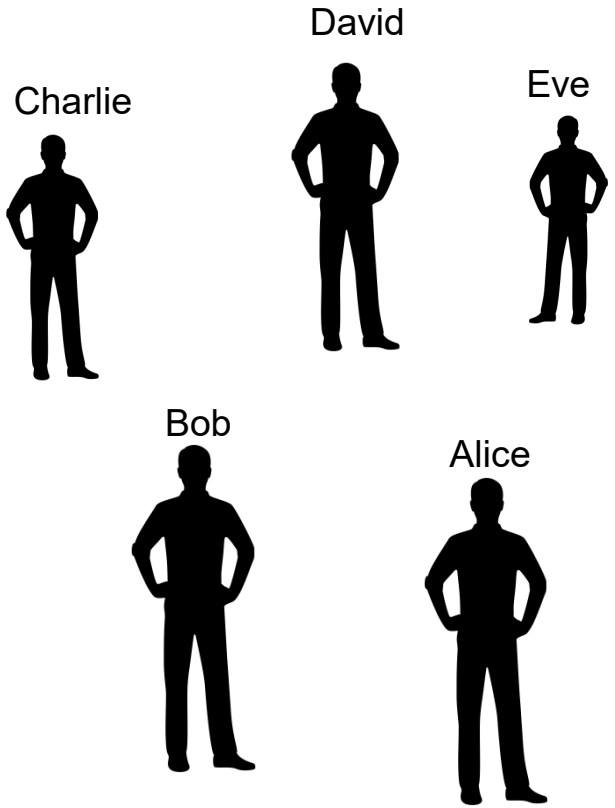
CS459/698 Privacy, Cryptography, Network and Data Security

Blockchain

Fall 2025, Tuesday/Thursday 8:30-9:50am

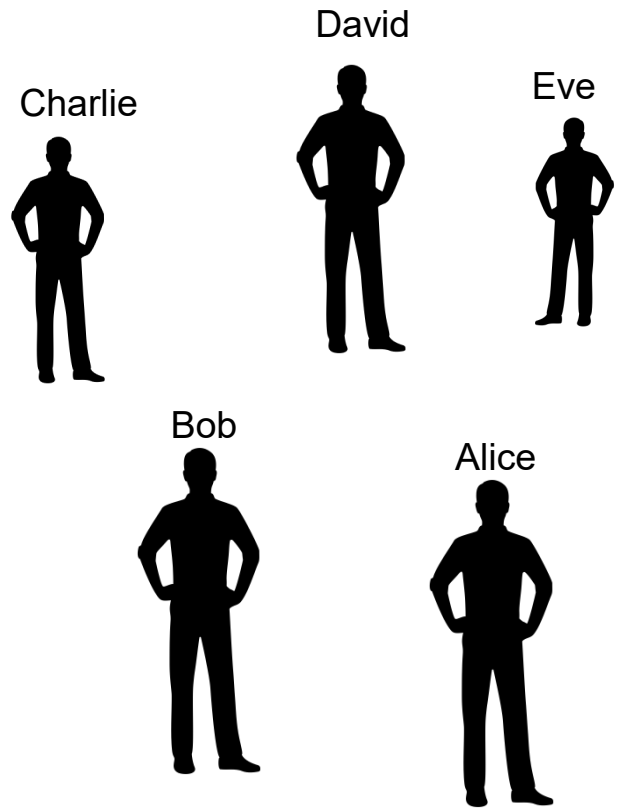
Byzantine Fault Tolerance (BFT) in Decentralized Systems

Problem: Who is the best Chess Player?



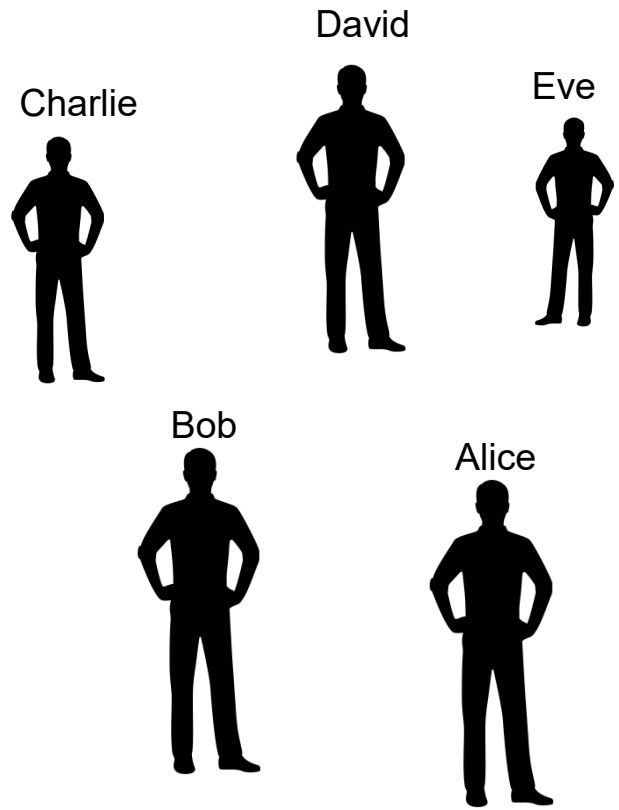
Problem: Who is the best Chess Player?

