CS251 Fall 2023

(cs251.stanford.edu)

Incentives and Accountability in Consensus: Proof-of-Stake

Ertem Nusret Tas

Recap: Security for SMR

Let LOG_t^i denote the log learned by a client i at time t.

Then, a **secure** SMR protocol satisfies the following guarantees:

Safety (Consistency): Similar to agreement!

- For any two clients i and j, and times t and s: either $LOG_t^i \preccurlyeq LOG_s^j$ is true or $LOG_s^j \preccurlyeq LOG_t^i$ is true or both (Logs are consistent).

No double spend

Liveness: Similar to validity and termination!

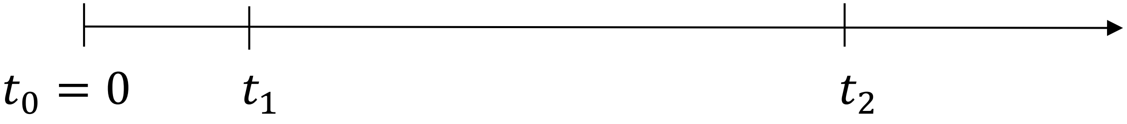
- If a transaction tx is input to an honest replica at some time t, then for all clients i, and times $s \geq t + T_{conf}$: $tx \in LOG_s^i$.

No censorship

Recap: Why is safety important?

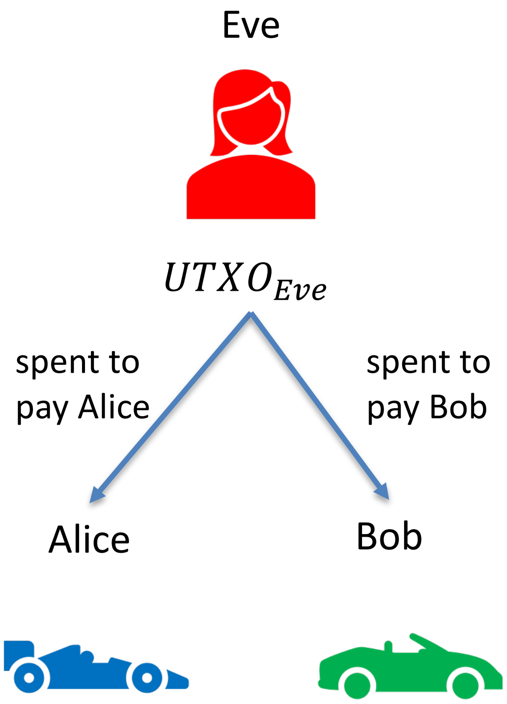
Suppose Eve has a UTXO.

- tx_1: transaction spending Eve's UTXO to pay to car vendor Alice.
- tx_2: transaction spending Eve's UTXO to pay to car vendor Bob.



- Alice's ledger at time t_1 contains tx_1: $LOG_{t_1}^{Alice} =< tx_1 >$
- Alice thinks it received Eve's payment and sends over the car.

- Bob's ledger at time t_2 contains tx_2: $LOG_{t_2}^{Bob} =< tx_2 >$
- Bob thinks it received Eve's payment and sends over the car.



Recap: Why is safety important?

Suppose Eve has a UTXO.

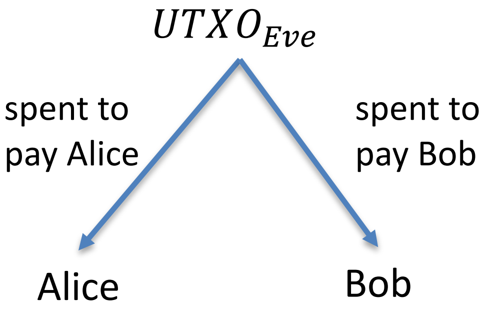
- tx_1: transaction spending Eve's UTXO to pay to car vendor Alice.
- tx_2: transaction spending Eve's UTXO to pay to car vendor Bob.

Eve

$t_0 = 0$ t_1 safety violation t_2

- Alice's ledger at time t_1 contains tx_1: $LOG_{t_1}^{Alice} =< tx_1 >$
- Alice thinks it received Eve's payment and sends over the car.

- Bob's ledger at time t_2 contains tx_2: $LOG_{t_2}^{Bob} =< tx_2 >$
- Bob thinks it received Eve's payment and sends over the car.



Double-spend → inconsistent ledgers → safety violation!

Safety → no double-spend!

Recap of the Last Lecture

- Sybil Attack
  - Adversary impersonates many different nodes to outnumber the honest nodes.
- Sybil Resistance
  - Proof-of-Work, Proof-of-Stake, and Proof-of-Space.
- Bitcoin and Nakamoto Consensus
  - Longest chain rule + k-deep confirmation rule
- Consensus in the Internet Setting
  - Sybil resistance and <u>dynamic availability: liveness under changing participation</u>.
- Security for Bitcoin
  - Nakamoto's private attack and forking
- Incentives in Bitcoin

Incentives in Bitcoin

How does Bitcoin *incentivize* miners to participate in consensus and mine new blocks?

- Block rewards – currently 6.25 Bitcoin – halved every 210,000 blocks – halved ~4 years
- Transaction fees

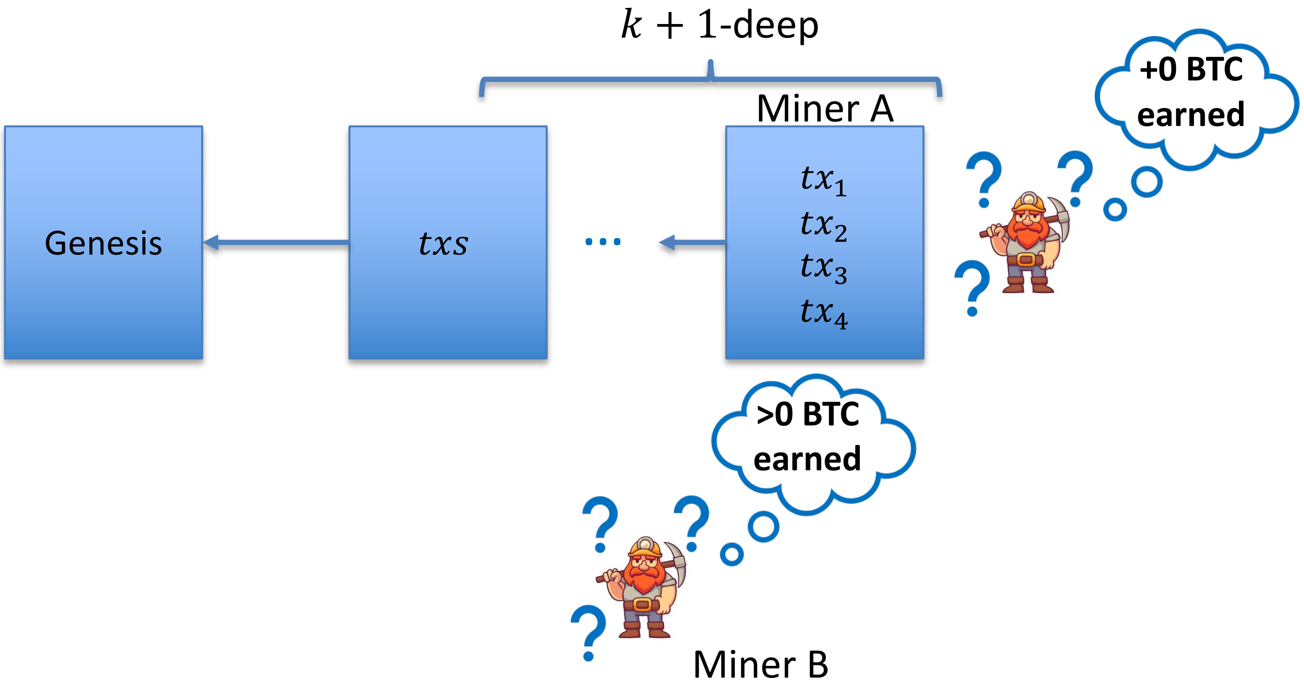
How does a miner capture these rewards?

- The first transaction in a Bitcoin block is called the **coinbase transaction**.
- The coinbase transaction can be created by the miner.
- Miner uses it to collect the block reward and the transaction fees.

Can these *incentives* guarantee *honest* participation?

- Not necessarily!
- Selfish mining attack!
- (See the optional slides if interested in the details.)

Incentives in Bitcoin



Transaction fees:

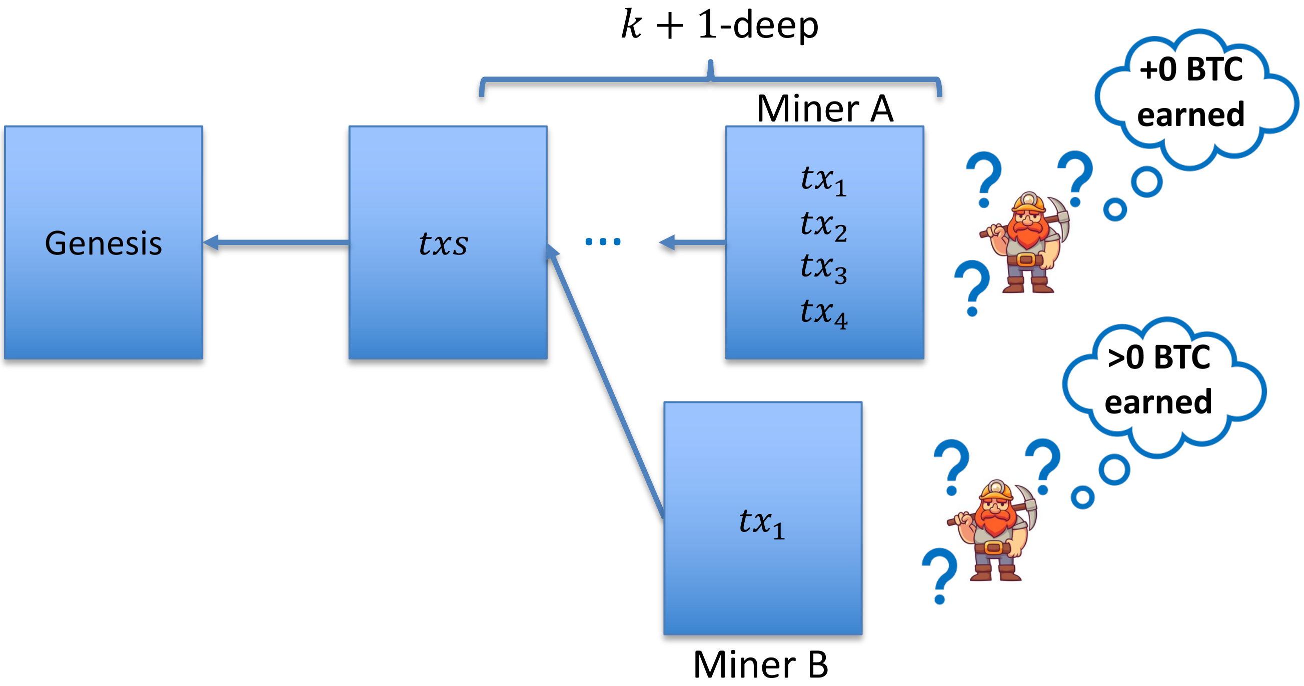
tx_1: 4 BTC

tx_2: 3 BTC

tx_3: 2 BTC

tx_4: 1 BTC

Incentives in Bitcoin



Transaction fees:

