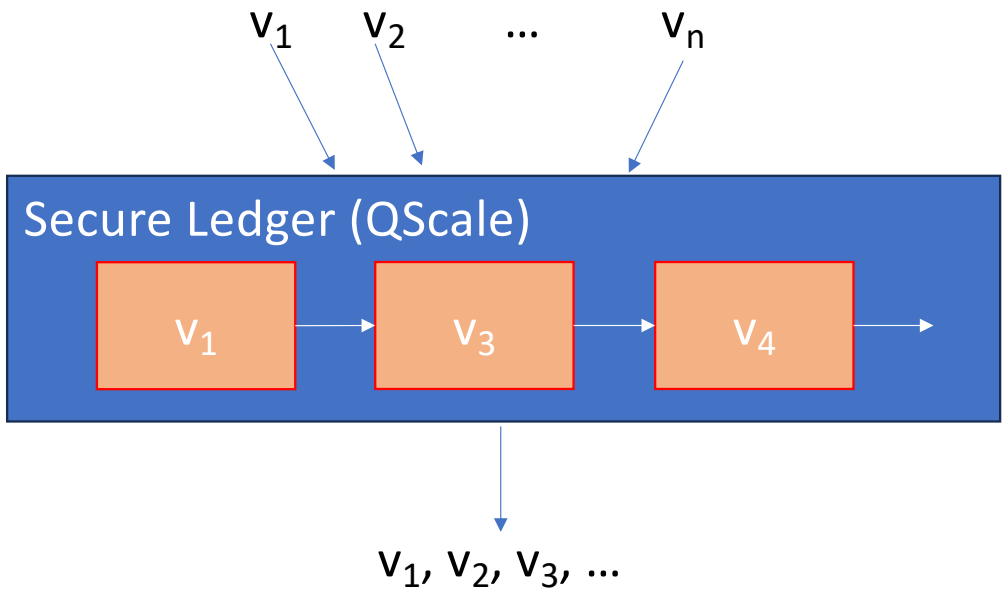


- Correct leader: Byzantine processes can stay silent and create a liveness issue
- Byzantine leader: Byzantine processes can create a safety issue
- In both cases, a PBFT-like view change protocol is executed

A Key Limitation

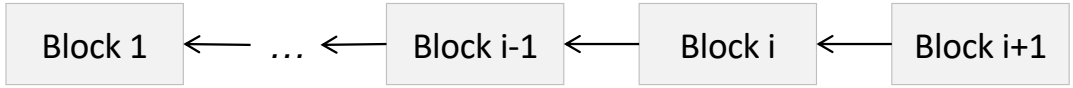
- ProBFT follows a propose/decide abstraction (single-shot consensus)
- Blockchains typically employ a chained consensus approach





PoW-based Public Ledgers (e.g., Bitcoin)

- Typically implemented through Nakamoto Consensus or its variations
- Key ideas:
  - Anyone can participate in the network
  - A block is added to the blockchain if a **cryptographic puzzle** (PoW) is solved
  - New blocks are disseminated in a peer-to-peer network through gossip
  - If multiple proposals for extending the chain are received (forks), the longest proposal is used (forks are pruned)



Nakamoto Consensus Security (as originally stated)

Adversary controls ~30% of the CPU

```
q=0.3
z=0     P=1.0000000
z=5     P=0.1773523
z=10    P=0.0416605
z=15    P=0.0101008
z=20    P=0.0024804
z=25    P=0.0006132
z=30    P=0.0001522
z=35    P=0.0000379
z=40    P=0.0000095
z=45    P=0.0000024
z=50    P=0.0000006
```

Wait for more blocks for confirmation

~2^{-10} (10 bits of security)

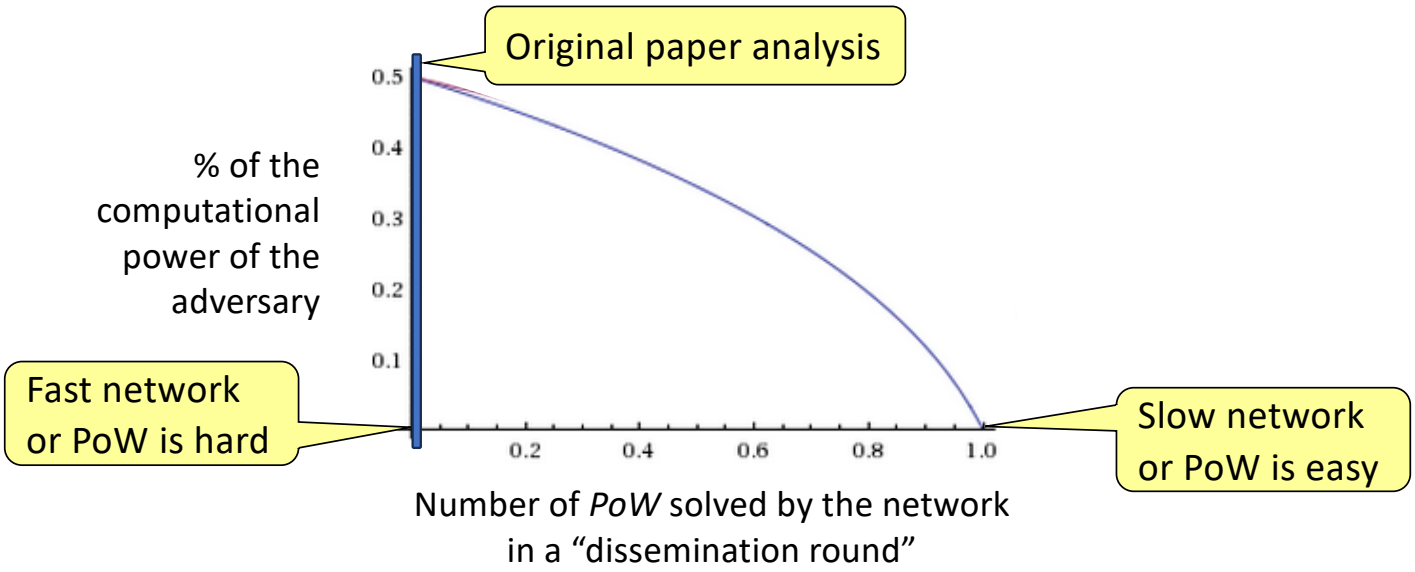
```
P < 0.001
q=0.10    z=5  ~50 minutes
q=0.15    z=8
q=0.20    z=11
q=0.25    z=15
q=0.30    z=24
q=0.35    z=41
q=0.40    z=89
q=0.45    z=340 >56 hours
```

