Byzantine Agreement & SMR with Sub-Quadratic Communication

Idit Keidar, Technion

Shout Out

Not a COINcidence: Sub-Quadratic Asynchronous Byzantine Agreement WHP.
Shir Cohen, Idit Keidar, and Alexander Spiegelman

Expected Linear Round Synchronization: The Missing Link for Linear Byzantine SMR.
Oded Naor and Idit Keidar

Byzantine Agreement (BA)

- Consensus among *n* processes
- Up to *f* can be controlled by an adversary and act arbitrarily

- A building block for State Machine Replication (SMR)

New Frontiers for BA & Byzantine SMR

- Permissioned blockchains – shared ledger
- Other FinTech infrastructures

Quorum
diem
HYPERLEDGER
MultiChain
ethereum
r3.corda
ORBS
parity
Algorand™

BA Has Been Around for Four Decades

[Pease, Shostak, Lamport 1980], [Lamport, Pease, Shostak 1980]

- 2500+, 7000+ citations, resp.
- Traditional use-cases – a handful of processes

Traditional BFT According to James Mickens

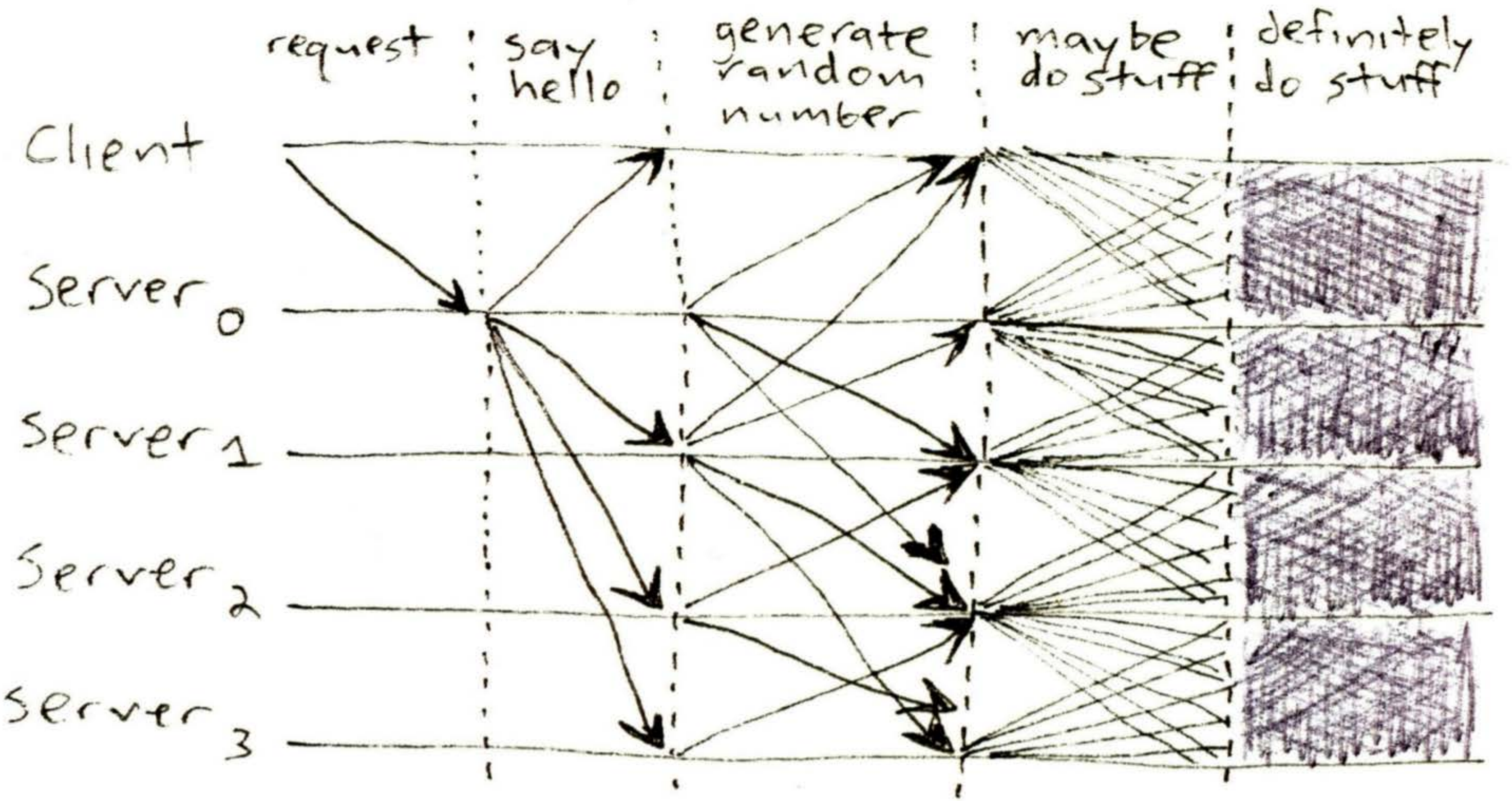


Figure 1: Typical Figure 2 from Byzantine fault paper: Our network protocol

Scalability Challenges

- Synchrony vs. asynchrony
  - Latency bounds defined in minutes
  - But deterministic fault-tolerant asynchronous consensus is impossible [Fisher, Lynch, Paterson 1985]

- Communication (word) complexity (of all processes together)
  - $\Omega(n^2)$ lower bound
    In the worst-case, in deterministic algorithms, regardless of synchrony [Dolev and Reischuk 1985]

Making It Scale

VRFs

- Assume asynchrony
- Solve BA
  with high probability (WHP)
  (probability of being correct
  tends to 1 as $n \to \infty$)

Threshold signatures

- Assume eventual synchrony
- Solve deterministic SMR
- Reduce *expected* complexity in
  some *optimistic* cases

Not a COINcidence: Sub-Quadratic Asynchronous Byzantine Agreement WHP

Shir Cohen, Idit Keidar, Alexander Spiegelman

DISC 2020

Contribution

The first sub-quadratic asynchronous BA WHP algorithm

- $\tilde{O}(n)$ word complexity and $O(1)$ expected time
- Safety and Liveness properties are gurenteed WHP
- Binary BA

- Previous sub-quadratic works made synchrony assumptions

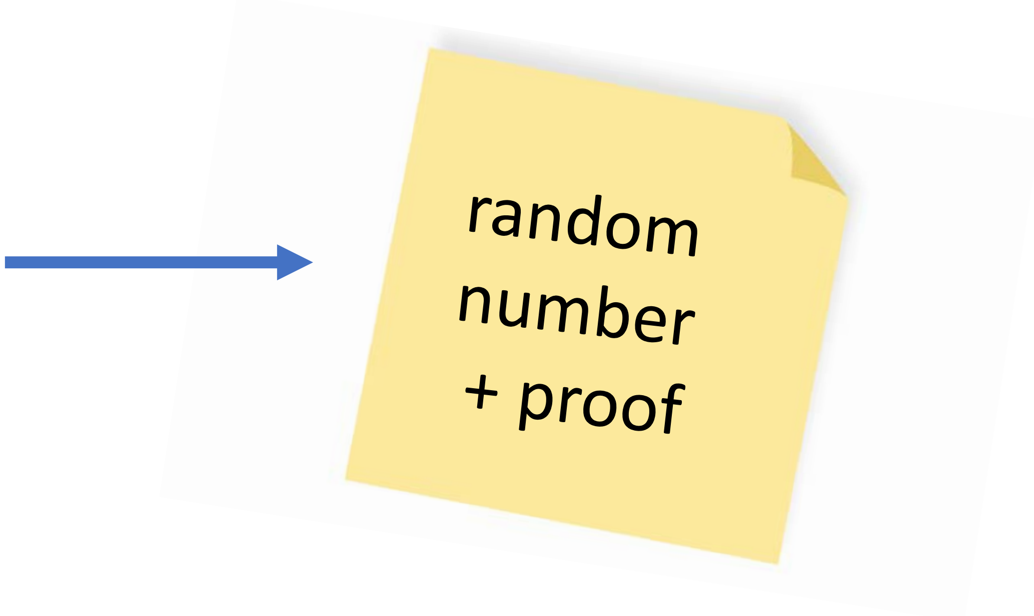
[King and Saia 2011], Algorand [Gilad et al. 2017]

Model

- Asynchronous
- n processes (permissioned)
- Up to f Byzantine processess for $n \approx 4.5f$
- Trusted PKI
  - Inherent for sub-quadratic algorithms
    [Abraham et al. 2019] [Blum et al. 2020] [Rambaud 2020]
- Delayed adaptive adversary:
  - Can use the contents of a message m sent by a correct process for scheduling a message m' only if $m \rightarrow m'$

Verifiable Random Function (VRF)

- A pseudorandom function that provides a proof of its correct computation
- For a secret key sk with a matching public key pk
  - $VRF_{sk}(x)$ is a random value
  - Verifiable using pk



Use VRFs for

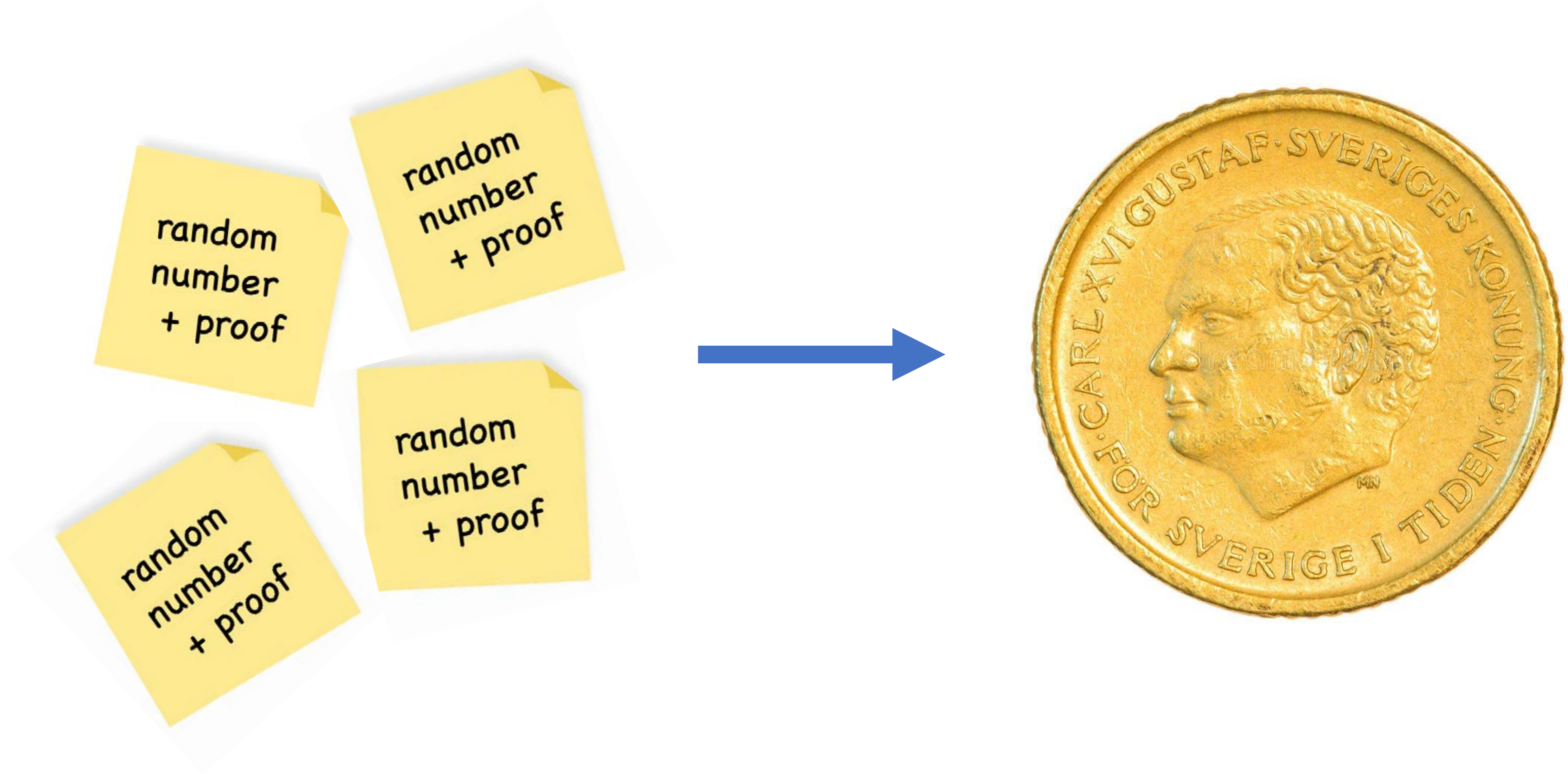
1. Flipping a shared coin
    - First step: $O(n^2)$ word complexity
2. Committee sampling
    - Cryptographic sortition
    - Reduces word complexity to $O(n \log n)$

