Making Byzantine Consensus Live

By Manuel Bravo, Gregory Chockler, Alexey Gotsman

Presented by Che-Yu Chang, Yan-Da Chen, Ling-Yuan Chen

October 24th Fall 2022

Department of Computer Science

UC Davis

Outline

- Liveness
- Partial Synchrony
- Synchronizer Specification
- FastSync
  - Switching view
  - Relaying Wishes

Byzantine Consensus

- Several replicas need to agree on a single value
- Some replicas may be malicious: $\leqq$ f out of $n = 3f + 1$
- Byzantine consensus: decision finality, but permissioned system - fixed set of participants
- Blockchain consensus *(proof-of-work, proof-of-stake)*: no hard finality guarantees, but permissionless system - anyone can participate

The Liveness Problem

- Many complications arise from ensuring liveness: protocol has to be responsive $\Rightarrow$ make a decision
- Can not make consensus both safe and live in an asynchronous network = no bound on message delivery time
  $\Rightarrow$ Can not tell a crashed replica from a slow one
- Provide safety and liveness only under synchrony

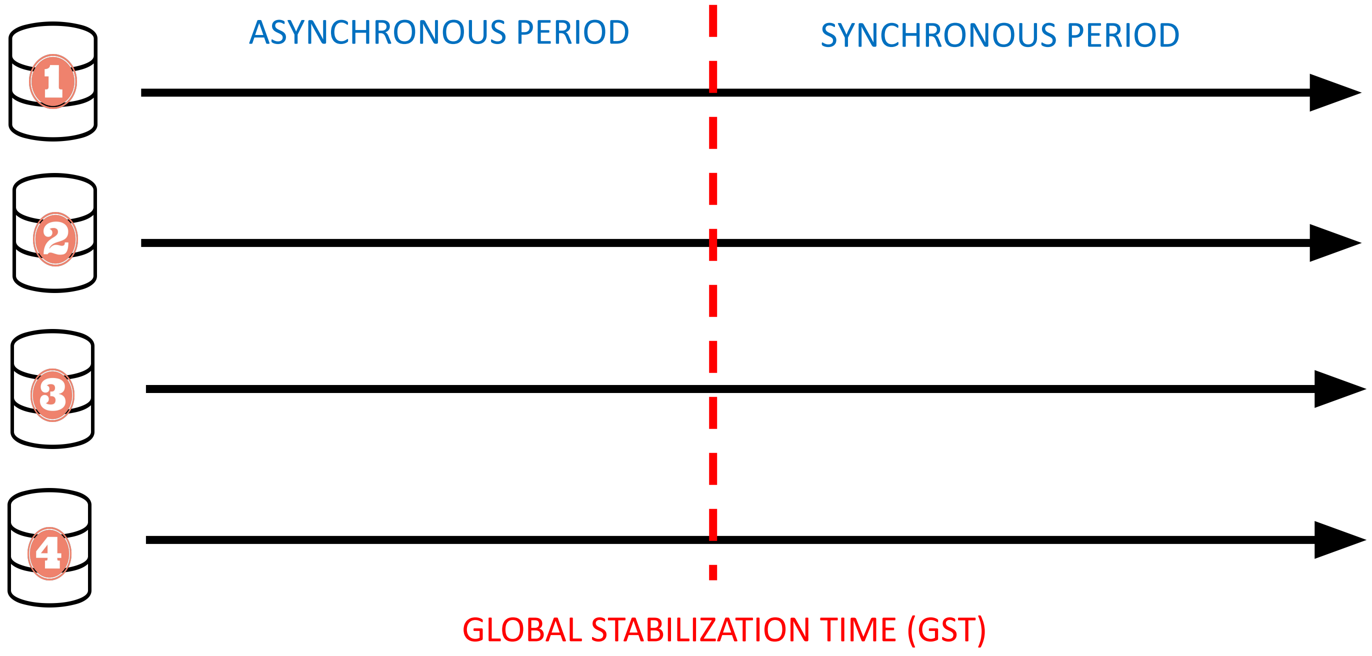
Partial Synchrony

- **Synchronous model:** there exists some known finite time bound δ. For any message sent, the adversary can delay its delivery by at most δ

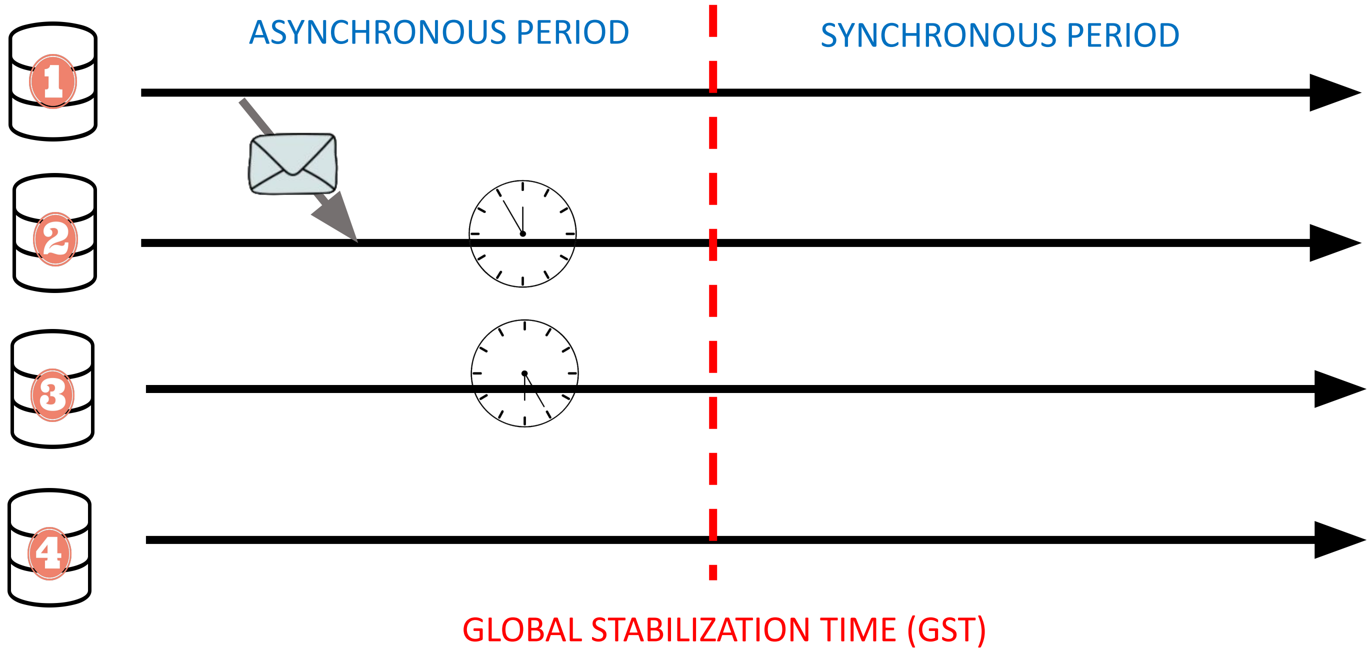
- **Asynchronous model:** for any message sent, the adversary can delay its delivery by any finite amount of time.

- **Partial synchrony model:** middle ground between these two models. The assumption is that there exists some finite time bound δ and a special event called GST (Global Stabilization Time).

Partial Synchrony

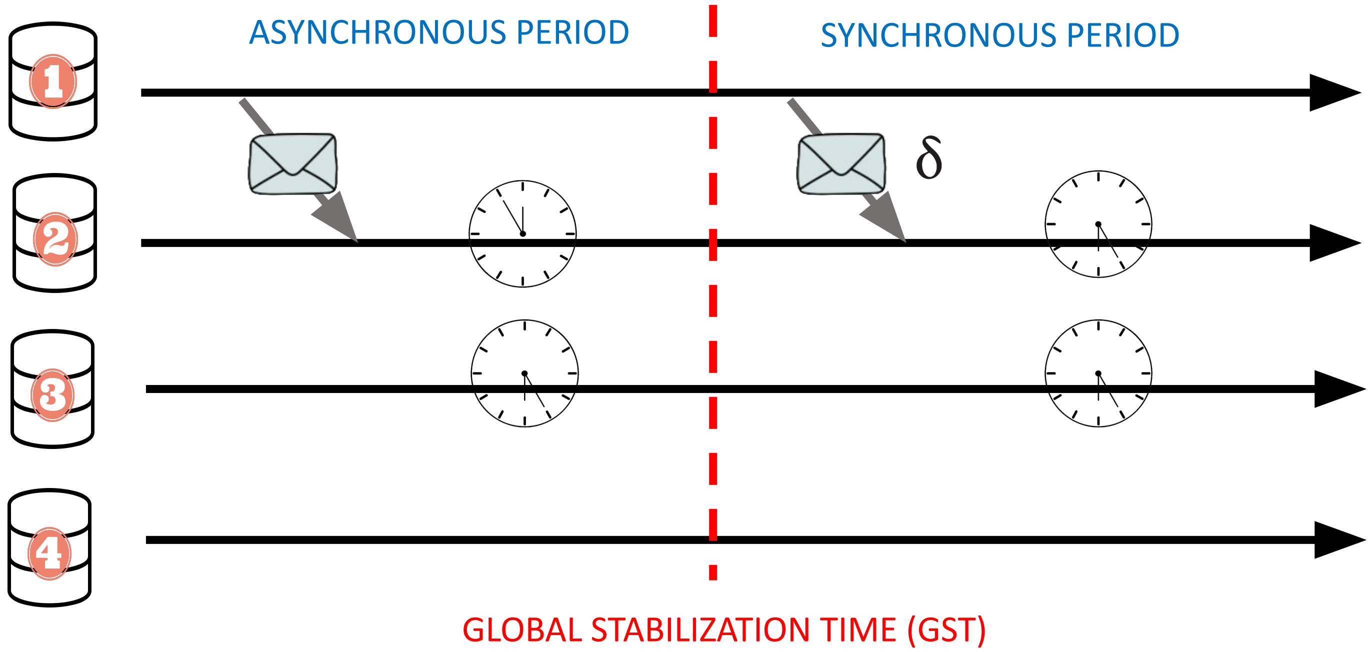


Partial Synchrony



- Messages delayed or lost
- Replica clocks out of sync: synchronized using messages

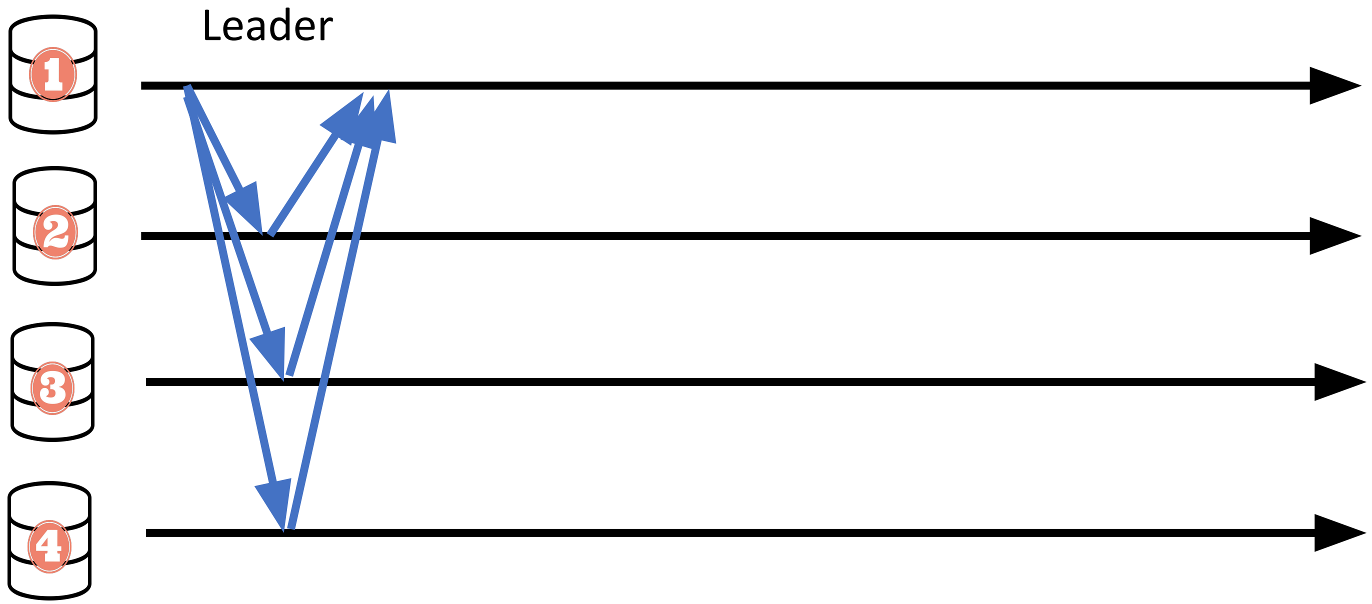
Partial Synchrony



- Messages delayed or lost
- Replica clocks out of sync: synchronized using messages

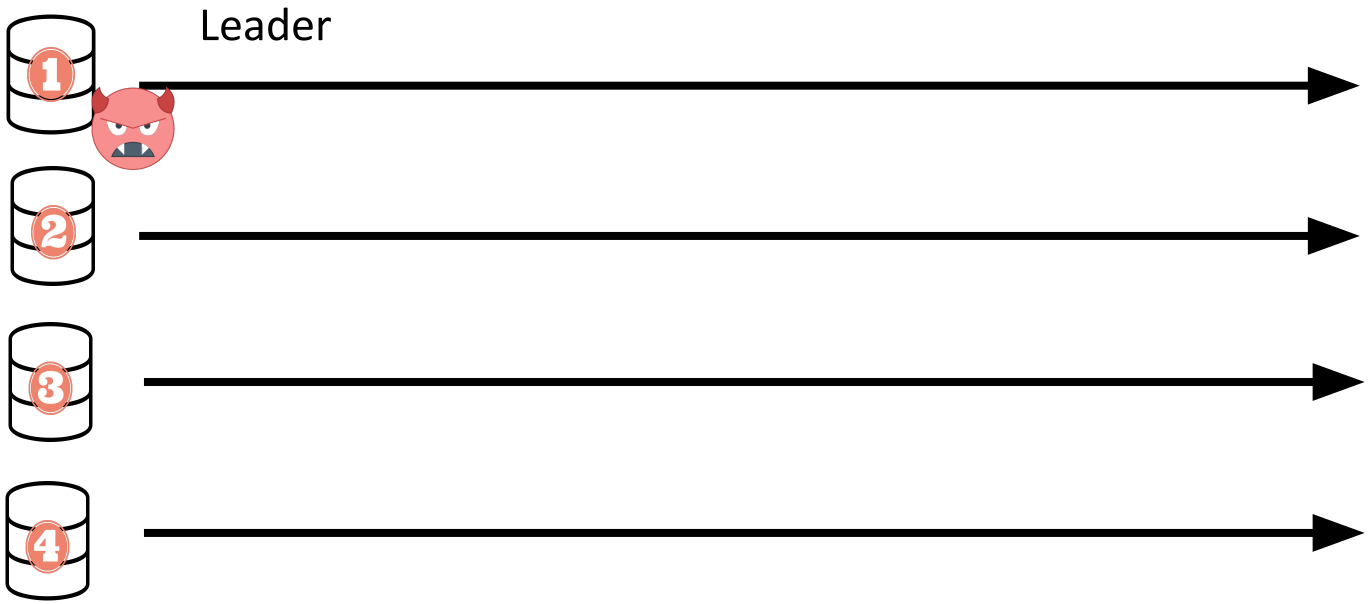
- Messages between non-faulty replicas delivered within and unknown time δ
- Replica clocks track real time.