Adversary controls more CPU

https://bitcoin.org/bitcoin.pdf (Section 11)

Actually, it is worse…

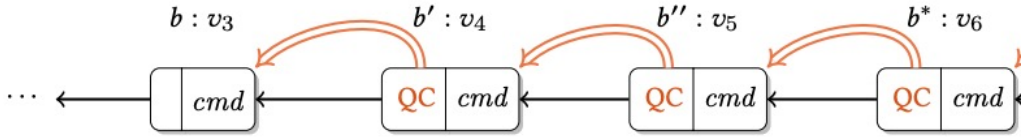
NC works **if the adversary controls less than half of the total computing power** and the network disseminates data instantaneously

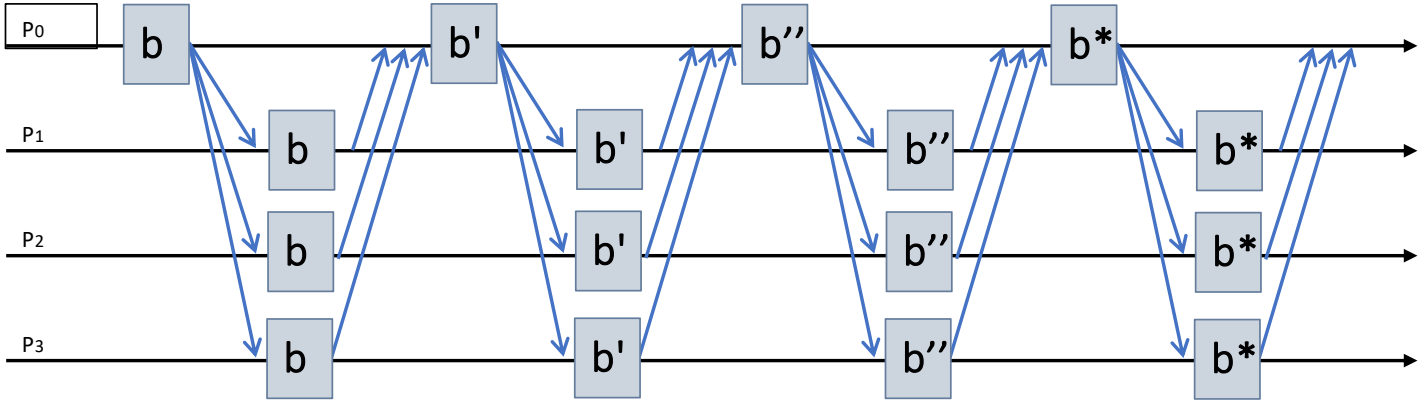


Blockchain-inspired Quorum-based Protocols

- <u>Chained consensus (CC)</u>
  - Simple block approval protocol
  - Commit rule
  - Chain selection rule
- Used in Proof-of-Stake or permissioned blockchains

HotStuff [PODC'19]





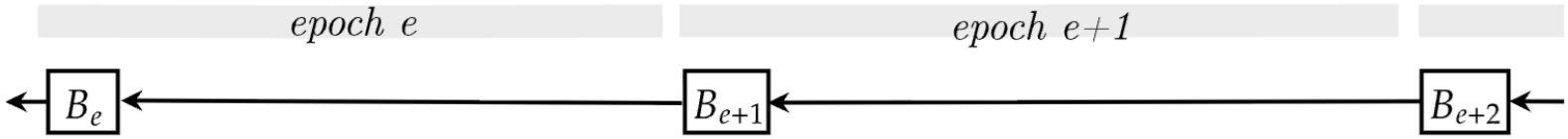
We want a probabilistic protocol…

- Without all-or-nothing guarantees (as in NC)
- Resistant to reasonable adversaries (as in NC)
- In which users decide the finality requirements of their transactions (as in NC)
- Without using PoW or PoS (as in CC)
- **Efficient and with good "performance" in favourable executions**

We want a protocol that runs with minimal configuration, can adapt to network conditions, and keeps ordering blocks in a blockchain. Users see their transactions ordered and decide when they are safe enough.

QScale: Probabilistic Chained Consensus

- The protocol works in **fixed-duration epochs** (e.g., 12 seconds)
- Each epoch has a designated **leader**, defined in advance, i.e., $l_e = \Pi(e)$
- The leader proposes a **block to be certified** in the epoch, extending its current certified chain



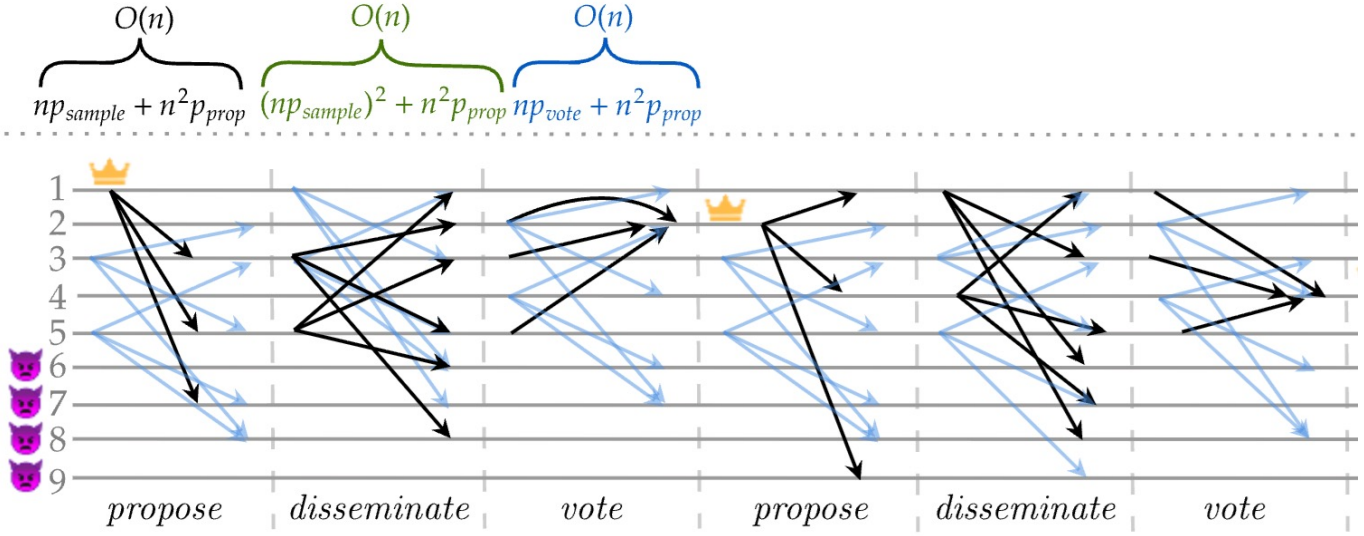
- Probabilistic methods are used to address two issues:
  - Decrease the number of messages exchanged (and comm. complexity)
  - Alleviate the bandwidth bottleneck at leaders