Following Algorand [Gilad et al. 2017]

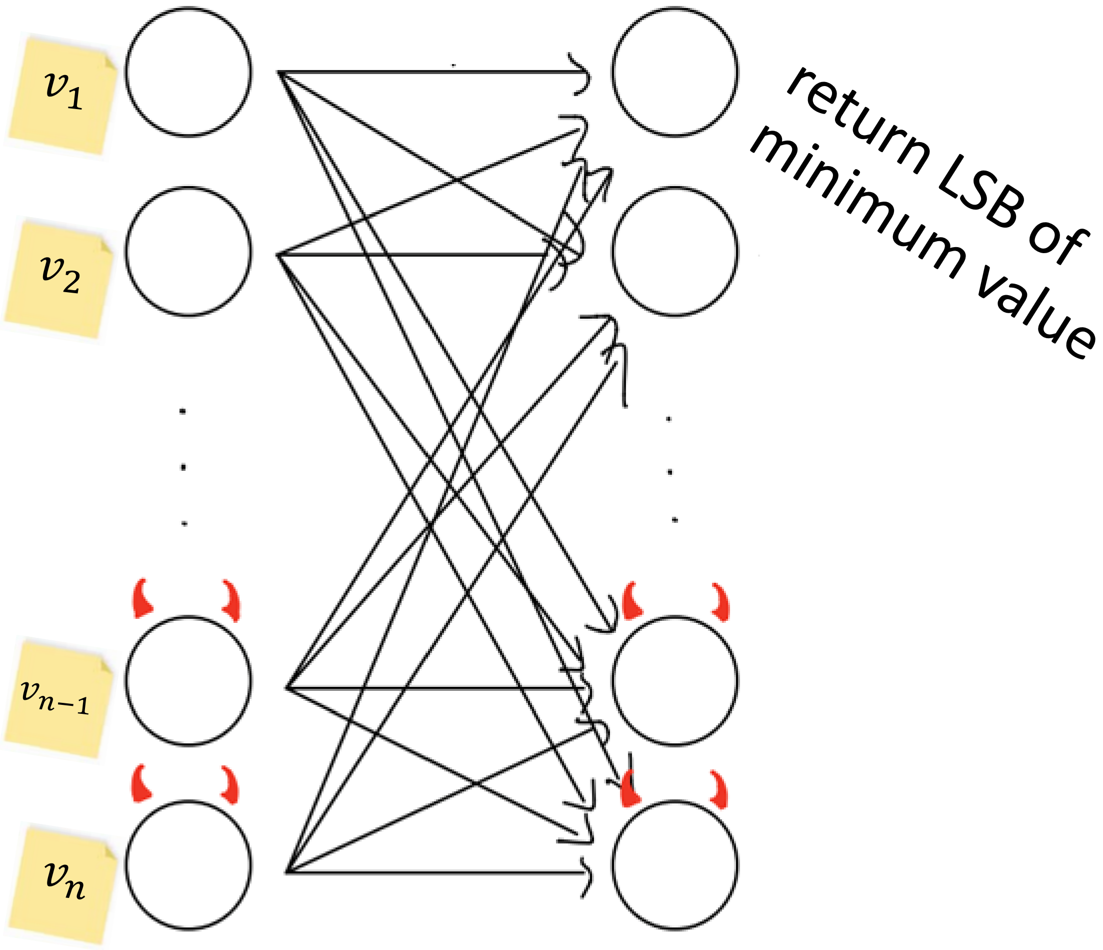
Shared Coin with Success Rate ρ

All correct processes output b with probability at least ρ, for any value $b \in \{0,1\}$

Shared Randomness

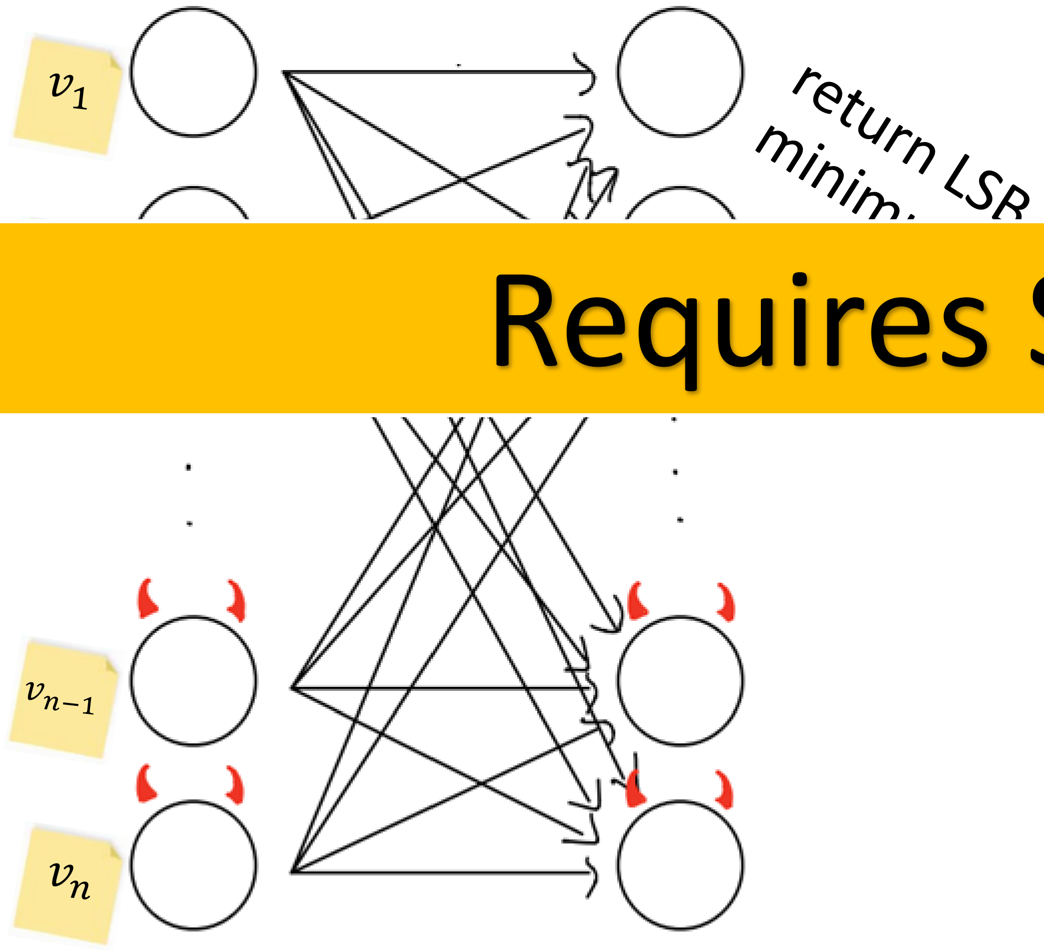


Background: A Simple VRF-Based Shared Coin



- Synchronous [Micali 2017]
- If the minimum VRF is of a correct process, all agree
  - With probability $\geq \frac{2}{3}$