Leader-driven Consensus



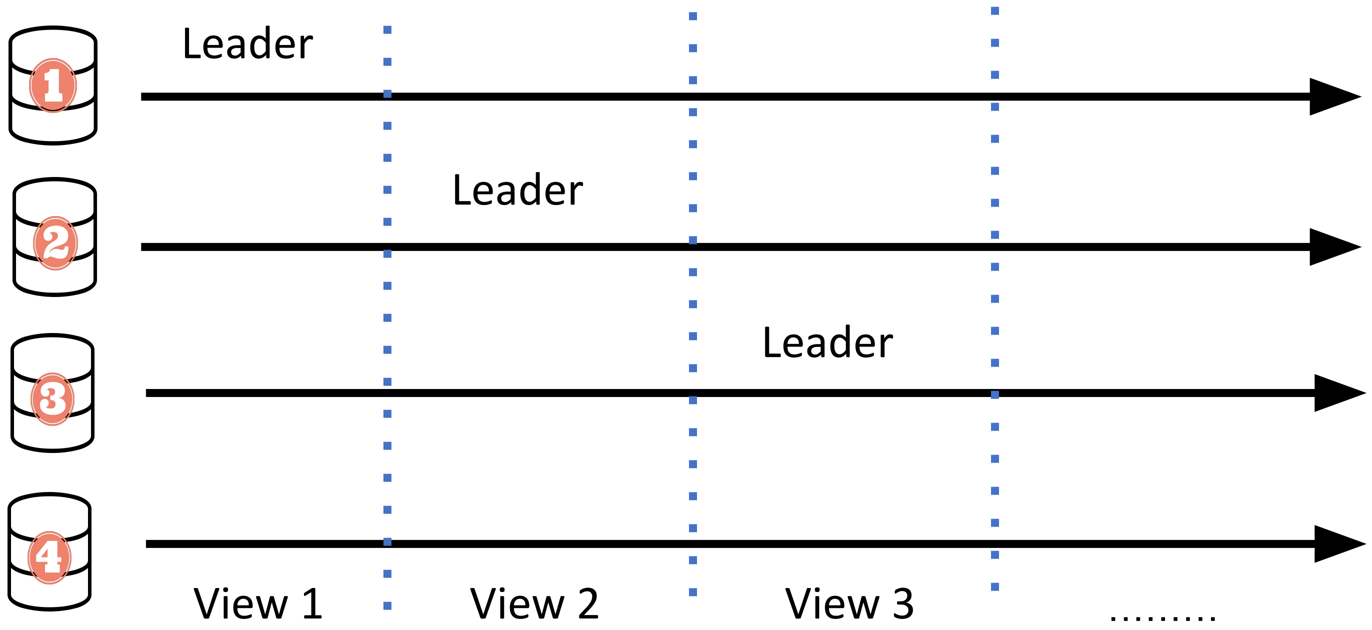
- Replicas: can vote to accept a value
- Leader: proposes a value to vote on
- Decision: 2f + 1 votes out of 3f + 1

Failed Votes



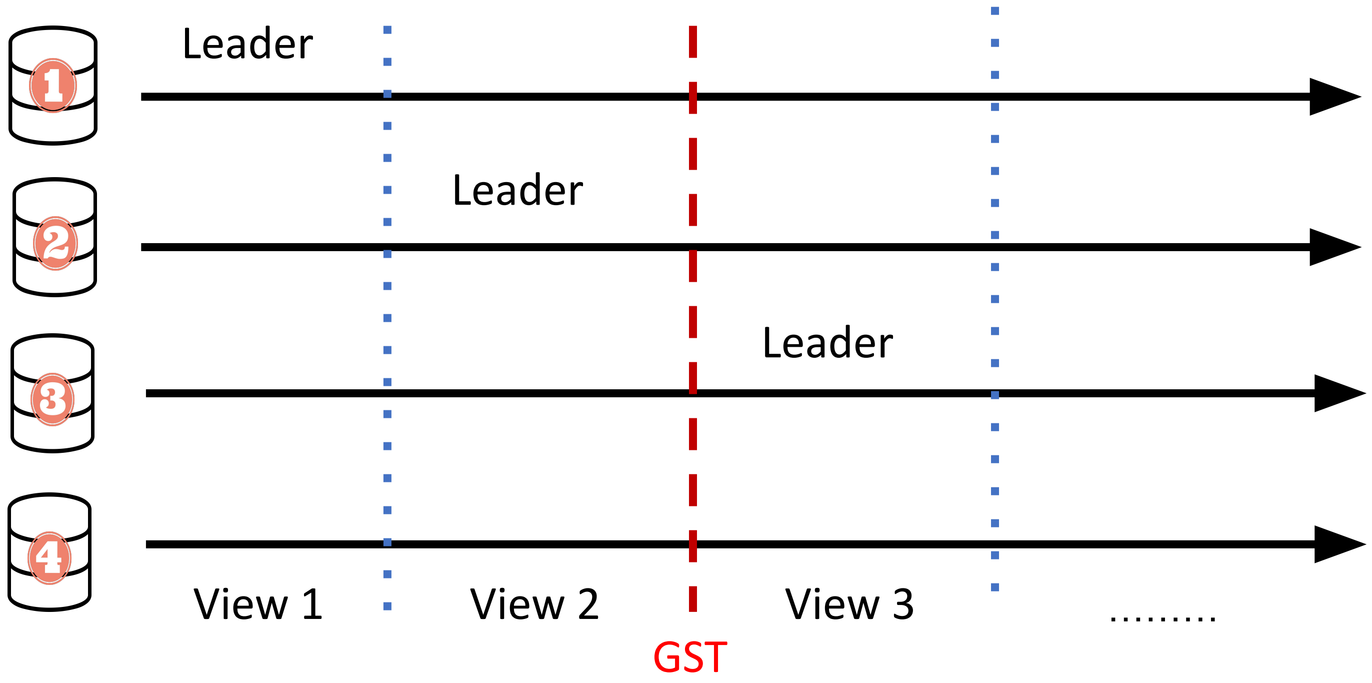
- Votes may fail: faulty leader or asynchronous network
- Can not tell the difference between the two: have to be able to change the leader

Views



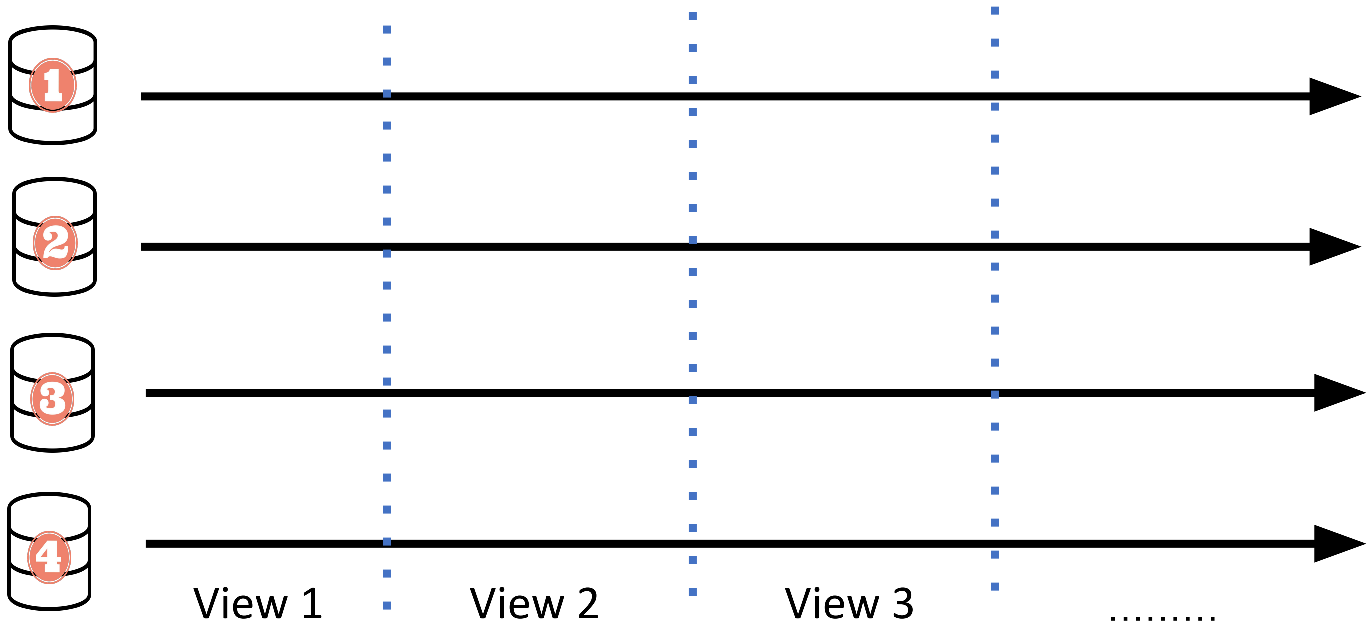
- Divide the execution into views, each with a fixed leader: view % n

Views



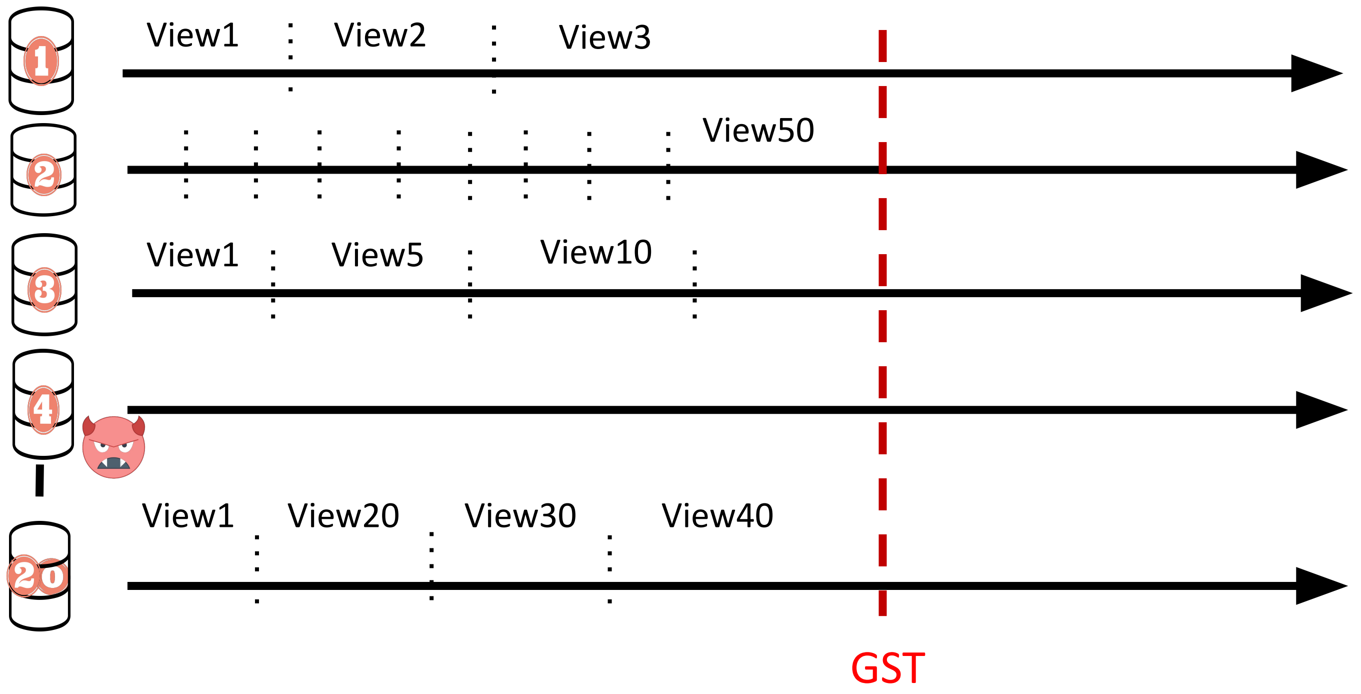
- Divide the execution into views, each with a fixed leader: view % n
- Eventually hit a correct leader and after GST messages get through, then we can terminate

View Synchronization



UCDAVIS COMPUTER SCIENCE

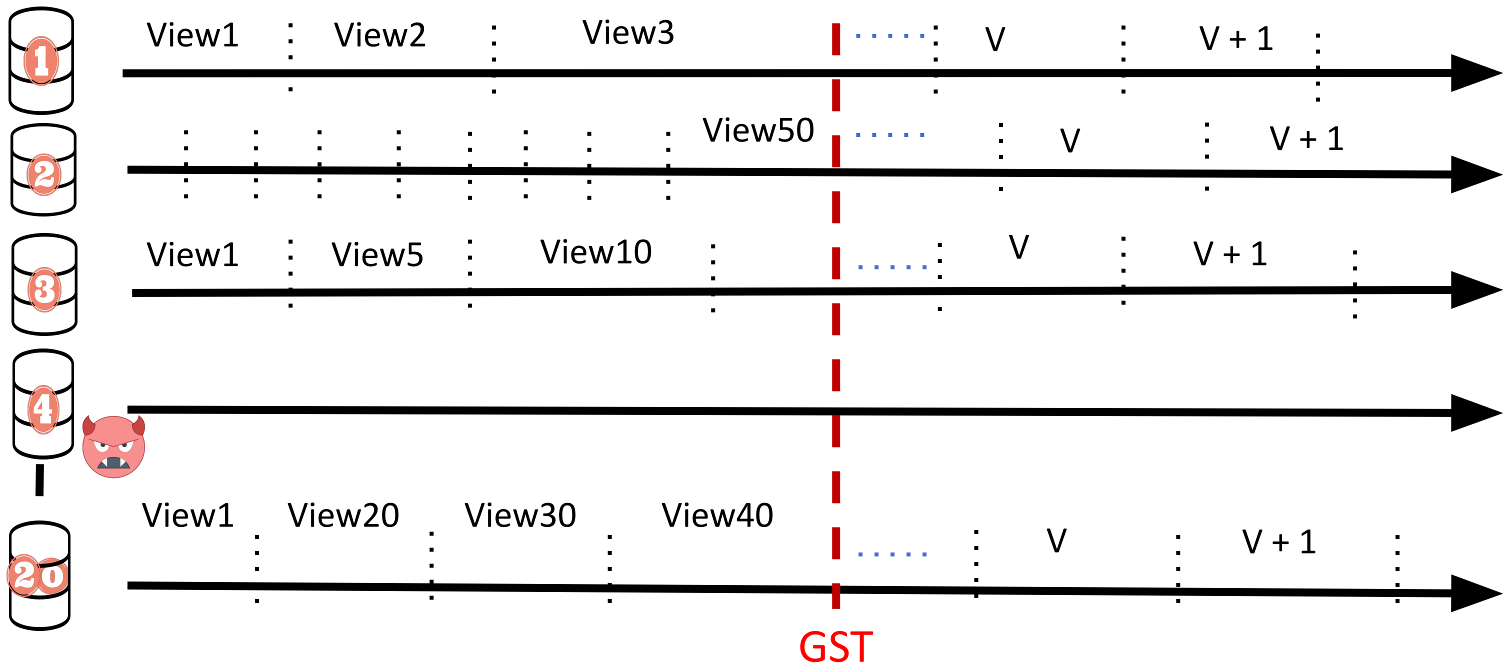
View Synchronization



- Before GST: clocks out of sync, messages delayed or lost
- After GST: bring all non-faulty replicas into the same view
- Despite malicious replicas trying to disrupt this

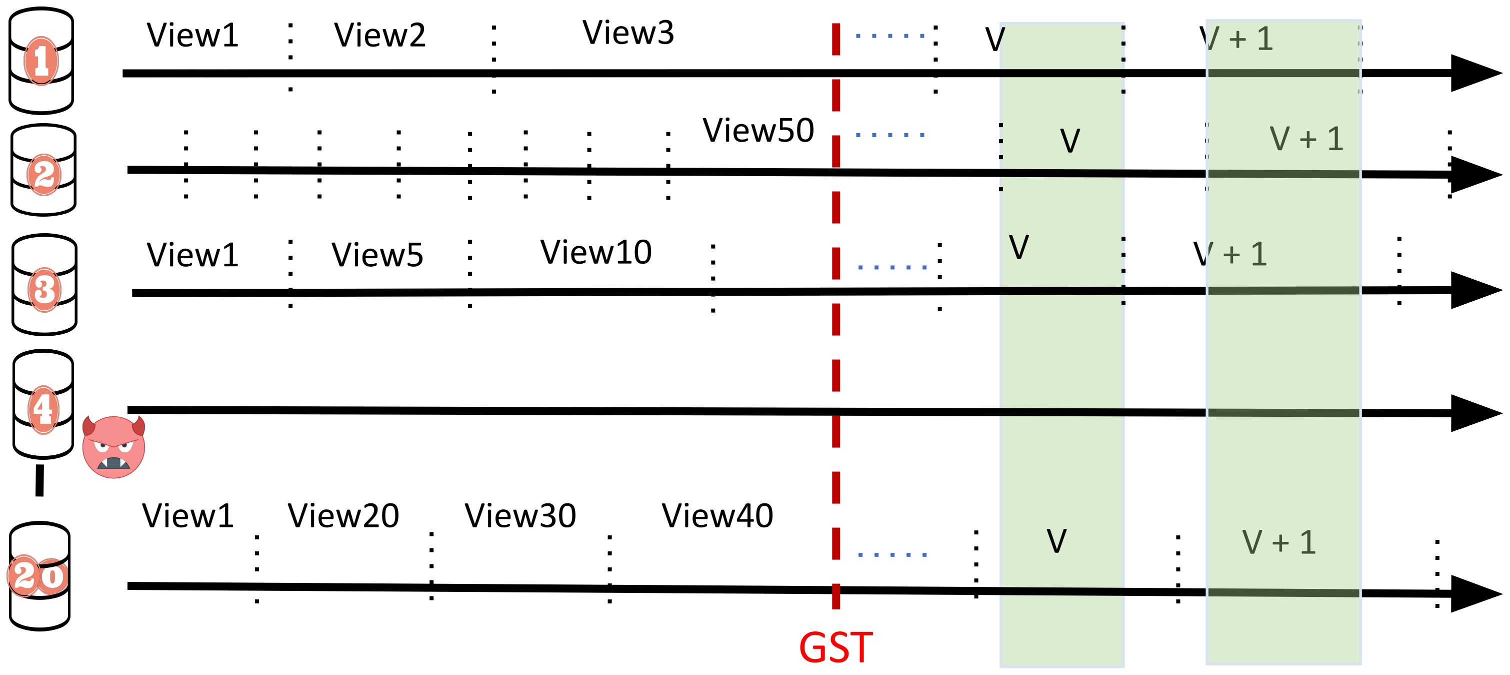
UCDAVIS COMPUTER SCIENCE

Synchronizer Specification



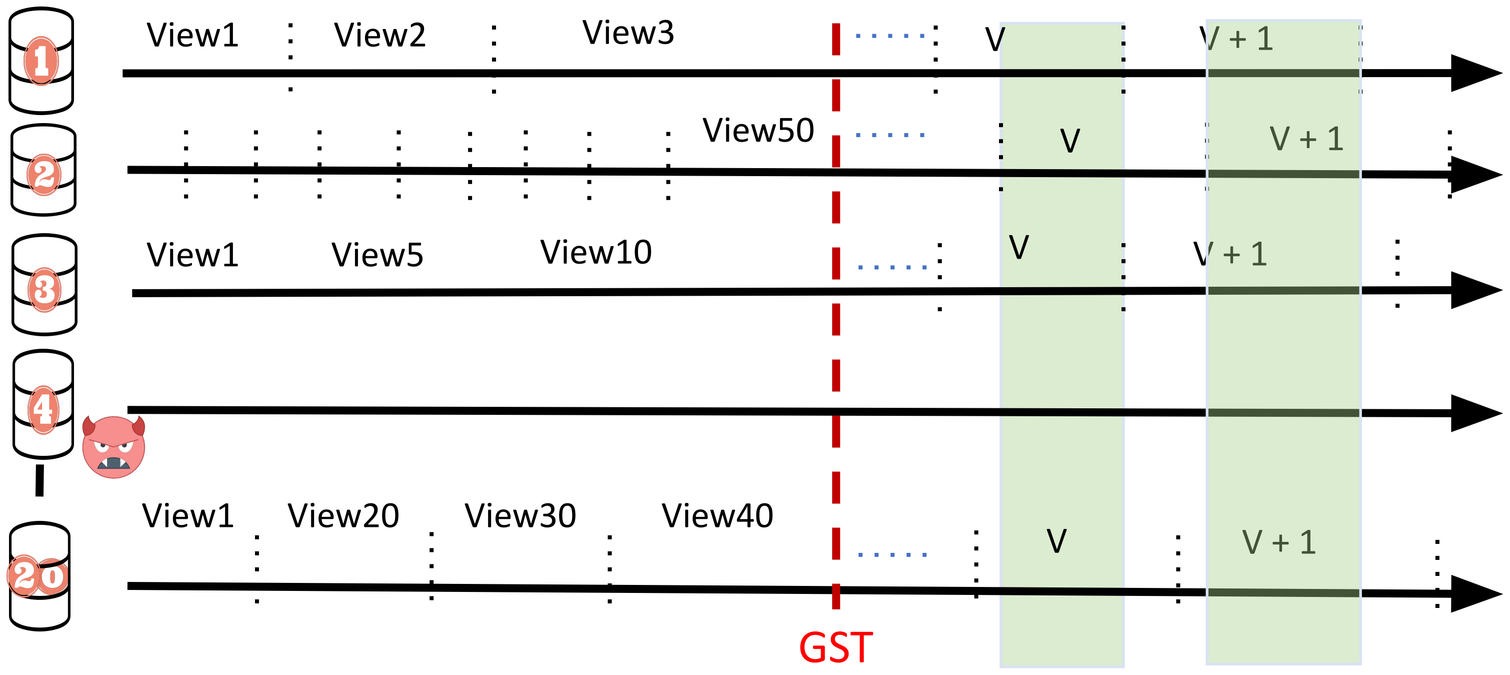
- ∃ view V: all non-faulty replicas enter every view >= V after GST and overlap in it for a non-zero time
- Leaders rotate round-robin ⇒ keep hitting correct leaders