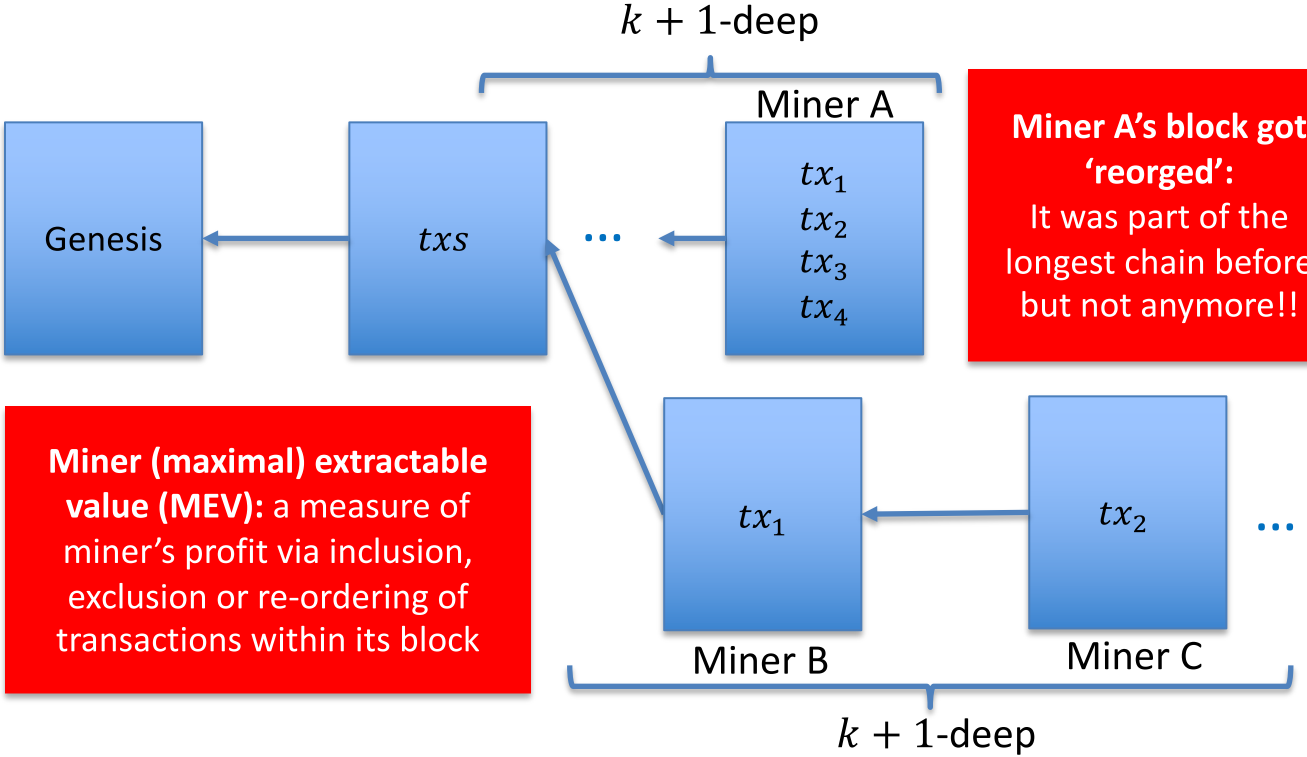
tx_1: 4 BTC

tx_2: 3 BTC

tx_3: 2 BTC

tx_4: 1 BTC

Incentives in Bitcoin



Miner A's block got 'reorged':
It was part of the longest chain before but not anymore!!

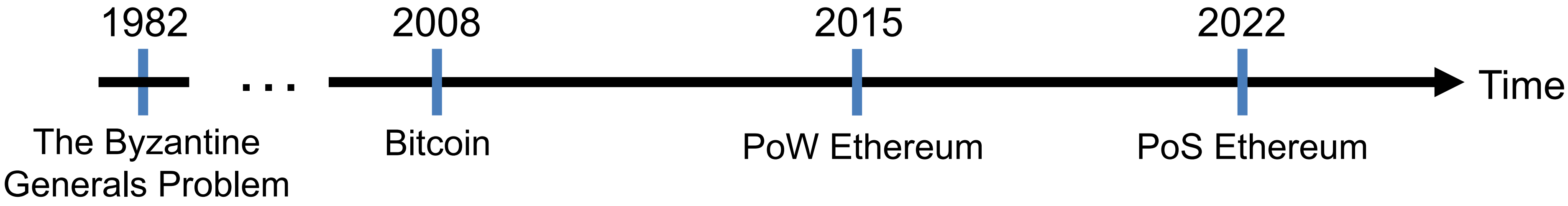
Transaction fees:
tx_1: 4 BTC
tx_2: 3 BTC
tx_3: 2 BTC
tx_4: 1 BTC
Total MEV: 10 BTC

Miner (maximal) extractable value (MEV): a measure of miner's profit via inclusion, exclusion or re-ordering of transactions within its block

Miners have incentive to violate the protocol!
Miners violate the protocol
→ No safety
→ Double-spend!

Need to think about incentives!!

From Bitcoin to Proof-of-Stake



Consensus in the Internet Setting
- Sybil resistance
- Dynamic availability
    - (Liveness under changing part.)

Block rewards (carrot)
- ➢ to incentivize participation!

➢ Consensus in the Internet Setting
- ➢ Sybil resistance
- ➢ Dynamic availability

➢ Block rewards (carrot)

➢ Finality and accountable safety

➢ Slashing (stick)
- ➢ to punish protocol violation!

The Byzantine Generals Problem (1982)
Bitcoin: A Peer-to-Peer Electronic Cash System (2008)
Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform. (2015)
Combining GHOST and Casper (2020)

A few words on Proof-of-Stake (PoS)

In a Proof-of-Stake protocol, nodes <u>lock up</u> (i.e., stake) their coins in the protocol to become <u>eligible to participate in consensus</u>.

The more coins staked by a node…

- Higher the probability that the node is elected as a leader.
- Larger the weight of that node's actions.

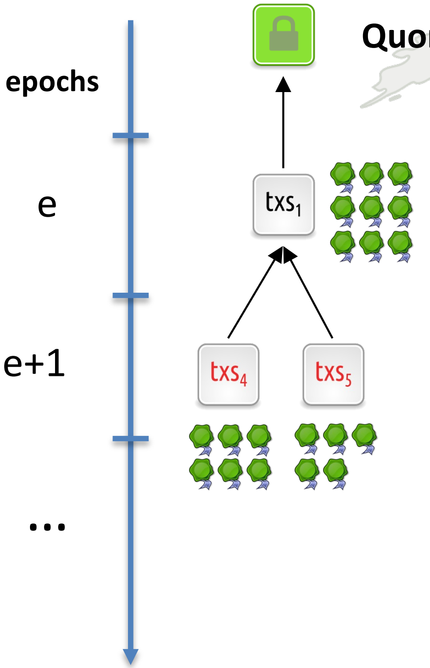
If a node is caught doing an adversarial action (e.g., sending two values), it can be punished by burning its locked coins (stake)! This is called *slashing*.

Thus, in a Proof-of-Stake protocol, nodes can be held ***accountable*** for their actions (unlike in Bitcoin, where nodes do not lock up coins).

A Simple (PBFT-style) PoS Protocol*

Need votes from $\frac{2}{3}$ **of total number of nodes** $(\boldsymbol{n})$ for finality:

Quorum size = $\frac{2}{3}$ of total number of nodes (n)



(9 nodes)

Alice's log: $txs_1\, txs_4$

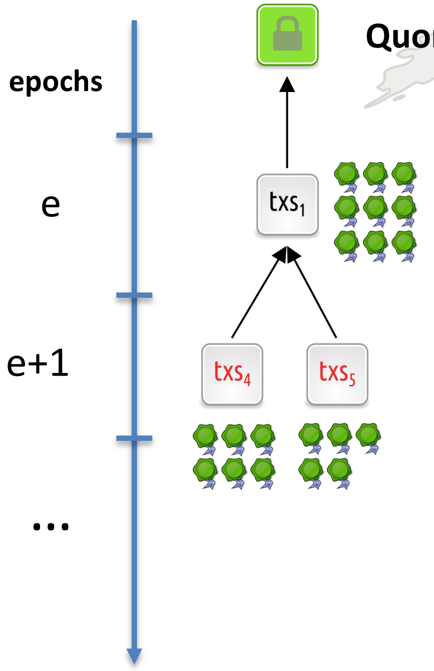
Vote by a node: Signature of the node on the block and the epoch

PBFT: Practical **B**yzantine **F**ault **T**olerance. (1999)

A Simple (PBFT-style) PoS Protocol*

Need votes from $\frac{2}{3}$ **x total number of nodes** $(\boldsymbol{n})$ for finality:

Quorum size = $\frac{2}{3}$ x total number of nodes (n)