QScale System Model

- Committee of *n* nodes with less than *n*/3 compromised
- Static adversary: Byzantine nodes are defined before the system starts
- Partial synchrony
  - The system is asynchronous until an unknown Global Stabilisation Time
  - After that, it becomes synchronous with known network latency bounds
- PKI infrastructure for signing messages (permissioned system)
- Local source of randomness used to implement a Verifiable Random Function (VRF), to generate provable random numbers

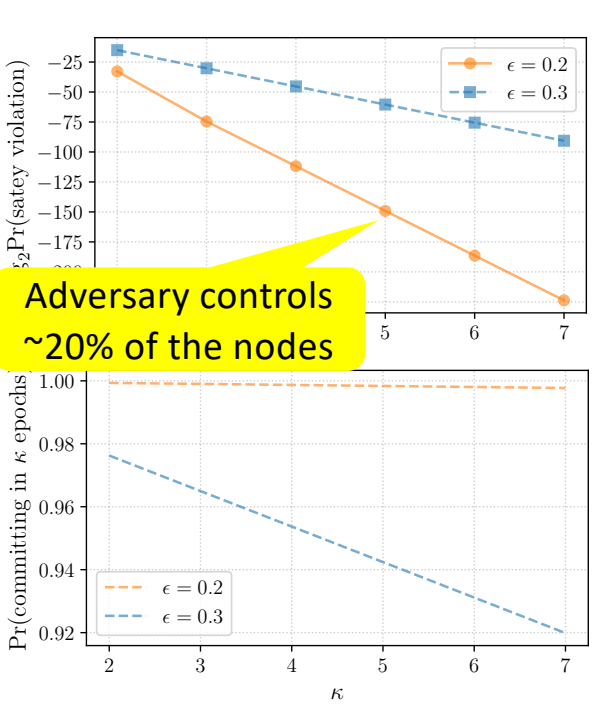
QScale in Action

- Each epoch has three phases: *propose*, *disseminate*, and *vote*
  - Messages are sent to random samples (VRF) in the first two phases
  - A random sortition (VRF) selects who will send votes to the next leader
  - The next leader waits for a quorum of approvals to certify a block
- As in NC, QScale assumes the block payload is disseminated through gossip

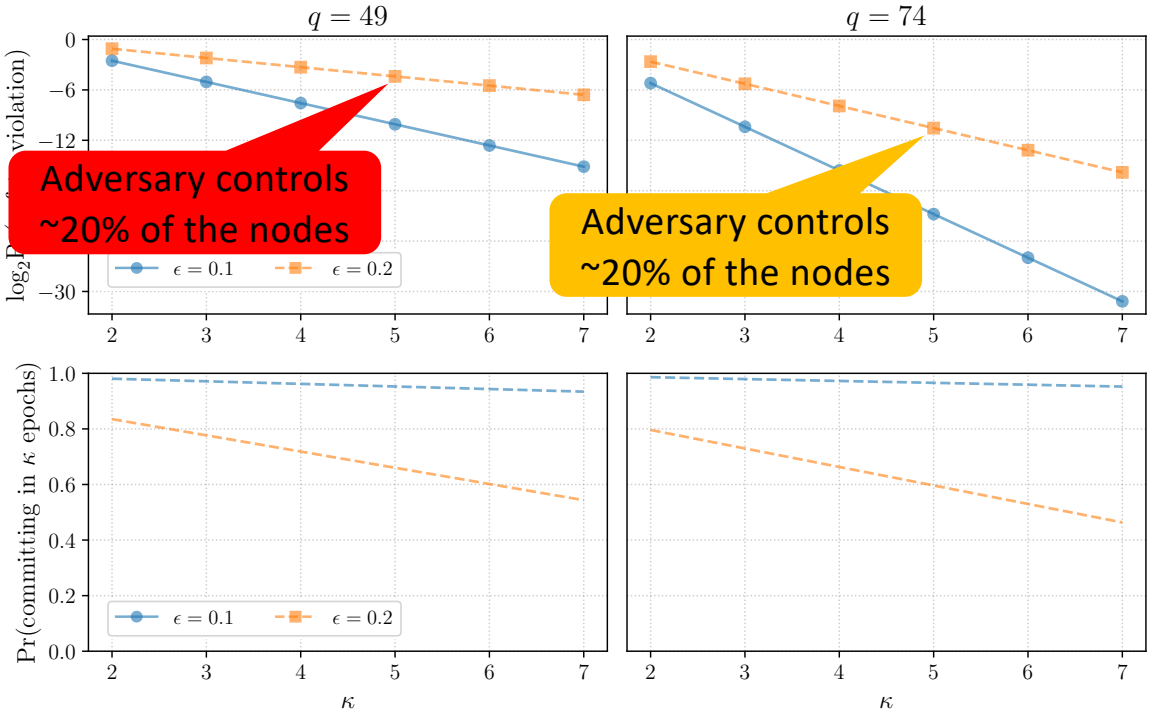


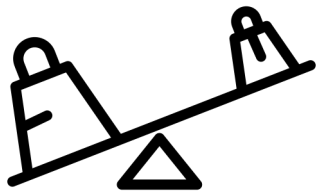
Concrete Probability Numbers (n = 500)