Synchronizer Specification



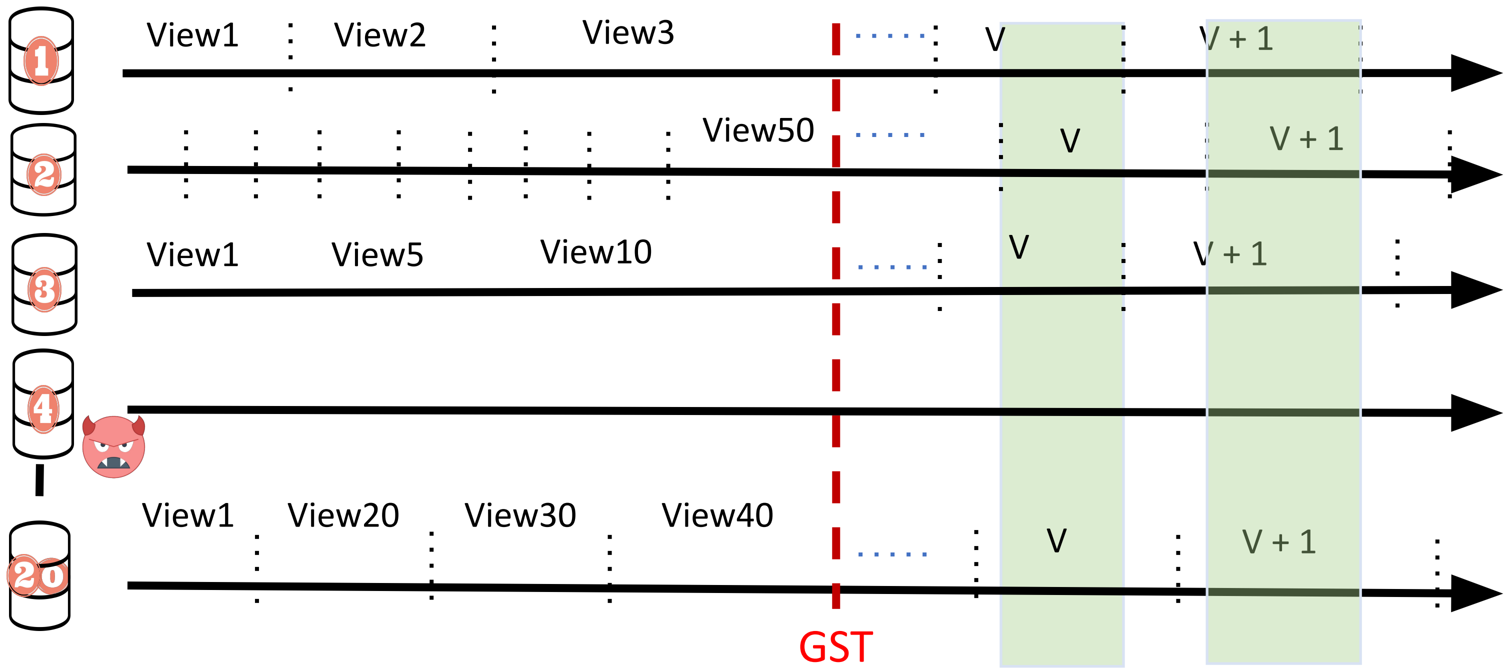
- ∃ view V: all non-faulty replicas enter every view >= V after GST and overlap in it for a non-zero time
- Leaders rotate round-robin ⇒ keep hitting correct leaders

Synchronizer Specification



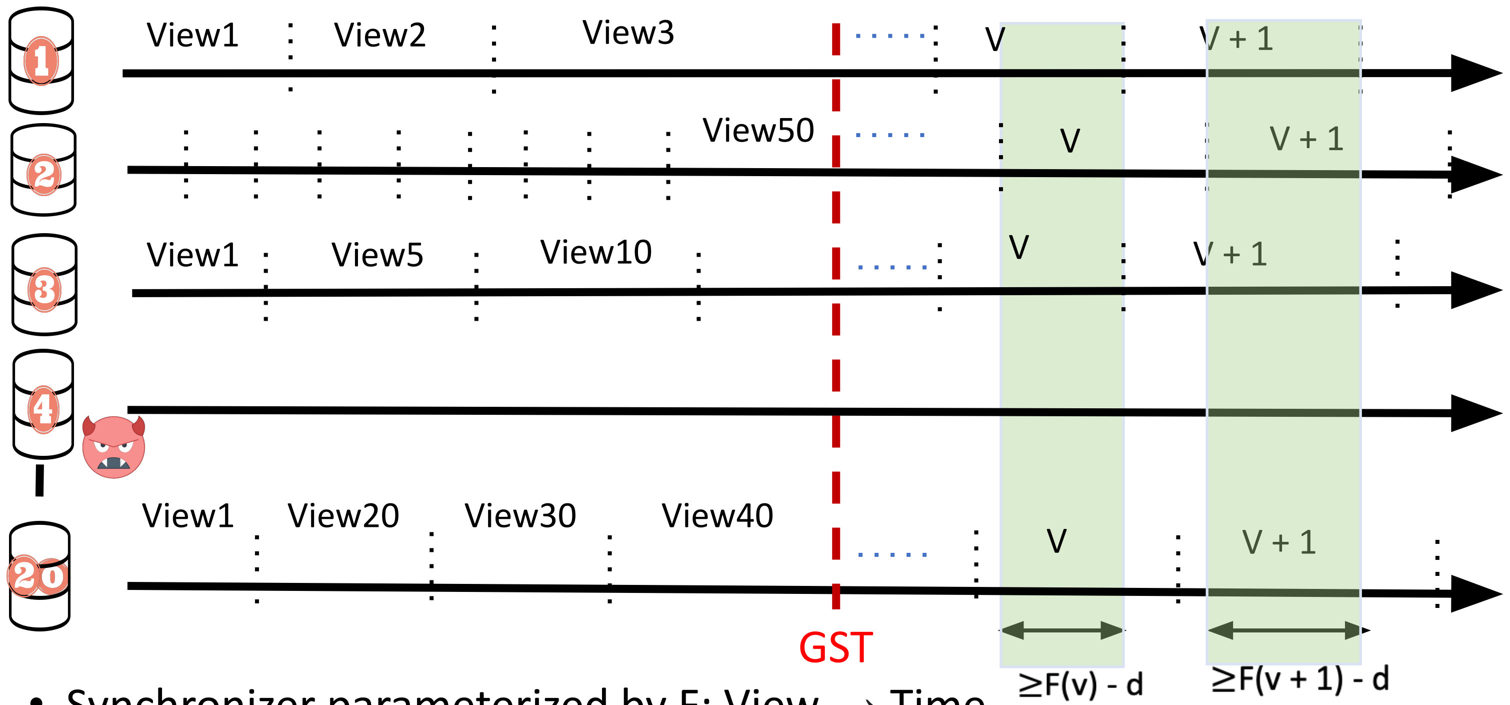
- To decide, replicas need to stay in the view long enough to complete voting
- Don’t know δ ⇒ keep increasing view duration until it’s long enough

Synchronizer Specification



- Synchronizer parameterized by F: View ⟶ Time
- Monotonic and unboundedly increasing: F(v) = c^v or F(v) = c*v

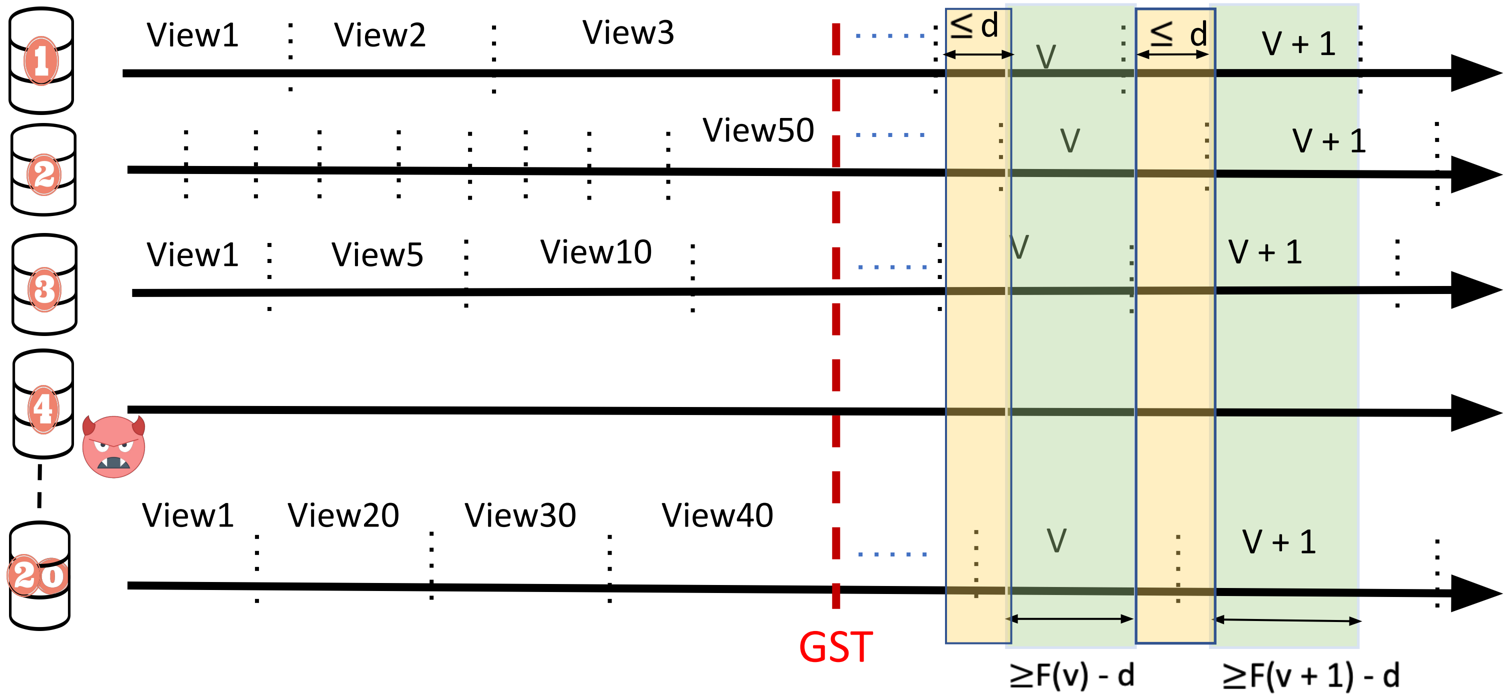
Synchronizer Specification



- Synchronizer parameterized by F: View ⟶ Time
- Monotonic and unboundedly increasing: $F(v) = c^v$ or $F(v) = c*v$
- Replicas overlap in view v for at least $F(v) - d$ (e.g., $d = 2\delta$)
- Eventually hit a long enough view with a correct leader ⇒ decide

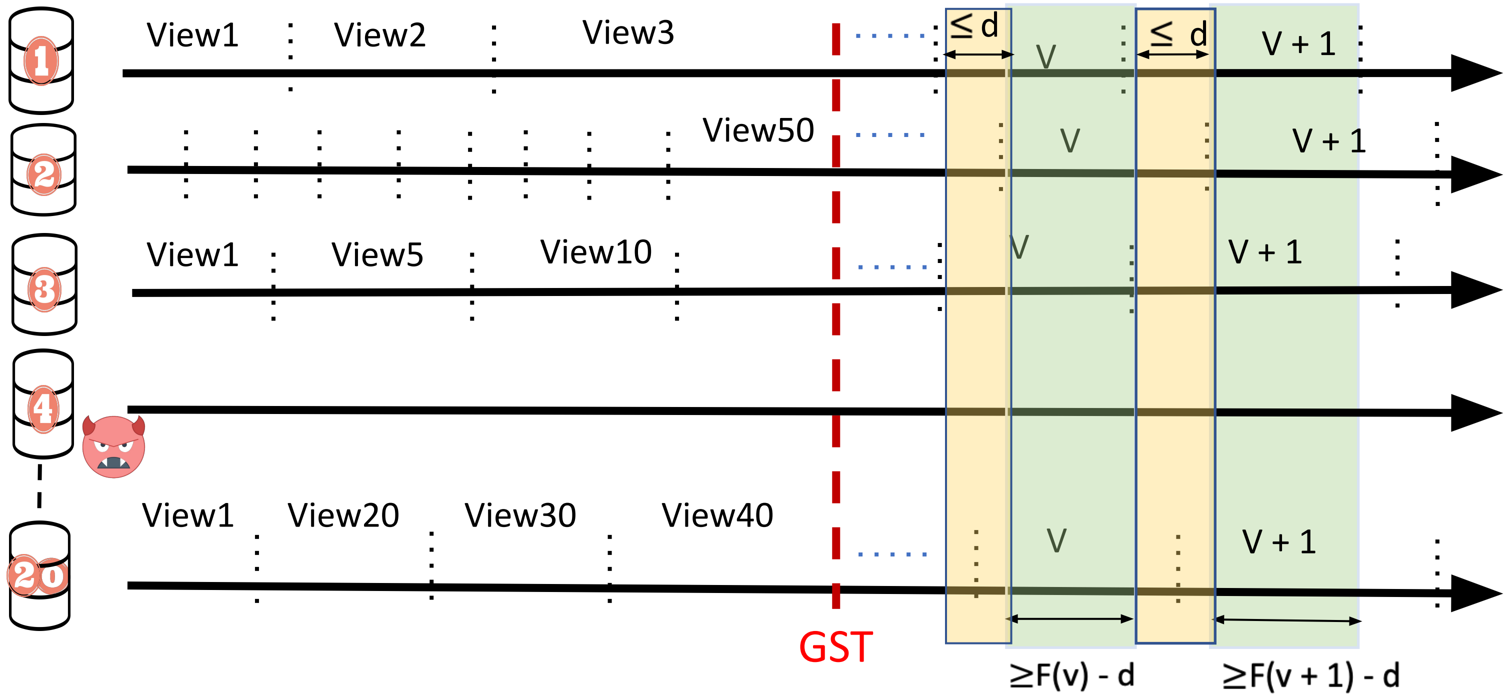
UCDAVIS COMPUTER SCIENCE

Synchronizer Specification



- Non-faulty replicas enter each view within d (e.g., d = 2δ)

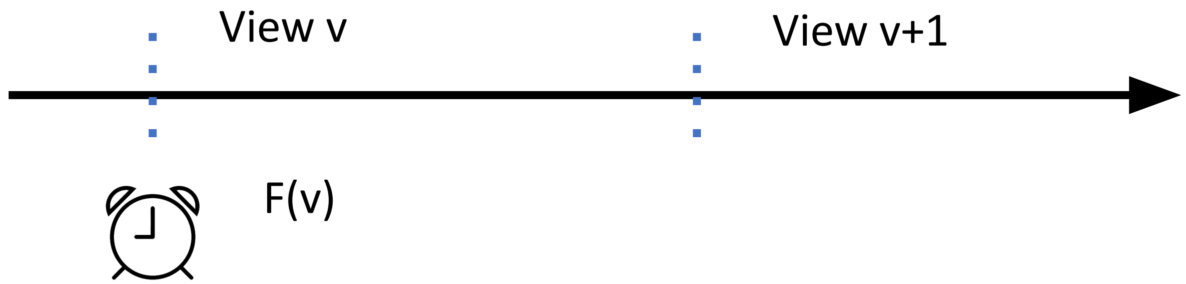
View Specification



- Proving liveness of several protocols: Hotstuff, Tendermint, PBFT

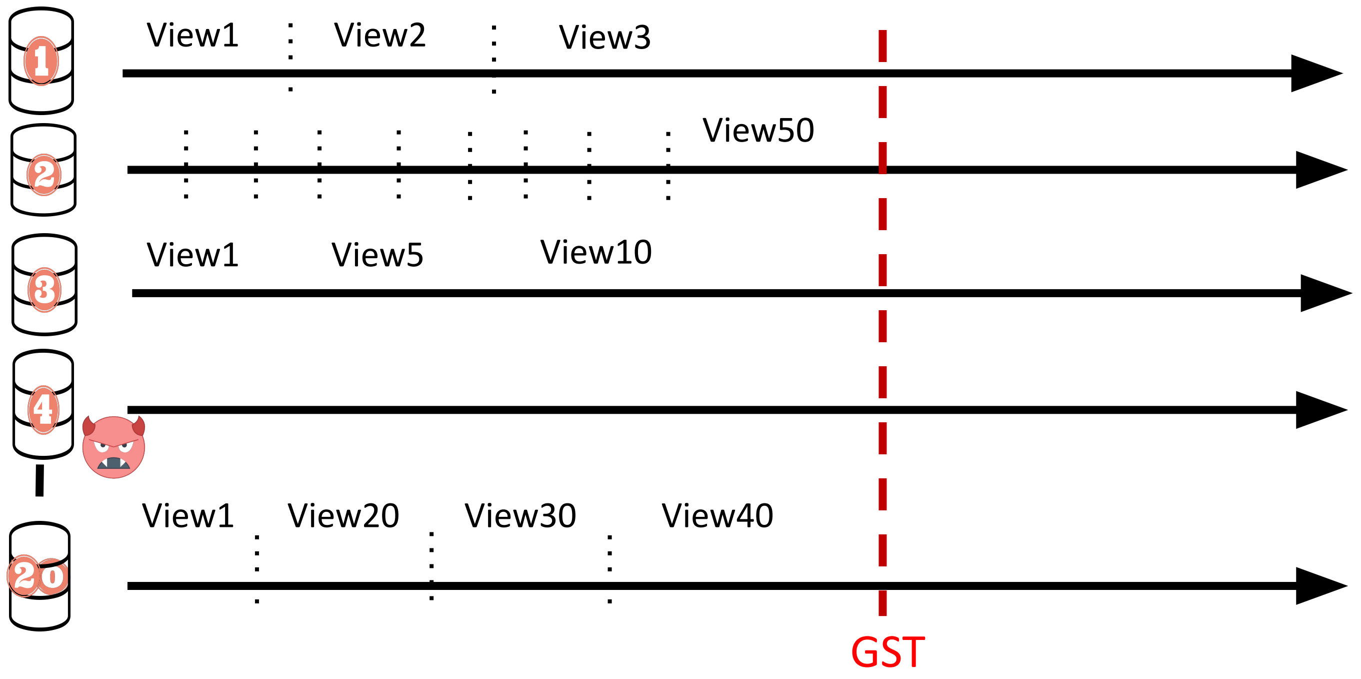
UCDAVIS COMPUTER SCIENCE

FastSync Synchronizer



- Upon entering a view, set a timer for F(v)
- Try to switch views when the timer expires