- Goal: Record game results
  - Alice vs Bob / 2025-02-19T14:30:00Z / Bob Won



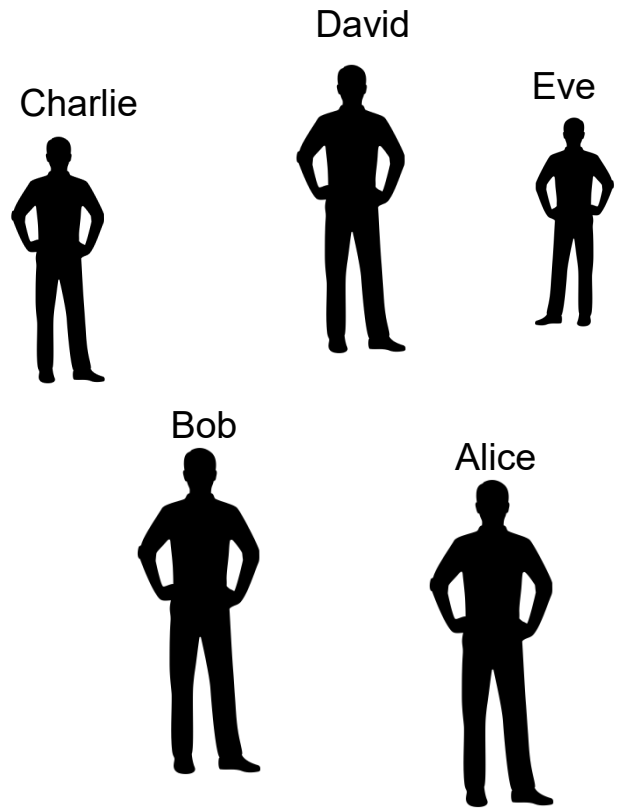
Problem: Who is the best Chess Player?

- Goal: Record game results
  - Alice vs Bob / 2025-02-19T14:30:00Z / Bob Won
- Security Challenges:
  - Confidentiality: ✅
    - Not an issue, data can be accessible for anyone.



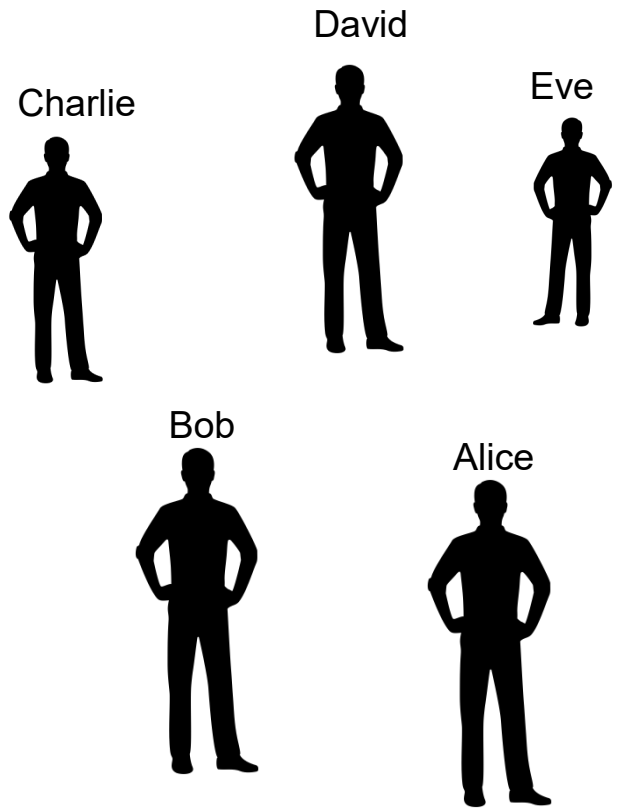
Problem: Who is the best Chess Player?

- Goal: Record game results
  - Alice vs Bob / 2025-02-19T14:30:00Z / Bob Won
- Security Challenges:
  - Confidentiality: ✅
    - Not an issue, data can be accessible for anyone.
  - Availability: ❌
    - Records might get lost.
    - Data must be accessible for all.



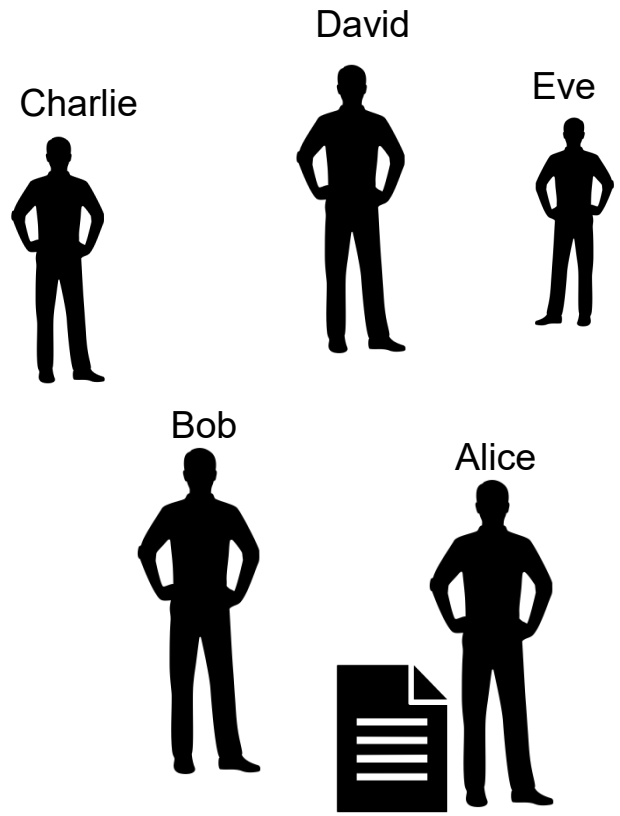
Problem: Who is the best Chess Player?