Good Network
(Synchrony)



Bad Network
(Asynchrony)



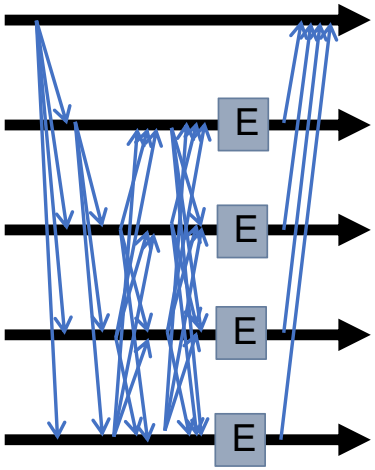


Weighted Quorums

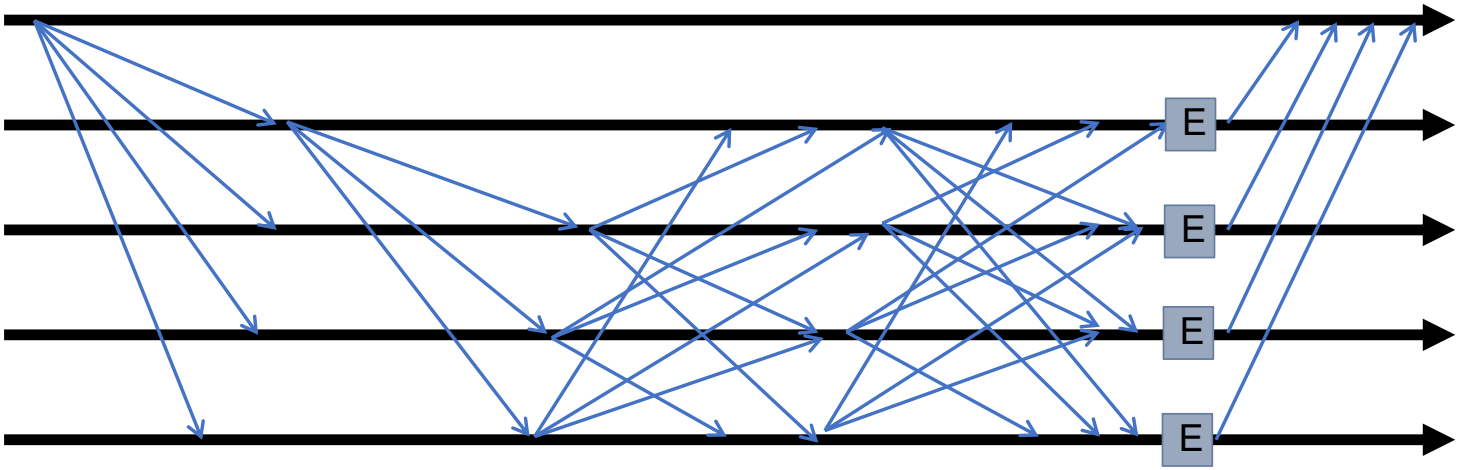
WHEAT, AWARE, and Mercury

Fast Wide-Area Byzantine Replication

- PBFT [OSDI'99] (a latency-optimal) protocol in different networks

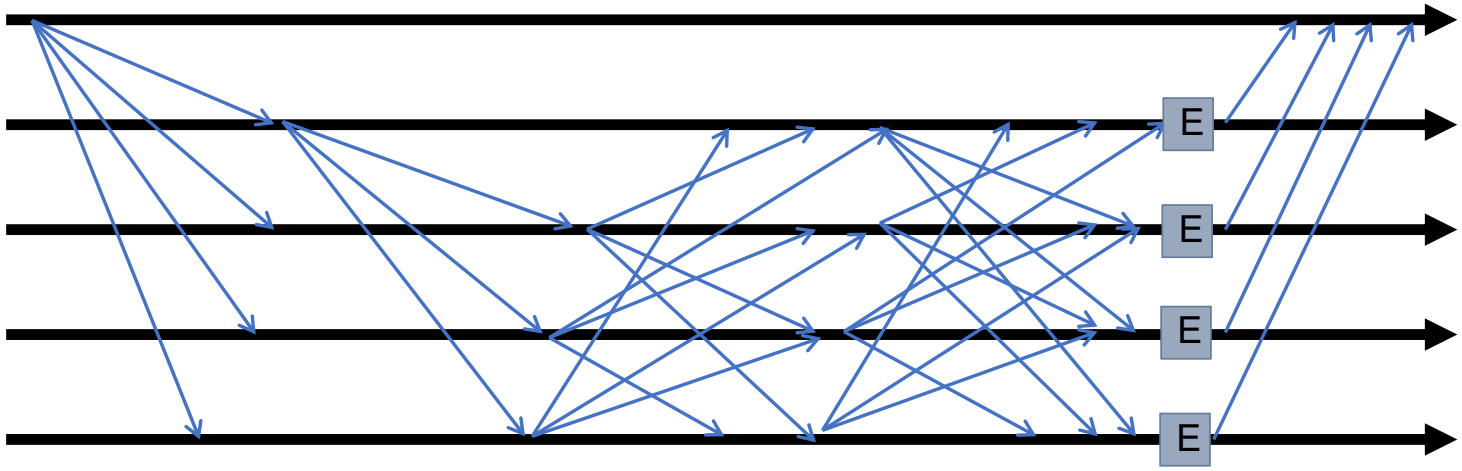


Nodes in the same data center



Geographically-dispersed nodes

Wide-Area Byzantine Consensus



- Given the replica locations, the links speed have a physical limit, i.e., they can never be better than **67% of the speed of light**
  - $Latency_{a \rightarrow b} = d_{a \rightarrow b}/0.67c$
- Knowing the latencies of the links, it is possible to calculate the **physical latency lower bound** of the protocol
  - Using the fastest quorum and best positioned leader

Problem: 0.67c is the best possible link speed, not the actual link speed

Question: Is it possible to reach/surpass the physical latency lower bound?

YES, but we need to CHEAT (weighted quorums)!