FastSync Synchronizer



- At GST replicas may be in different views
- Timers are not enough: replicas will have to communicate

Switching Views

- When a timer for view v expires, try to enter v+1

  ⇒ broadcast WISH(v+1)

- Enter view v+1 when enough replicas express a similar wish

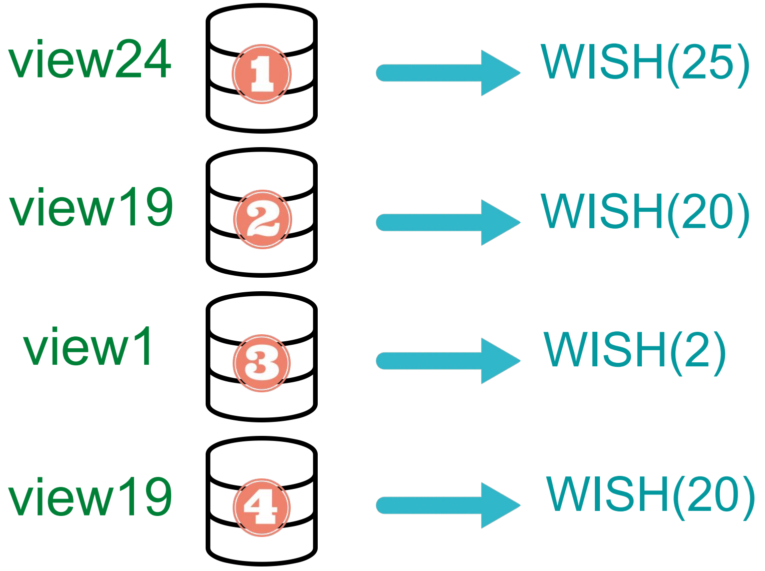
  E.g., WISH(v+1)

- Enough replicas = 2f+1 out of 3f+1 replicas

  ⇒ guard against disruption by faulty replicas

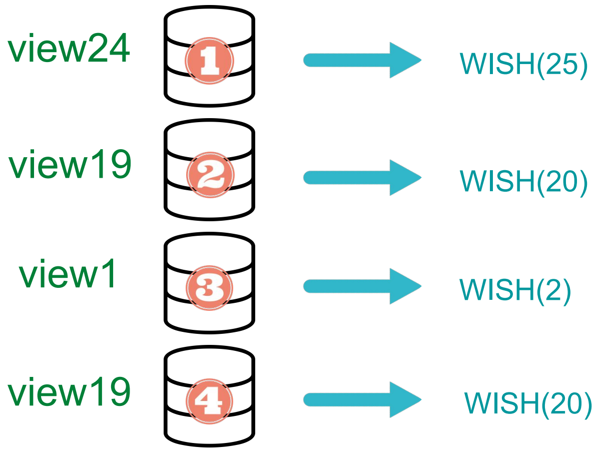
UCDAVIS
COMPUTER SCIENCE

Switching Views



UCDAVIS COMPUTER SCIENCE

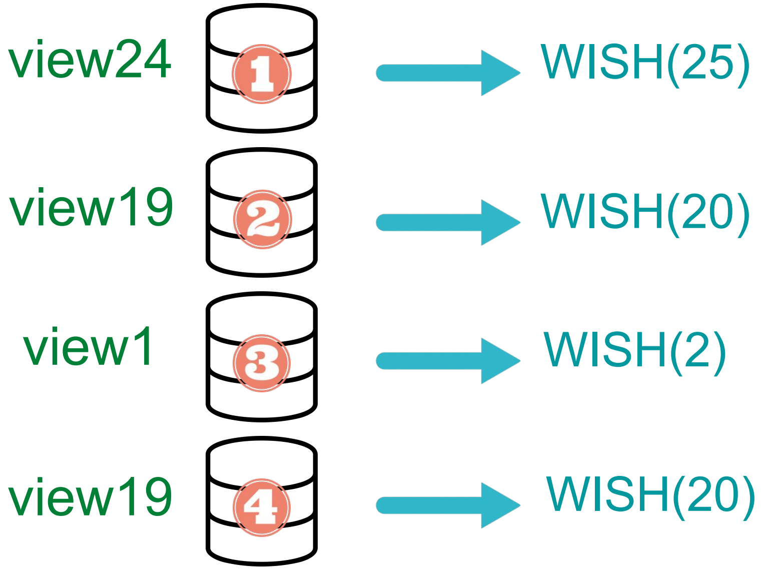
Switching Views



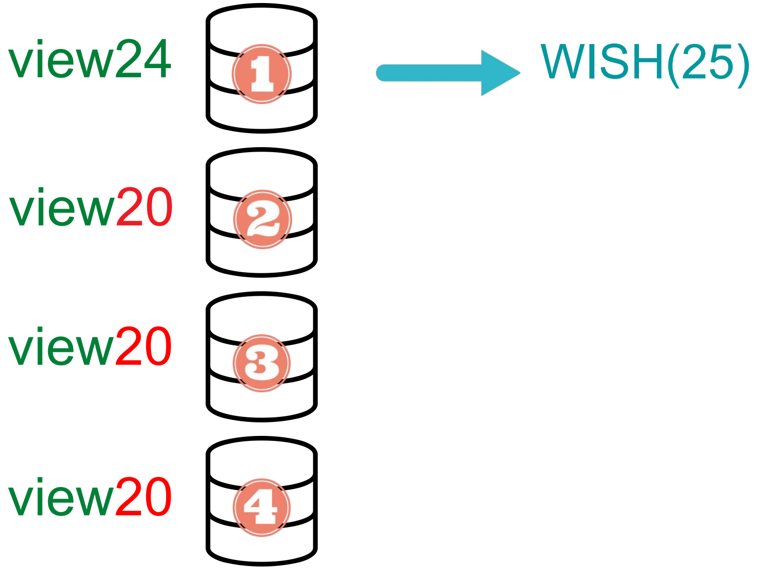
25	20	2	20

- Bounded space: maintain an array with the highest WISH received from each replicas
- When received 2f+1 WISHes for views > your (view), enter directly to the minimal WISH view in them

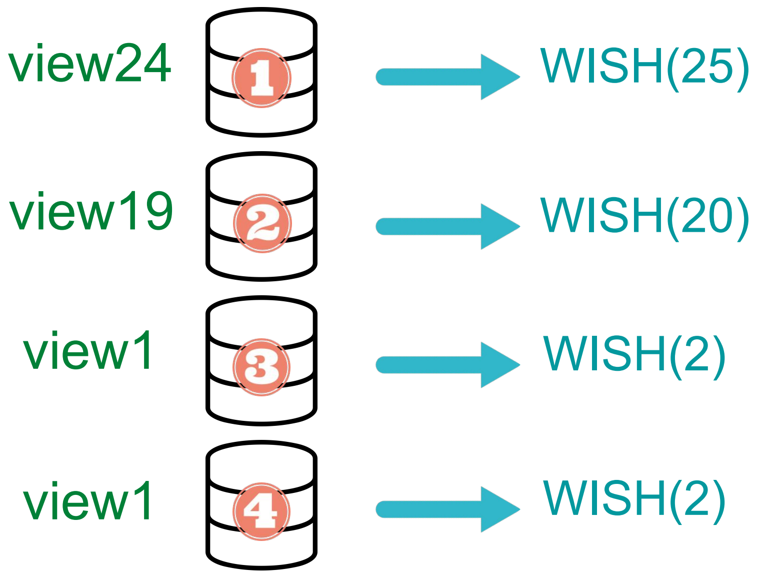
Switching Views



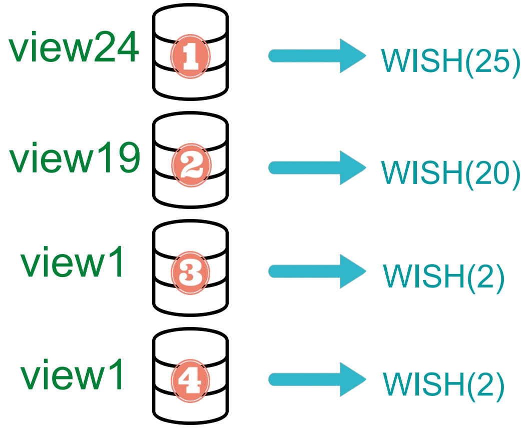
Switching Views



Relaying WISHes



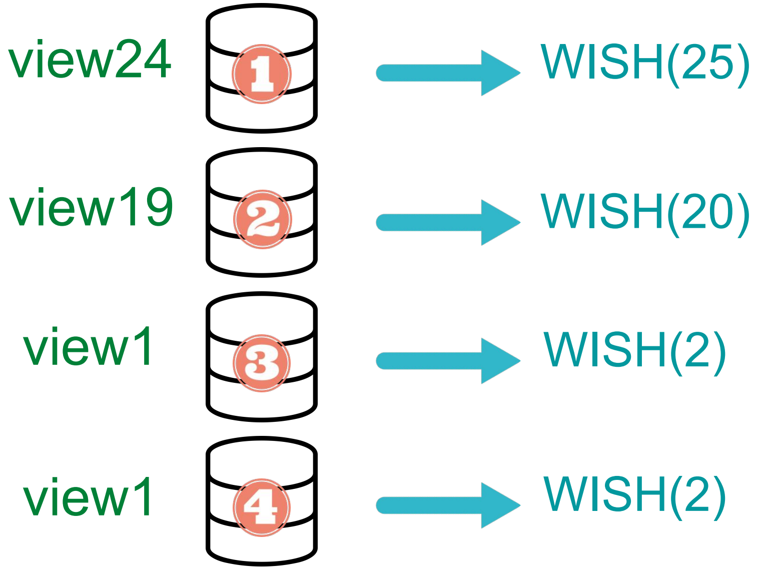
Relaying WISHes



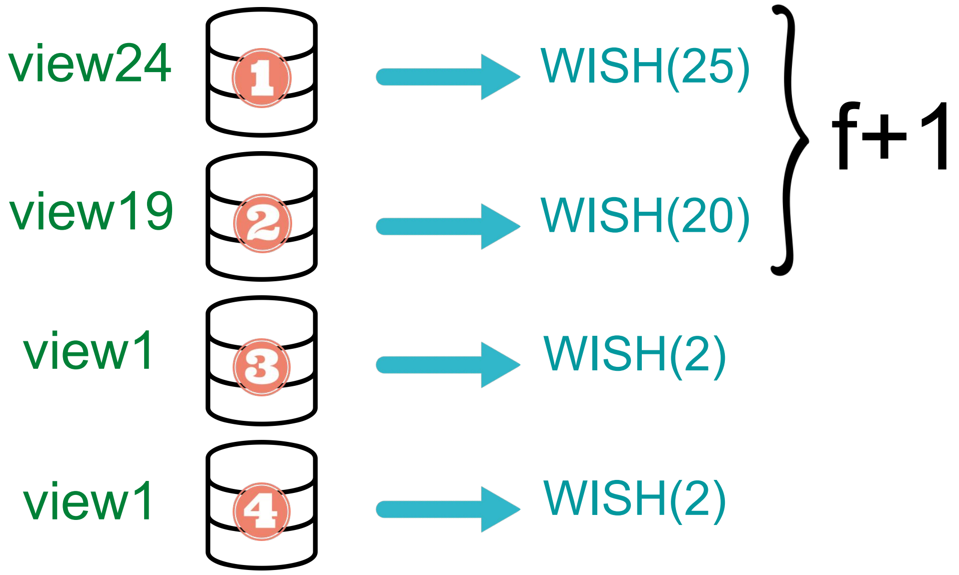
25	20	2	2

- f+1 WISHes > the view you're trying to enter (WISH)
  ⇒trying to enter the minimal view in them and relay the corresponding WISH