Votes for txs_4	Votes for txs_5
$\geq \frac{2}{3}n$	$\geq \frac{2}{3}n$

Alice's log: $txs_1\ txs_4$

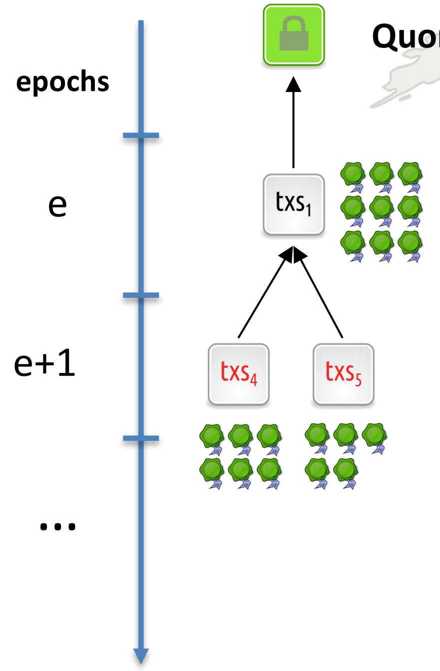
Vote by a node: Signature of the node on the block and the epoch

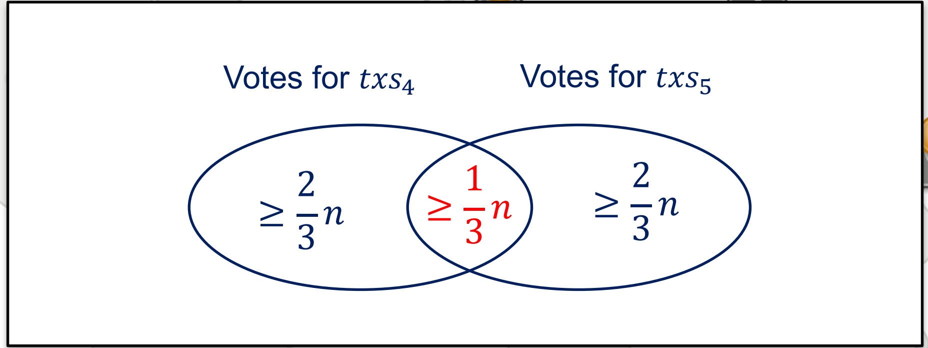
PBFT: Practical **B**yzantine **F**ault **T**olerance. (1999)

A Simple (PBFT-style) PoS Protocol*

Need votes from $\frac{2}{3}$ **x total number of nodes (n)** for finality:

Quorum size = $\frac{2}{3}$ x total number of nodes (n)





Alice's log: $txs_1\ txs_4$

$\geq \frac{1}{3}n$ of nodes must have voted twice (once for txs_4 and once for txs_5)

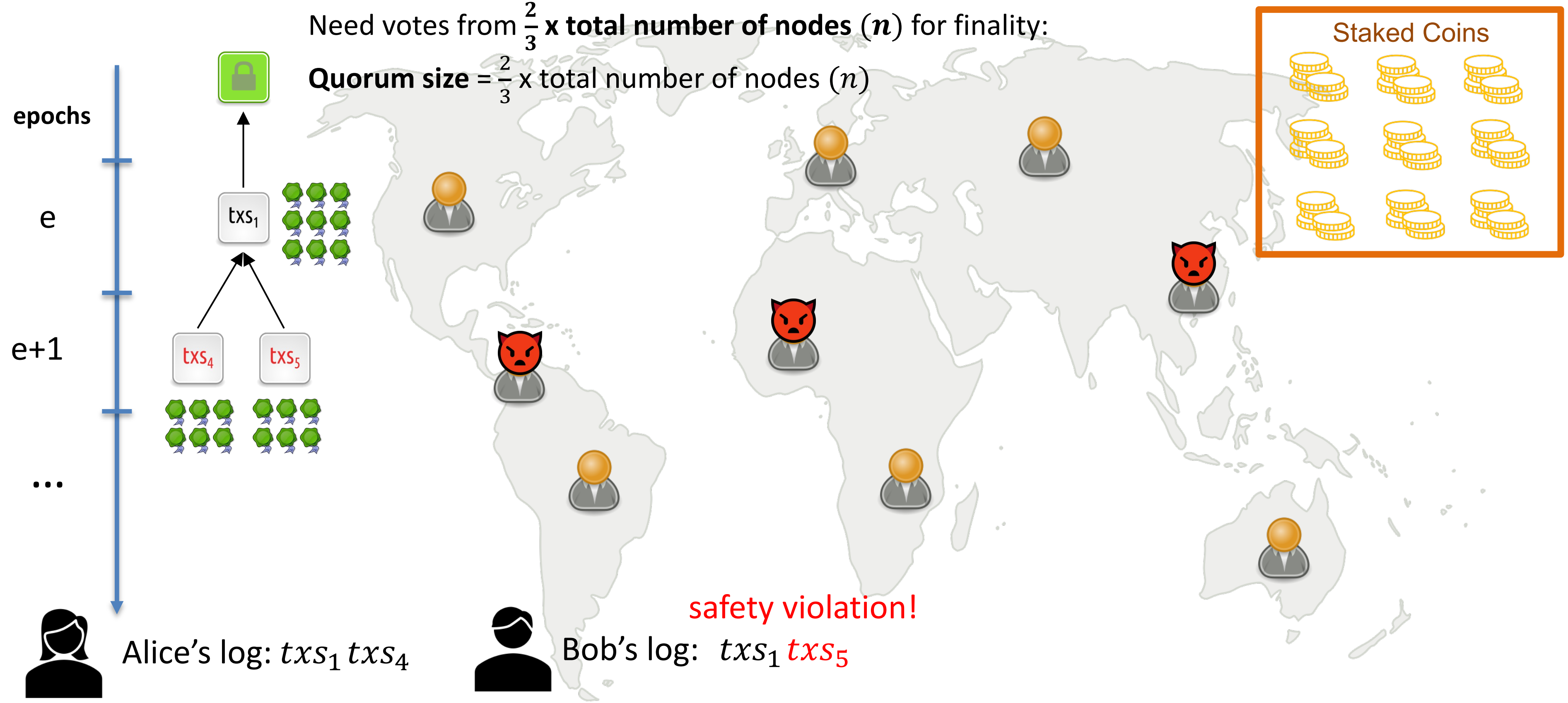
Therefore, $\geq \frac{1}{3}n$ nodes are adversarial

Safety when # adversarial nodes $< \frac{1}{3}$ x total number of nodes

A Simple (PBFT-style) PoS Protocol*

Need votes from $\frac{2}{3}$ **x total number of nodes** $(\boldsymbol{n})$ for finality:

Quorum size = $\frac{2}{3}$ x total number of nodes (n)



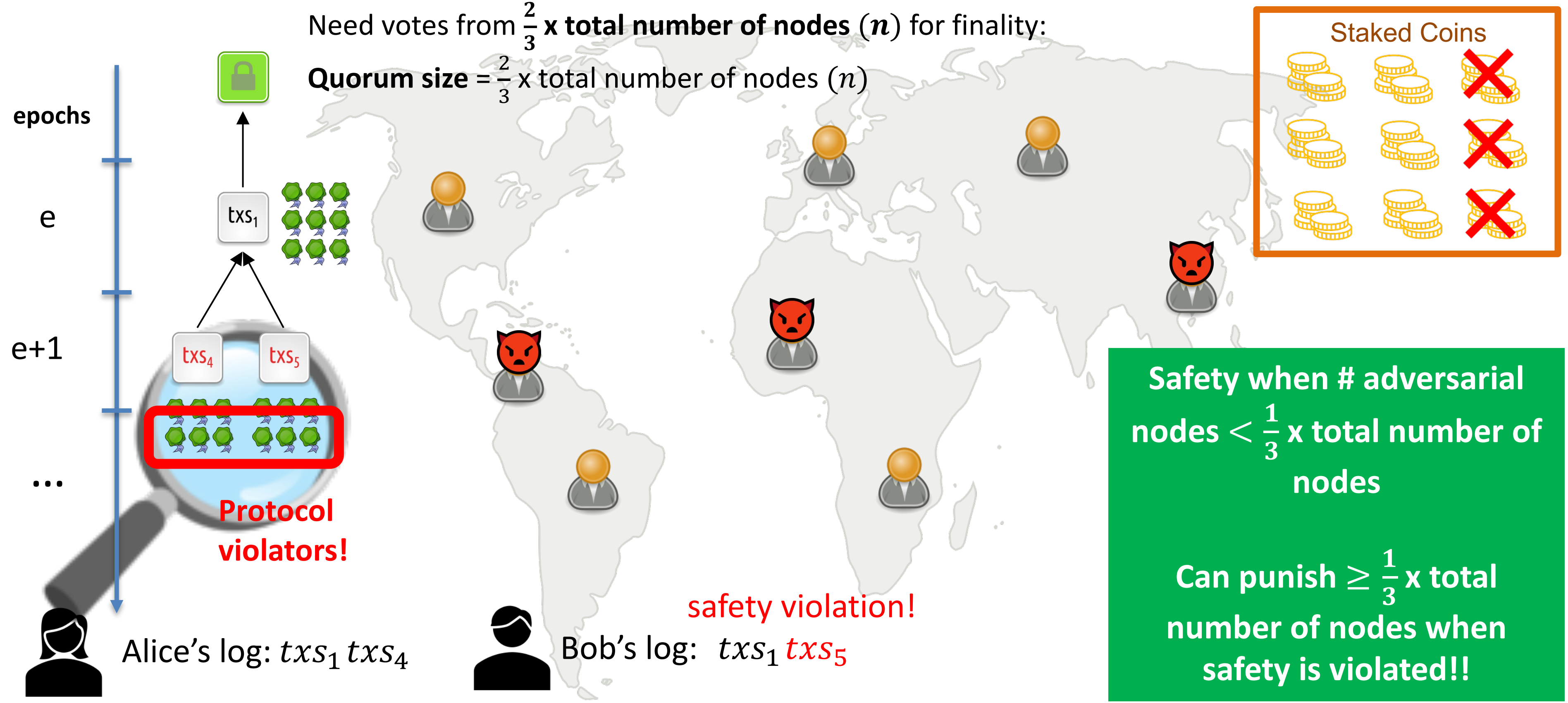
Alice's log: $txs_1\ txs_4$

Bob's log: $txs_1\ txs_5$

A Simple (PBFT-style) PoS Protocol*

Need votes from $\frac{2}{3}$ **x total number of nodes** $(\boldsymbol{n})$ for finality:

Quorum size = $\frac{2}{3}$ x total number of nodes (n)



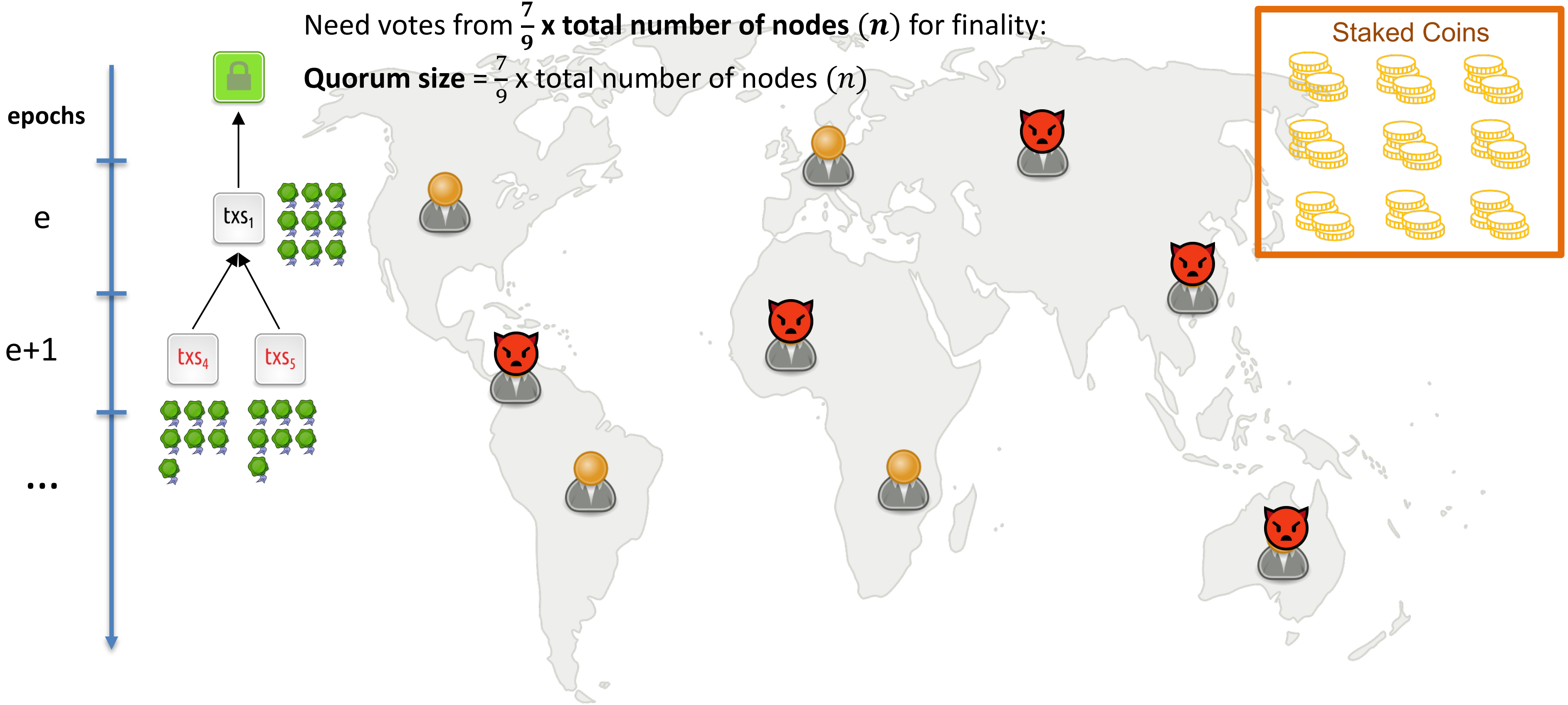
Safety when # adversarial nodes $< \frac{1}{3}$ x total number of nodes

Can punish $\geq \frac{1}{3}$ x total number of nodes when safety is violated!!

A Simple (PBFT-style) PoS Protocol*

Need votes from $\frac{7}{9}$ **x total number of nodes** (n) for finality:

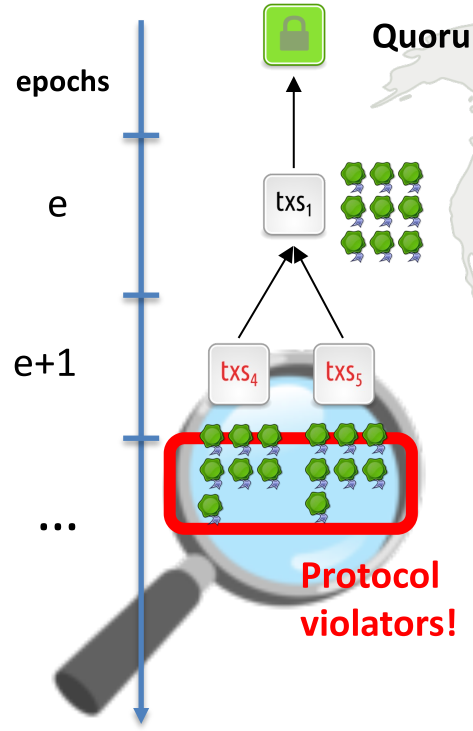
Quorum size = $\frac{7}{9}$ x total number of nodes (n)



A Simple (PBFT-style) PoS Protocol*

Need votes from $\frac{7}{9}$ **x total number of nodes** ($\boldsymbol{n}$) for finality:

Quorum size = $\frac{7}{9}$ x total number of nodes (n)



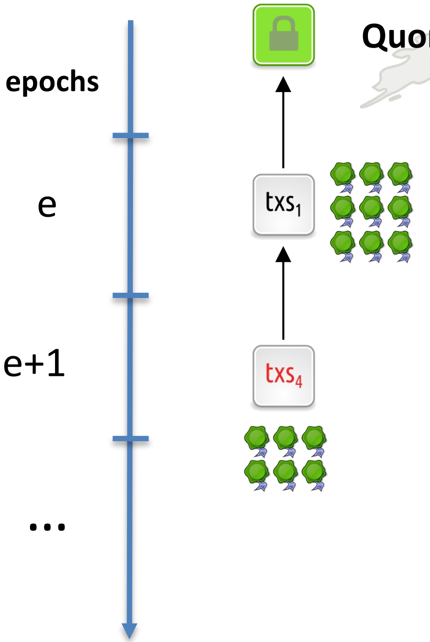
Safety when # adversarial nodes $< \frac{5}{9}$ x total number of nodes

Can punish $\geq \frac{5}{9}$ x total number of nodes when safety is violated!!

A Simple (PBFT-style) PoS Protocol*

Need votes from $\frac{2}{3}$ **x total number of nodes** $(\boldsymbol{n})$ for finality:

Quorum size = $\frac{2}{3}$ x total number of nodes (n)



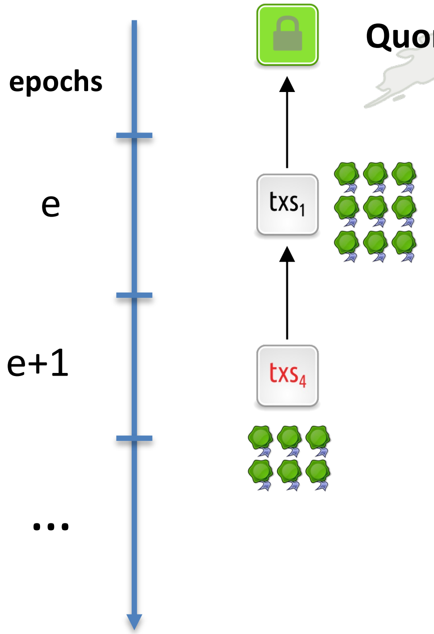
Alice's log: txs_1

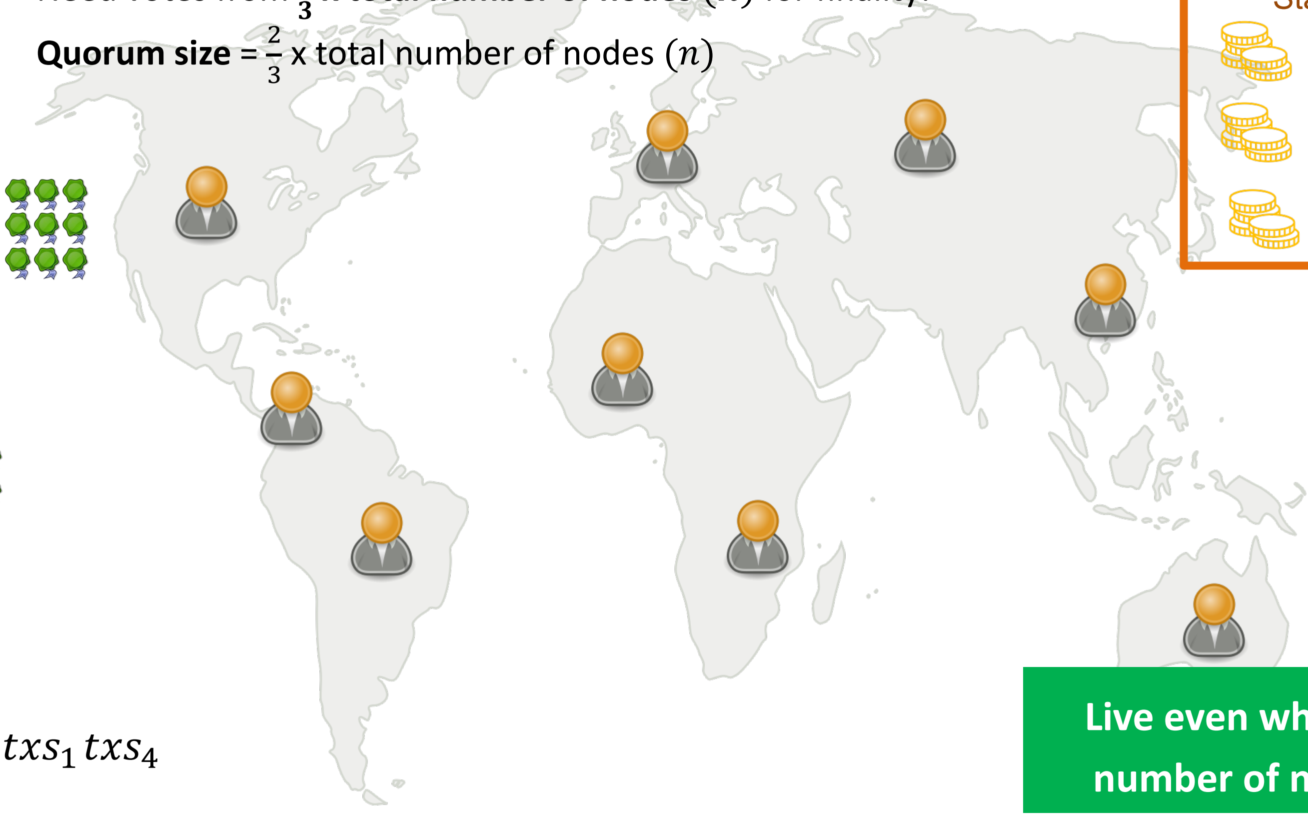
Not live when $\frac{1}{3}$ of total number of nodes crash