Background: A Simple VRF-Based Shared Coin

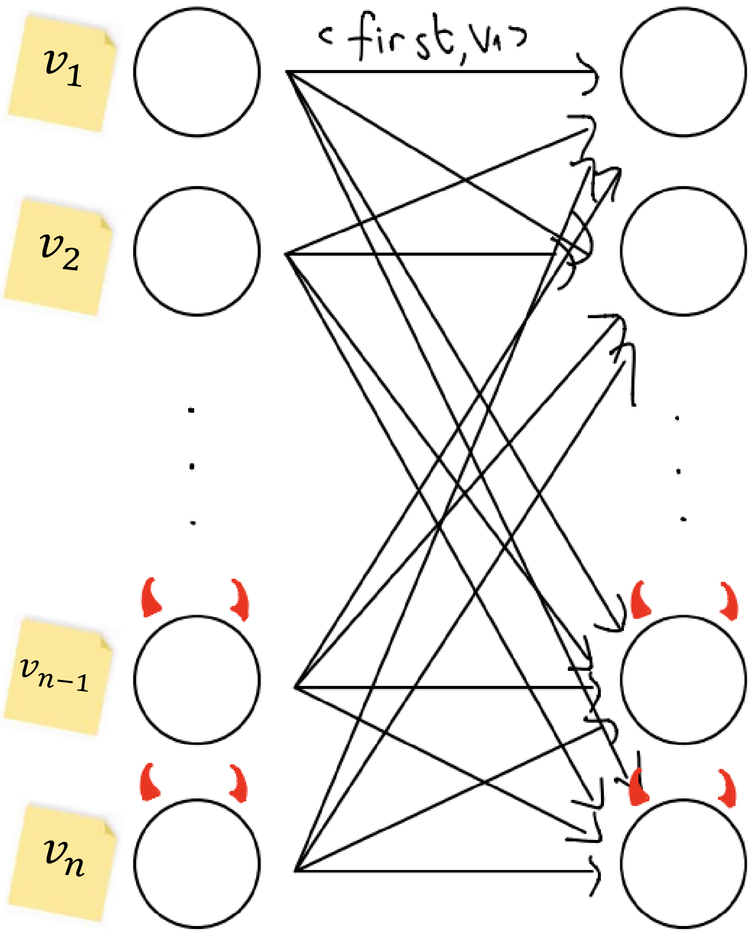


- Synchronous [Micali 2017]
- If the minimum VRF is of a

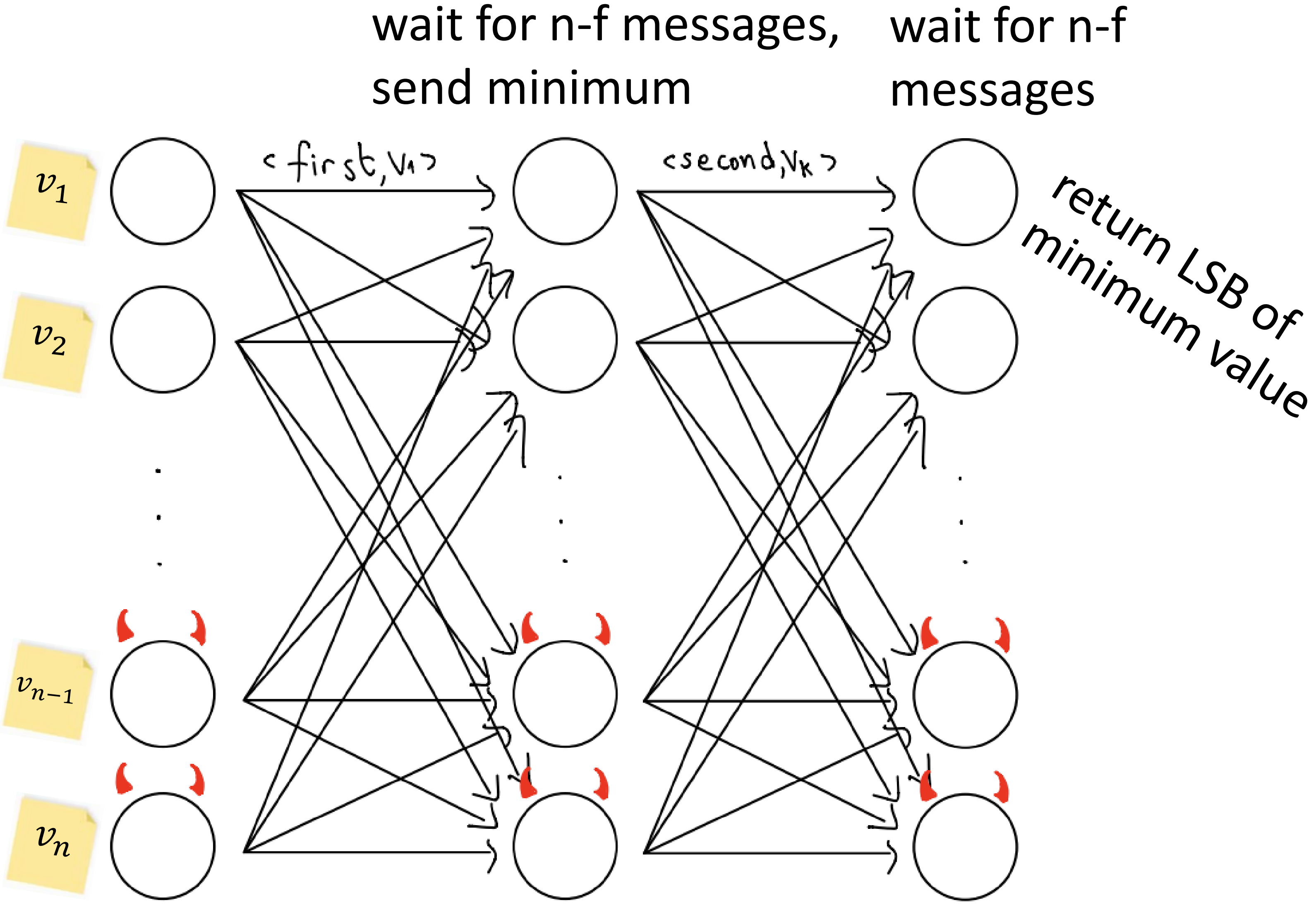
Requires Synchrony

Asynchronous Shared Coin – Take 1

wait for n-f messages



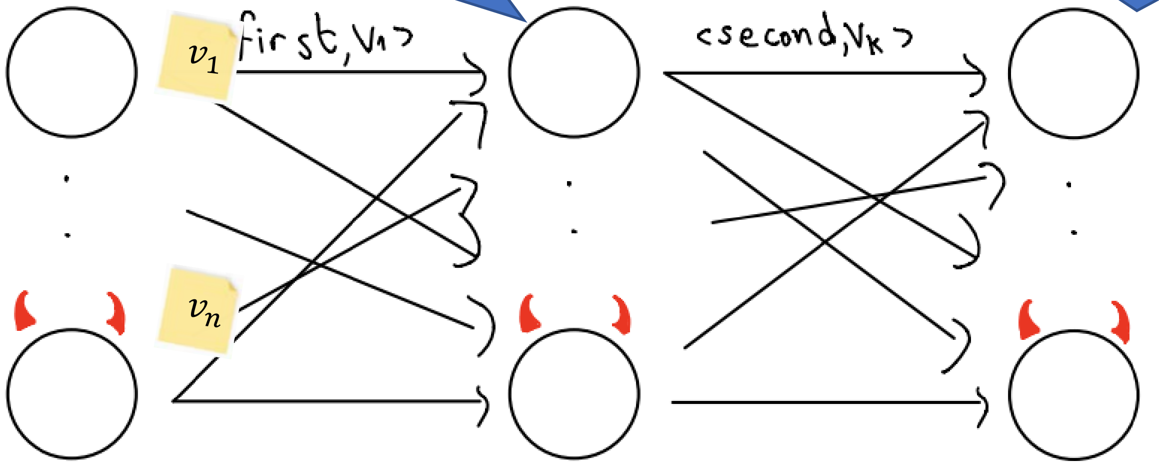
Asynchronous Shared Coin – Take 1



Asynchronous Shared Coin - Analysis

a value that reaches $f + 1$ correct processes is *common*

common values reach all correct processes



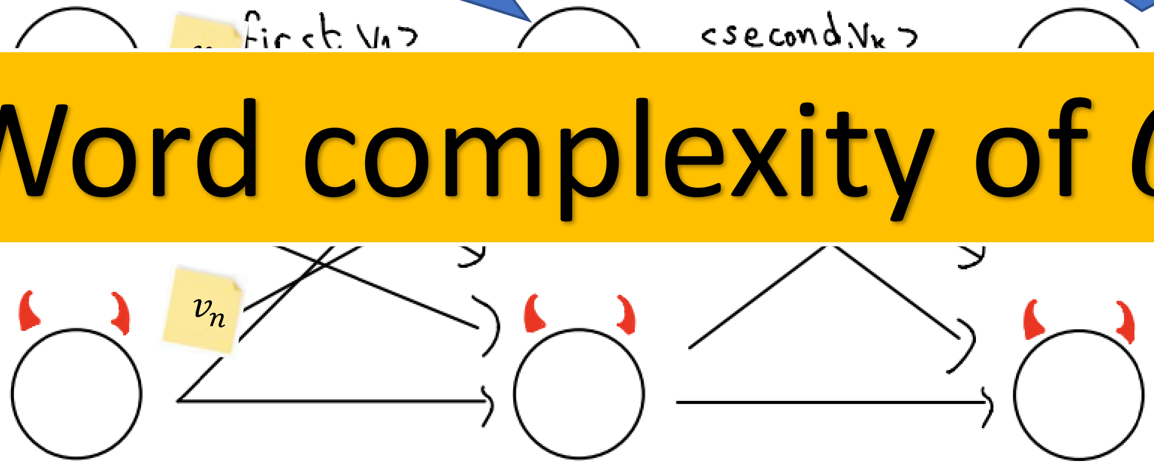
- We prove:
  - $\Omega(\epsilon)$ bound the number of common values
  - our adversary “commits” to them in advance

⇒With a constant probability, the global minimum is common

Asynchronous Shared Coin - Analysis

a value that reaches $f + 1$ correct processes is *common*

common values reach all correct processes



Word complexity of $O(n^2)$

- We prove:
  - $\Omega(\epsilon)$ bound the number of common values
  - our adversary “commits” to them in advance

⇒With a constant probability, the global minimum is common

Use VRFs for

1. Flipping a shared coin
   - First step: $O(n^2)$ word complexity
2. Committee sampling
   - Cryptographic sortition
   - Reduces word complexity to $O(n \log n)$

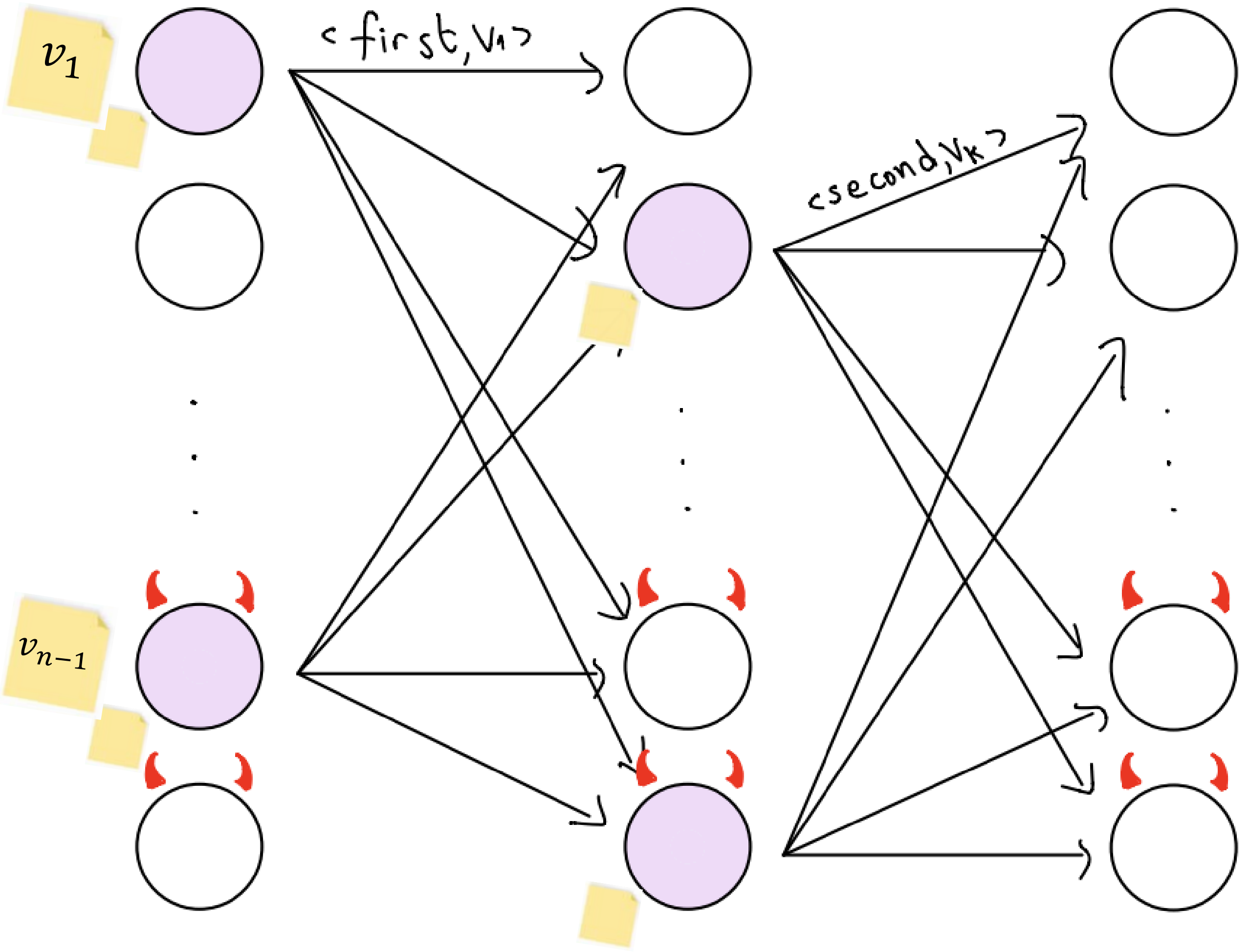
Following Algorand [Gilad et al. 2017]

