ETHEREUM 2.0 & REVIEW SESSION

Kai Mast

CS639/839

Spring 2023

ANNOUNCEMENTS

- Project 2b out
  - Due 3/31
- Midterm tomorrow at 5:45pm
  - Biochemistry 1120
  - Bring your student ID
- Submit project proposals by the end of the week

TODAY'S AGENDA

- Recap of Consensus
- Ethereum 2.0

- Break?

- Review Session

RECAP: NAKAMOTO CONSENSUS

Random Block Creation

Decide what transactions are (potentially) part of the main chain

Rate-limit how many blocks are created

+

Fork-Choice Rule

Decide between conflicting blocks
Detect invalid blocks

BLOCK CONFIRMATION

- Blocks “confirm” the predecessors
  - Miners/block creators vote implicitly by extending a particular fork

- In Bitcoin/Ethereum 1.0 “depth” of a block determines confirmation certainty

RECAP: PROOF OF STAKE VS. PROOF OF WORK

PoS and PoW are used to **pick a block creator** and to **prevent Sybil attacks**

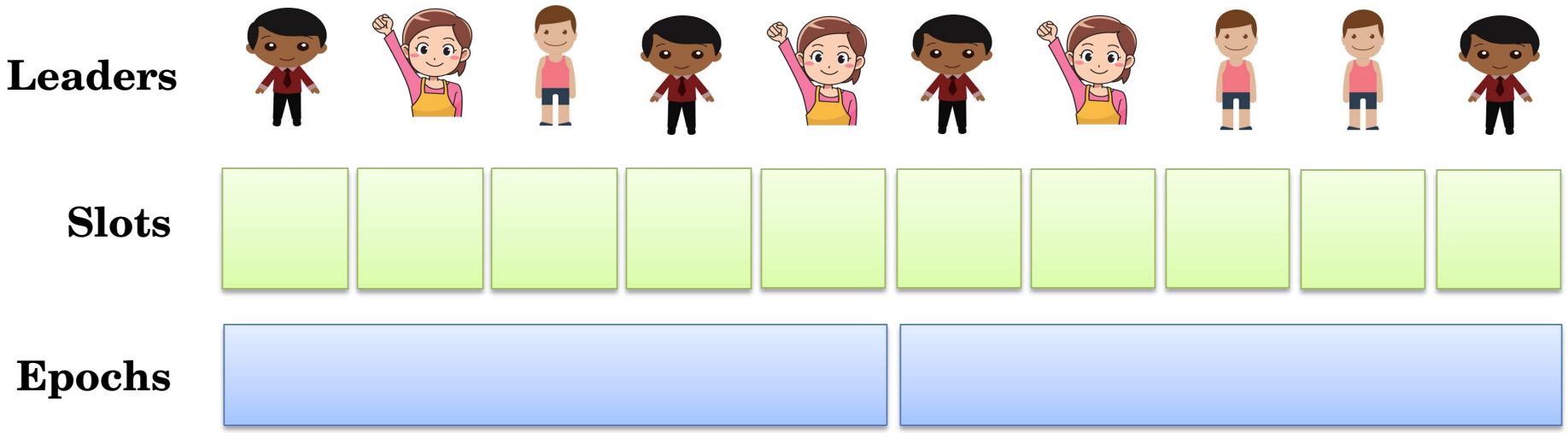
Proof of Work

- Chance of creating a block proportional to mining power
- Fully public, anyone with a CPU/GPU can join

Proof of Stake

- Chance of creating a block or voting power proportional to account balance
- Requires some amount of stake to participate
- Note, Terminology: Miner => Validator/Staker

RECAP: OUROBOROS



- Each slot is of fixed time length
- Each slot has exactly one randomly-selected leader
- Leader schedule is generated at the beginning of an epoch

RECAP: SYNCHRONICITY MODELS

Synchronous

- There is a known finite time bound in which network messages are delivered
- Assumed, e.g., by Bitcoin and Ouroboros 1.0

Partially Synchronous

- Similar to synchronous but there can be periods of time where the network is slower or partitioned
  - During these times we only guarantee safety not liveness
- Assumed, e.g., by Algorand and Ethereum 2.0