Relaying WISHes

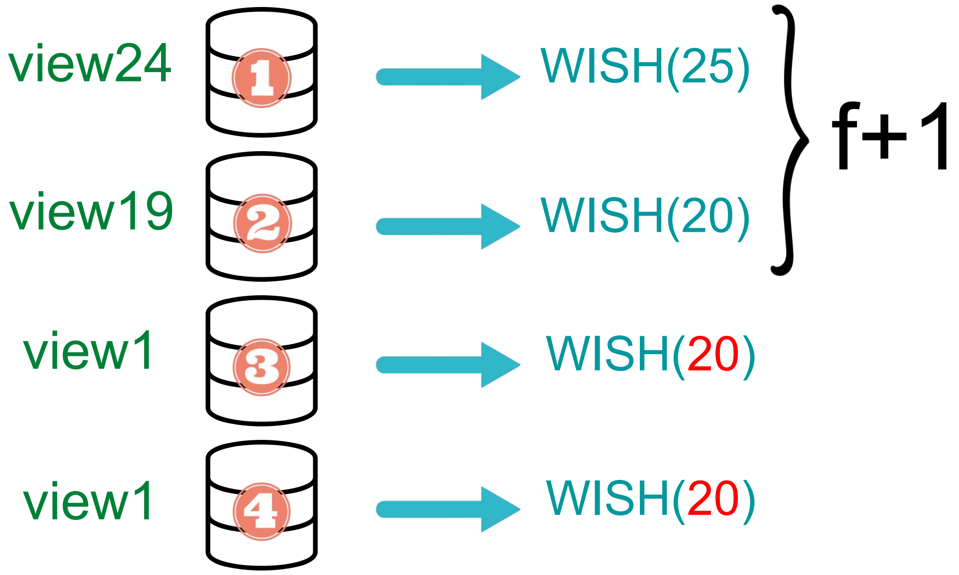


Relaying WISHes

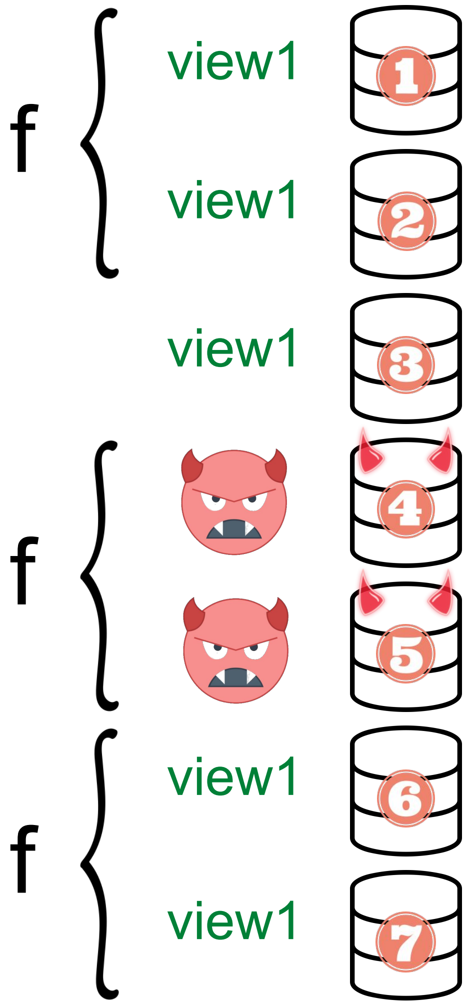


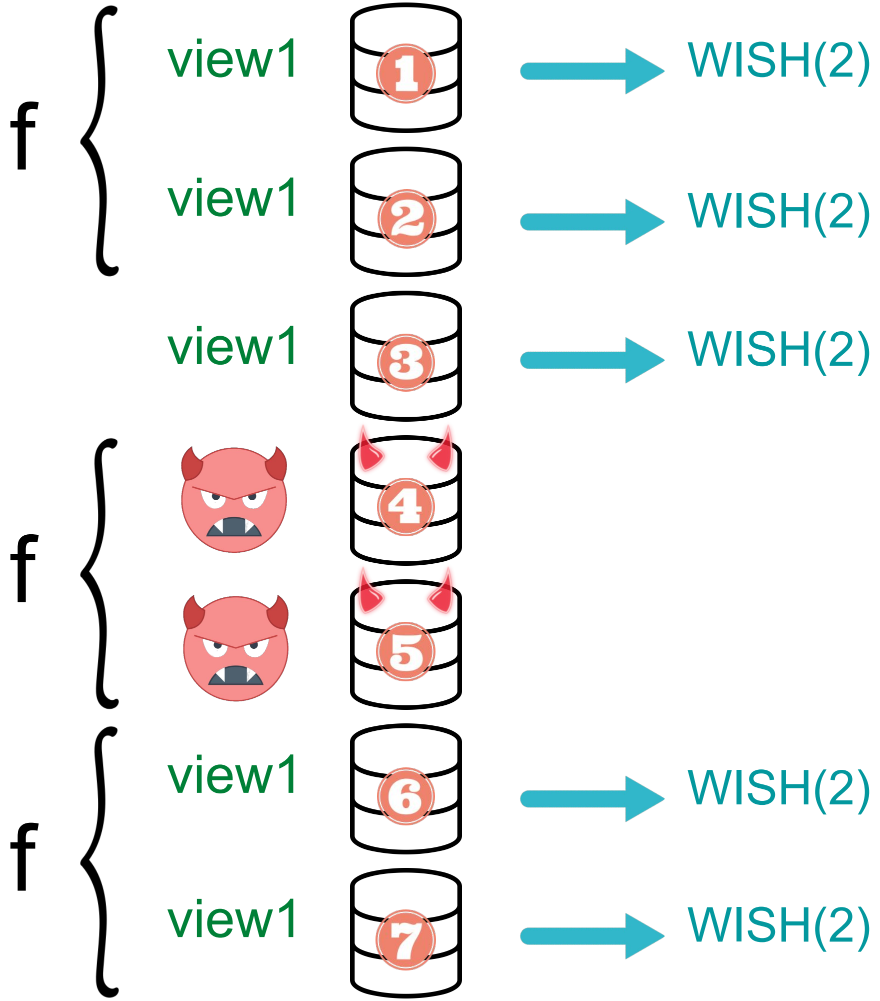
UCDAVIS COMPUTER SCIENCE

Relaying WISHes

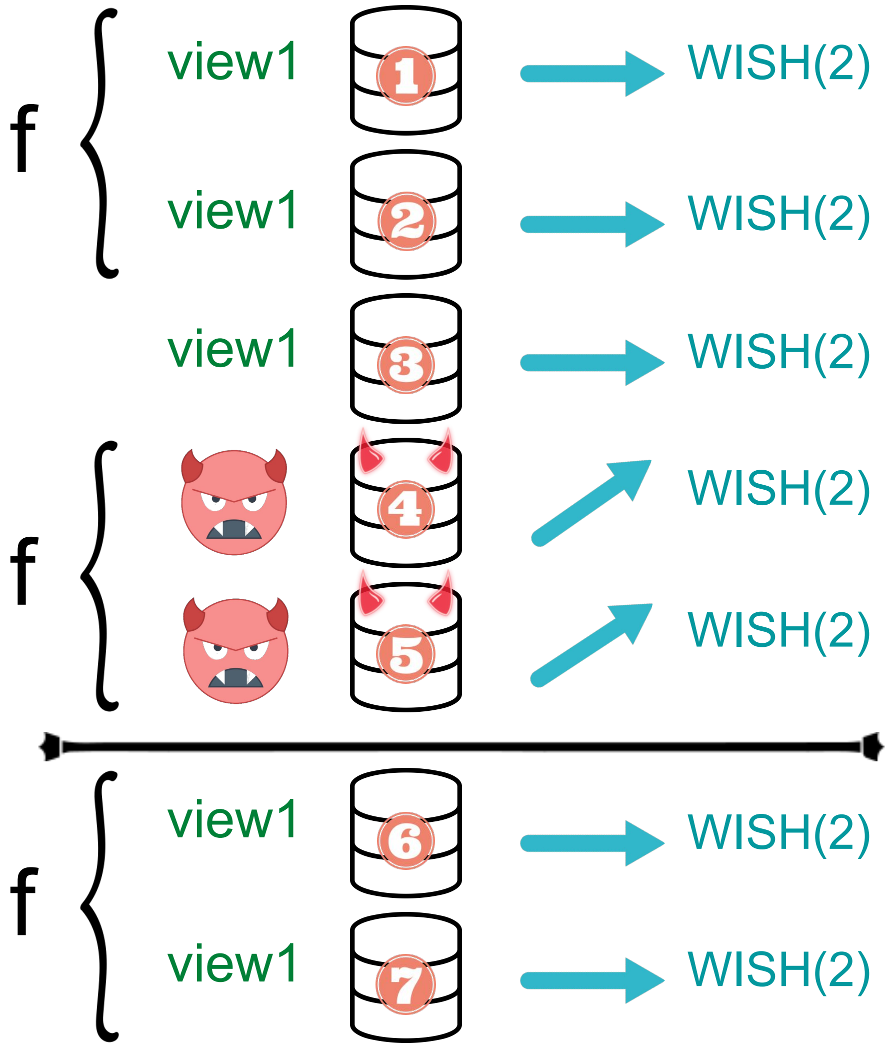


UCDAVIS COMPUTER SCIENCE


f
view1
1
view1
2
view1
3
f
4
5
f
view1
6
view1
7

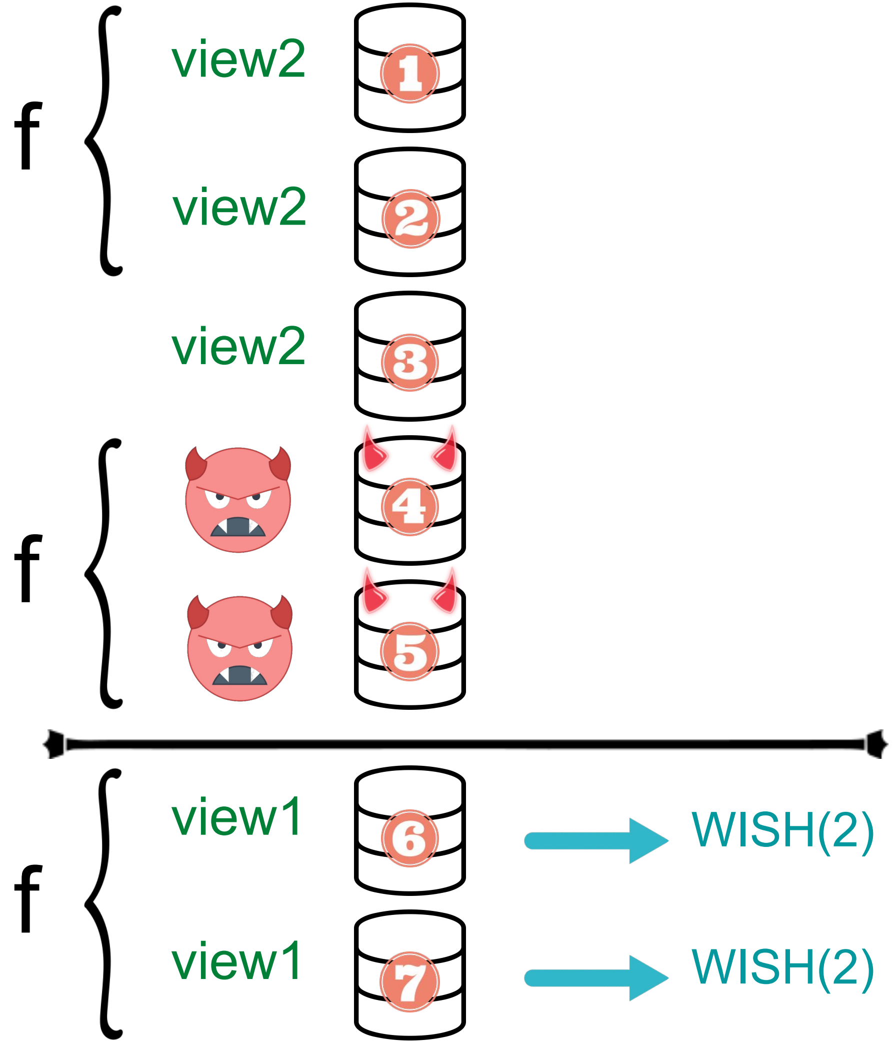


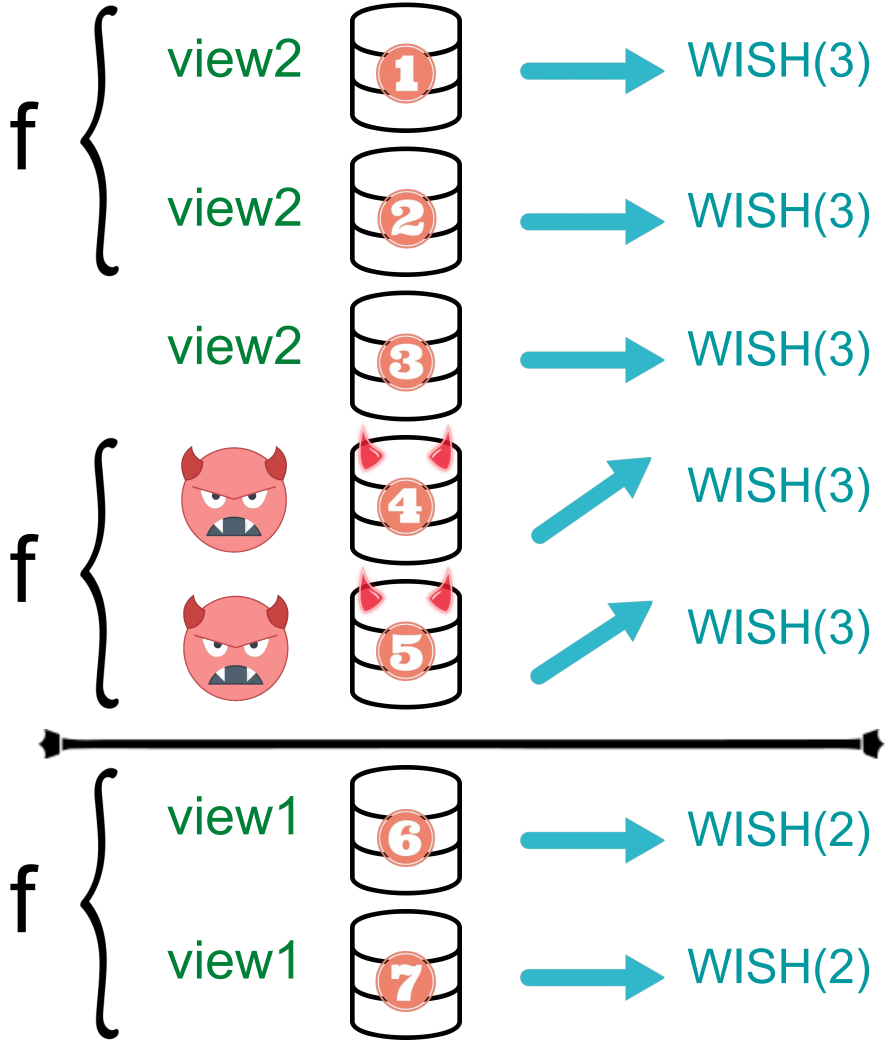
UCDAVIS COMPUTER SCIENCE

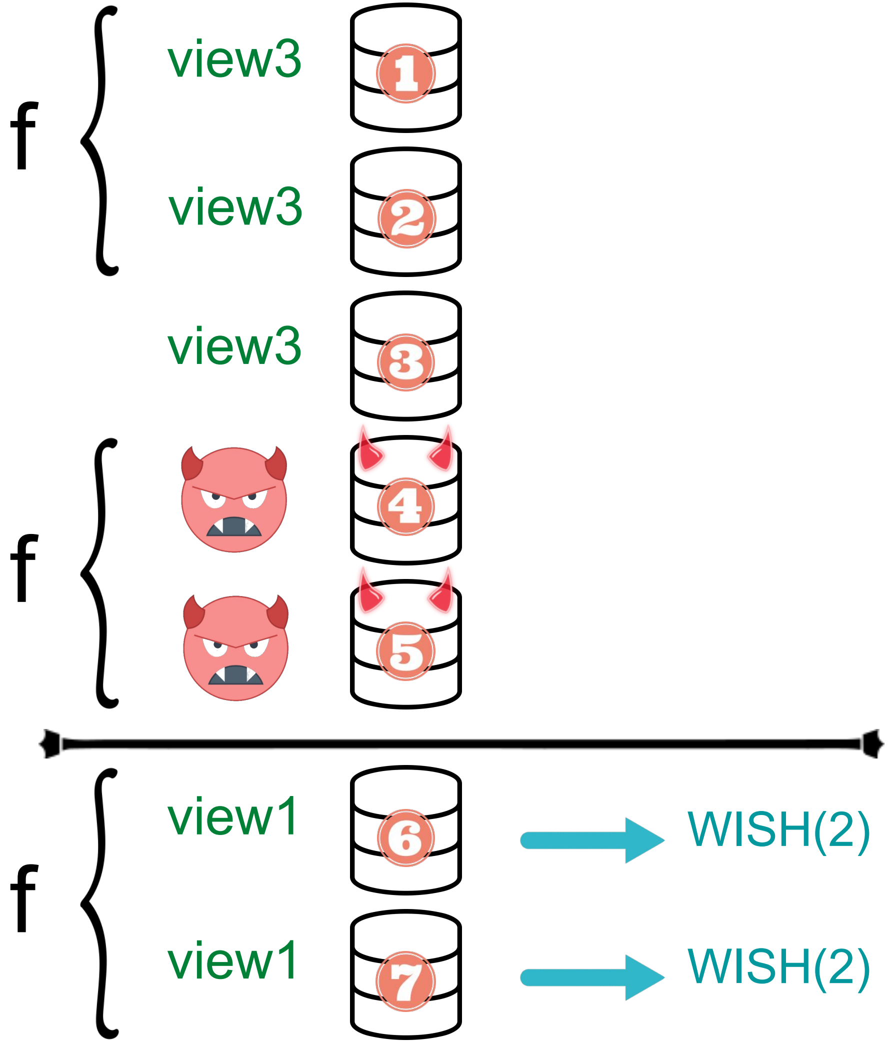


Faulty replicas send WISHes to 1~3, but not 6~7

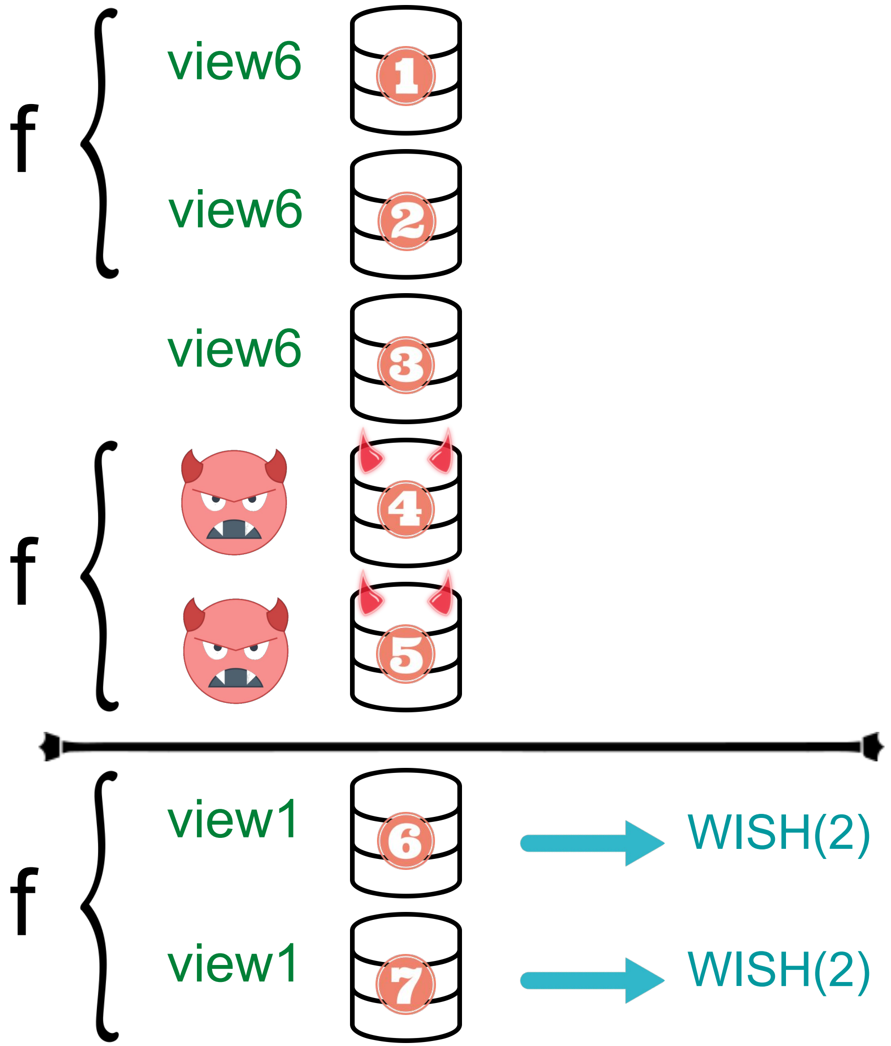
Replicas 1~3 are partitioned from 6~7

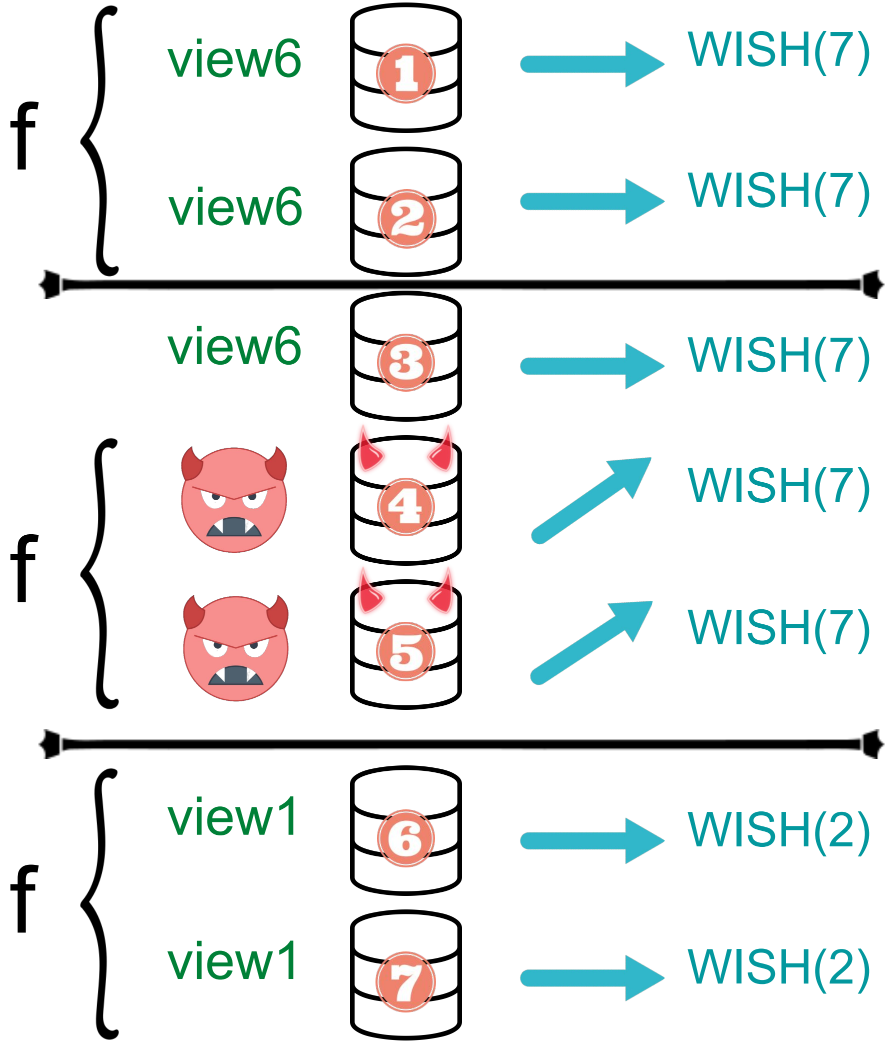

f
view2
1
view2
2
view2
3
f
4
5
f
view1
6
WISH(2)
view1
7
WISH(2)


f
view2
1
WISH(3)
view2
2
WISH(3)
view2
3
WISH(3)
f
4
WISH(3)
5
WISH(3)
f
view1
6
WISH(2)
view1
7
WISH(2)