A Simple (PBFT-style) PoS Protocol*

Need votes from $\frac{2}{3}$ **x total number of nodes** ($\boldsymbol{n}$) for finality:

Quorum size = $\frac{2}{3}$ x total number of nodes (n)





Alice's log: $txs_1\, txs_4$

Live even when $\frac{1}{3}$ x total number of nodes crash

A Simple (PBFT-style) PoS Protocol*

- **Sybil resistance mechanism** determines how to select the nodes that are eligible to participate in consensus and propose/vote for transactions/blocks.
- **Consensus protocol** specifies the instructions for honest nodes so that given a set of eligible nodes with sufficiently many being honest, safety and liveness are satisfied.

Sybil resistance mechanism: Consensus protocol (SMR):	Proof-of-Work	Proof-of-Stake
Nakamoto consensus (longest chain) satisfies dynamic availability	Bitcoin PoW Ethereum	Ouroboros
PBFT-style (with votes) satisfies finality and accountable safety	??	PoS Ethereum* Simple PBFT-style PoS protocol

Ouroboros Genesis: Composable Proof-of-Stake Blockchains with Dynamic Availability. (2019)

Accountable Safety

In a protocol with resilience of *n/3:*

- The protocol is secure (safe & live) if there are less than *n/3* adversarial nodes.
- **Example:** The simple proof-of-stake protocol.

In a protocol with *accountable safety resilience* of *n/3:*

- The protocol is secure if there are less than *n/3* adversarial nodes.
- If there is ever a safety violation, all observers of the protocol can provably identify (i.e., catch) *at least n/3* adversarial nodes as protocol violators.
- No honest node is ever identified (no false accusation).
- **Examples:** The simple proof-of-stake protocol , PBFT, Tendermint, HotStuff …

Casper the Friendly Finality Gadget. (2017)
BFT Protocol Forensics (2021)

Accountable Safety

Accountable safety is a stronger notion than just security.

	0 – $n/3$	$n/3$ – $2n/3$	$2n/3$ – 1
Resilience of n/3	Safety & Liveness ☺	No Safety or Liveness ☹	No Safety or Liveness ☹
Accountable safety resilience of n/3	Safety & Liveness ☺	• No liveness ☹ • If safety is violated, catch and punish adversarial nodes ☺	

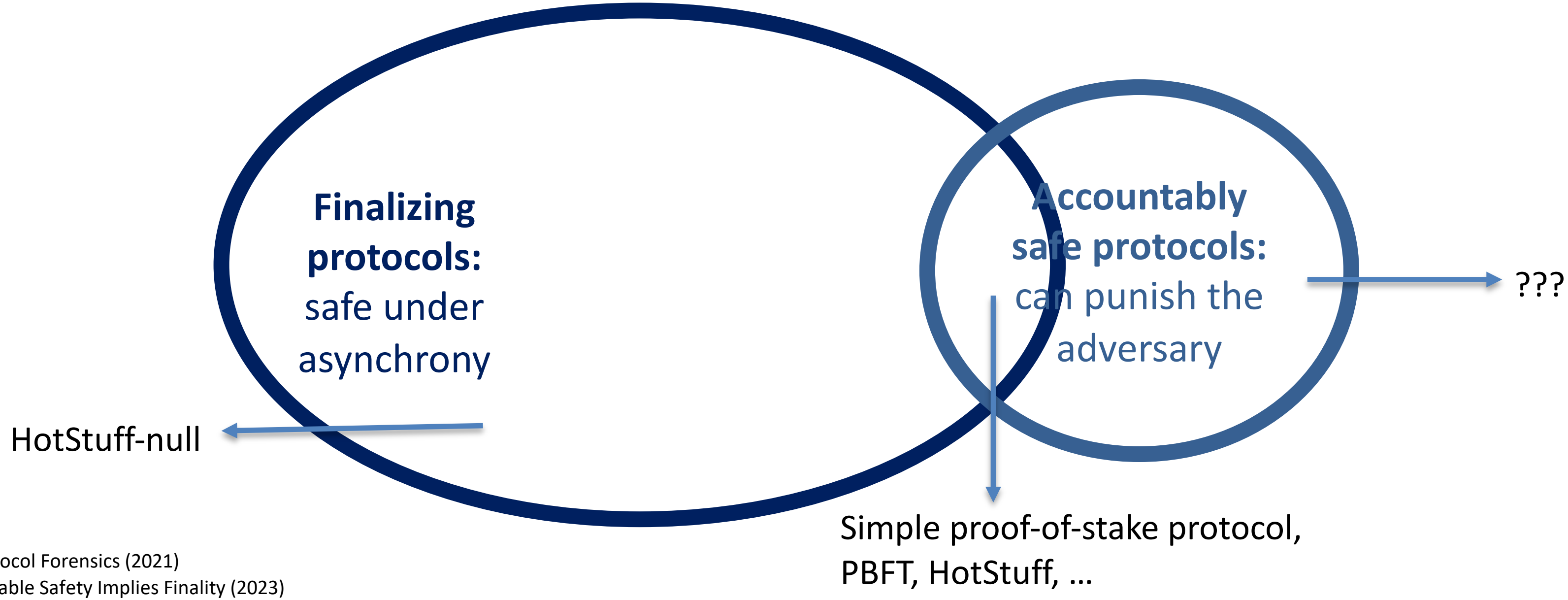
Finality

- We say that a protocol provides *finality* with resilience $\frac{n}{3}$ if it preserves safety during periods of asynchrony, when there are less than $\frac{n}{3}$ adversarial nodes.
  - **Recall:** under asynchrony, messages can be delayed *arbitrarily* for a finite time.
  - **Example:** The simple proof-of-stake protocol, PBFT, Tendermint, HotStuff …
- Interestingly, in *most* protocol providing *finality*, transactions can be *finalized* much faster than they can be *confirmed* in Bitcoin.
  - No need to wait for k=6 blocks (1 hour)!

Accountability implies Finality

Accountability implies Finality:

Accountable safety (with resilience $\frac{n}{3}$) implies **finality** (with resilience $\frac{n}{3}$).

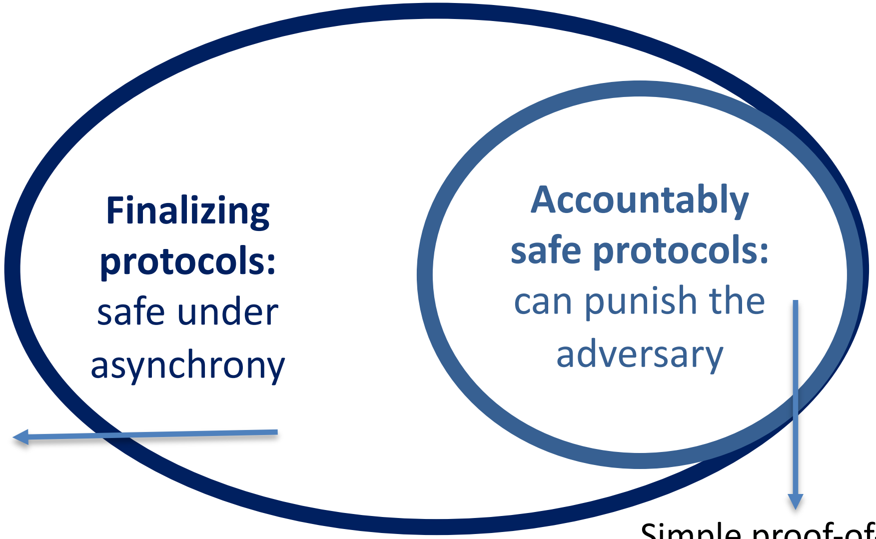


BFT Protocol Forensics (2021)
Accountable Safety Implies Finality (2023)

Accountability implies Finality

Accountability implies Finality:

Accountable safety (with resilience $\frac{n}{3}$) implies **finality** (with resilience $\frac{n}{3}$).



(Accountable safety:) if the protocol can punish at least $\frac{n}{3}$ adv. nodes after a safety violation (and is safe when there are less than $\frac{n}{3}$ adv. nodes),

Then **(Finality:)** it must be safe when there are less than $\frac{n}{3}$ adv. nodes even under asynchrony.