Asynchronous

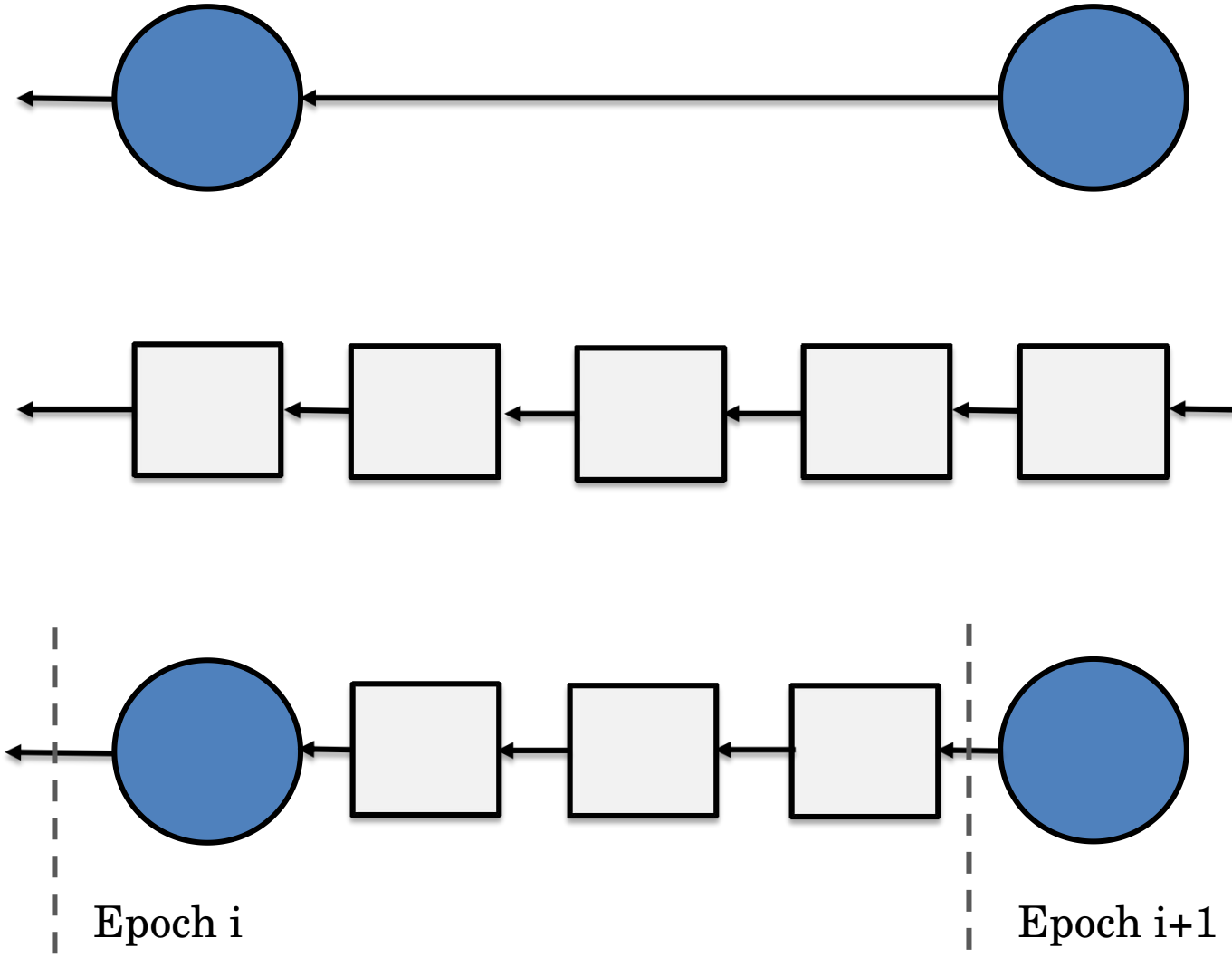
- No known bound on when network messages are delivered
- No protocol we cover in this class assumes this

THE ETHERUM 2.0 CHAIN

Consensus Chain
- Or “Beacon Chain”
- Uses Proof of Stake
- Large Block intervals

Execution Chain
- “Classic” Ethereum without PoW
- Small Block Intervals

Merged Chain
- Blocks are grouped into epochs
- Consensus is achieved on “epoch boundary blocks”
- Block creators are picked similar to Ouroboros



GASPER: THE ETH2.0 CONSENSUS PROTOCOL

Gasper = GHOST + Casper

GHOST: Greediest Heaviest Observed SubTree

- Blocks reference "uncles" as well as their parents
- Reduces the number of forks when using high block intervals
- GHOST is used to pick the longest chain in Gasper

Casper: The "friendly finality gadget" (Casper FFG)

- Provides absolute certainty that a block is accepted by a probabilistic network
- Allows for faster confirmation times
- Finality allows pruning state (we know state is immutable at some point, so competing forks and transaction history is not needed anymore)

STAKE AND SLASHING

Validators lock up stake

- Determines their voting power
- Serves as a deposit

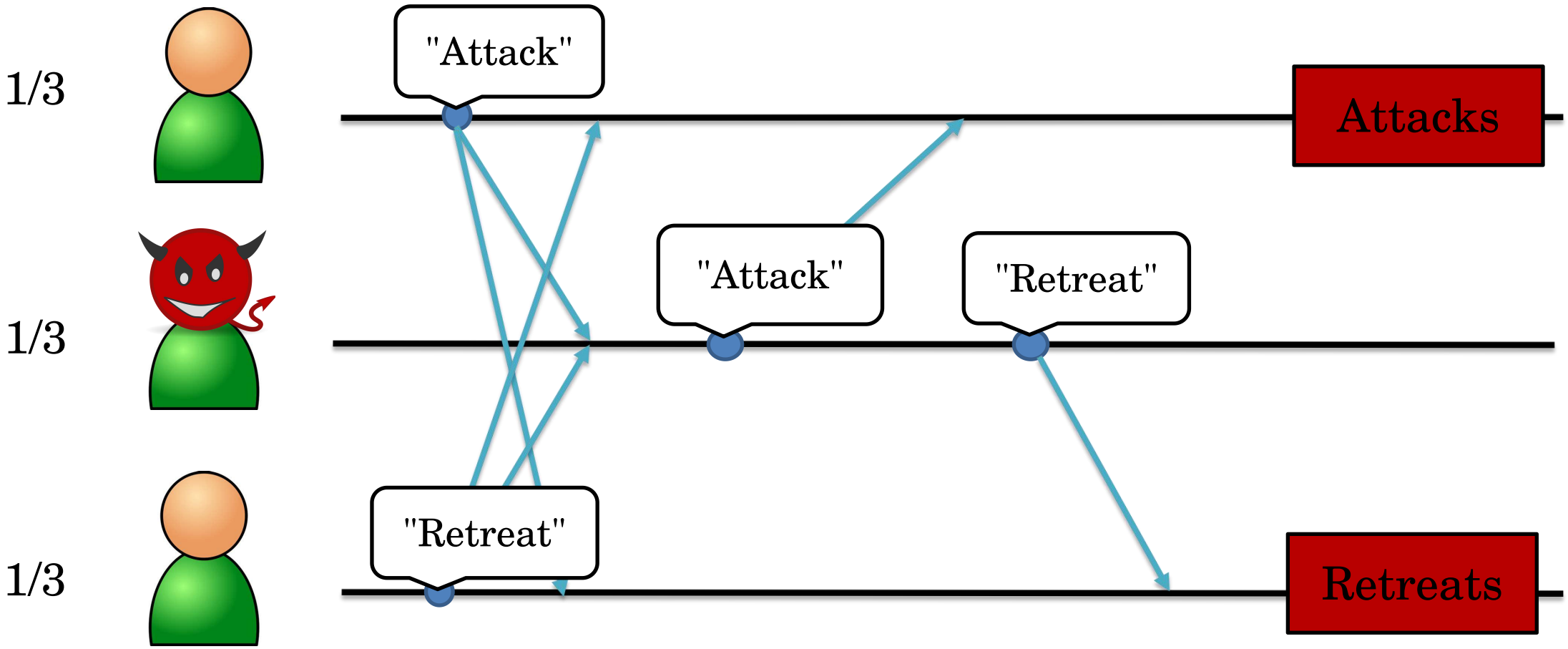
Validators that do not follow the protocol will lose some of their stake ("slashing")