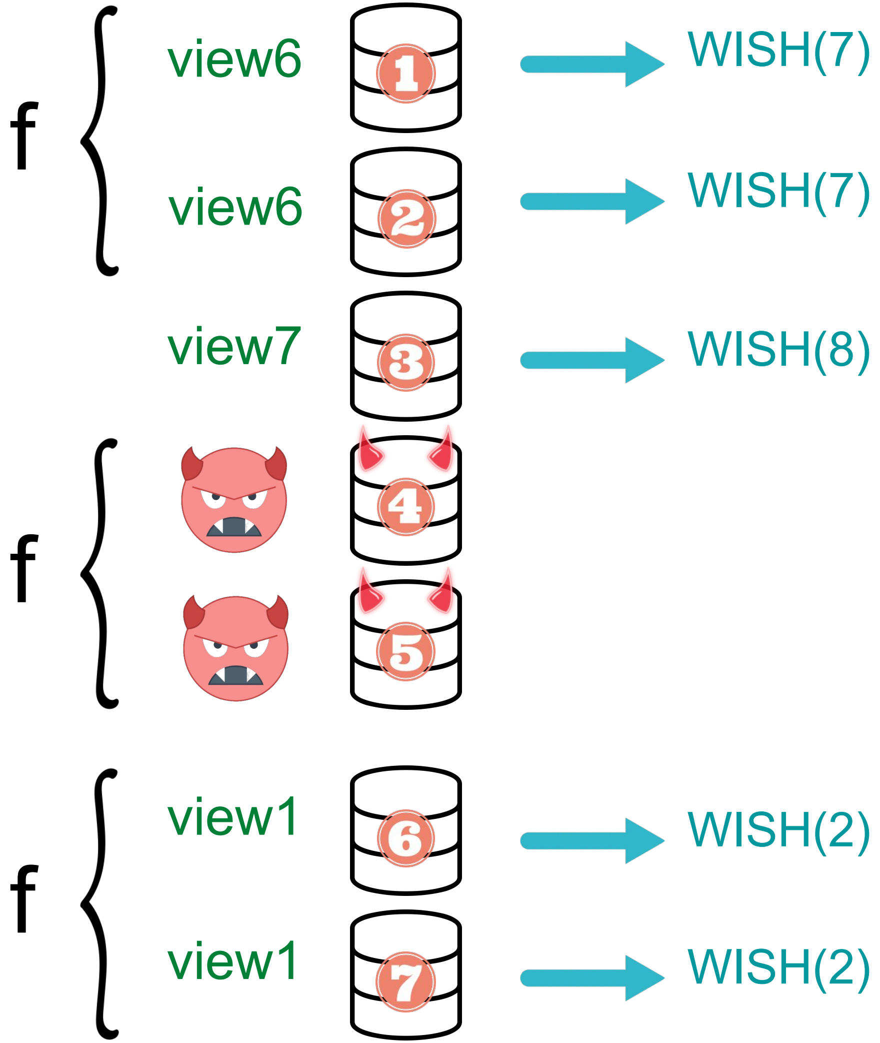


UCDAVIS COMPUTER SCIENCE


f
view6
1
view6
2
view6
3
f
4
5
f
view1
6
WISH(2)
view1
7
WISH(2)