BFT Protocol Forensics (2021)
Accountable Safety Implies Finality (2023)

Holy Grail of Internet Scale Consensus

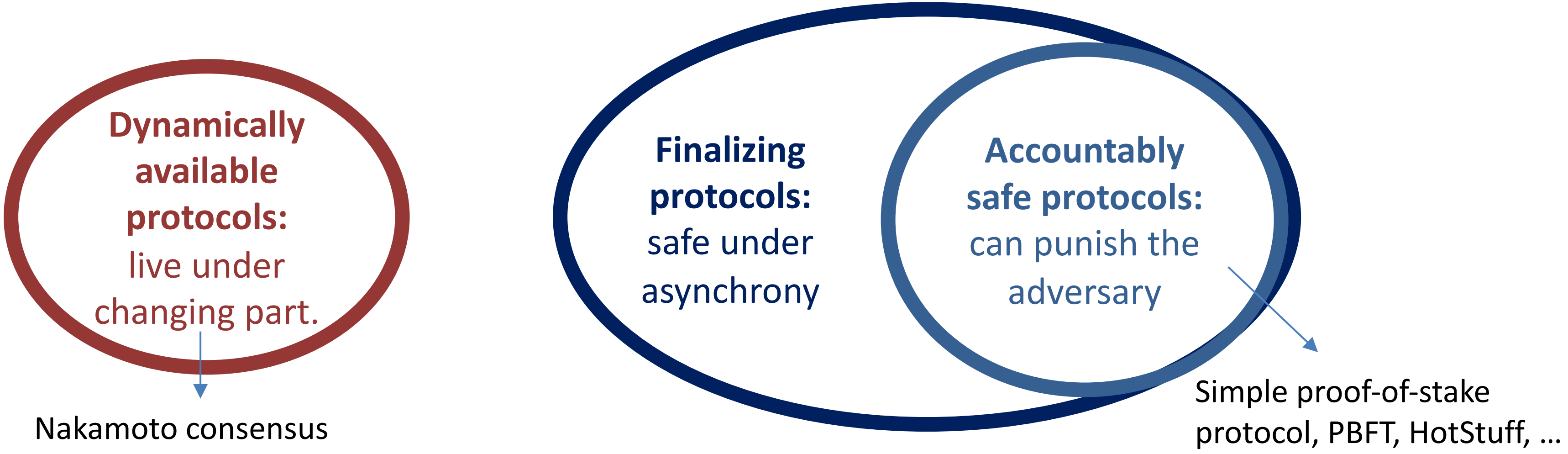
- We want Sybil resistance: Proof-of-Work or Proof-of-Stake…
- We want dynamic availability so that…
  - Transactions continue to be confirmed and processed even when there is low participation.
  - Satisfied by Nakamoto consensus.
- We want finality and **accountable safety** so that…
  - Finality: There cannot be safety violations (double-spends) during asynchrony.
  - **Accountable safety:** Nodes can be held accountable for their actions.
  - Satisfied by our simple proof-of-stake protocol, PBFT, HotStuff, …
- Let’s focus on having dynamic availability and finality for now…

Holy Grail of Internet Scale Consensus

Is there a SMR protocol that provides both dynamic availability and finality with any resilience?

No: Blockchain CAP Theorem!!

CAP: Consistency, **A**vailability, **P**artition tolerance

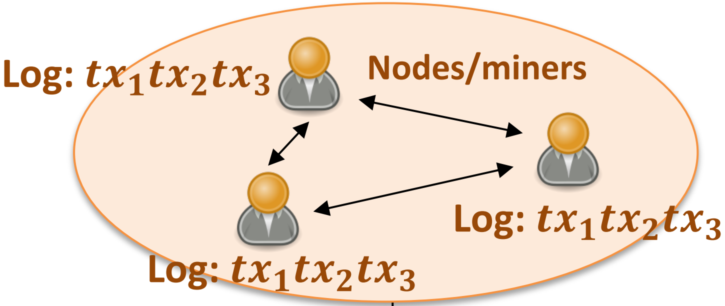


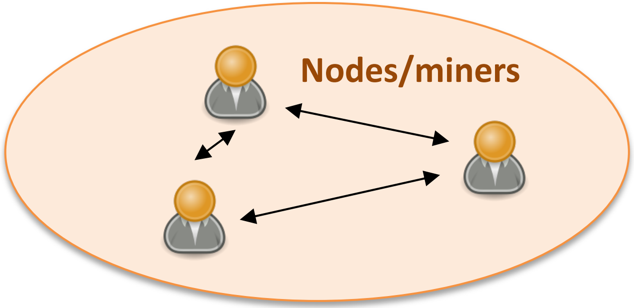
Resource Pools and the CAP Theorem (2020)

Blockchain CAP Theorem

For contradiction, suppose our SMR protocol has both dynamic availability and finality.

World 1





"I didn't hear from the other nodes; they are probably offline."

Correct log: $tx_1 tx_2 tx_3$

Client: Alice

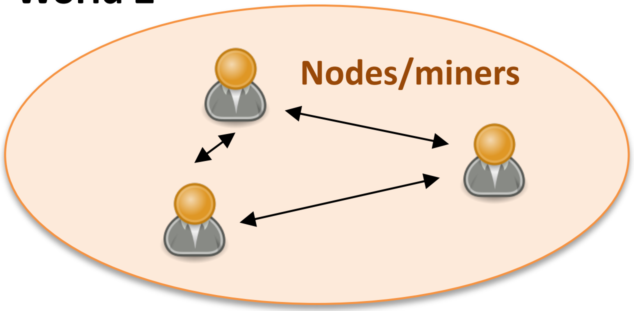
Log learned by Alice: $tx_1 tx_2 tx_3$

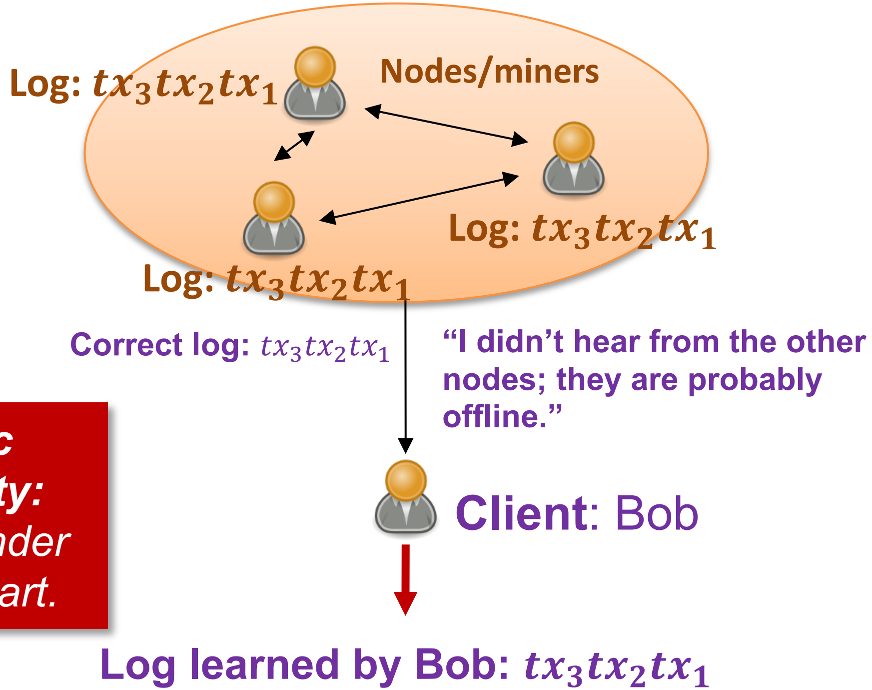
Dynamic Availability:
Liveness under changing part.

Blockchain CAP Theorem

For contradiction, suppose our SMR protocol has both dynamic availability and finality.

World 2



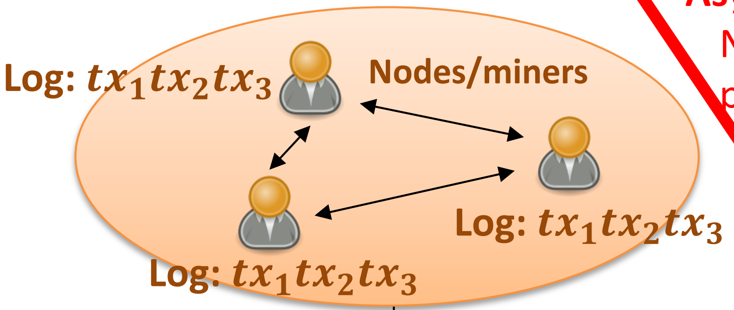


Dynamic Availability:
Liveness under changing part.

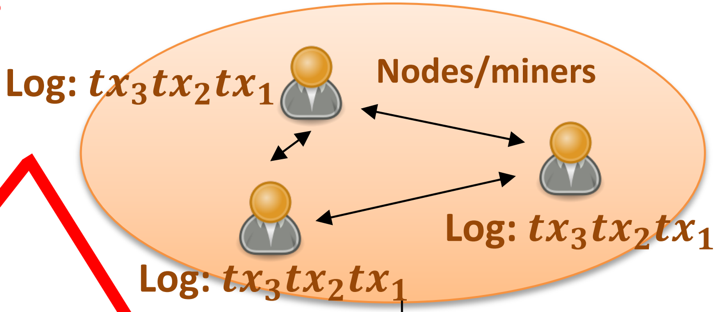
Blockchain CAP Theorem

For contradiction, suppose our SMR protocol has both dynamic availability and finality.

World 3



Asynchrony:
Network partition



"I didn't hear from the other nodes; they are probably offline. **I am in world 1.**"

Correct log: $tx_1tx_2tx_3$

Correct log: $tx_3tx_2tx_1$

"I didn't hear from the other nodes; they are probably offline. **I am in world 2.**"

Client: Alice

Safety violation!
No safety under asynchrony!
No finality!

Client: Bob

Log learned by Alice: $tx_1tx_2tx_3$

Log learned by Bob: $tx_3tx_2tx_1$

Resolution: Nested Ledgers/Chains

Single chain: tx_1, tx_2, tx_3, …

- Finality: Safe under asynchrony
- Dynamic availability: Live under changing participation

Impossible! Due to the CAP Theorem!

Accountable finalized chain

- **Prefix property:** Prefix of the available chain.
- Accountably safe under asynchrony.
- Live once the network becomes synchronous and if enough nodes are online.
- Not live under low participation.

Available chain

- Safe and live under synchrony and changing participation.
- Not safe under asynchrony.

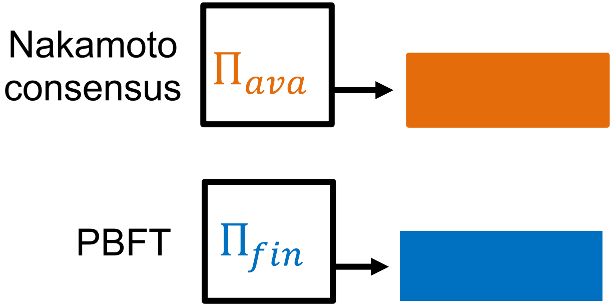
Ebb-and-Flow Protocols: A Resolution of the Availability-Finality Dilemma (2020)

Resolution: Nested Ledgers/Chains

Accountable finalized chain	Available chain

- **Prefix property:** Prefix of the available chain.
- Accountably safe under asynchrony.
- Live once the network becomes synchronous and if enough nodes are online.
- Not live under small participation.