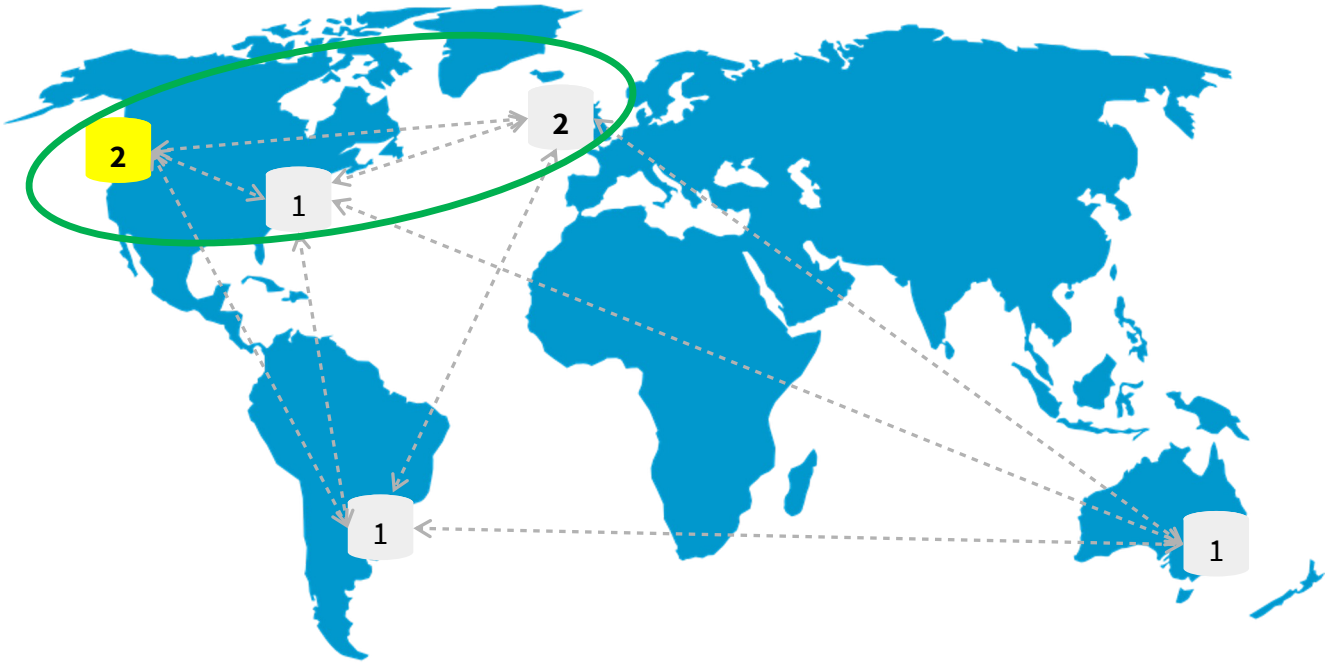
WHEAT: WeigHt-Enabled Active replicaTion

[SRDS'15]

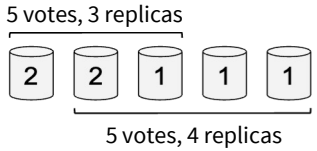
- Use optimizations that lead to significant latency reduction:
  - Single leader in the best-connected site
  - Tentative executions (from PBFT)
  - Employs smaller quorums (weighted replication)

- Weighted replication: safe voting assignment scheme for SMR
  - **Uses Δ extra replica(s) for quorum formation**
  - Improves latency by enabling more choice upon quorum formation
  - Needs to preserve quorum intersection and tolerance to *f* faulty replicas

Weighted BFT Replication



N=4, f=1, **Δ=1 (extra)**



Weighted quorums,
One set of 3 out-of 5
and any set 4 out-of 5

Weighted BFT Replication

- Every replica has a number of votes (its weight)
- Quorum properties:
  - **Consistency**: All quorums that hold Q votes intersect by at least one correct replica
  - **Availability**: There is always a quorum available in the system that holds Q votes
  - **Safe minimality**: There exists at least one minimal quorum in the system
- How to distribute weight?

Weighted BFT Replication

Define the number of replicas *u* that hold $V_{max} > 1$ votes, without violating *f*