- Only works in protocols where the stake is locked up
  - So, not in Ouroboros or Algorand

For a correct Ethereum 2.0 chain, less than 1/3 of the stake is "slashable"

- At most 1/3 of the stake is controlled by an attacker
- Slightly guarantee than just saying f<1/3
  - Even rational nodes have an incentive behave correctly because of slashing

SUPERMAJORITIES IN BYZANTINE CONSENSUS



Need a majority greater than 2/3 to prevent this!

PIGEONHOLE PRINCIPLE

- Assume at most f malicious voters/validators, where $f < n/3$
- A majority is $2f+1$ and we need at least $3f+1$ total stake
- Even if f nodes vote for conflicting proposals, only one proposal can reach a supermajority

ATTESTATIONS IN ETH2.0

Stakers vote on a block by issuing an attestation

- w.l.o.g., we assume each staker has the same fraction of the stake
- In the real world
  - validates have different stakes (and voting power)
  - stake can vary over time

Attestations are broadcast and eventually included in a block

ATTESTATION STATES OF BLOCK

Proposed

- Block has been created but has not received enough attestations yet

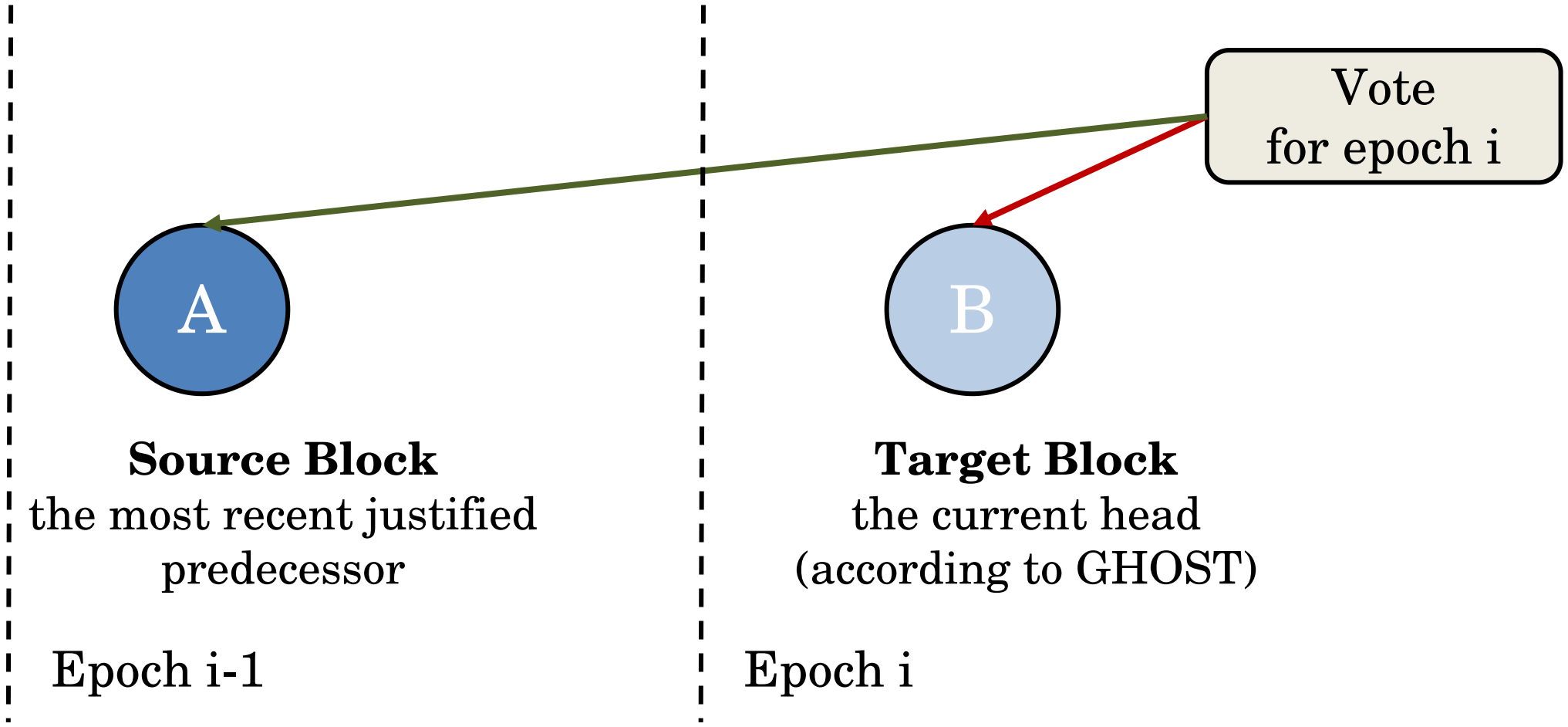
Justified

- A supermajority has approved the block
- But, existence of the “justified”-decision might not be known to everyone