- Goal: Record game results
  - Alice vs Bob / 2025-02-19T14:30:00Z / Bob Won
- Security Challenges:
  - Confidentiality: ✅
    - Not an issue, data can be accessible for anyone.
  - Availability: ❌
    - Records might get lost.
    - Data must be accessible for all.
  - Integrity: ❌
    - Nobody can be fully trusted.
    - Data should not be manipulated.



Who is the best Chess Player?

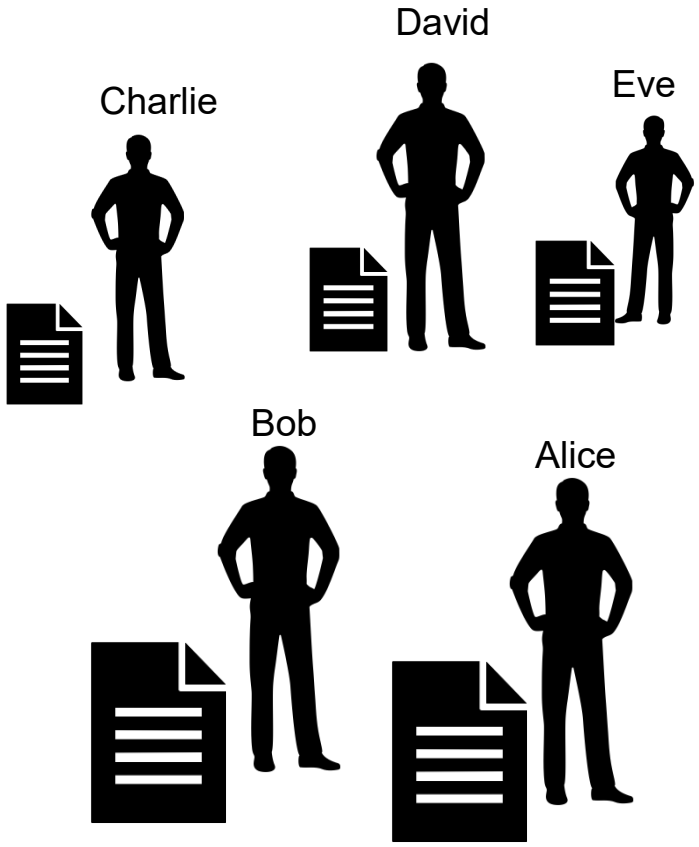
- Solution 1:
  - Trust Alice for recording results.
- Risks:
  - Availability: ❌
    - Alice might lose her notebook
  - Integrity: ❌
    - Alice might cheat



Who is the best Chess Player?

- Solution 2:
  - Each member independently records all the results.
- Risks:
  - Availability: ✅

    * Ignoring the unlikely scenario where all notebooks go missing.
  - Integrity: ❌
    - How to ensure all books stay synced and up-to-date?



Who is the best Chess Player?