Committee Sampling

- Use the VRF to sample O(log n) processes to a committee in each round
- Replace all-to-all rounds with committee-to-all rounds

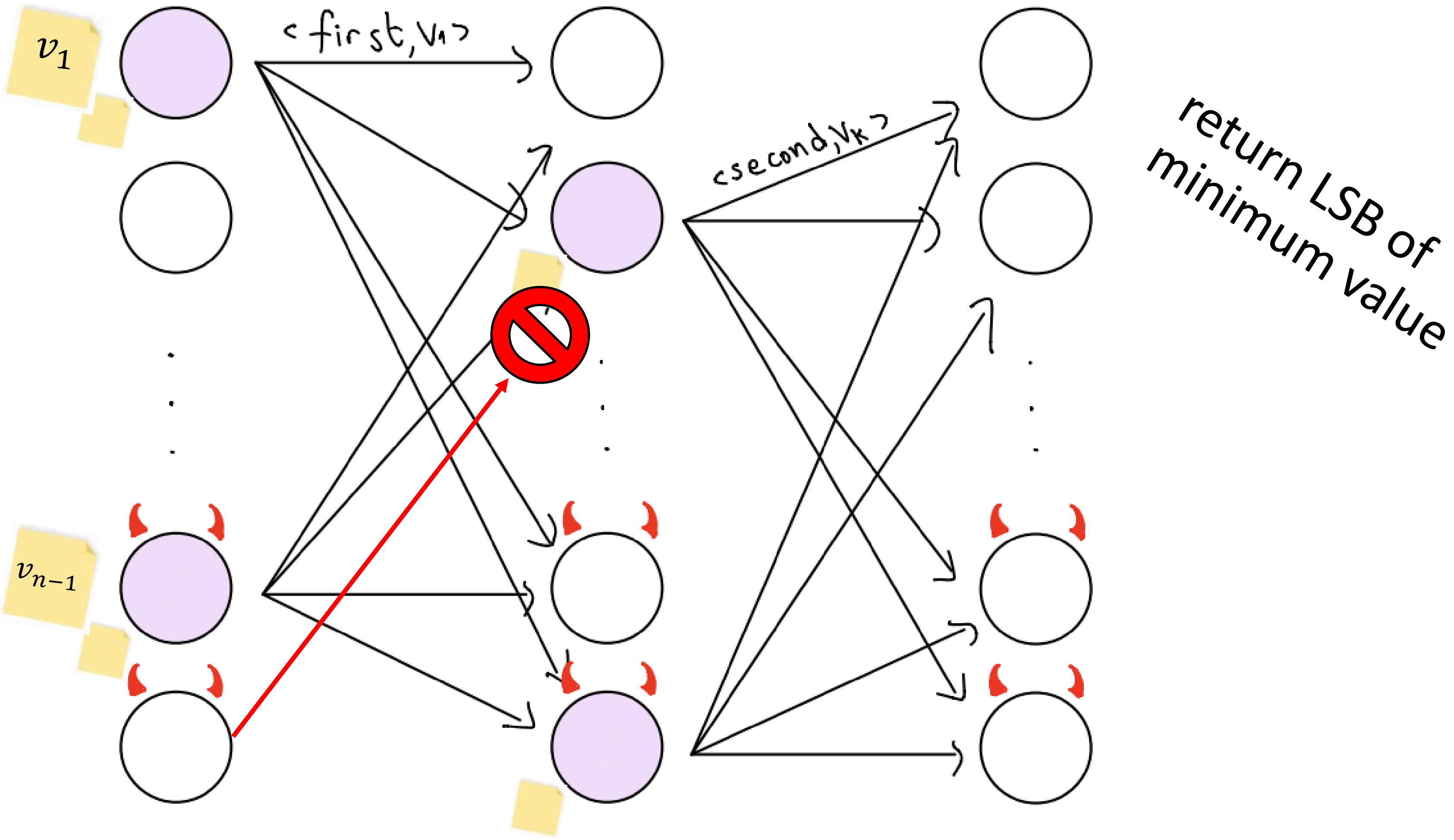
- Evading the adversary:
  - Use a new committee in each round
  - Send to all since committees are unpredictable
  - By Chernoff bounds, “not too many” faulty processes in each committee

Shared Coin – Take 2

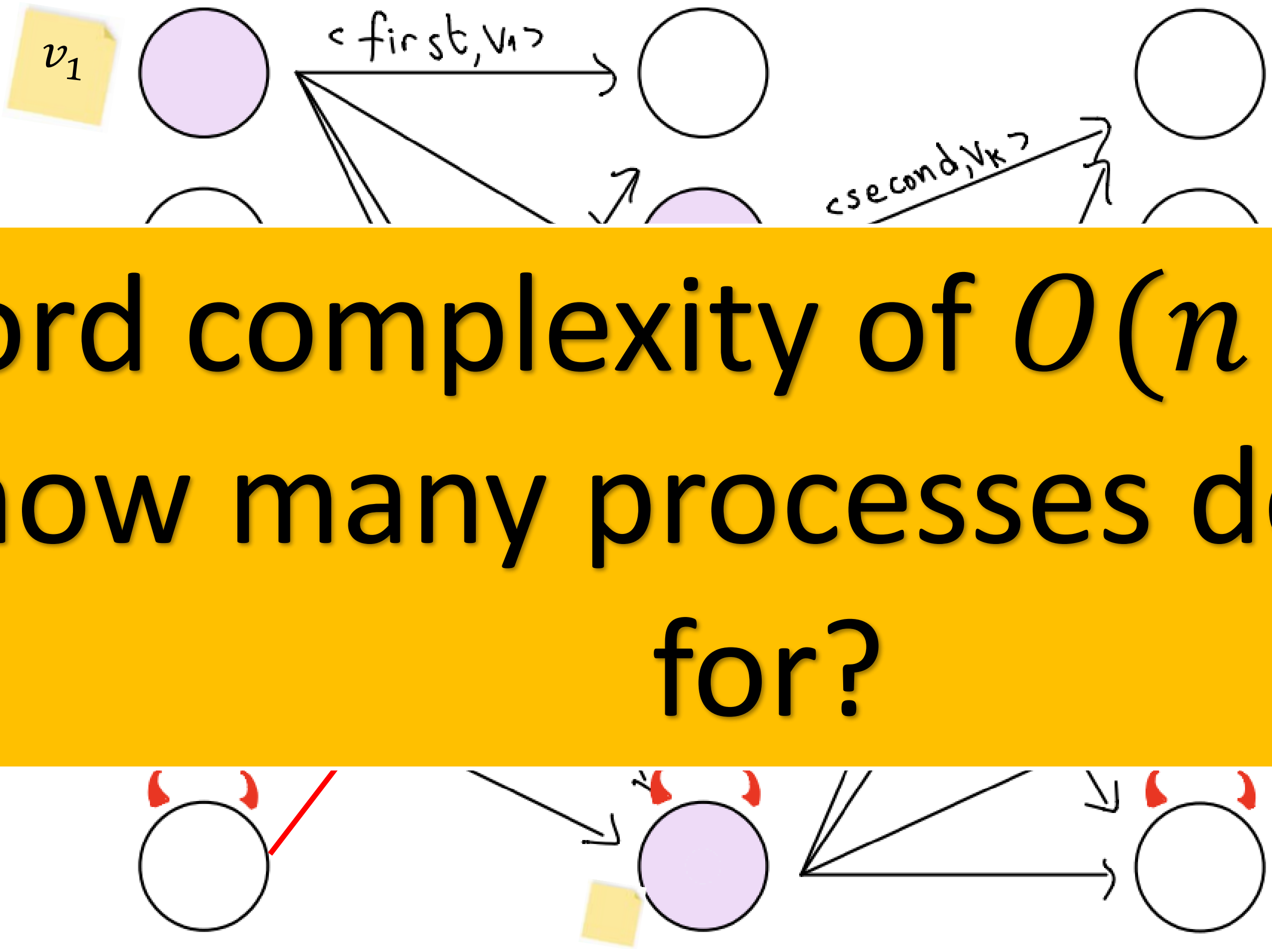


return LSB of minimum value

Shared Coin – Take 2



Shared Coin – Take 2



Word complexity of $O(n\ log\ n)$,
but how many processes do we wait for?

Committee Sampling in Asynchronous Model

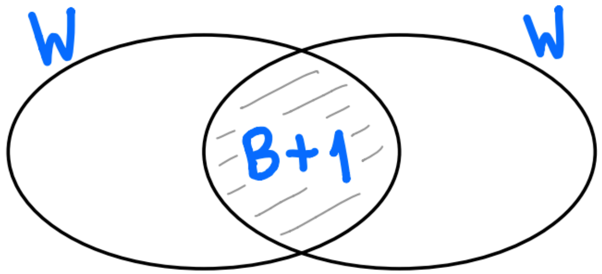
- Committee based protocols cannot wait for $n - f$ processes. Instead, they wait for W processes.