Crash fault tolerance

$$n = 2f + 1 + \Delta$$

$$N_v = \sum V_i = 2F_v + 1$$

$$Q_v = F_v + 1$$

$$u = f$$

Byzantine fault tolerance

$$n = 3f + 1 + \Delta$$

$$N_v = \sum V_i = 3F_v + 1$$

$$Q_v = 2F_v + 1$$

$$u = 2f$$

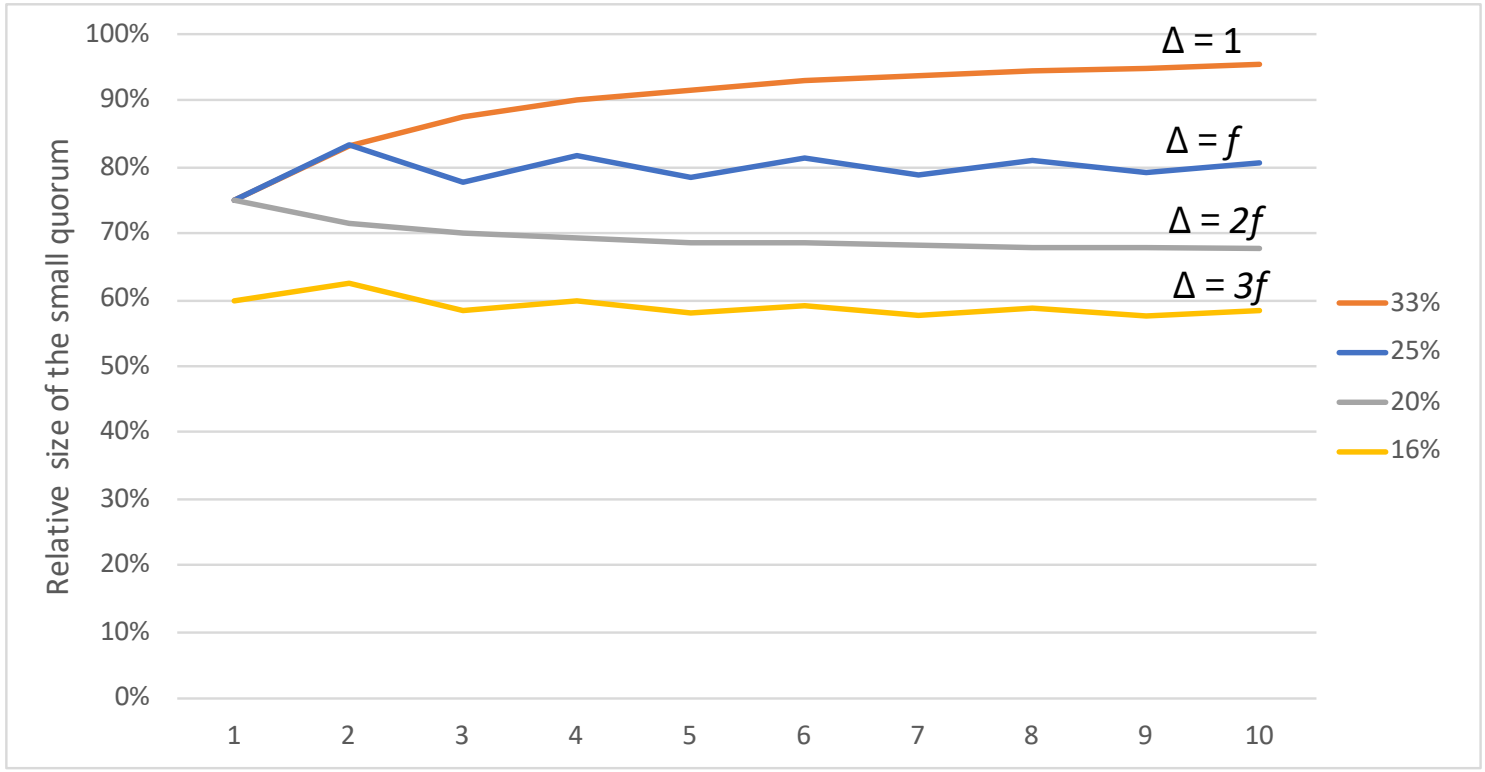
$$F_v = \Delta + f$$

$$V_{max} = \frac{\Delta + f}{f} = 1 + \frac{\Delta}{f}$$

Input: ***f*** **and** **Δ**

Output: ***u*** **and** V_{max}

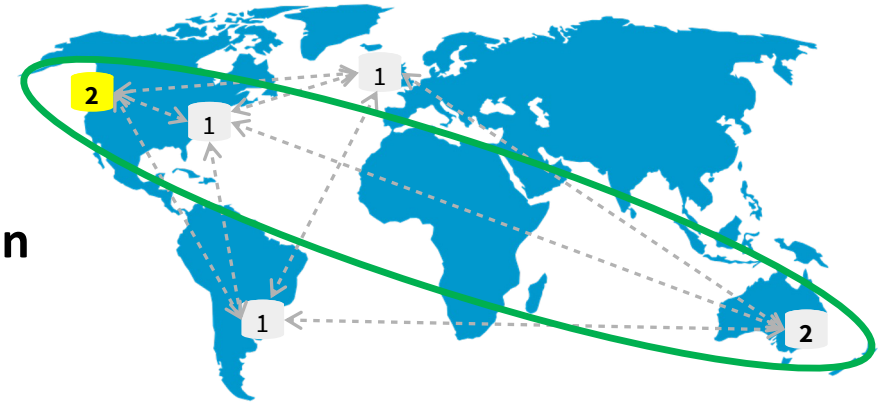
Size of fast quorums with different *f* and Δ



AWARE: Adaptive Wide-Area REplication

[TDSC'22]

The benefit of weighted replication depends on **choosing an optimal weight configuration**

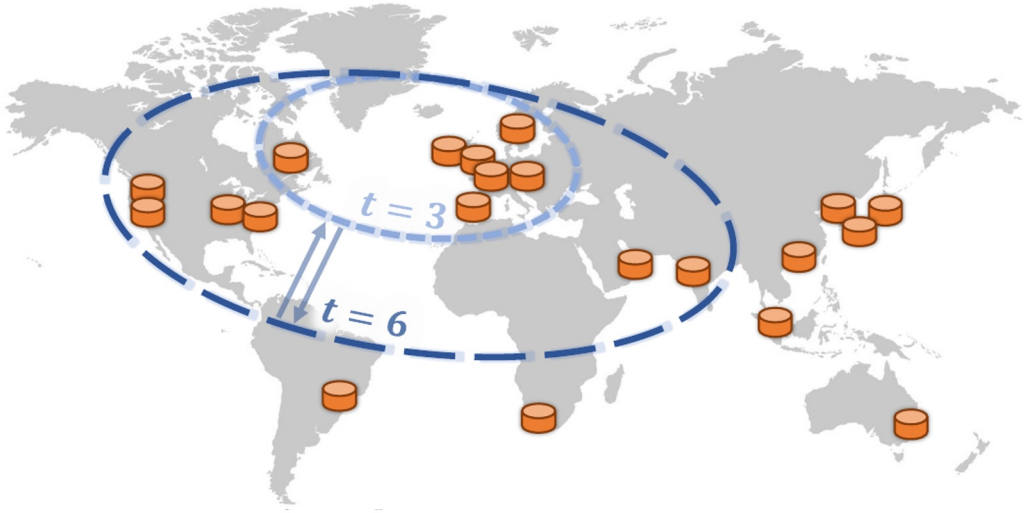


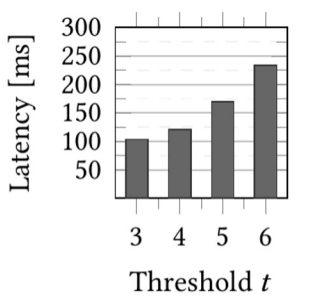
- The environment of the system (i.e., network characteristics) may **change at runtime** (e.g., due to network reconfigurations or a DDoS attack)

AWARE enables a geo-replicated consensus-based system to **adapt to its environment!**

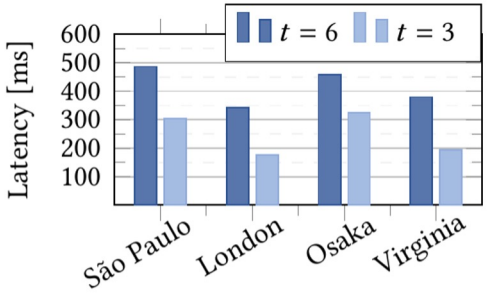
How to chase lightspeed consensus

- **Key observation**: using smaller quorums can accelerate consensus
- **Size of quorums** depends *on the* configured resilience threshold ***t***
- **Weighted replication** is also crucial for reaching smaller quorums





(b) Consensus latency *vs.* resilience threshold.