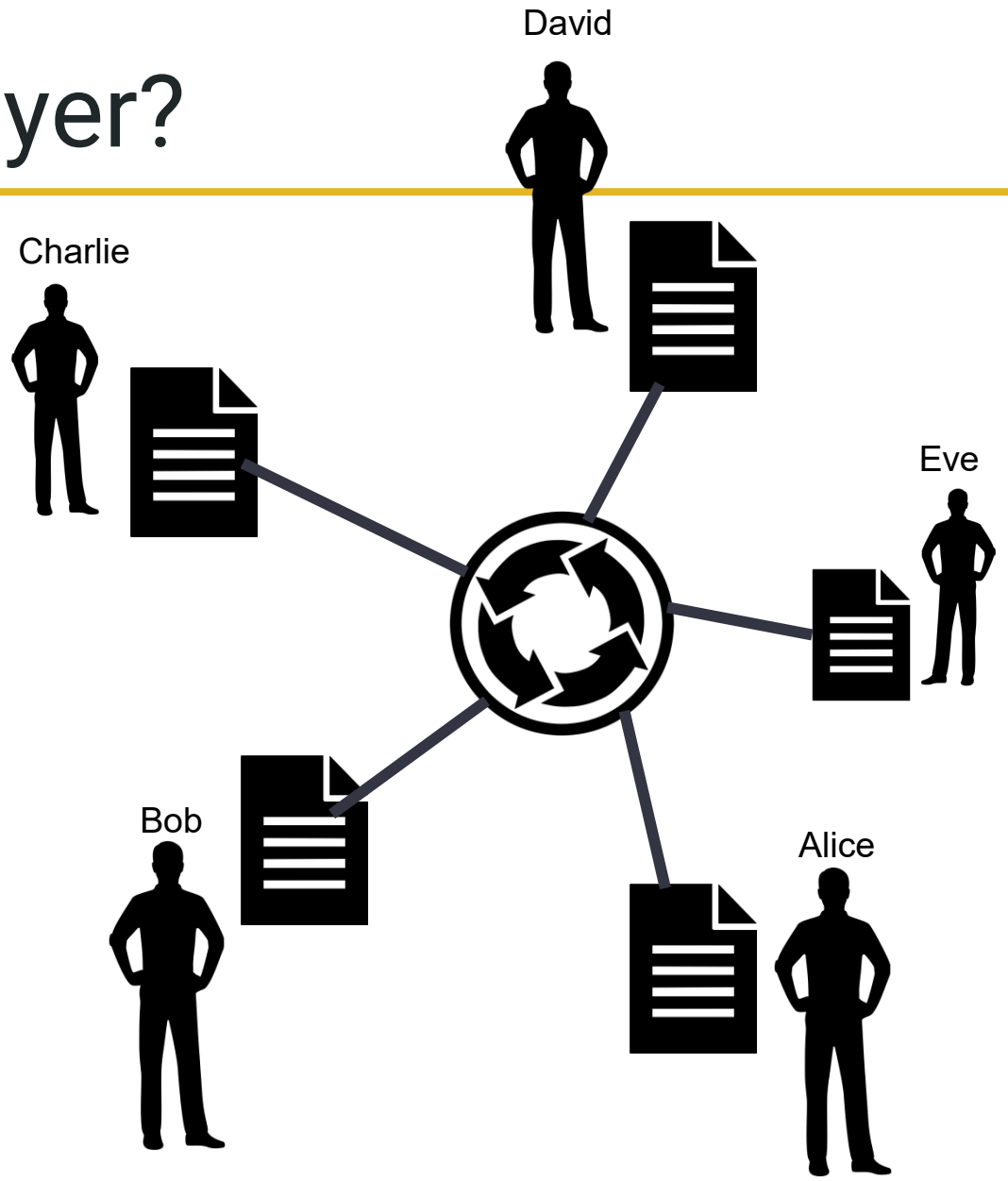
- Solution 3:
  - Consensus: Each member records all results by following a protocol to keep the data synchronized. (Called: Consensus Protocol)

Risks:

  - Availability: ✅

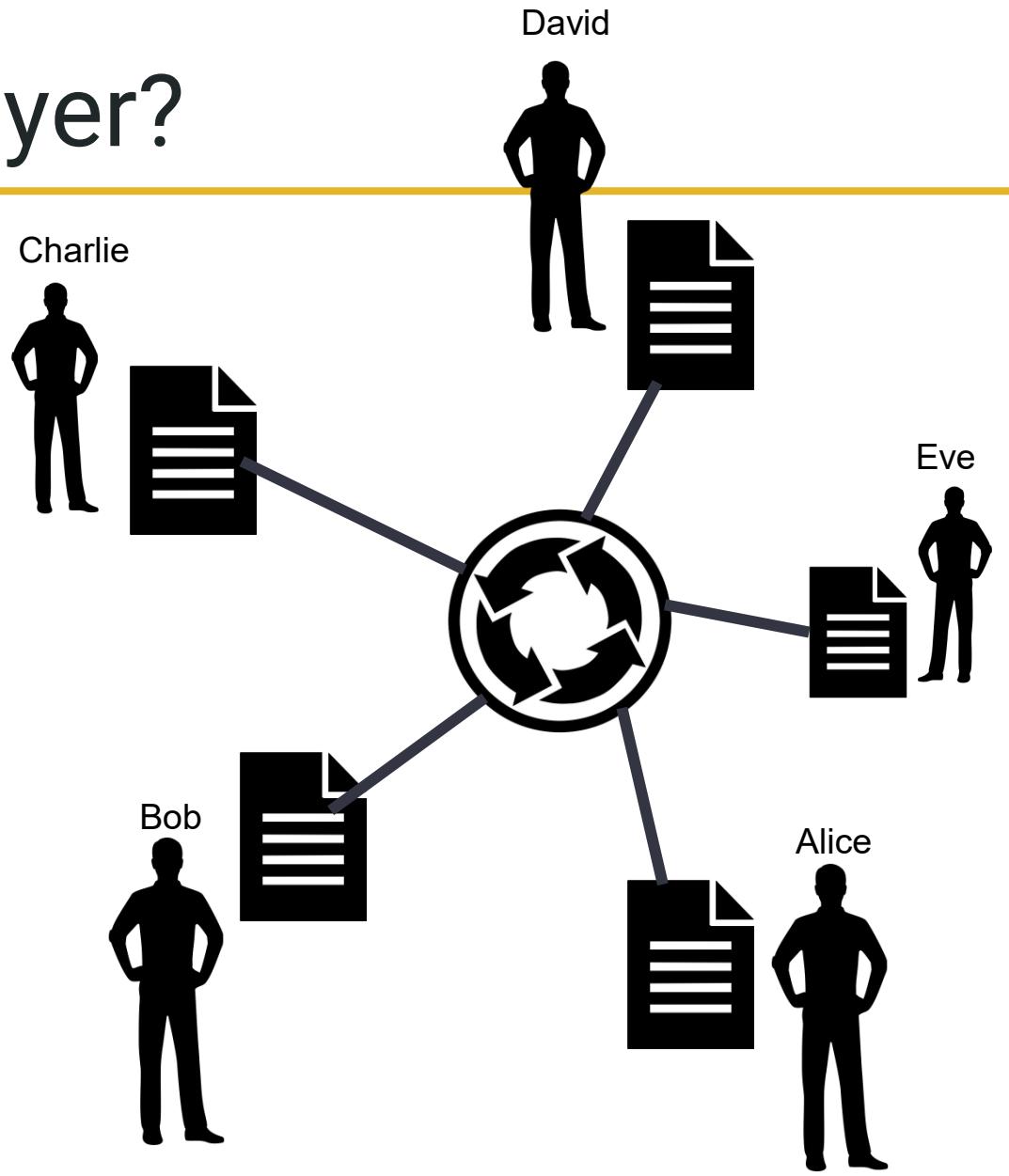
    * Ignoring the unlikely scenario where all notebooks go missing.