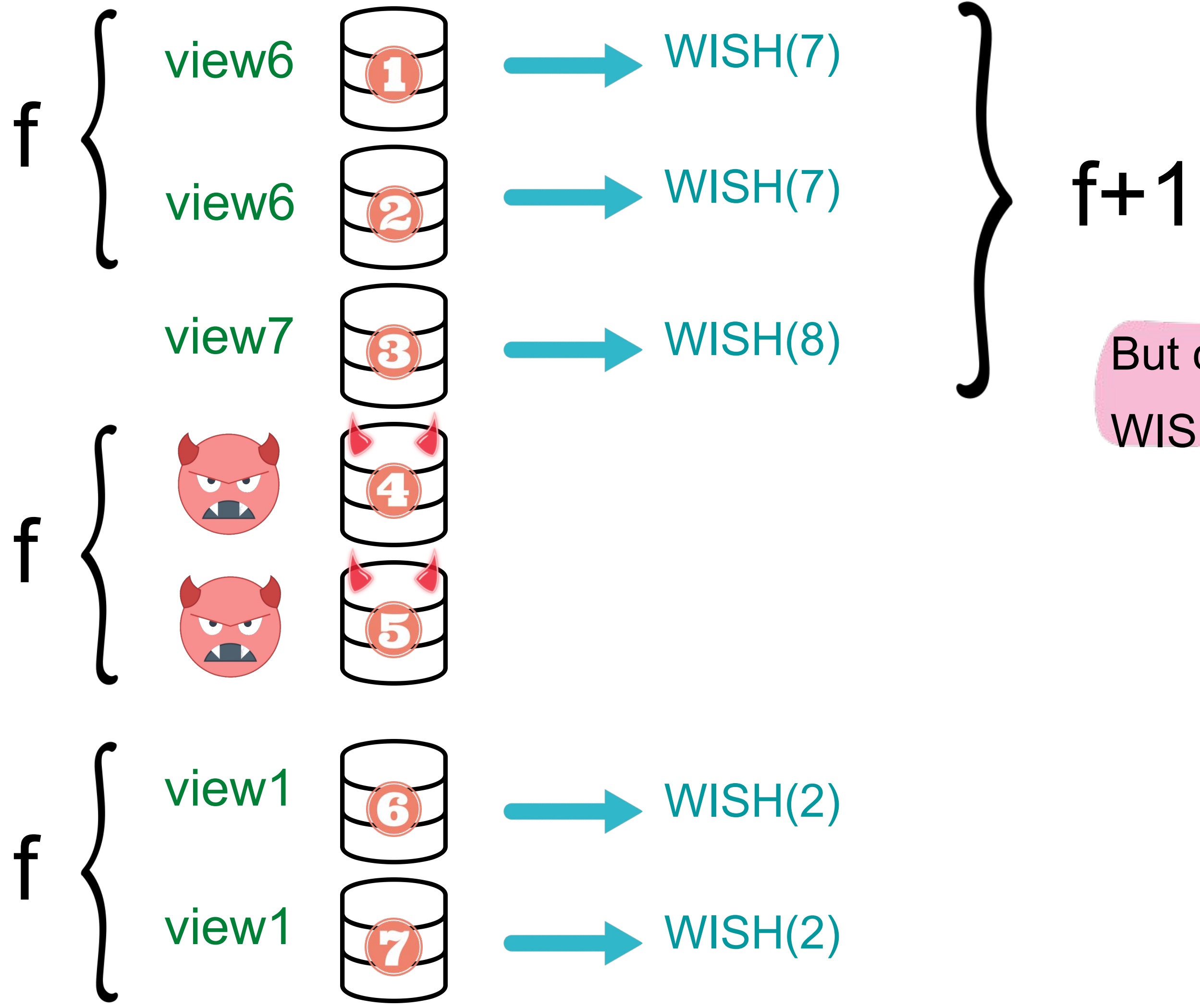


Faulty replicas send WISHes to replica 3 only



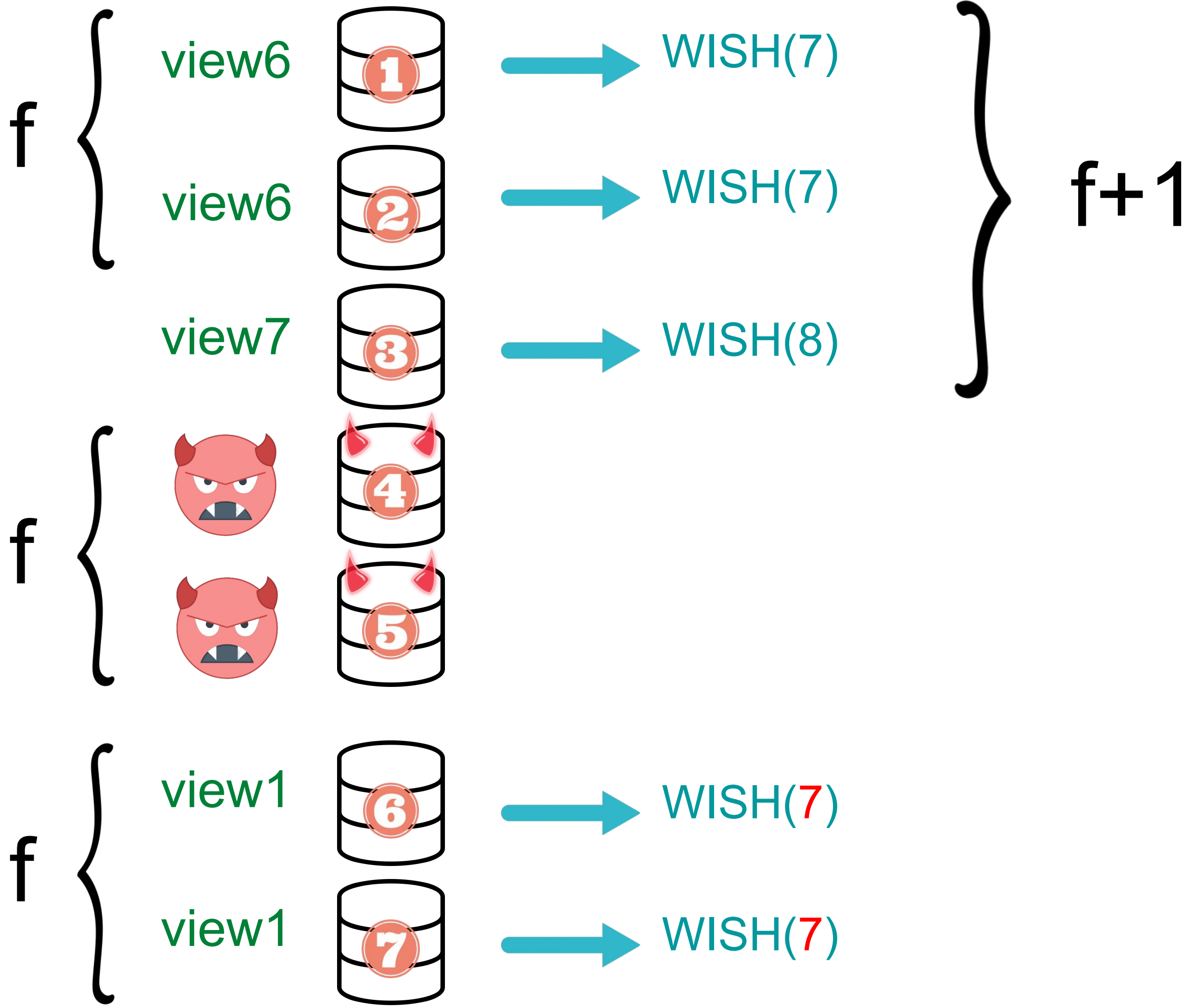
GST: WISHes get through

No matching f+1 WISHes
⇒ Bracha-like syncroniser
would get stuck

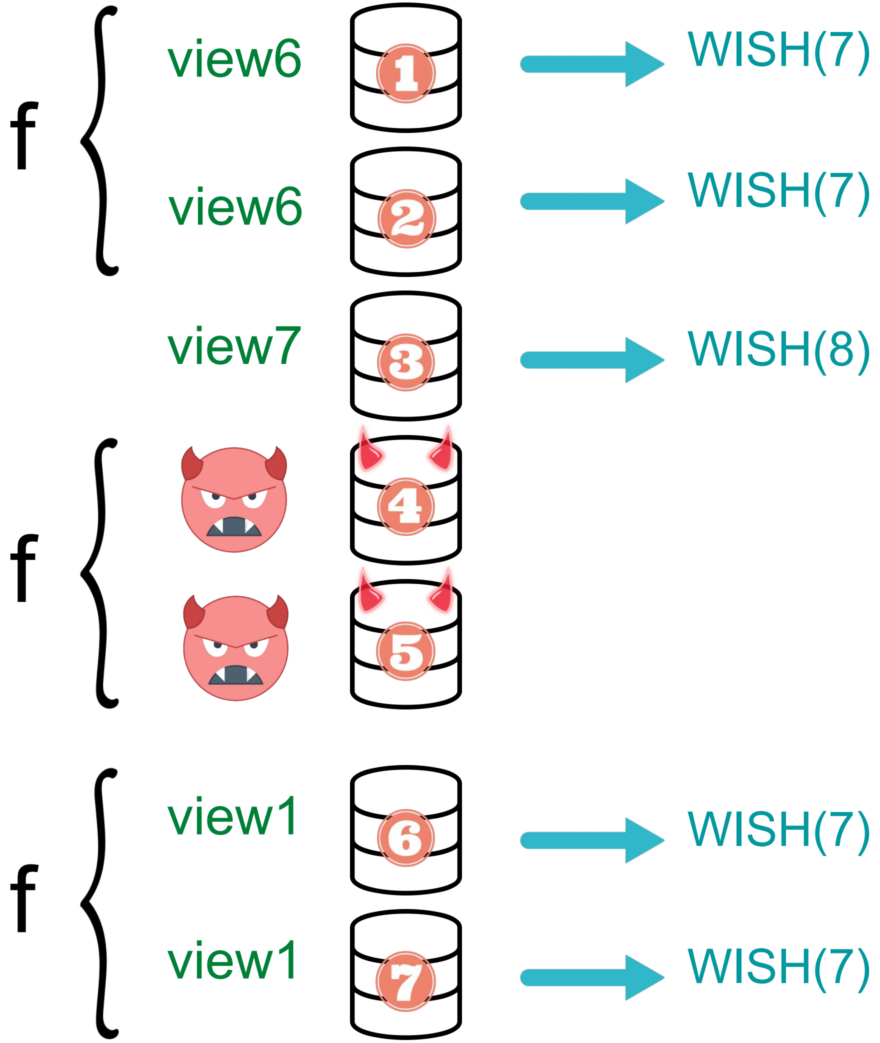


But can relay the minimal WISH of f+1

UCDAVIS COMPUTER SCIENCE

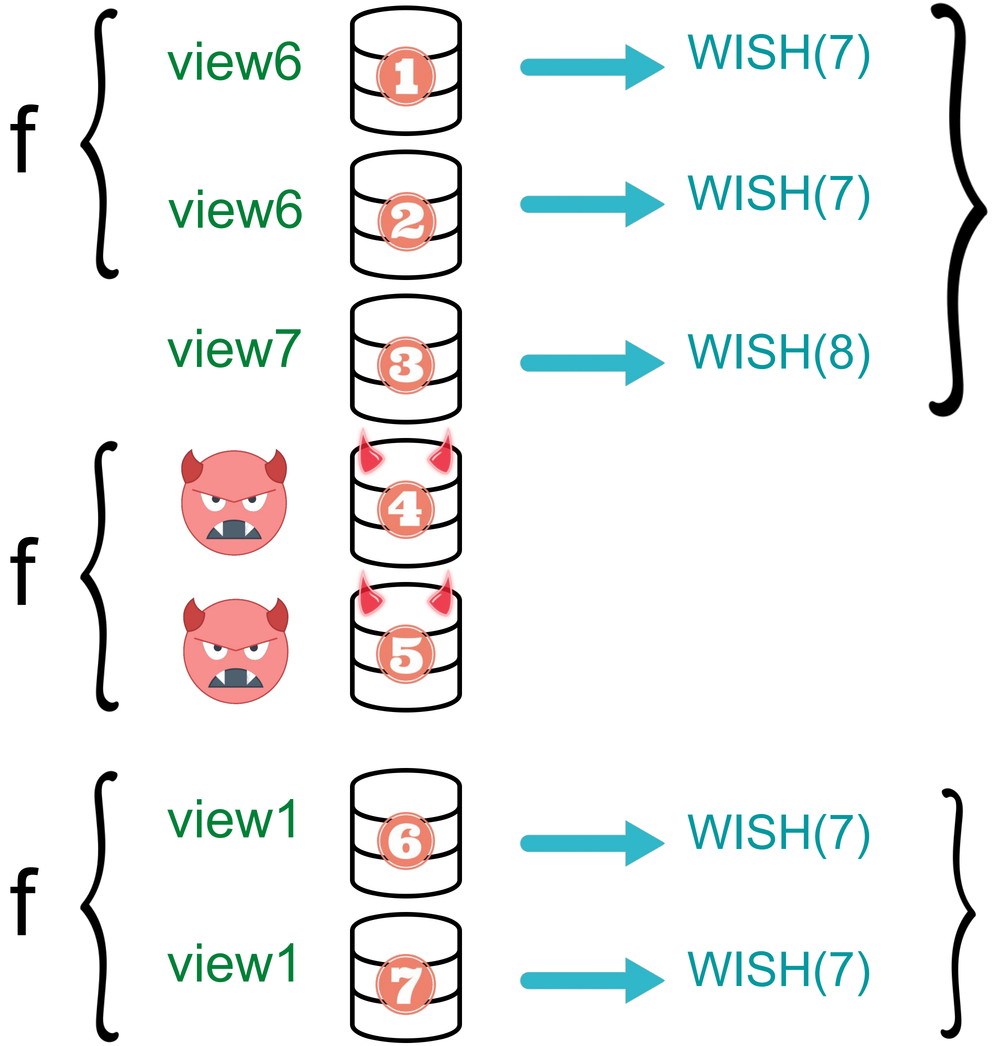


UCDAVIS COMPUTER SCIENCE

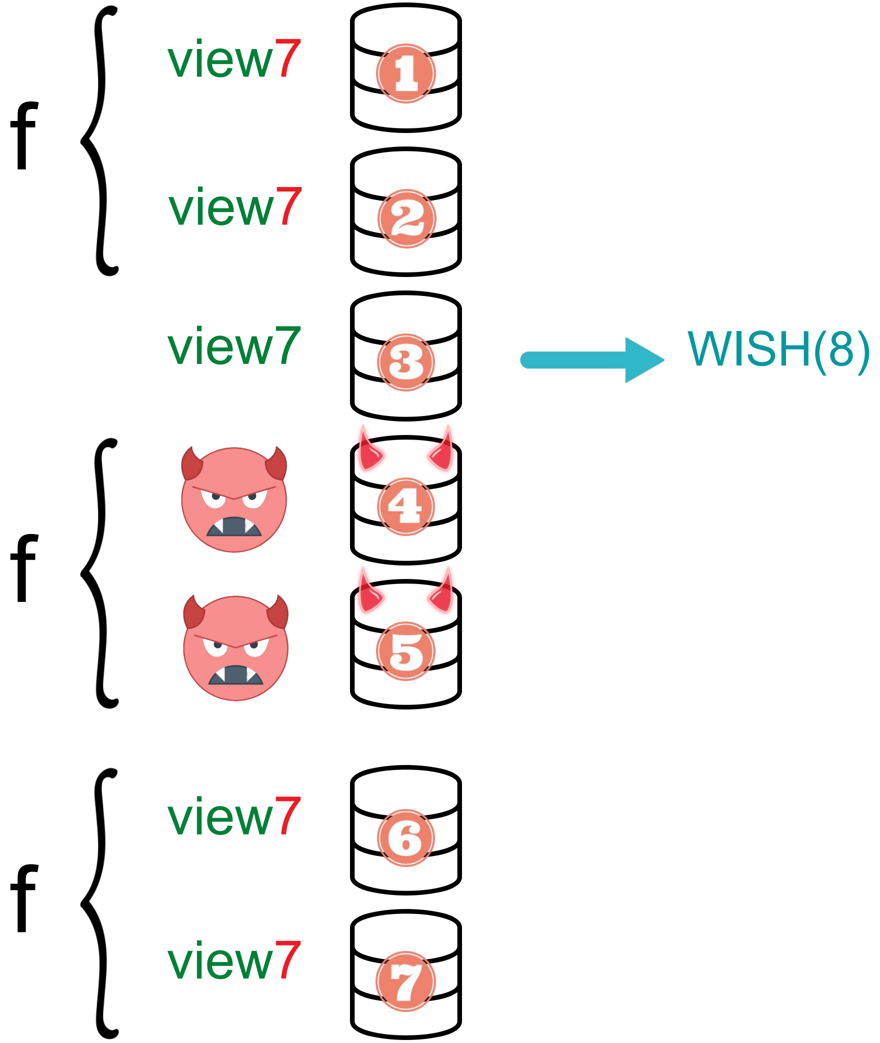


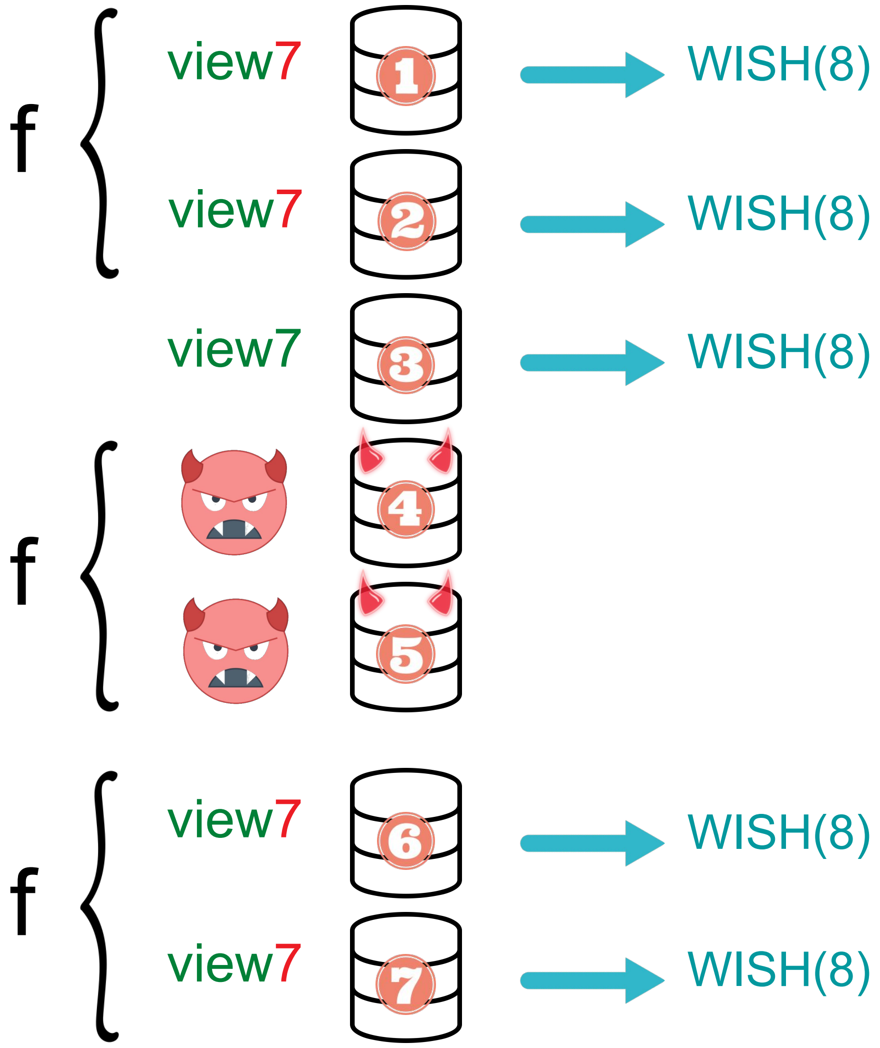
No matching 2f+1 WISHes

⇒ Bracha-like synchronizer would get stuck again

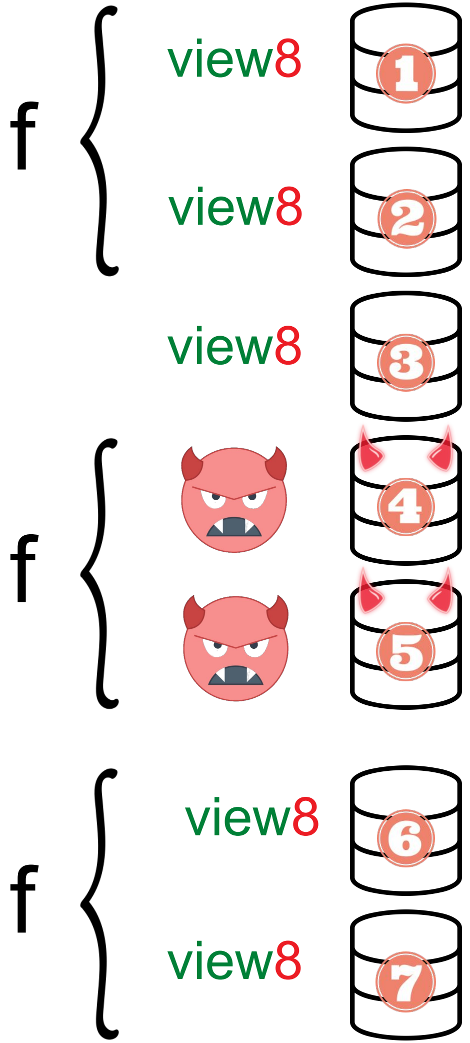


But can enter the view in the minimal WISH of 2f+1


f
view7
view7
view7
WISH(8)
f
f
view7
view7
1
2
3
4
5
6
7



UCDAVIS COMPUTER SCIENCE



All correct replicas converge on the same view.

FastSync Synchronizer

- View synchronization mechanics hidden under the spec

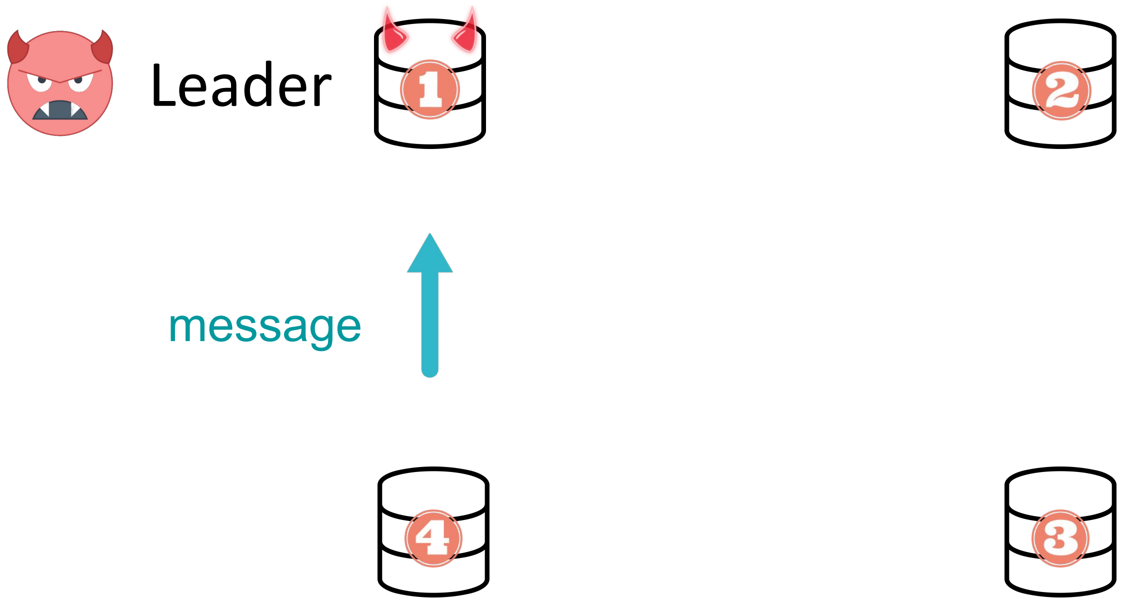
  → Can be proved liveness of different protocols

- FastSync + some of consensus protocols run in bounded space(PBFT, Hotstuff)

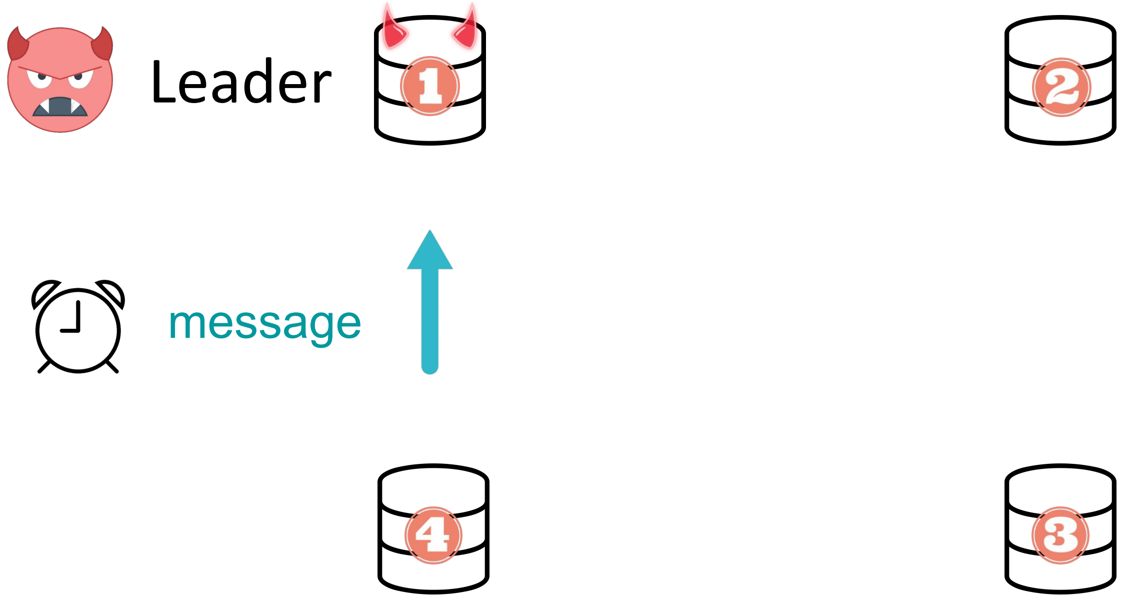
- Implementation has nice latency characteristics →
  Allows establishing tight latency bounds for latency.