Finalized

- A supermajority has approved the block and also approved its direct successor
- Or one of the blocks ancestors is finalized

ATTESTATIONS IN ETH2.0

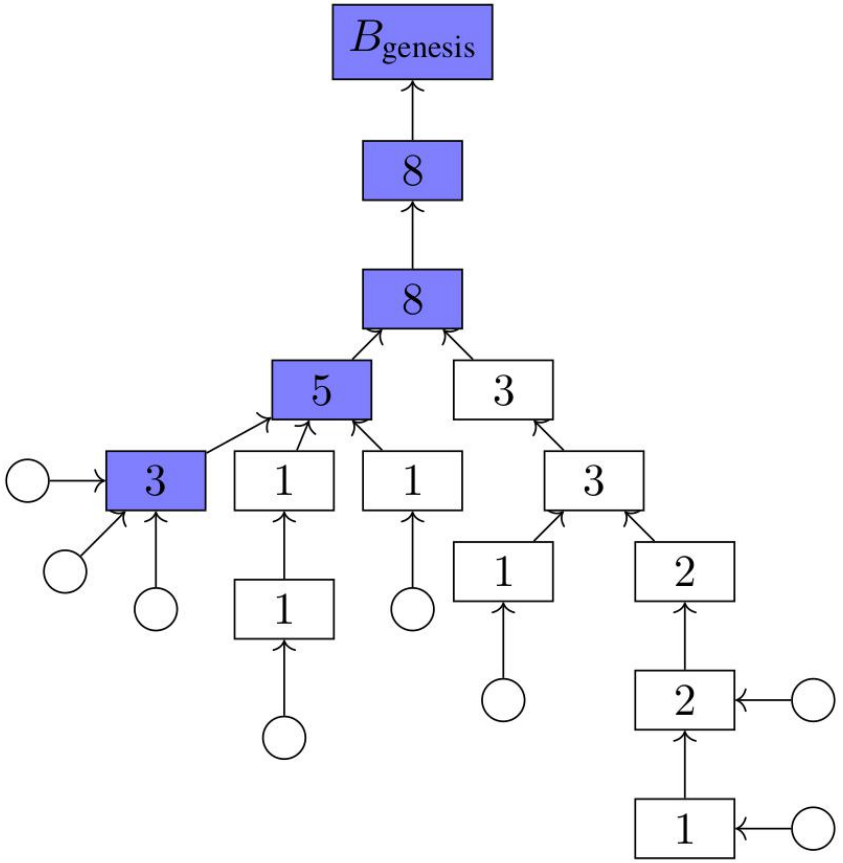


Votes are contained within other blocks of the same epoch (not shown here)

RESOLVING FORKS IN GASPER

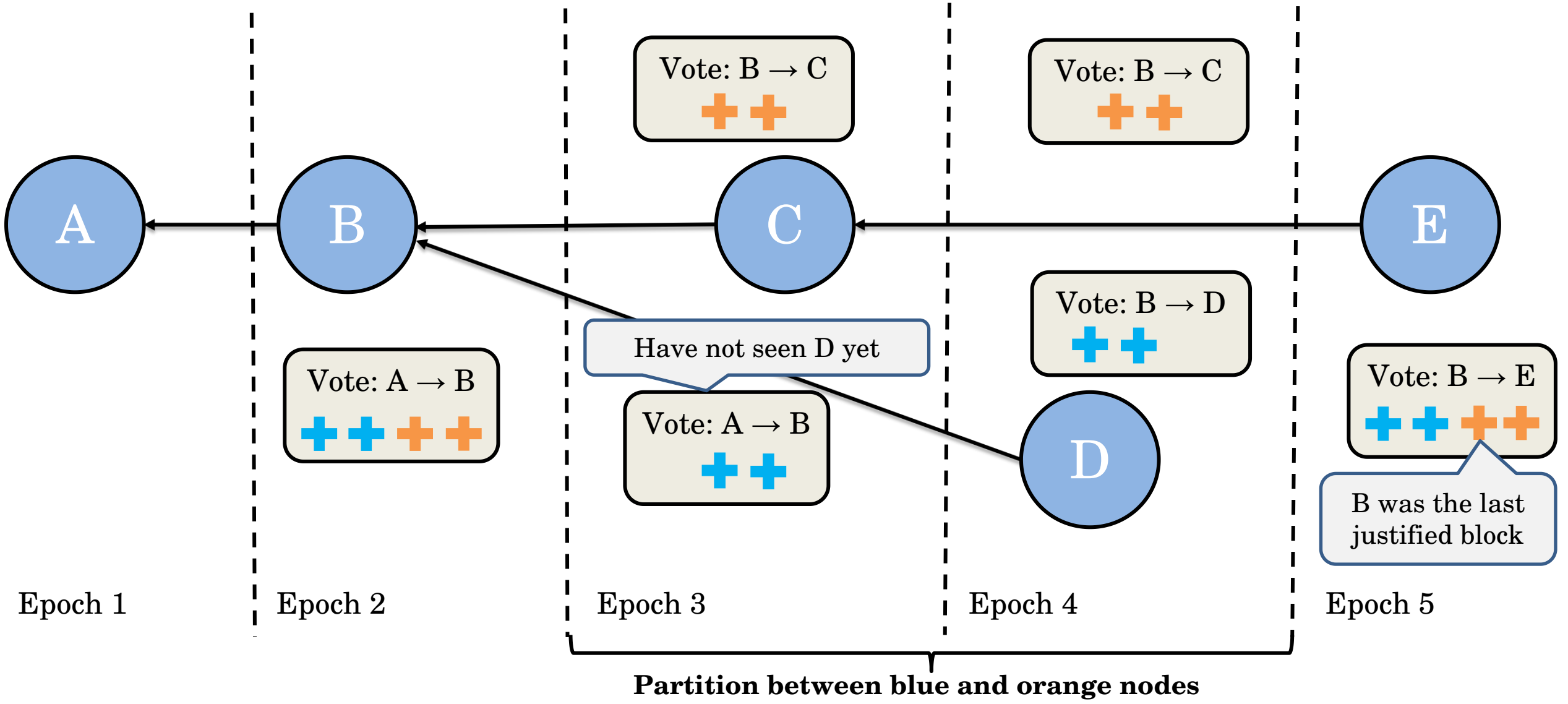
Gasper relies on Last Message Driven GHOST (LMD GHOST) to choose between forks