  - Integrity: ✅
    - An attempt to change the personal notebook is not enough to change the reality.



Who is the best Chess Player?

Q: What happens if some people collude? What is the minimum number of nodes required to alter reality?



Fault Tolerance Level: No Fault Tolerance (NFT*)

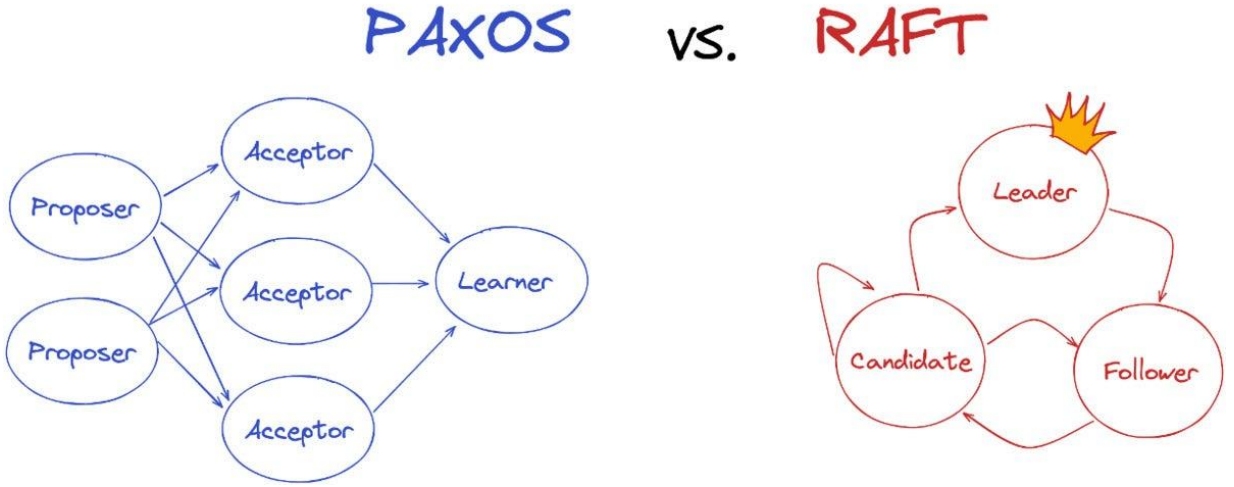
- **Definition:** The system fails completely if any node crashes, behaves incorrectly, or acts maliciously.

- **Example:** early versions of Napster relied on a single source for a file, its failure results in data loss.

Fault Tolerance Level: Crash Fault Tolerance (CFT)

- **Definition:** Handles the case of “if a node crashes”. No single point of failure. No single source of truth.

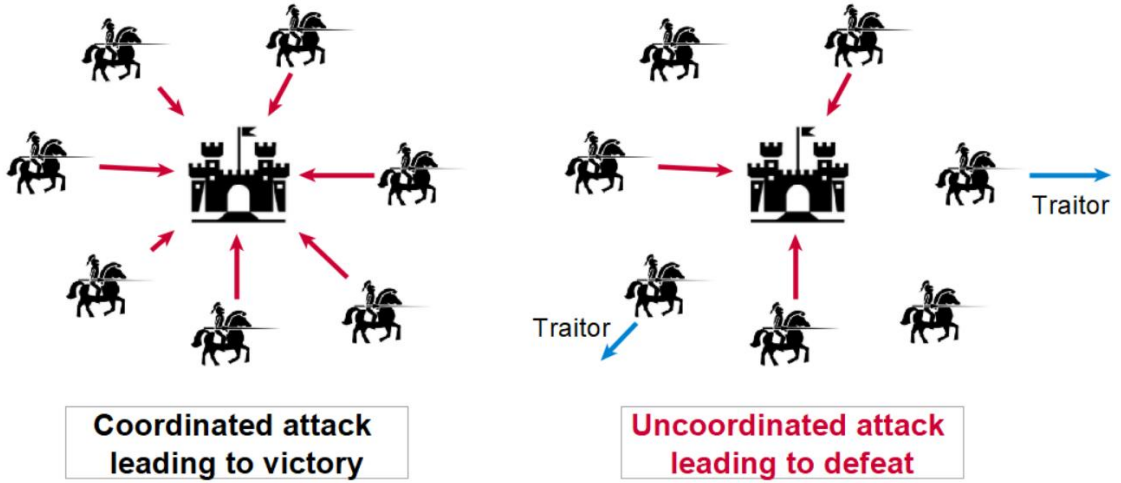
- **Examples:** Raft, Paxos



Fault Tolerance Level: Byzantine Fault Tolerance (BFT)

- **Definition:** Also handles the case of “if a node is byzantine (e.g., acting maliciously, unpredictably, or dishonestly)”. Inherently addresses CFT.