- We choose W, B so that using Chernoff bounds, WHP:

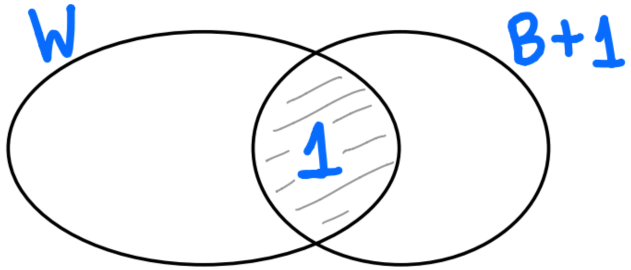
1. At least W processes in each committee are correct
2. At most B processes in in each committee are Byzantine

Committee Sampling in Asynchronous Model

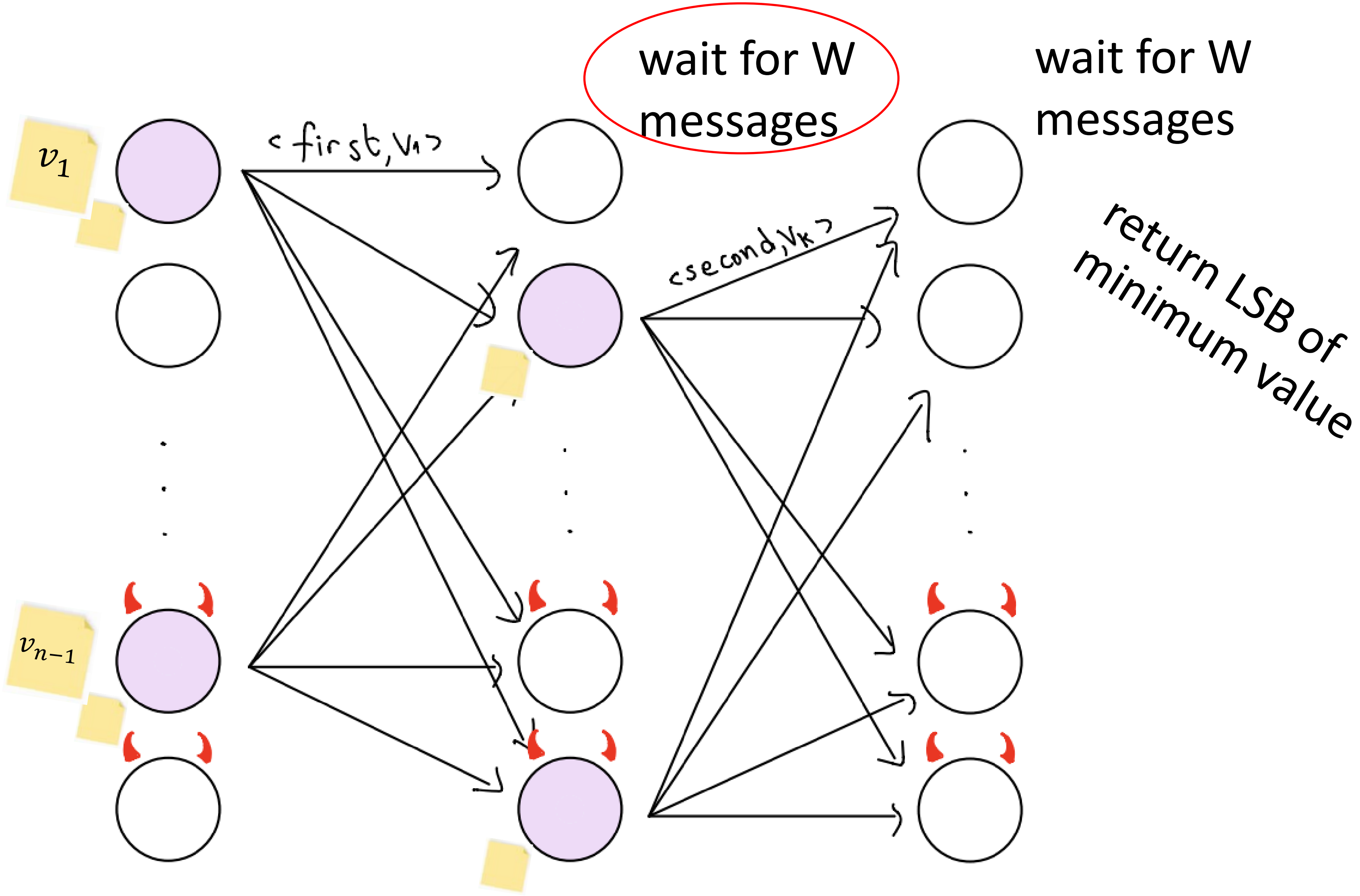
3. Every two subsets in a committee of size W intersect by at least $B + 1$ processes



4. Every two subsets in a committee of size W and $B + 1$ intersect by at least 1 process



Shir Cohen's Shared Coin



From Coin Flipping to (Binary) BA WHP

Approver +  → BA WHP

- Approver based on [Bracha 1987] – reliable broadcast
  - But with committee sampling
- BA based on [Mostefaoui et al. 2015]

Approver

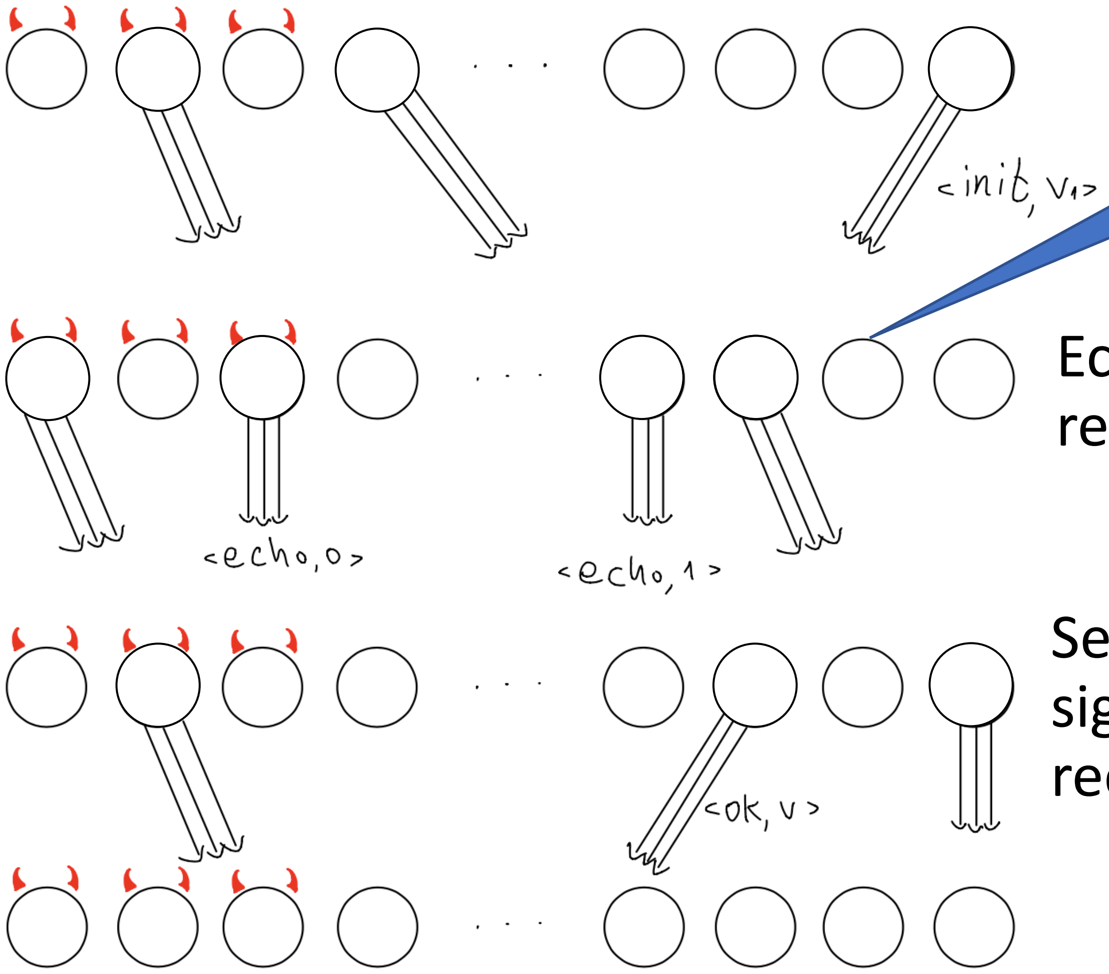
API: *approve*$_i$*(v*$_i$*)* returns a set of values

We assume *approve* is called with at most two different values

WHP the following hold:

- Validity: If all correct processes invoke $approve(v)$ then the only possible return value of correct processes is $\{v\}$
- Graded agreement: If correct processes return both $\{v\}$ and $\{w\}$ then $v = w$
- Termination: If all correct processes invoke *approve* then it returns with a non-empty set at all of them