- Safe and live under synchrony and dynamic participation.
- Not safe under asynchrony.



Safety-favoring client: trusts acc. finalized chain

Liveness-favoring client: trusts available chain

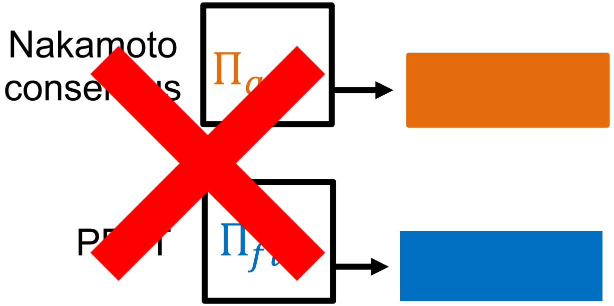
Client chooses!

Can interact with each other thanks to the prefix property!!

Ebb-and-Flow Protocols: A Resolution of the Availability-Finality Dilemma (2020)

Resolution: Nested Ledgers/Chains

Accountable finalized chain	Available chain
• **Prefix property:** Prefix of the available chain. • <u>Accountably safe</u> under <u>asynchrony</u>. • Live once the network becomes synchronous and if enough nodes are online. • Not live under small participation.	• Safe and live under synchrony and dynamic participation. • Not safe under asynchrony.



Ledgers can be inconsistent!
No prefix property!

Safety-favoring client: trusts acc. finalized chain

Liveness-favoring client: trusts available chain

Client chooses!

Can interact with each other thanks to the prefix property!!

Ebb-and-Flow Protocols: A Resolution of the Availability-Finality Dilemma (2020)

Resolution: Nested Ledgers/Chains

Safety-favoring client: trusts acc. finalized chain

Liveness-favoring client: trusts available chain

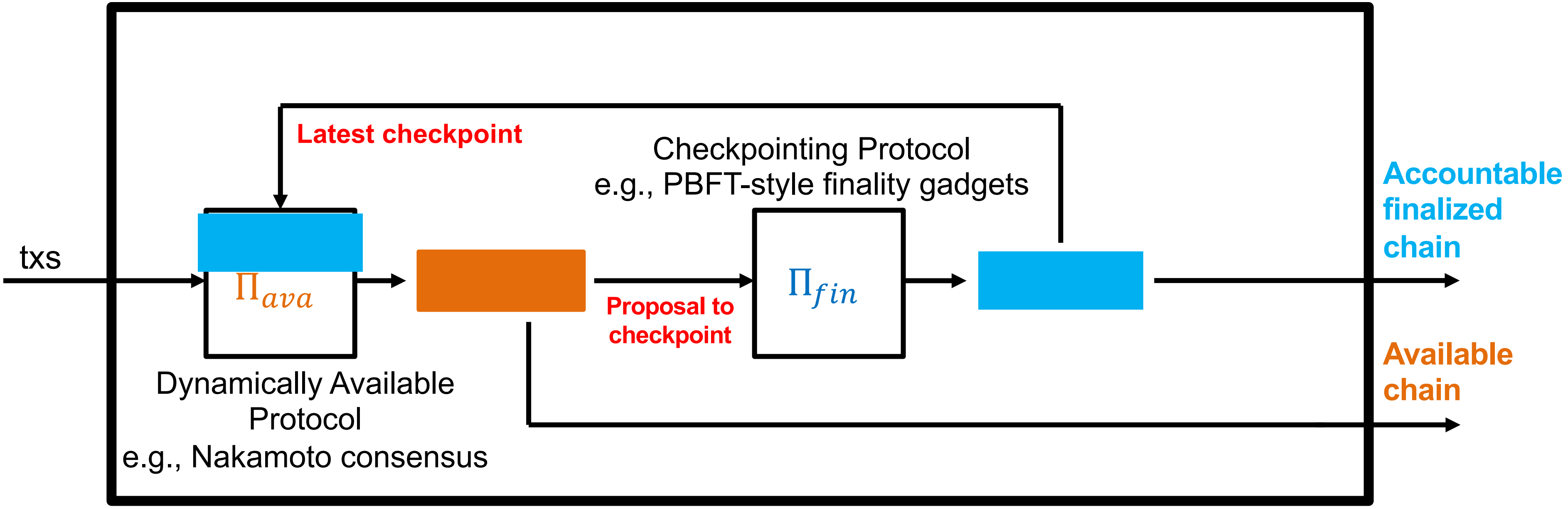
Client chooses!

- When the participation seems low at the weekend, it can either be that participation is actually low due to nodes taking time off or there is in fact a network partition.

- In this case, the boba vendor is willing to follow the available chain and risk a safety violation (and some double spend) due to a partition, since its transactions are of less value. By following the available chain, it can in turn keep selling boba at the weekends. Indeed, most of the time, there will not be a network partition, and participation will be low at the weekends due to nodes taking time off.

- However, the car vendor's transactions have large value, and the car vendor cannot afford even one double spend! Therefore, it will follow the accountable, finalized chain that never has safety violations, but stops when there is low participation, e.g., at the weekends. This is fine since the car vendor has few transactions and can afford to wait the weekend. Indeed, on Monday, the accountable, finalized chain regains its liveness with higher participation.

How to obtain the nested chains?



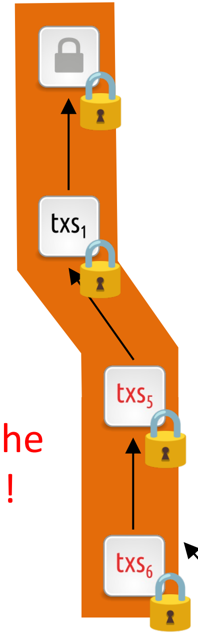
Combining GHOST and Casper. (2020)

How to obtain the nested chains?

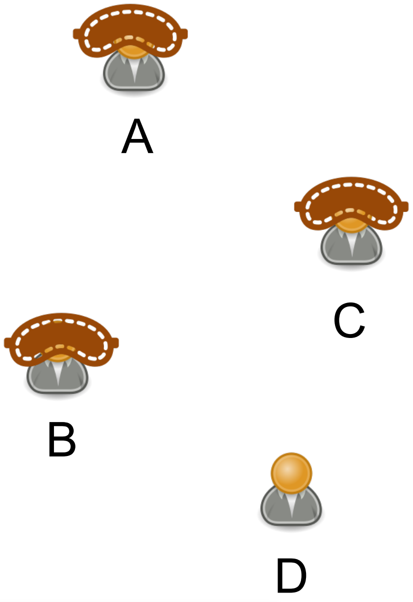
Nested Chains

Orange: available (full) chain

Blue: accountable, final (prefix) chain



Always extend the last checkpoint!!



Checkpointing Protocol

Propose “txs5”

C votes “txs5”

B votes “txs5”

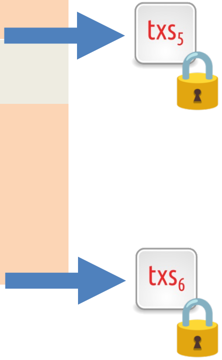
D votes “txs5”

Propose “txs6”

A votes “txs6”

C votes “txs6”

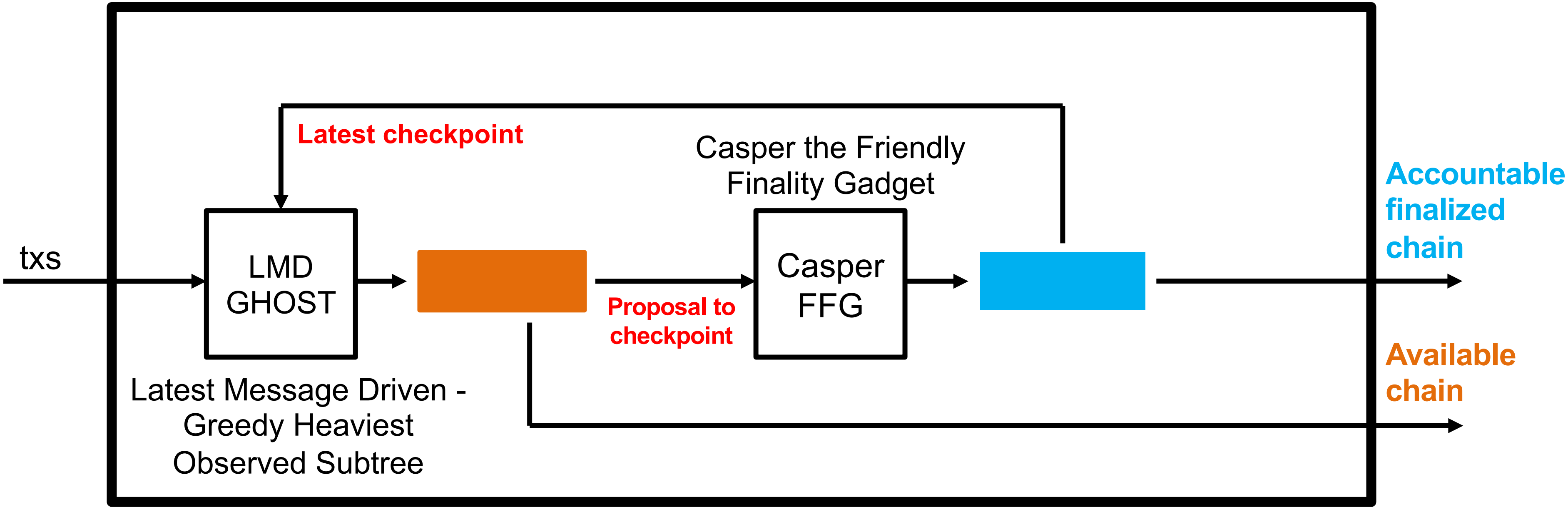
D votes “txs6”