- **Examples:**
  - **PBFT (Practical Byzantine Fault Tolerance)**: Strong BFT (guaranteed with ≤1/3 malicious nodes)
  - **Nakamoto Consensus (PoW):** Probabilistic BFT (depends on mining power: >50% of mining power must be honest)



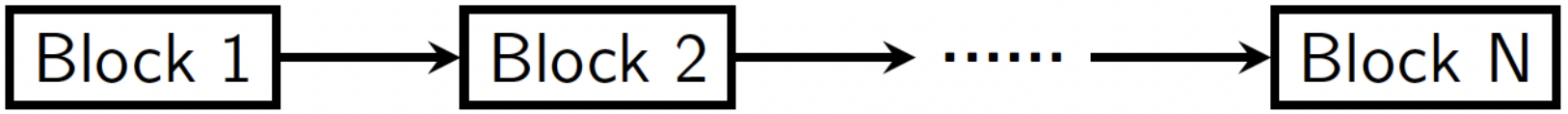
To create BFT decentralized systems we need:

- A BFT Consensus Mechanism
  - An algorithm to reach agreement
  - Solutions: PoW, PoS etc.
- A ledger
  - To record and secure agreed information (Chess Results).
    - Our solution: **Blockchain**

Blockchain

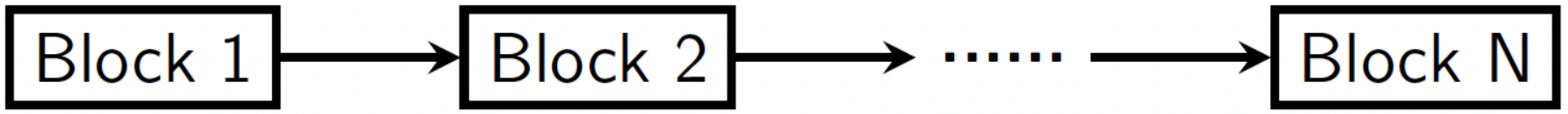
What is a blockchain?

- A blockchain is ... a chain of blocks!



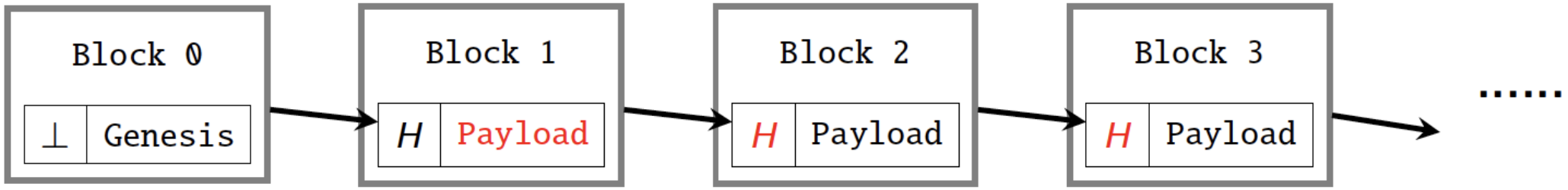
What is a blockchain?

- A blockchain is ... a chain of blocks!



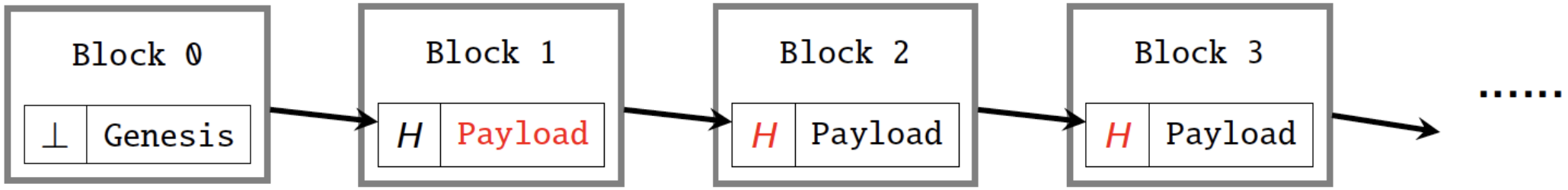
  - What is the purpose of it?
  - What does chaining mean here?
    - A Linked list? A Data structure?
  - What goes into these blocks?
    - A fixed format?