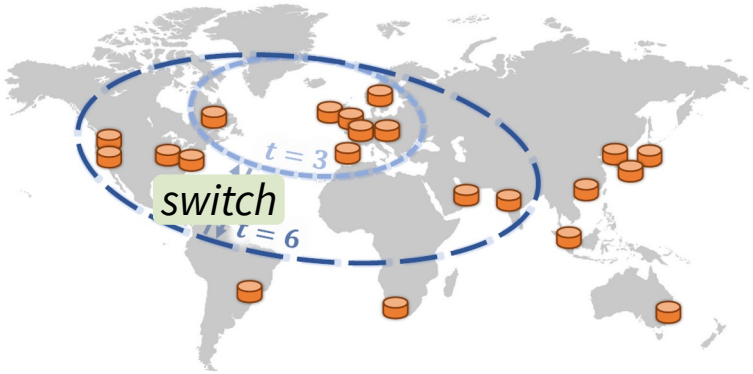
(c) End-to-end transaction latencies observed by clients in different regions.

Mercury key ideas

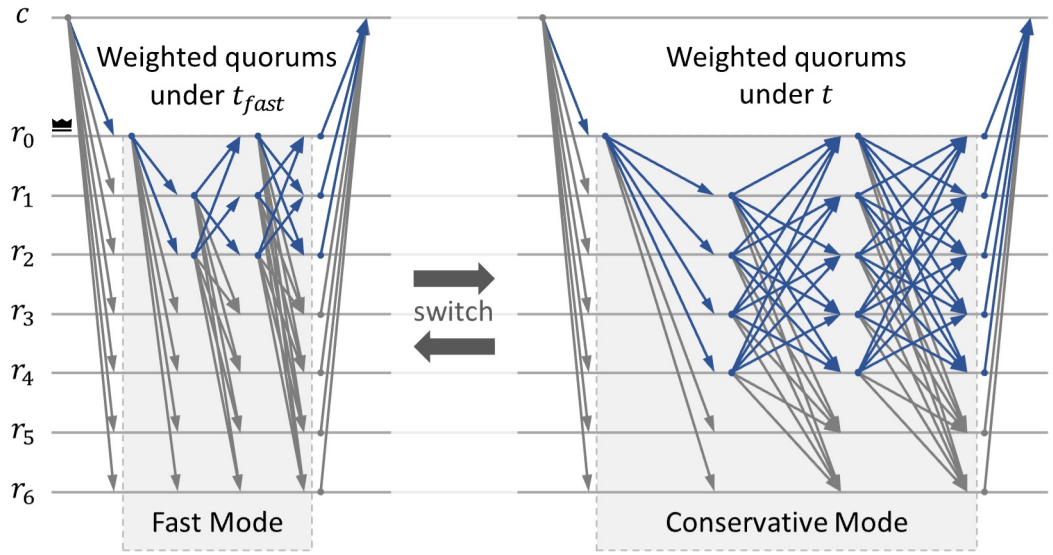
BFT-SMaRt [DSN'14] **+**
Adaptive weighted replication +
BFT Forensics support

Core idea:

- Two modes of operations
- Use of smaller quorums a tolerating lower resilience threshold ($t_{fast} < n/6$)
- Detect failures and reconfigure the system to the conservative mode tolerating $t < n/3$



(a) Weighted quorums sizes with $t = 6$ and $t = 3$.



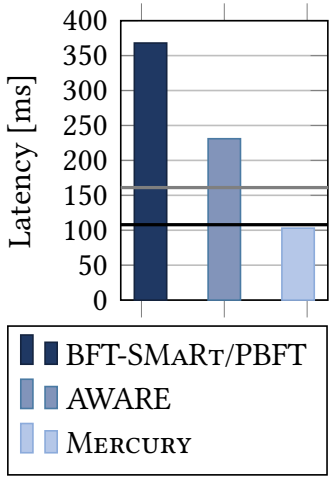
MERCURY Acceleration on AWS (*n*=21)