UCDAVIS COMPUTER SCIENCE

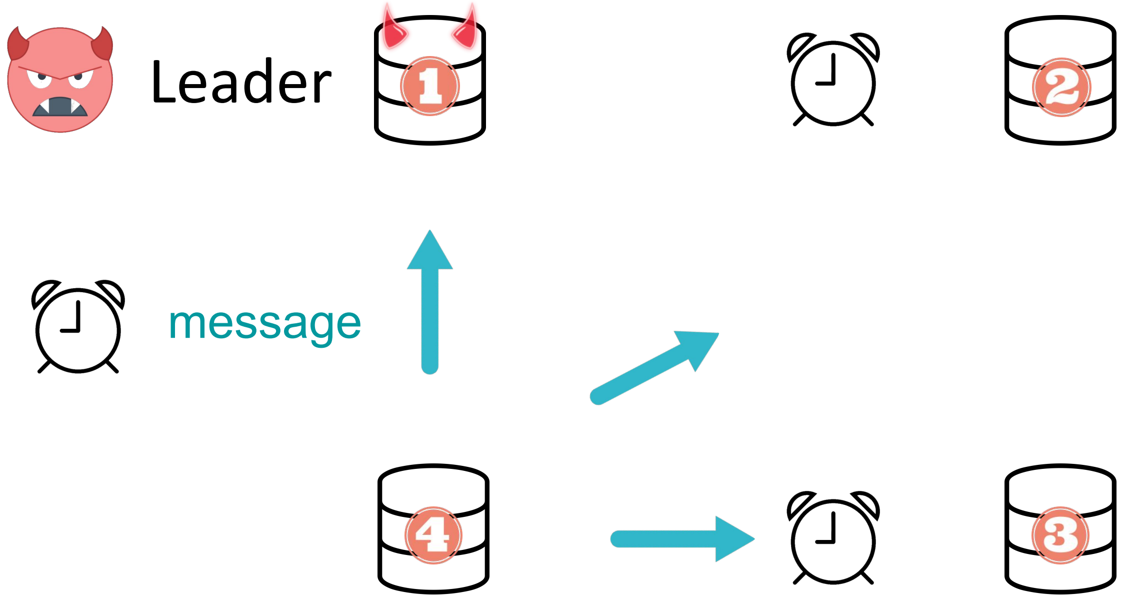
Ensuring Leader-based Protocol Liveness



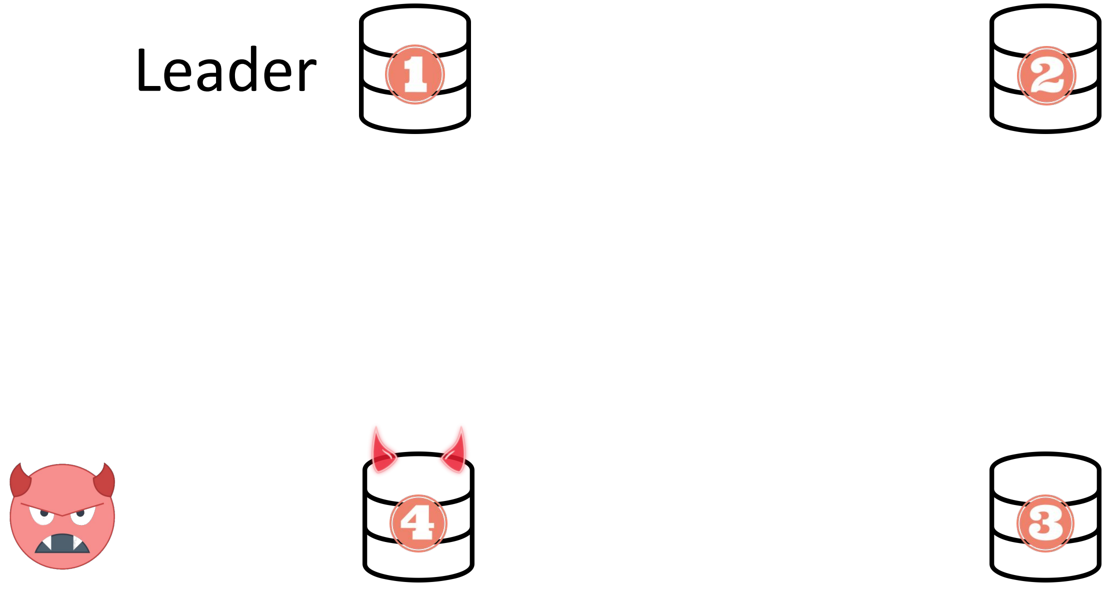
Ensuring Leader-based Protocol Liveness



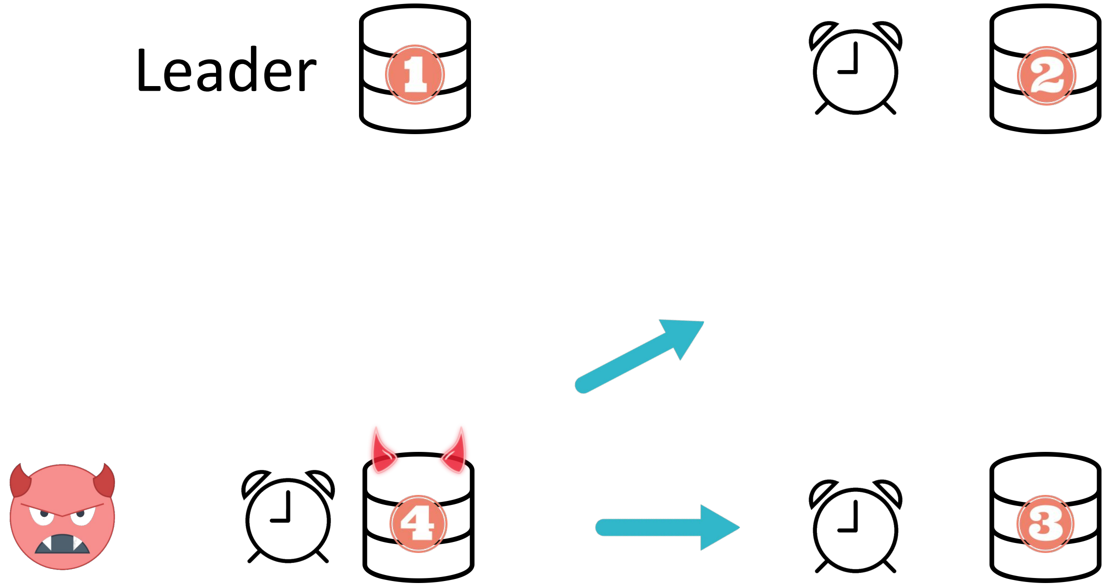
Ensuring Leader-based Protocol Liveness



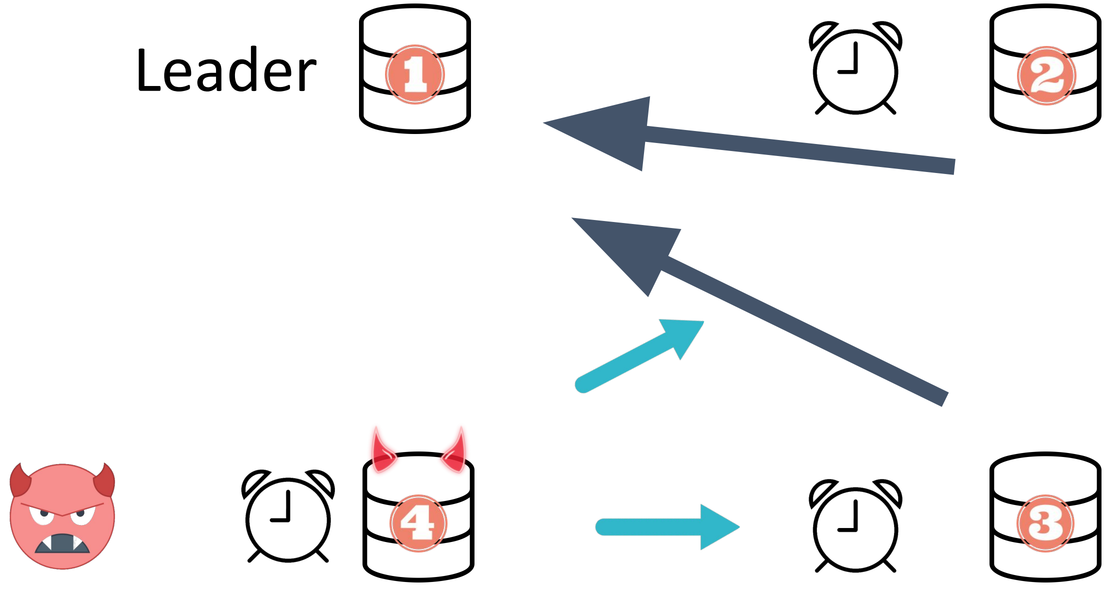
Ensuring Leader-based Protocol Liveness



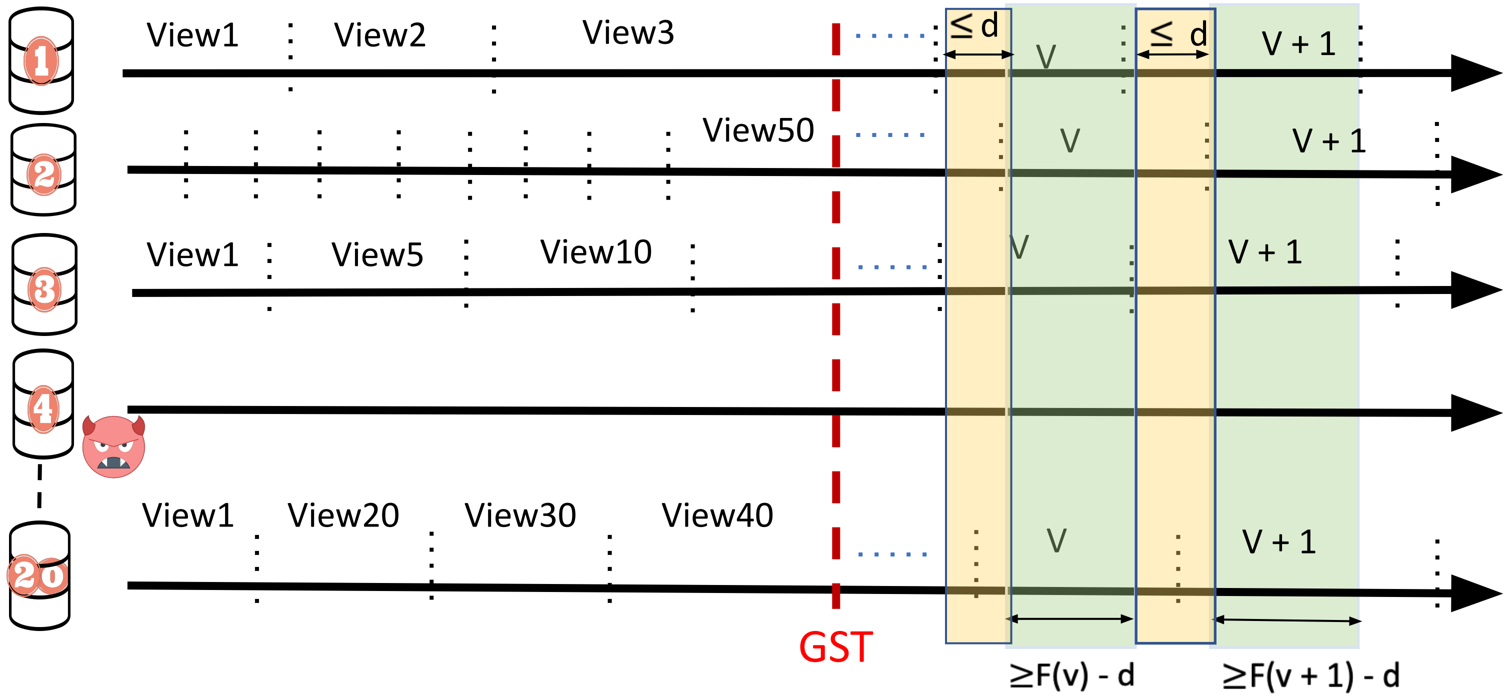
Ensuring Leader-based Protocol Liveness



Ensuring Leader-based Protocol Liveness

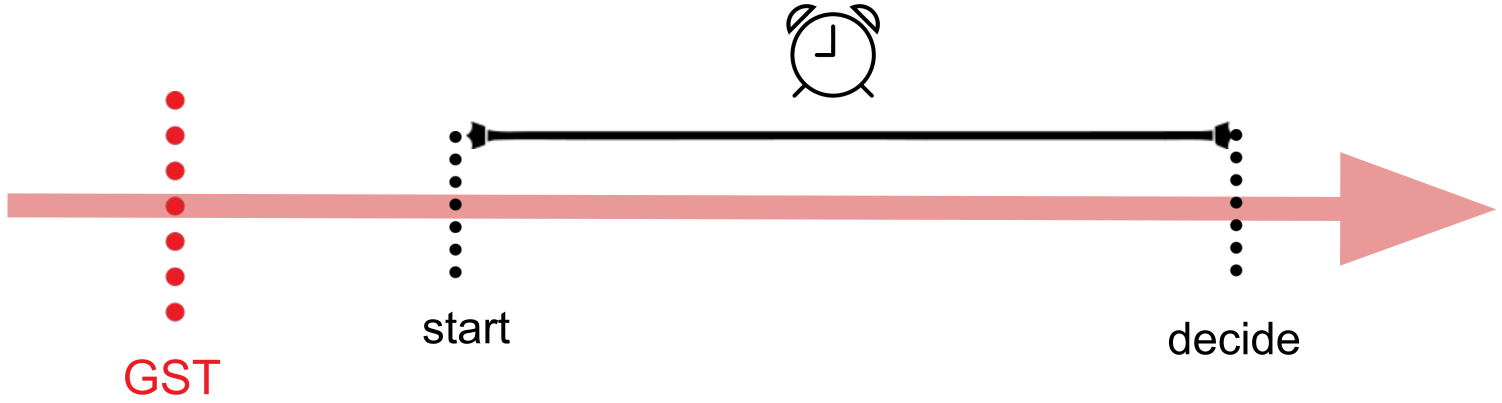


Synchronizer Specification



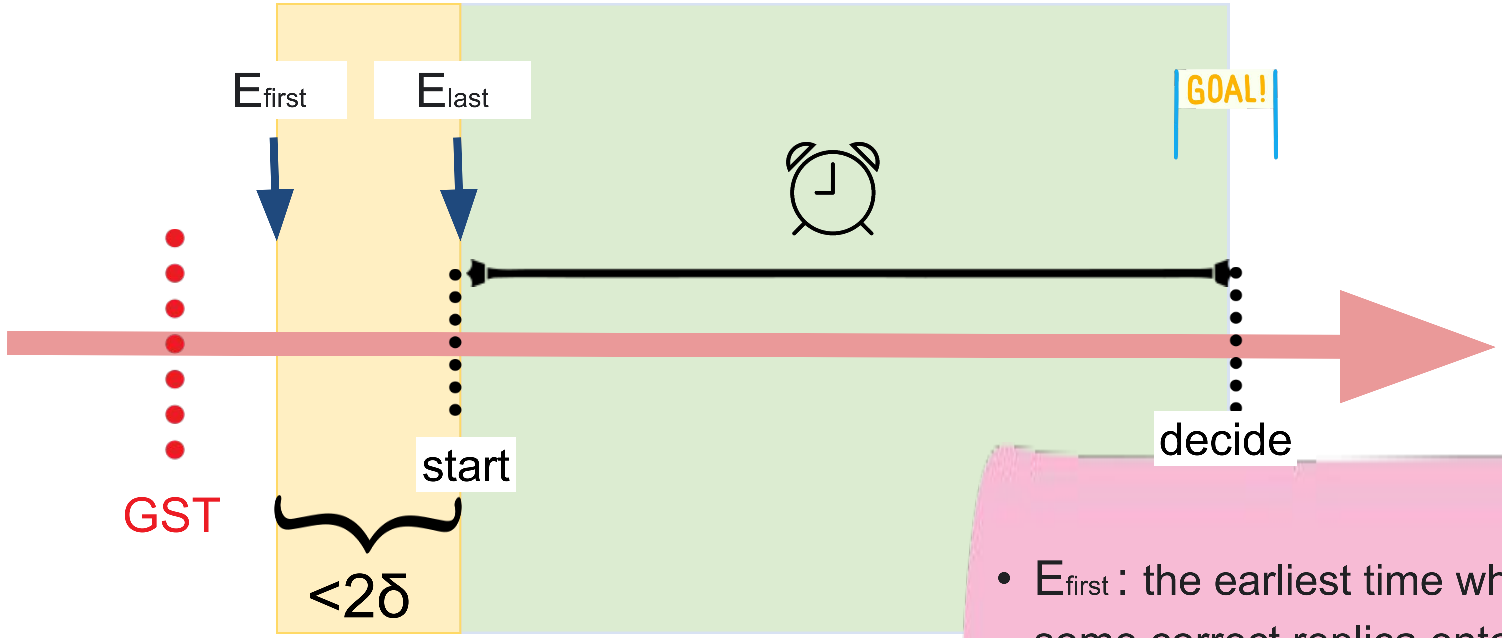
- Non-faulty replicas enter each view within d (e.g., d = 2δ)

Latency Bounds for Consensus



- Under favorable conditions:
  if the protocol executes during a synchronous period, F(1) is big enough and leader(1) is correct

Latency Bounds for Consensus

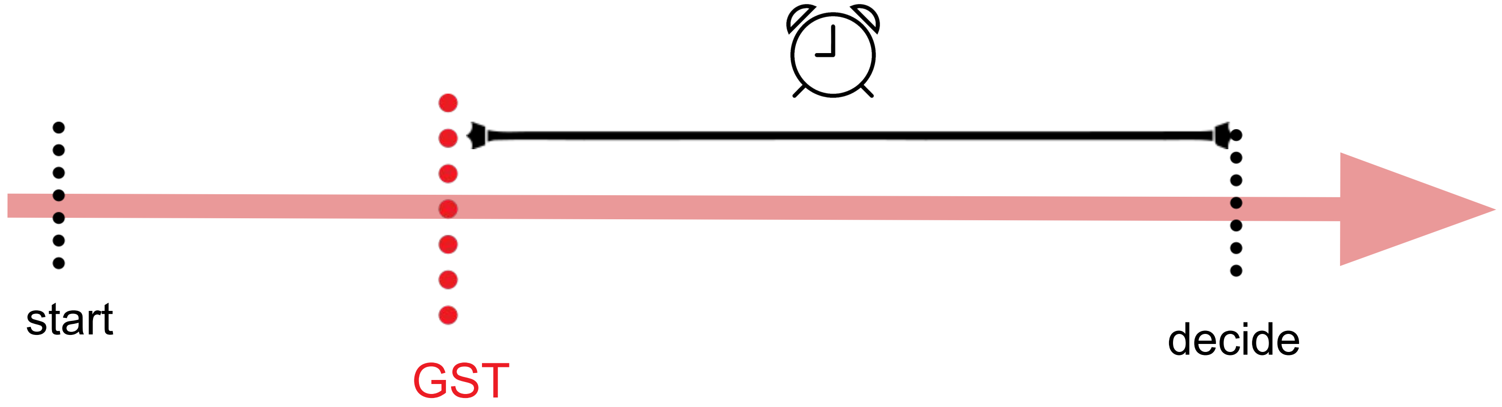


All correct replicas decide:

- HotStuff: no later than E_{last} + 4δ
- 2-phase HotStuff: no later than E_{last} + 3δ

- E_{first} : the earliest time when some correct replica enters v
- E_{last}: the latest time when some correct replica enters v

Latency Bounds for Consensus



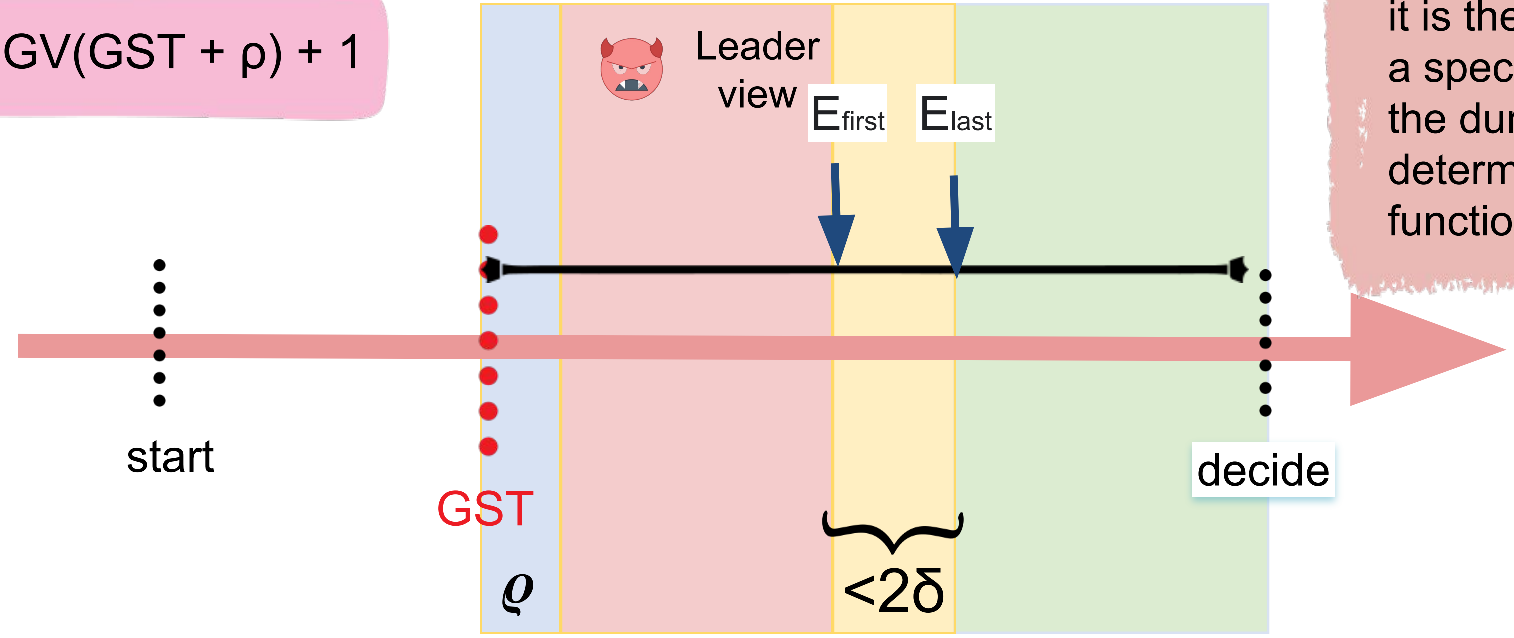
- Under unfavorable conditions:

  If the protocol starts during an asynchronous period

Latency Bounds for Consensus

v = GV(GST + ρ) + 1

Fp: When a replica enters a view where it is the leader, it sets a special timer for the duration determined by a function Fp



All correct replicas decide:

- HotStuff: no later than $$GST + \rho + \sum_{k=v-1}^{v+f-1} (F(k) + \delta) + 7\delta$$
- 2-phase HotStuff: no later than $$GST + \rho + \sum_{k=v-1}^{v+f-1} (F(k) + \delta) + f_p(v + f) + 5\delta$$

Thank you!