Basic chaining scheme: Version 1



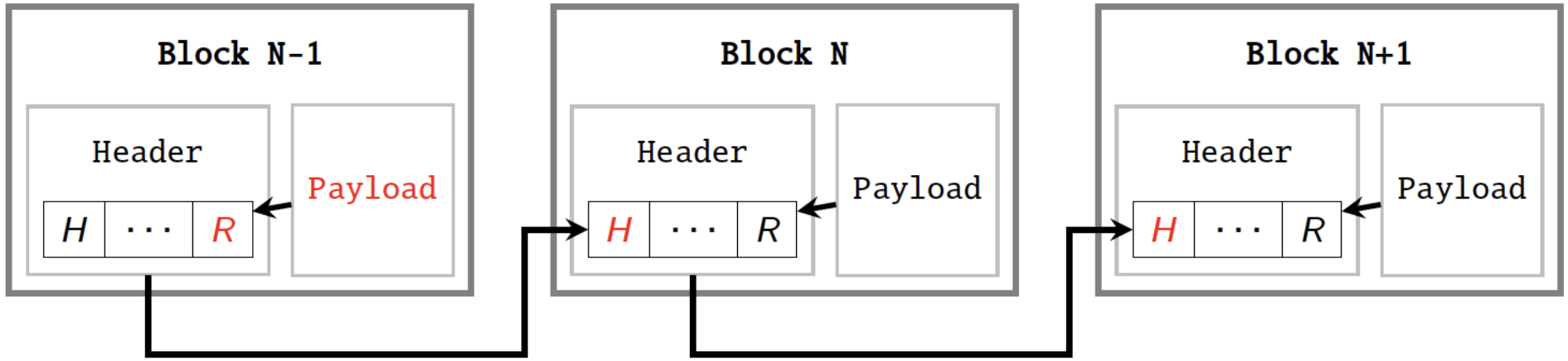
- Each block contains a cryptographic hash of the total body of the previous block.
- So, each block cryptographically depends on the previous block(s).

Basic chaining scheme: Version 1



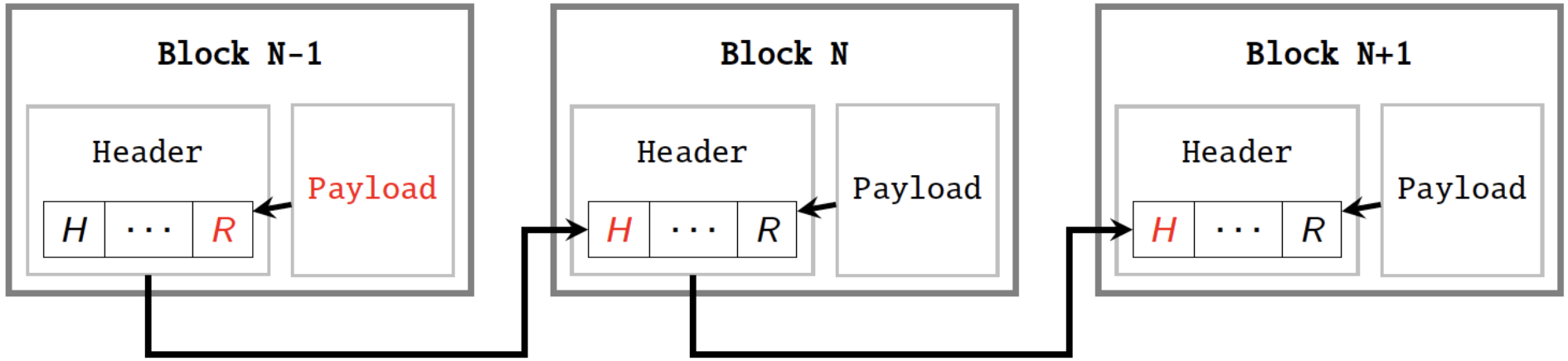
Q1: What happens if we modify the payload of "Block 1"?

Basic chaining scheme: Version 2



- Each block is split into two parts:
  - A *header* that contains at least two critical values:
    - *H:* A **cryptographic hash** of the previous block header
    - *R:* A **cryptographic hash** of the current block payload
  - *Payload* contains application-specific information

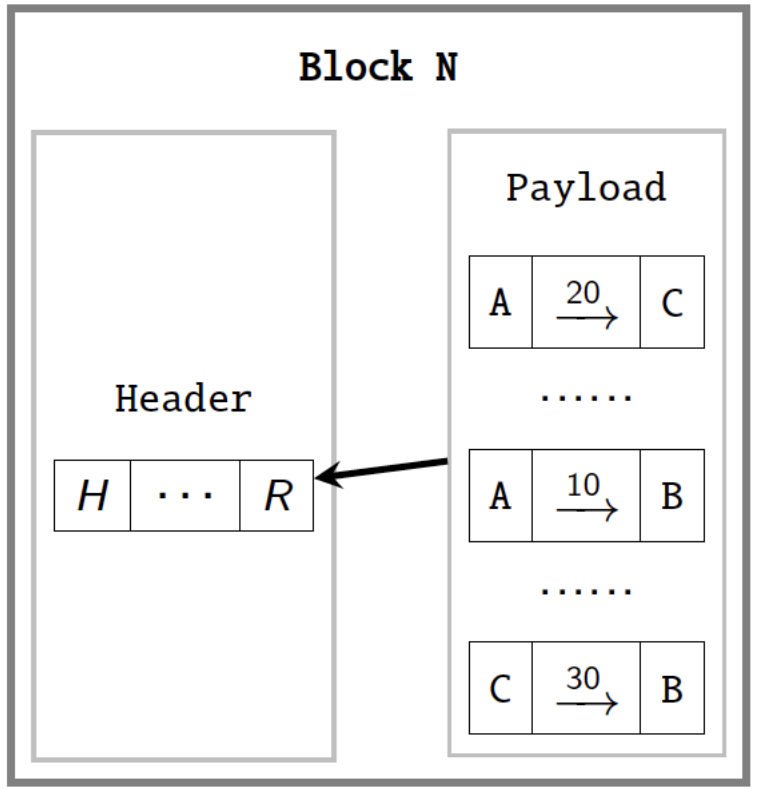
Basic chaining scheme: Version 2



Q: Why is this a better scheme?