Dynamic Availability: *Longest chain keeps growing.*

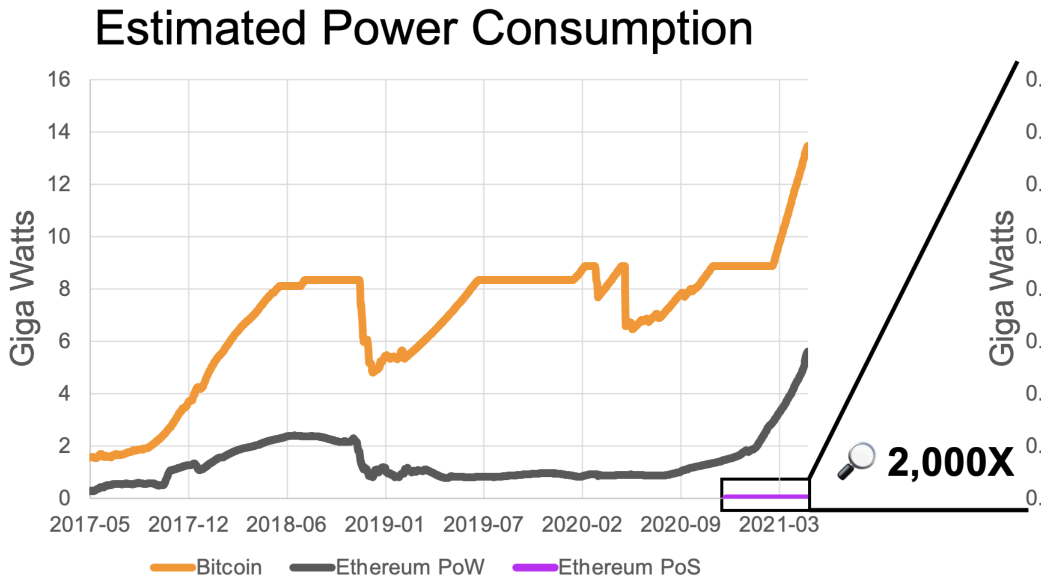
Finality: *Thanks to votes, checkpoints are safe even under asynchrony.*

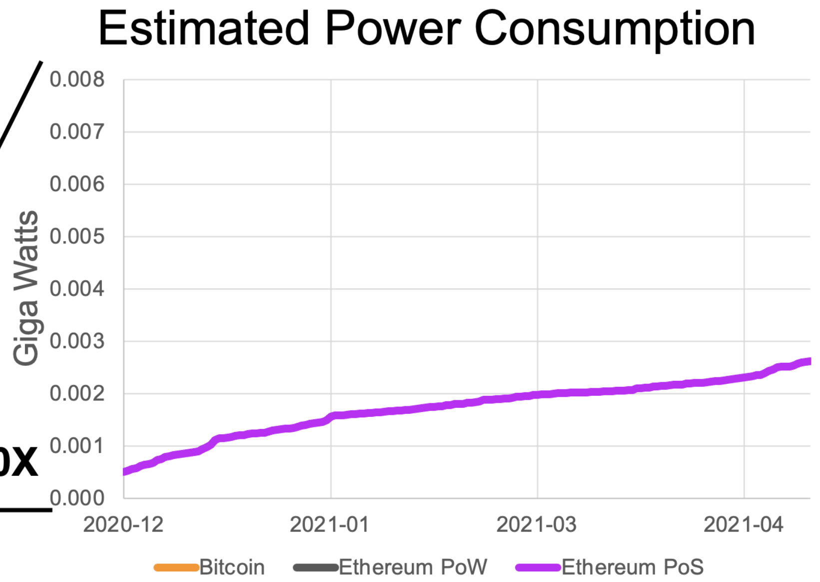
Ethereum



Combining GHOST and Casper. (2020)

A Greener Future for Blockchains?





Bitcoin and Ethereum PoW data taken from Digiconomist

Taken from the article "Ethereum's energy usage will soon decrease by ~99.95%" that appeared at the 'ethereum foundation blog' on May 18th 2021.

END OF LECTURE

Next lecture: interesting scripts,
wallets, and how to manage crypto assets

*A Note on the Simple PoS Protocol

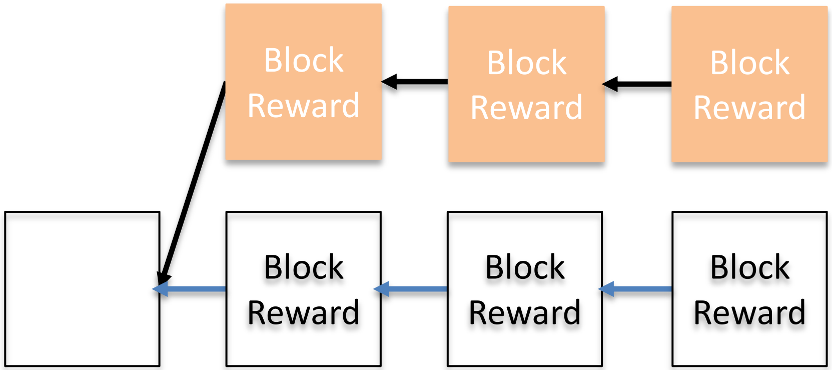
- This protocol is, in fact, **not secure**; because even though it satisfies safety, it does not satisfy liveness:
  - Suppose an adversarial epoch leader proposes two conflicting blocks and shows each block to different halves of the set of nodes.
  - In this case, each block gathers $\frac{1}{2}n$ votes, even though the quorum required for finality is $> \frac{2}{3}n$ votes. None of the blocks get finalized, and the protocol gets stuck.
- Resolving this situation requires a non-trivial improvement of the protocol, and is at the heart of PBFT, a secure SMR protocol, on which this simple protocol was based.
- The purpose of the simple (yet insecure) PoS protocol is to illustrate the core ideas in finalizing and accountably-safe SMR protocols, such as quorum intersection.
- Secure and modern PBFT-style protocols include Tendermint and HotStuff.

Optional Slides

Slides going forward is optional material and investigate the Selfish Mining Attack.

Selfish Mining Attack (Optional)

Attacker keeps its blocks private until sufficiently many honest blocks are mined.
It then publishes the hidden blocks to ‘reorg’ the honest blocks.



Majority is not Enough: Bitcoin Mining is Vulnerable (2013)

Selfish Mining Attack (Optional)

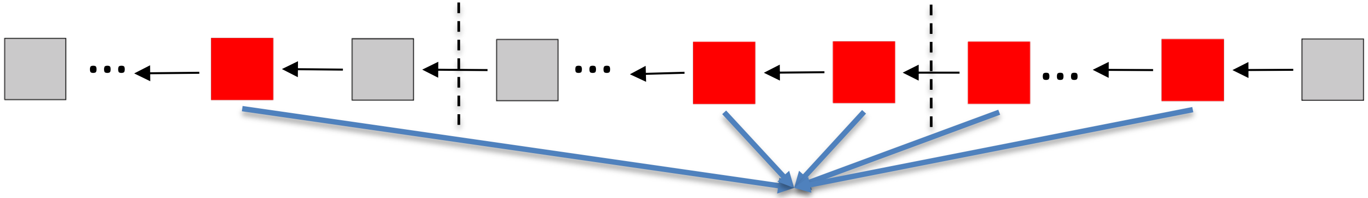
Suppose you hold β fraction of the mining power.

If you behave honestly, mining on the tip of the longest chain in your view and broadcasting your blocks as soon as they are mined…

- You mine $\sim\beta$ fraction of the blocks.
- You earn $\sim\beta$ fraction of the block rewards over Bitcoin's lifetime.

Note that the total amount of block rewards over Bitcoin's lifetime is fixed!

Selfish Mining Attack (Optional)



β **fraction:** adversary's blocks

Total fraction on the longest chain: 1

Remaining $1-\beta$ fraction: honest miners' blocks

Selfish Mining Attack (Optional)

If you do selfish mining…

You kick out $\sim\beta$ fraction of the mined blocks out of the longest chain.

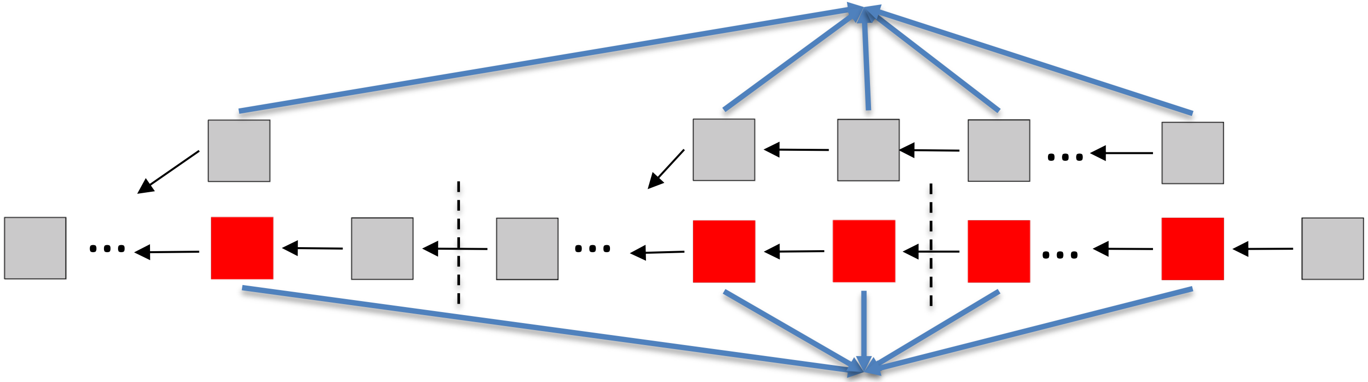
$\sim 1-\beta$ fraction of the mined blocks are in the longest chain.

You have mined $\sim \frac{\beta}{1-\beta}$ of the blocks in the longest chain.

You earn $\sim \frac{\beta}{1-\beta} > \beta$ fraction of the block rewards over Bitcoin's lifetime!

Selfish Mining Attack (Optional)

β **fraction:** honest miners' blocks displaced by the adversary's blocks



β **fraction:** adversary's blocks

Total fraction on the longest chain: $1 - \beta$

Remaining $1 - 2\beta$ **fraction:** honest miners' blocks that were not displaced by the adversary's blocks

Selfish Mining Attack (Optional)

Chain quality (fraction of honest blocks in the longest chain) of Bitcoin $\leq \frac{1-2\beta}{1-\beta}$

Is it possible to make Bitcoin incentive compatible and increase chain quality to β?

Yes!

Examples: Fruitchains (ε-Nash equilibrium), Colordag (ε-sure Nash equilibrium)

Fruitchains: A Fair Blockchain (2017)
Colordag: An Incentive-Compatible Blockchain (2022)