- GHOST: pick the heaviest branch of the tree
- Last Message Driven
  - A blocks weight is the sum of stake of all its attestations and attestations to its descendants
  - Only the most recently seen attestations count
- If there is a tie, all nodes deterministically break the tie based on the block hashes



- Numbers are weights
- Circles are last-seen attestations

NETWORK PARTITIONS IN ETH2.0: EXAMPLE

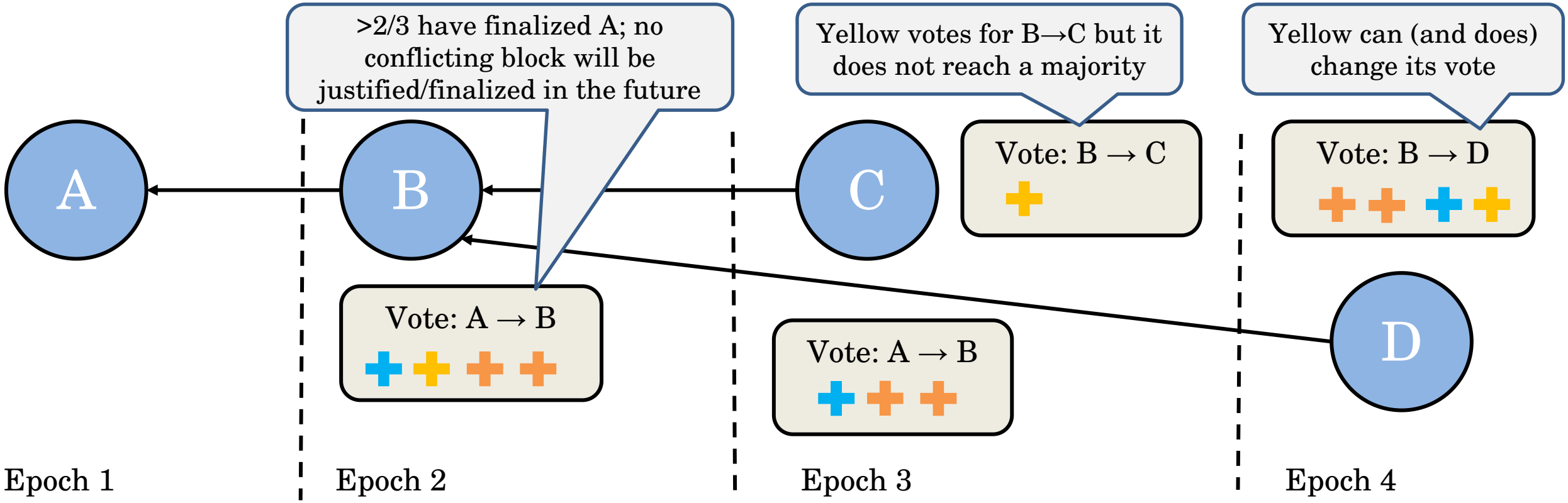


ATTESTATIONS AND THE PIGEONHOLE PRINCIPLE

Honest validators will vote for at most one block per epoch

Honest validators only vote for blocks that do not conflict with an already finalized or justified block, but

- But, may still change their vote as long as a block is not justified/finalized yet



BENIGN FAILURES IN ETH2.0

- Network Partitions
  - Participants get temporarily disconnected
  - See the partially synchronous model....
- Node crashes
  - Nodes stop execution and restart after some time
  - Similar to handle as a network partition

MALICIOUS FAILURES IN ETH2.0

Block Creators

- Can hide blocks (see selfish mining)
- Can create multiple blocks in one epoch (see “nothing at stake”)
- Can create invalid blocks