Approver Without Sampling



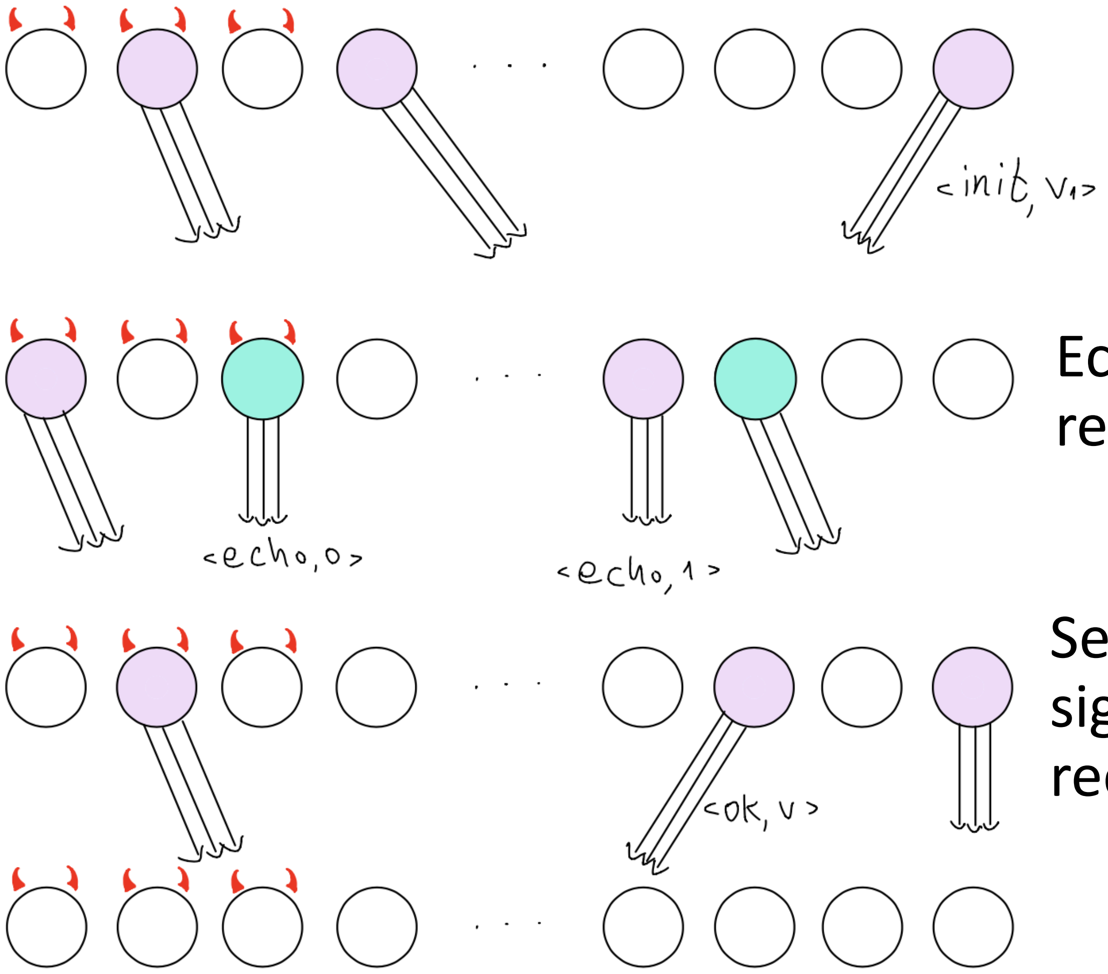
May speak twice

Echo v upon receiving f+1 v

Send <ok, v> with n-f signatures upon receiving n-f <echo, v>

Return the set of values in the first n-f ok messages

Approver 👍 With Sampling

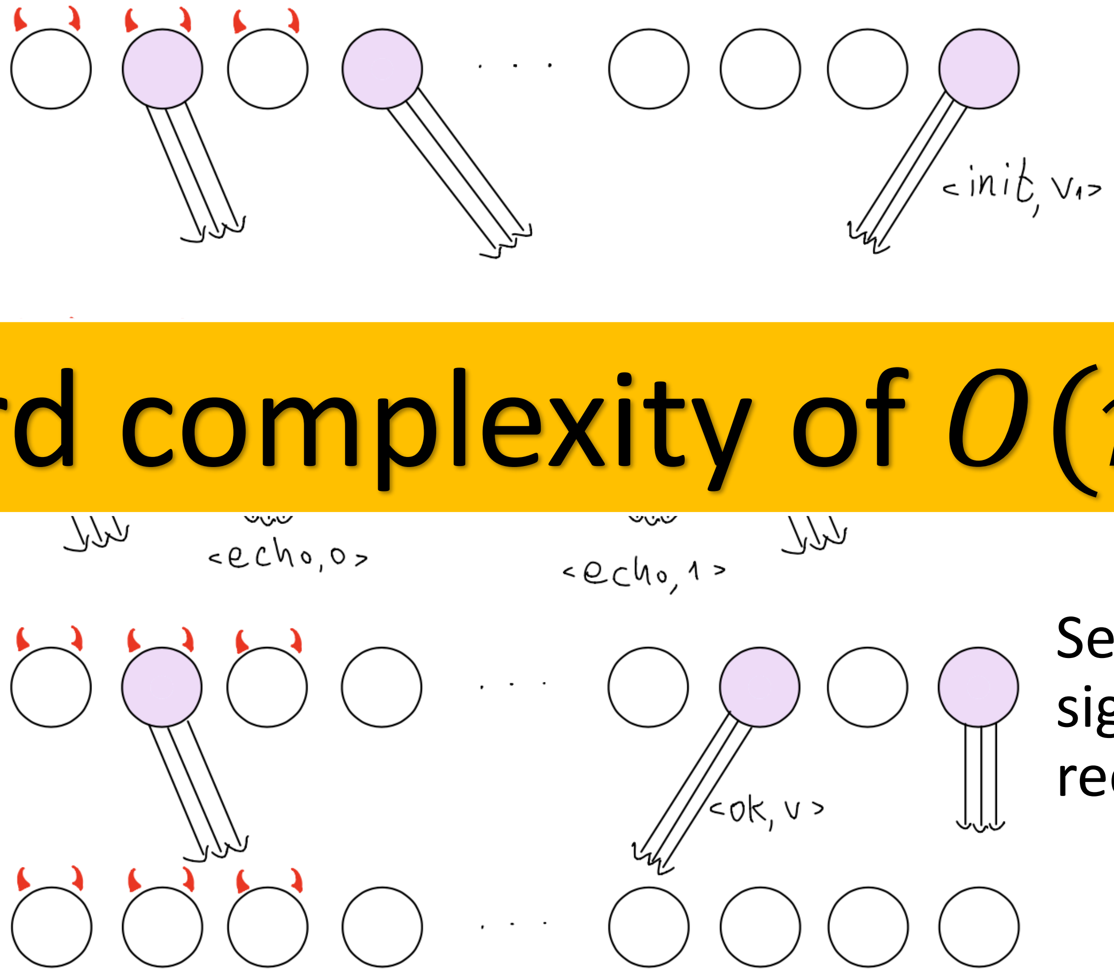


Echo v upon receiving B+1 v

Send <ok, v> with W signatures upon receiving W <echo, v>

Return the set of values in the first W ok messages

Approver 👍 With Sampling



Word complexity of $O(n \log^2 n)$

Send <ok, v> with W signatures upon receiving W <echo, v>

Return the set of values in the first W ok messages

From Coin Flipping to (Binary) BA WHP

- Approver based on [Bracha 1987] – reliable broadcast
  - But with committee sampling
- BA based on [Mostefaoui et al. 2015]

BA WHP

$est_i \leftarrow v_i$
$decision_i \leftarrow \bot$

for $r = 0, 1, \ldots$ **do**
    $vals \leftarrow \text{approve}(est_i)$
    if $vals = \{v\}$ for some v **then**
        $propose_i \leftarrow v$
    otherwise $propose_i \leftarrow \bot$

    $c \leftarrow \text{whp\_coin}(r)$
    $props \leftarrow \text{approve}(propose_i)$
    if $props = \{v\}$ for some $v \neq \bot$ **then**
        $est_i \leftarrow v$
        if $decision_i = \bot$ **then**
            $decision_i \leftarrow v$
    else
        if $props = \{\bot\}$ **then**
            $est_i \leftarrow c$
        else $\%props = \{v, \bot\}$
            $est_i \leftarrow v$

BA WHP

$est_i \leftarrow v_i$
$decision_i \leftarrow \bot$

for $r = 0, 1, \ldots$ **do**
  $vals \leftarrow \text{approve}(est_i)$
  if $vals = \{v\}$ for some v **then**
    $propose_i \leftarrow v$
  otherwise $propose_i \leftarrow \bot$

  $c \leftarrow \text{whp\_coin}(r)$

  $props \leftarrow \text{approve}(propose_i)$
  if $props = \{v\}$ for some $v \neq \bot$ **then**
    $est_i \leftarrow v$
    if $decision_i = \bot$ **then**
      $decision_i \leftarrow v$
  else
    if $props = \{\bot\}$ **then**
      $est_i \leftarrow c$
    else $\%props = \{v, \bot\}$
      $est_i \leftarrow v$

BA WHP

$est_i \leftarrow v_i$
$decision_i \leftarrow \bot$

Word complexity of $O(n \log^2 n)$

  $propose_i \leftarrow v$
 otherwise $propose_i \leftarrow \bot$

 $c \leftarrow \text{whp\_coin}(r)$

 $props \leftarrow \text{approve}(propose_i)$
 if $props = \{v\}$ for some $v \neq \bot$ **then**
 else
  if $props = \{\bot\}$ **then**
   $est_i \leftarrow c$
  else $\quad$ %$props = \{v, \bot\}$
   $est_i \leftarrow v$

Not a COINcidence Summary

- First formalization of randomly sampled committees using cryptography in asynchronous settings
- First sub-quadratic asynchronous shared coin and BA WHP algorithms
- Expected $\tilde{O}(n)$ word complexity and $O(1)$ expected time

Limitations:

- Binary consensus only
- Safety and liveness only WHP
- One-shot algorithm (not SMR)
- Non-optimal resilience – improved by [Blum et al. 2020]

Making It Scale

VRFs

Threshold signatures

- Assume asynchrony
- Solve BA
  with high probability (WHP)
  (probability of being correct
  tends to 1 as $n \to \infty$)

- Assume eventual synchrony
- Solve deterministic SMR
- Reduce *expected* complexity in some *optimistic* cases

Expected Linear Round Synchronization: The Missing Link for Linear Byzantine SMR

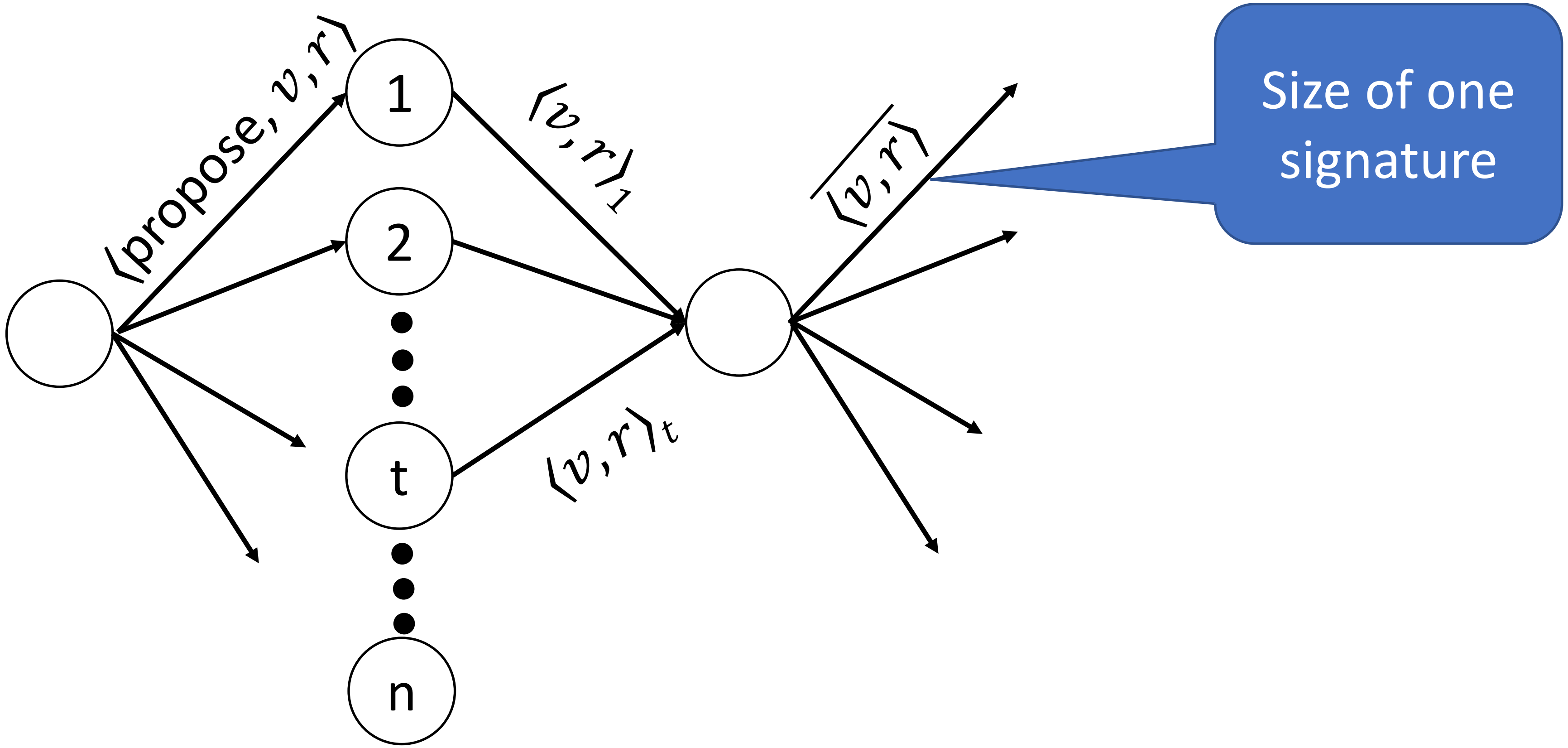
Oded Naor and Idit Keidar

DISC 2020

Model

- Eventual synchrony
  - Initially asynchronous
  - Synchronous after *Global Stabilization Time (GST)*
  - With latency bound δ
- Optimal resilience: $f < n/3$
  - For simplicity, assume $n=3f+1$
- Crypto: threshold signatures, PKI
- Shared source of randomness