Consensus Acceleration

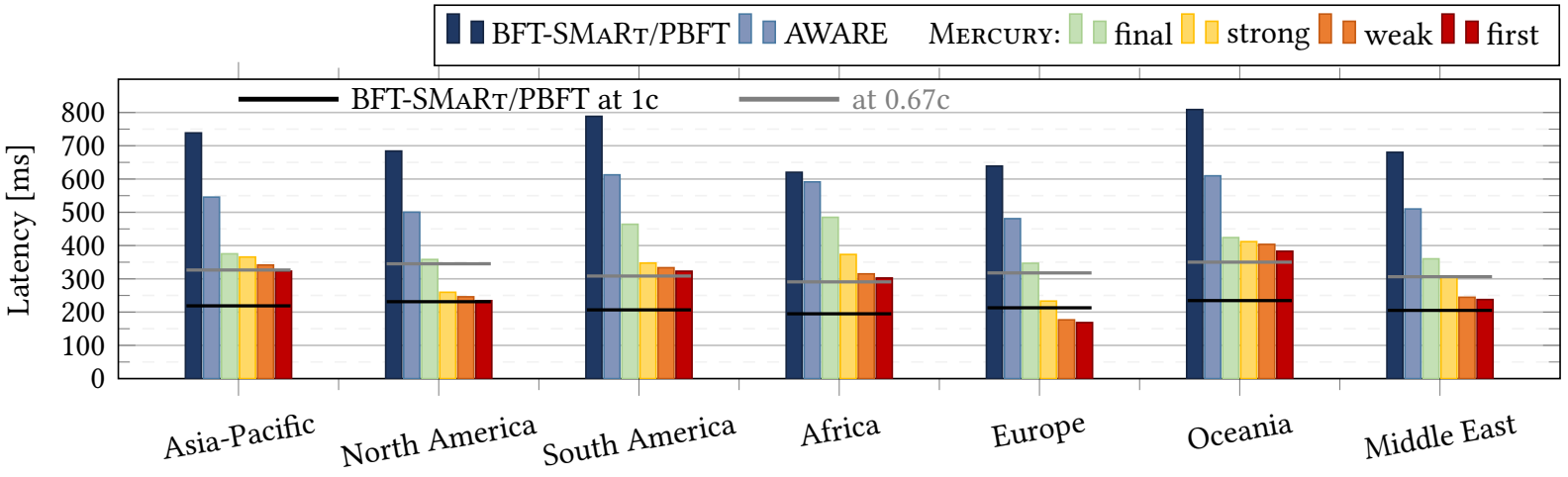
- **3.57×** faster decisions

Client End-to-End

- **1.87×** faster (final)
- Up to **2.90×** (speculation)



(a) Consensus latency.



(b) End-to-end latencies observed by clients in protocol executions with BFT-SMaRt, AWARE, and MERCURY. Client results are averaged over all regions per continent.

Open Questions

What's next?

- Probabilistic Protocols
  - How to make QScale responsive?
  - Can we make QScale adaptive?
  - Is it possible to solve consensus with fewer than *O(n)* messages?

- Weighted Replication
  - Can we provide other types of weight distribution schemes?
  - How to securely monitor and reassign weights in the presence of Byzantine nodes?

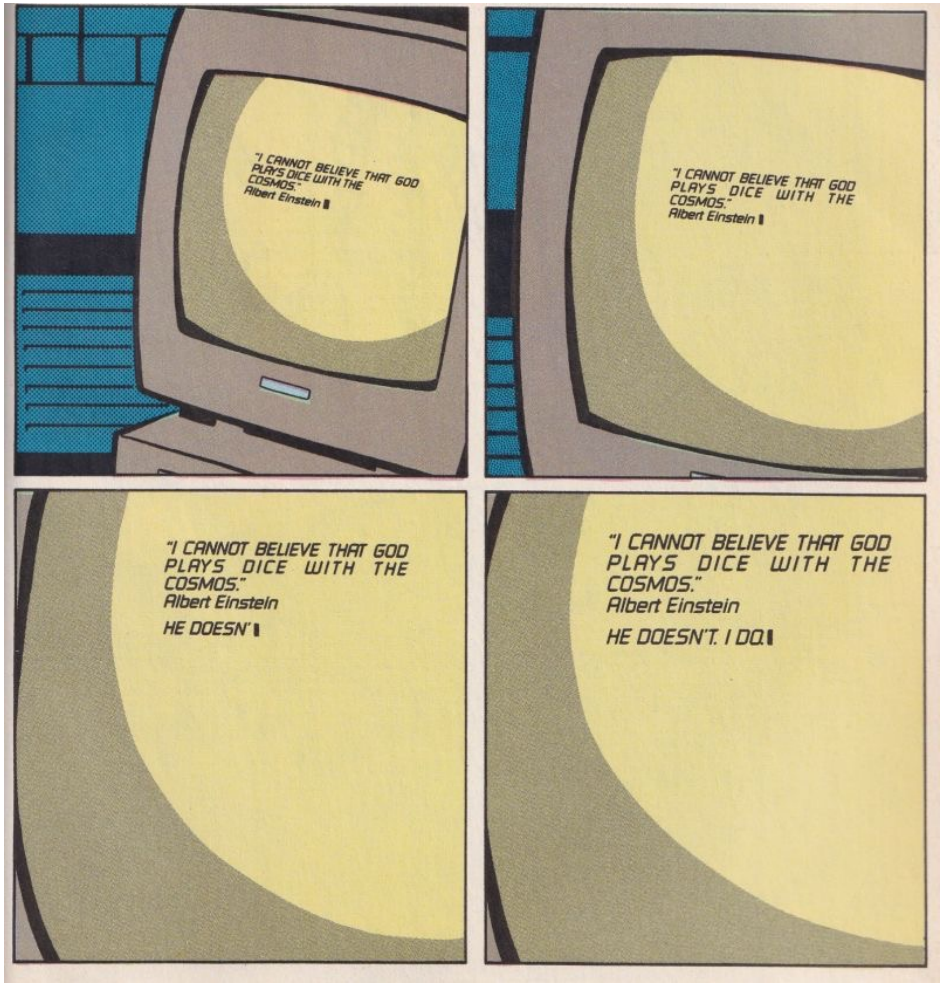
Questions?

- Alysson Bessani
  - anbessani@fc.ul.pt
  - www.di.fc.ul.pt/~bessani

Ciências ULisboa LASIGE

Fundação para a Ciência e a Tecnologia

This work is partially supported by FCT through the SMaRtChain project (2022.08431.PTDC), and the LASIGE Research Unit (UID/00408/2023).



Morrison & Troug, Animal Man 8, DC Comics, 1989.

References

Yin et al. **HotStuff: BFT Consensus with Linearity and Responsiveness**. PODC'19.

Avelãs et al. **Probabilistic Byzantine Fault Tolerance**. PODC'24.

Heydari et al. **QScale: Probabilistic Chained Consensus for Moderate-Scale Systems**. https://arxiv.org/abs/2510.01536. 2025.

Castro, Liskov. **Practical Byzantine Fault Tolerance**. OSDI'99.

Sousa, Bessani. **Separating the WHEAT from the Chaff: An Empirical Design for Geo-replicated State Machines**. SRDS'15.

Berger et al. **AWARE: Adaptive Wide-Area Replication for Fast and Resilient Byzantine Consensus**. IEEE TDSC'22.

Berger et al. **Chasing Lightspeed Consensus: Fast Wide-Area Byzantine Replication with Mercury**. Middleware'24.

Bessani et al. **State Machine Replication for the Masses with BFT-SMaRt**. DSN'14.