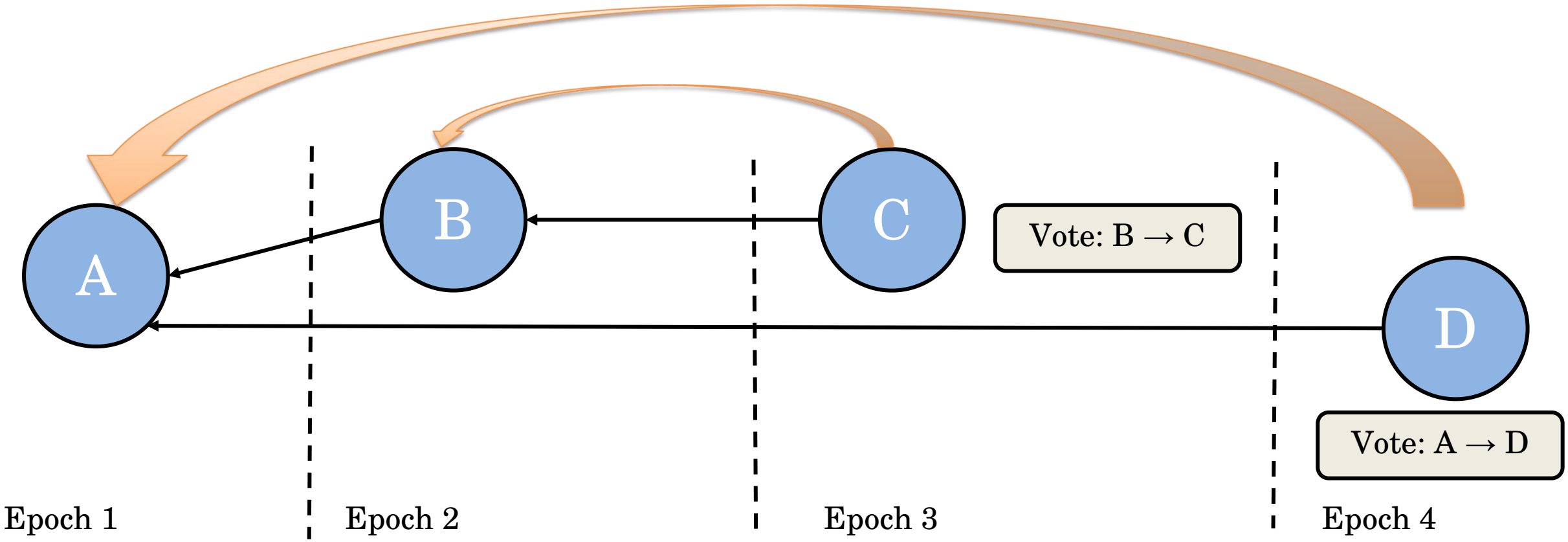
Validators

- Can create conflicting attestations (see “Byzantine generals”)
  - Multiple attestations in one epoch (simplest case)
  - Attestations that conflict with a previous attestation (more complex)

CONFLICTING ATTESTATIONS

Example: "Long-range attack", or "encompassing fork"

- Validator acknowledged B as justified in epoch 3
- But states in epoch 4 that A is the most recent justified block



SLASHING

Idea: Take (some of) the validators stake when they misbehave

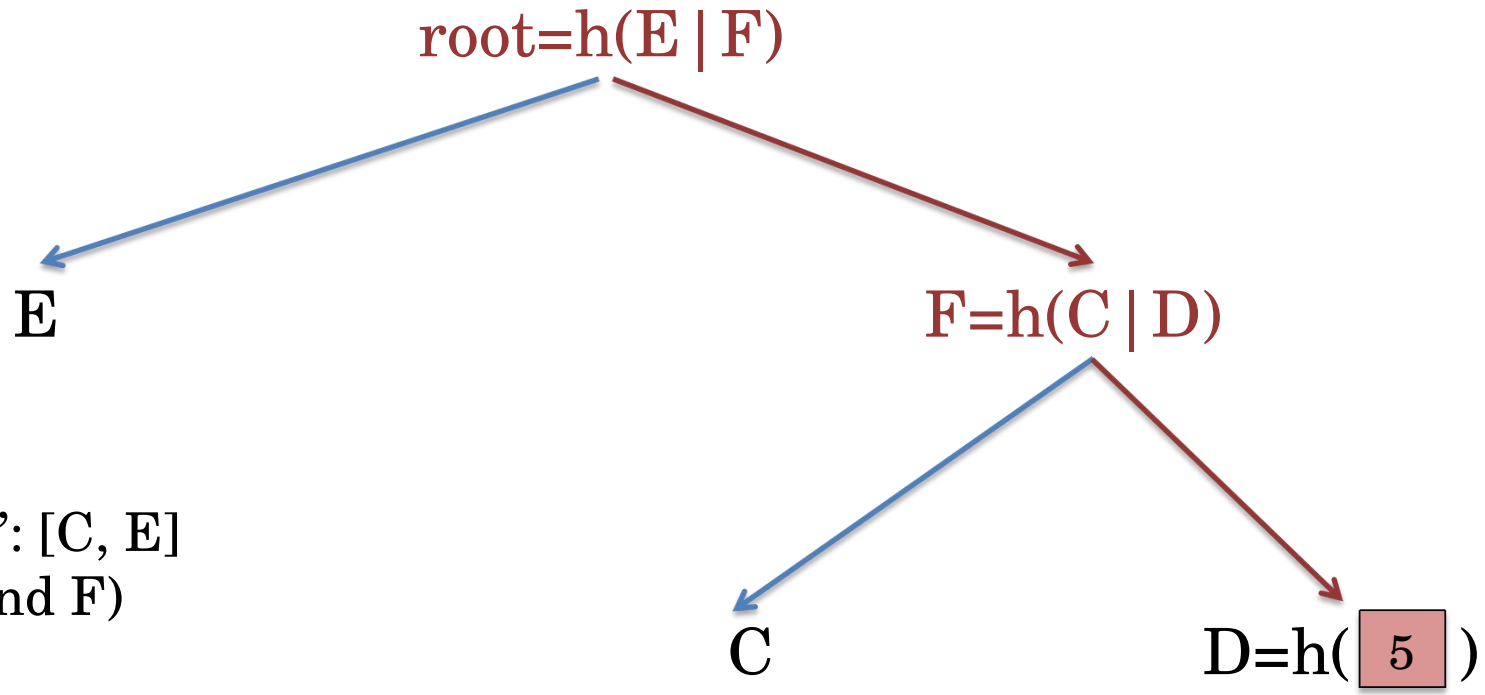
Height of penalty is calculated from

- the severity of misbehavior
  - should not penalize bugs the same as a coordinated attack