Threshold Signatures Reduce Communication



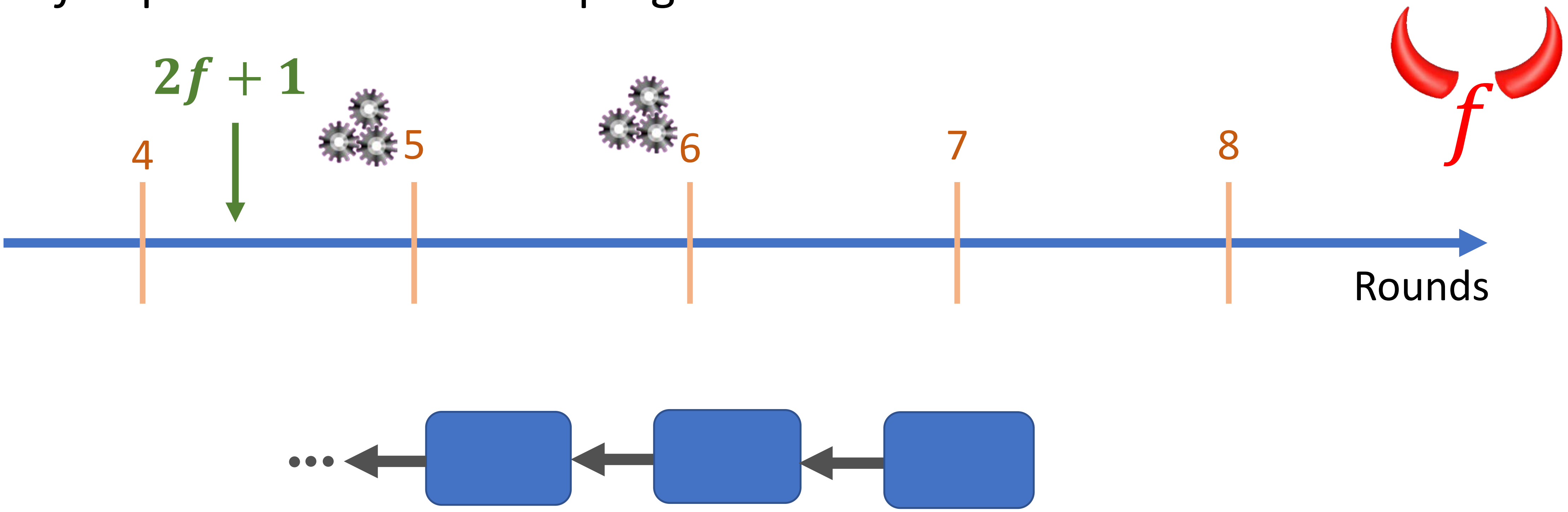
Byzantine SMR Communication Costs

Year	Protocol	Word complexity to reach a decision
1988	DLS	$O(n^3)$
1999	PBFT	$O(n^2)$
2007	Zyzzyva	$O(n^2)$
2016	Tendermint, Casper	$O(n)$
2017	Algorand	Committees
2018	HotStuff	$O(n)$
2019	LibraBFT	$O(n)$

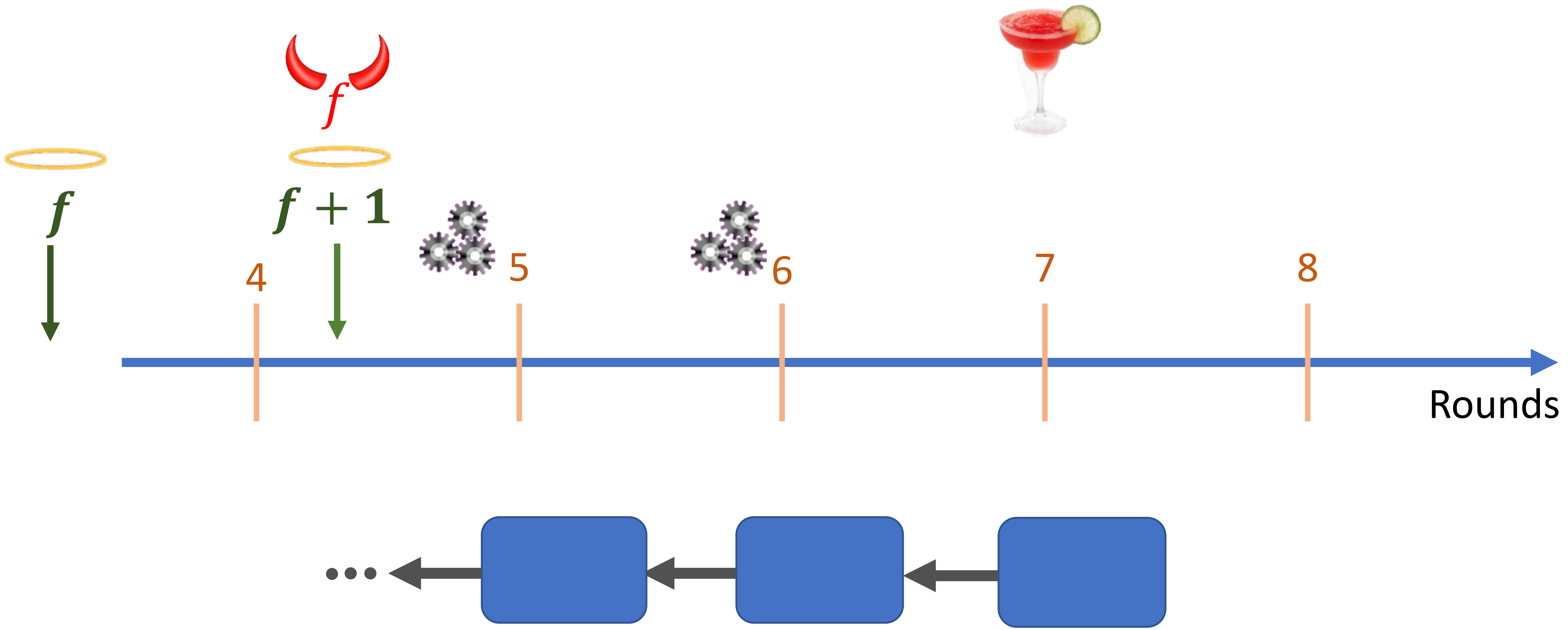
$O(n)$ once 2f+1 correct processes follow a correct leader

Eventually Synchronous Byzantine SMR

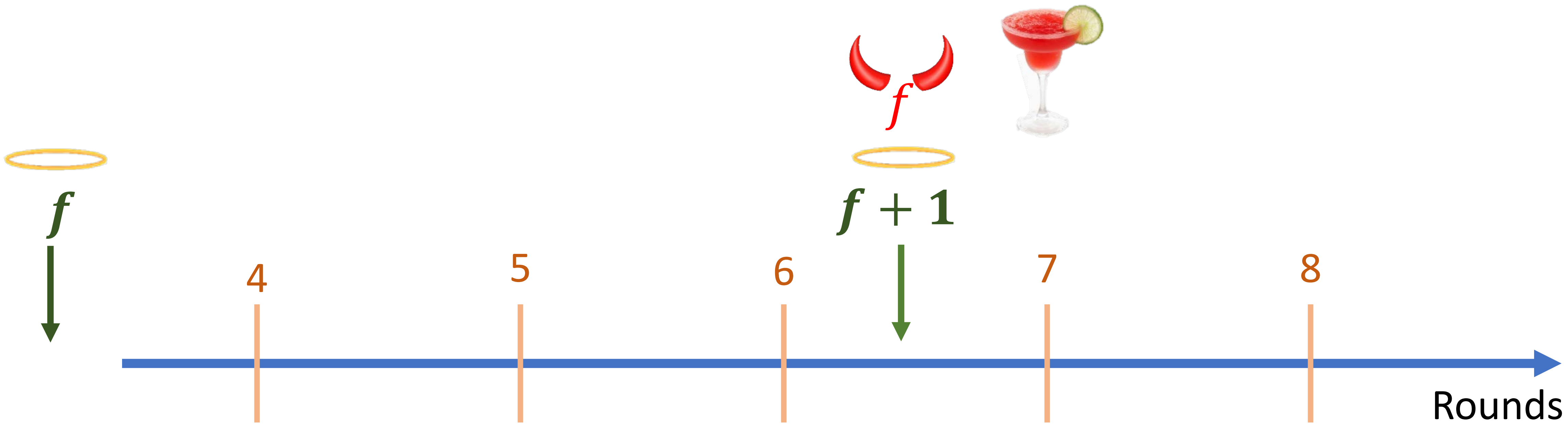
- Each process divides its time into rounds (aka views)
- $2f+1$ processes can make progress



An Alternative Run



Needed: Round Synchronization (RS)



Round Synchronization Makes SMR Live

- Theorem 4 from HotStuff [Yin et al. 2019]:

  "After GST, there exists a bounded time period T_f such that if all correct replicas remain in view v during T_f and the leader for view v is correct, then a decision is reached."