- the voting share of the validator
  - if you have more stake, the penalty will be higher

REVIEW SESSION

MERKLE PROOFS



Proof for “D=5 in A” ”: [C, E]
(can re-compute D, and F)
(root is stored)

We can verify a single data item by comparing its branch with the root of the tree

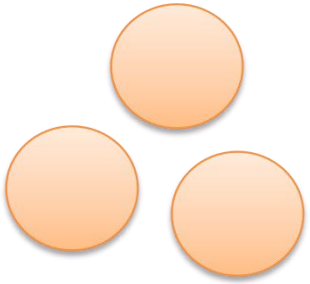
- Verifier only needs to have the root stored
- Verifier recomputes every node in the branch, and compares newly computed root with known root

UTXO MODEL

- UTXOs can be spent at most once
  - Usually requires access to a private key to be spent
- Transactions consume a number of UTXOs and create some number of new UTXOs
- Transactions input balance must be less or equal to the output balance
  - Difference is the transaction fee

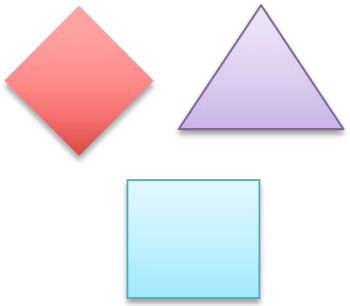
TOKENS

Two types of tokens exists



Fungible Tokens (or just “Tokens”):

- Tokens are interchangeable and separable
- E.g., shares of a company or a
- ERC20 is the standardized interface



Non-Fungible Tokens (NFTs):

- Each token is unique
- E.g., certificate of ownership of an asset
- ERC721 is the standardized interface for NFTs

DECENTRALIZED EXCHANGES

Two Types of DEXs: Liquidity Pools and Order Books

- Uniswap is a liquidity pool

Uniswap "crowdfunds" liquidity

- Participants pool currency/tokens in a smart contract
  - They receive some number of exchange-specific tokens in return

- When people trade through the contract, a fee is charged
  - Each pool participant gets share of the fee proportional to their pool contribution

- Exchange tokens can be converted back into liquidity *at the current exchange rate*

DIFFICULTY ADJUSTMENT

Pick mining difficulty based on observed block creation rate vs. expected block creation rate

- If mining power increases, also increase difficulty
- If mining power decreases, also decrease difficulty

Difficulty Adjustment Mechanism

- Defines how the difficulty is (re-)computed
- E.g., every k blocks or incrementally

“Heaviest Chain Rule”

- Chose winning chained based on total mining power needed to create it
  - Calculated by summing up difficulty values

SELFISH MINING

Idea: Selfish miners (SM) hide their blocks to gain an advantage

- Best case scenario: Honest miners mine on an outdated prefix of SM's fork

Make blocks public "at the last minute":

- E.g., when honest miners find a block in a competing fork
- Make as few blocks visible as public

Problems caused by selfish mining:

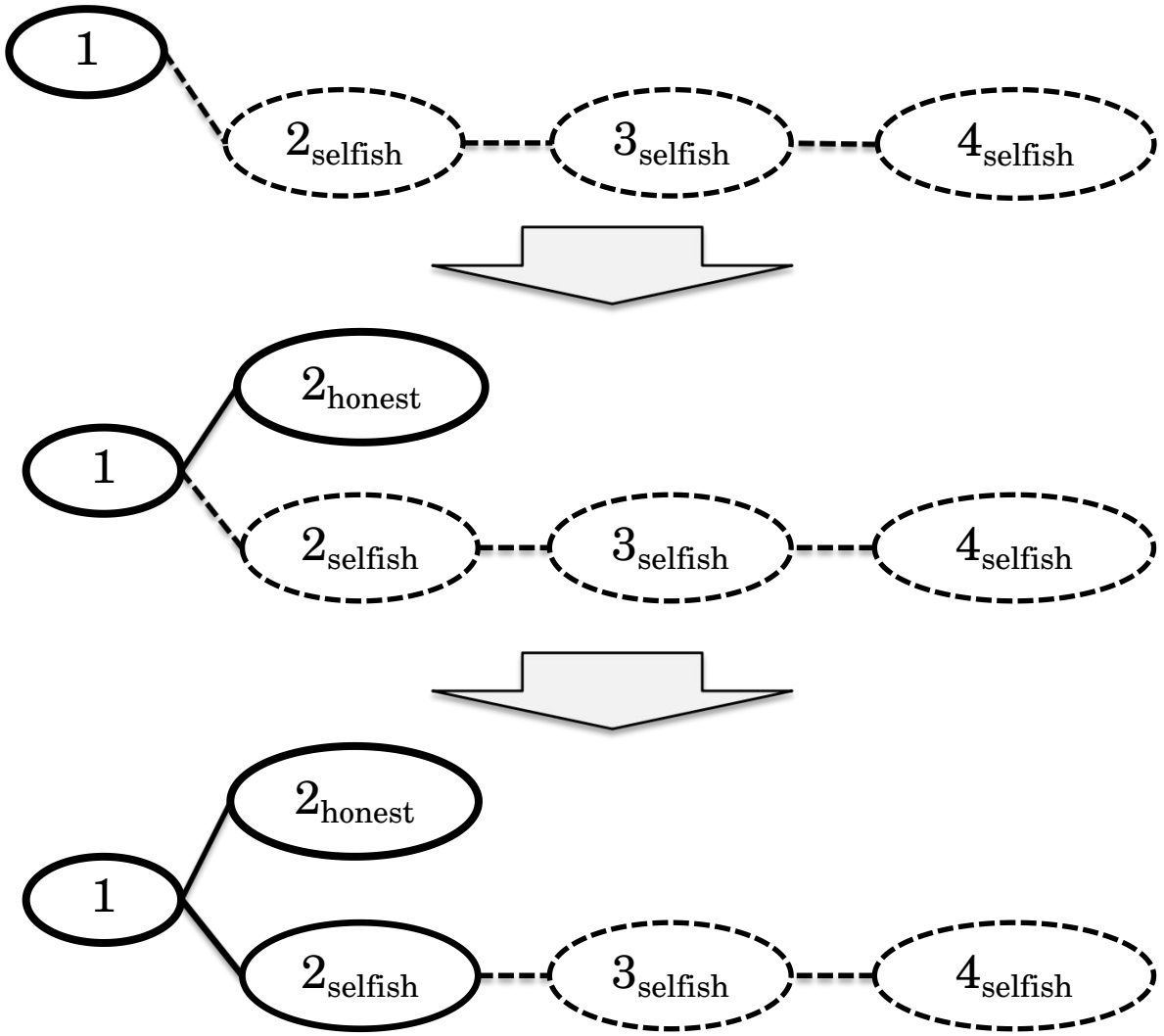
- Honest mining will be less profitable
- Throughput of the blockchain might be reduced
- Difficulty adjustment won't work correctly

SELFISH MINING: EXAMPLE

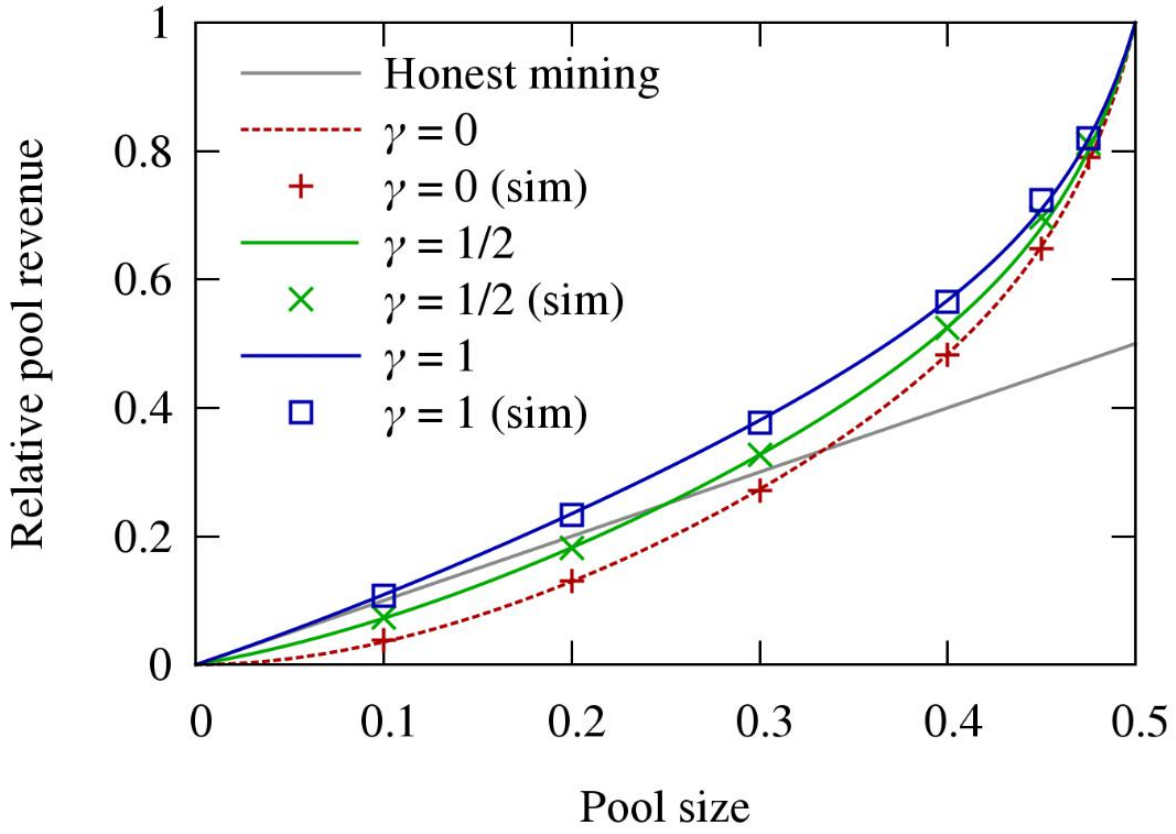
1) Private branch consist of three hidden blocks

2) Honest miners publish a block

3) Selfish miner reveals one of their blocks so there are two competing public branches



SELFISH-MINE REVENUE



Major takeaway

- Event with the proposed fix (γ=1/2) selfish mining is already profitable at 25%

- Bitcoin is less secure than previously assumed