- Formulated and solved as a separate problem
  HotStuff Pacemaker, Cogsworth [Naor et al. 2020], [Bravo et al. 2020]

The Round Synchronization Service

- Parametrized by a time period Δ (e.g., $= 4\delta$)
- Repeatedly outputs round-leader pairs $\langle r, p \rangle$
  - Enter round r with leader p
  - Rounds are monotonically increasing
  - Leaders are uniquely determined per round
- Guarantee:

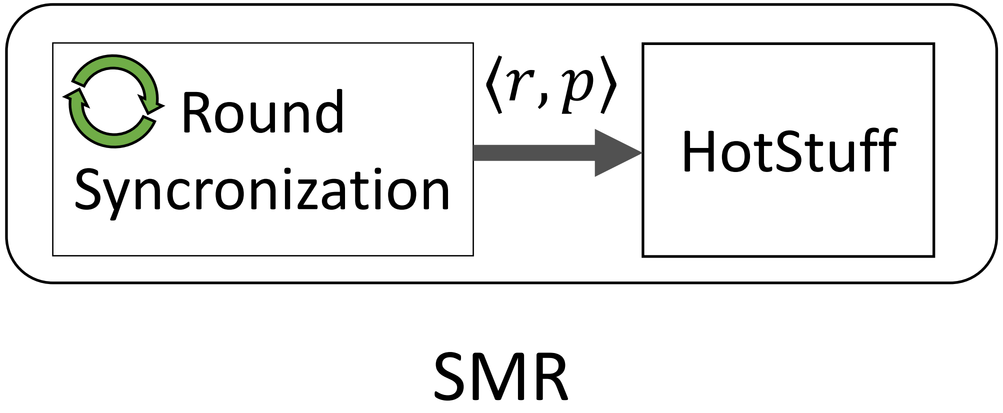
For any time t, there is a synchronization time $t_s \geq t$ so that all correct processes are in the same round with the same correct leader from time t_s for at least Δ

- The precondition needed for HotStuff's liveness theorem

RS is the Performance Bottleneck

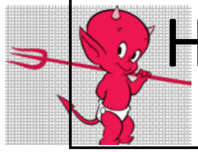
- After round synchronization with a correct leader, we have deterministic SMR
  - O(n) word complexity per decision
  - O(1) time per decision

HotStuff [Yin et al. 2019]
Tendermint [Buchman et al. 2018]
LibraBFT [Baudet et al. 2019]



- Our solution: RS with expected linear word complexity, constant time

Fast RS is the Key to SMR Performance

Round Syncronization +  HotStuff → SMR

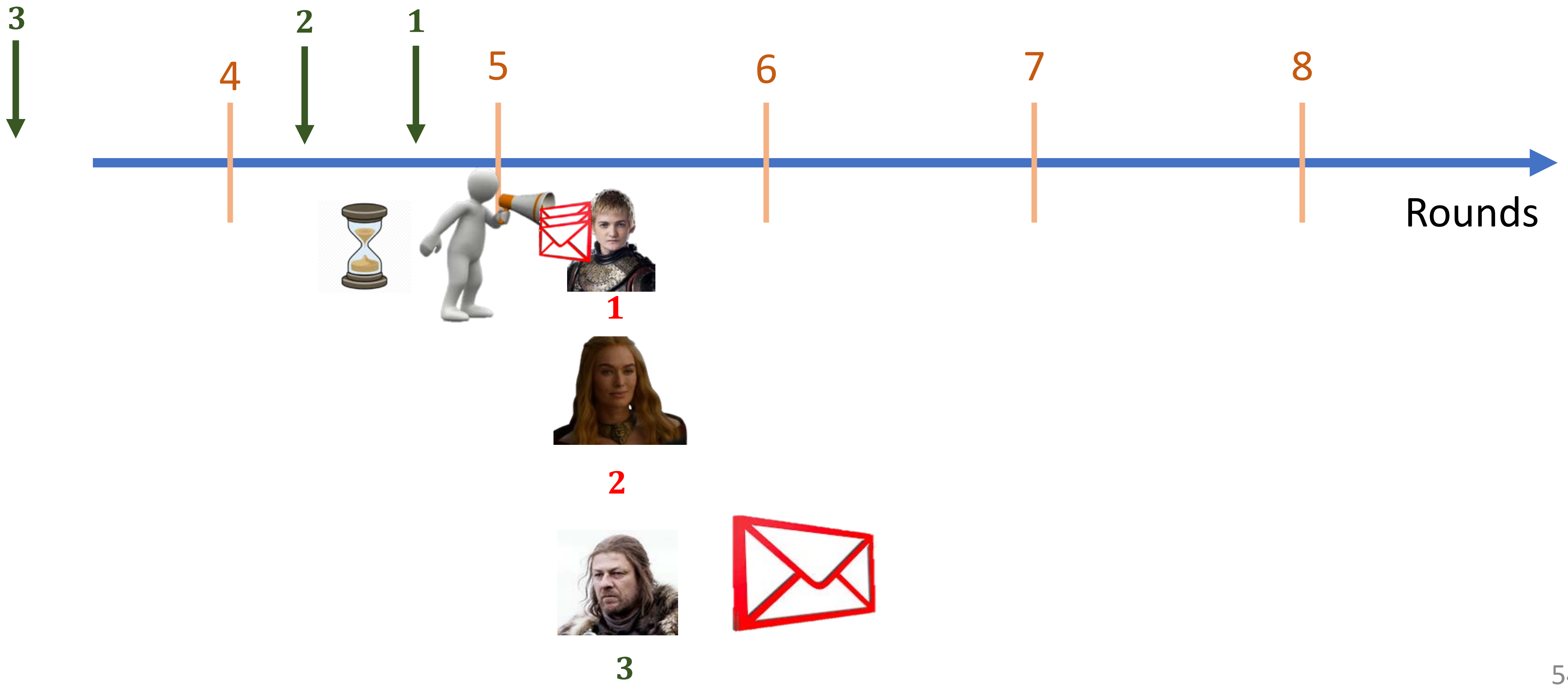
expected O(n) + O(n) = expected O(n)

- We get: deterministic SMR, after GST, each decision with
  - Expected O(n) word complexity, $O(n^3)$ worst-case
  - Expected O(1) time, $O(n^2)$ worst case

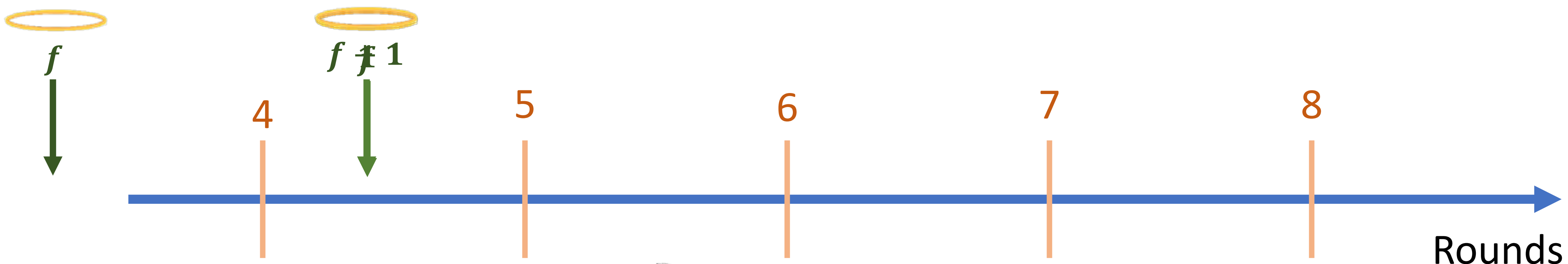
Relay-Based Round Synchronization

- In each round *r*, a designated relay is responsible for synchronizing the processes to this round *r*
- The relay collects threshold signatures to prove that enough processes proceed with it
- On timeout, switch to another relay
- Randomly permute relays in each round
  - In expected constant time, a correct relay is chosen

Relay-Based Round Synchronization



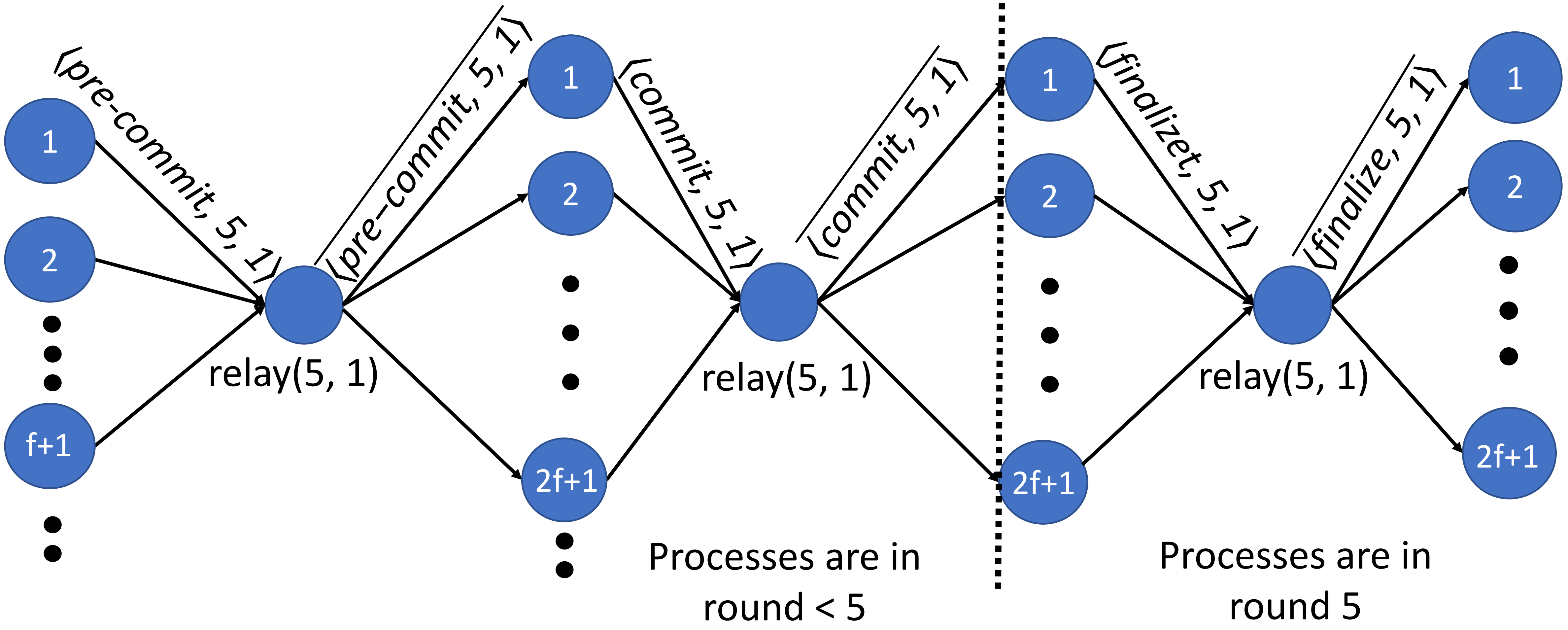
Byzantine Relays Can Split the Good Guys



First relay: 

- Solved by adding another protocol phase - finalize

Message Flow – Synchronize in Round 5



Round Synchronization Summary

- Formalize RS abstraction
- Byzantine RS with
  - Expected linear word complexity
  - Expected constant latency
- The missing ingredient for Byzantine SMR with expected linear word complexity
  - Per decision
  - HotStuff, LibraBFT

Conclusion

Sub-quadratic BA in two flavors:

1. Asynchronous, binary BA WHP
2. Eventually synchronous, multi-value SMR

Thank you!

Yes, it will scale!