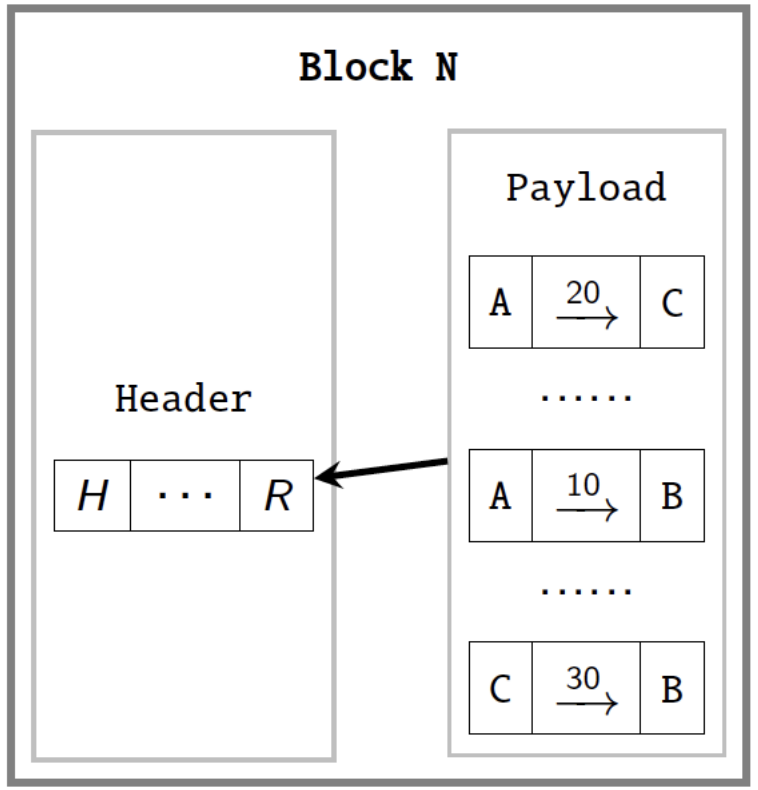
What goes into the payload?

- Anything depending on how you plan to use it:
  - Record vital information
    - **Example:** Chess Results
    - Alice vs Bob / 2025-02-19T14:30:00Z / Bob Won



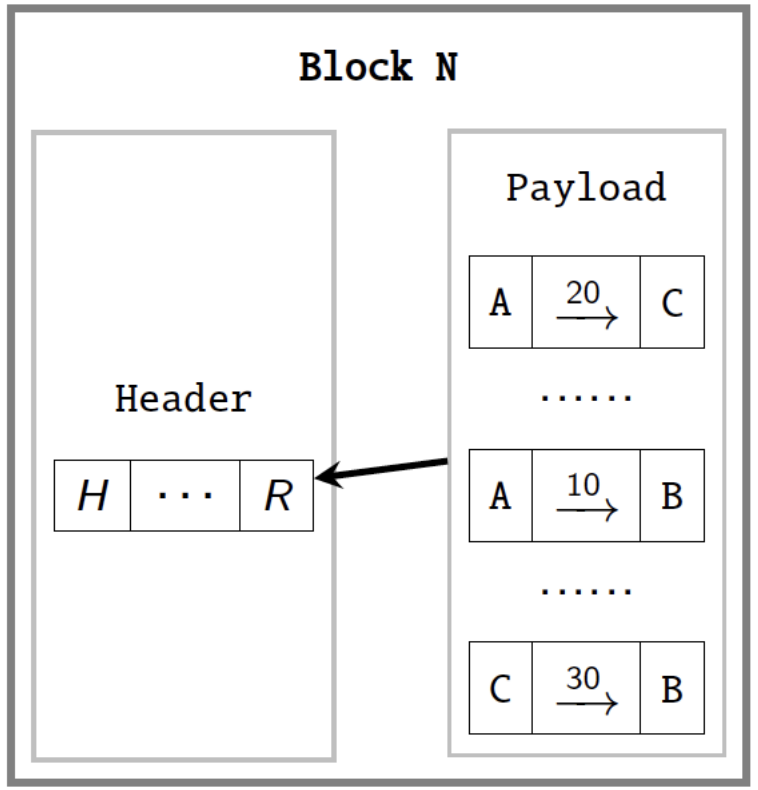
What goes into the payload?

- Anything depending on how you plan to use it:
  - Record vital information
    - **Example:** Chess Results
    - Alice vs Bob / 2025-02-19T14:30:00Z / Bob Won
  - Record Transactions
    - **Example:** Bitcoin
    - A sent TX to B



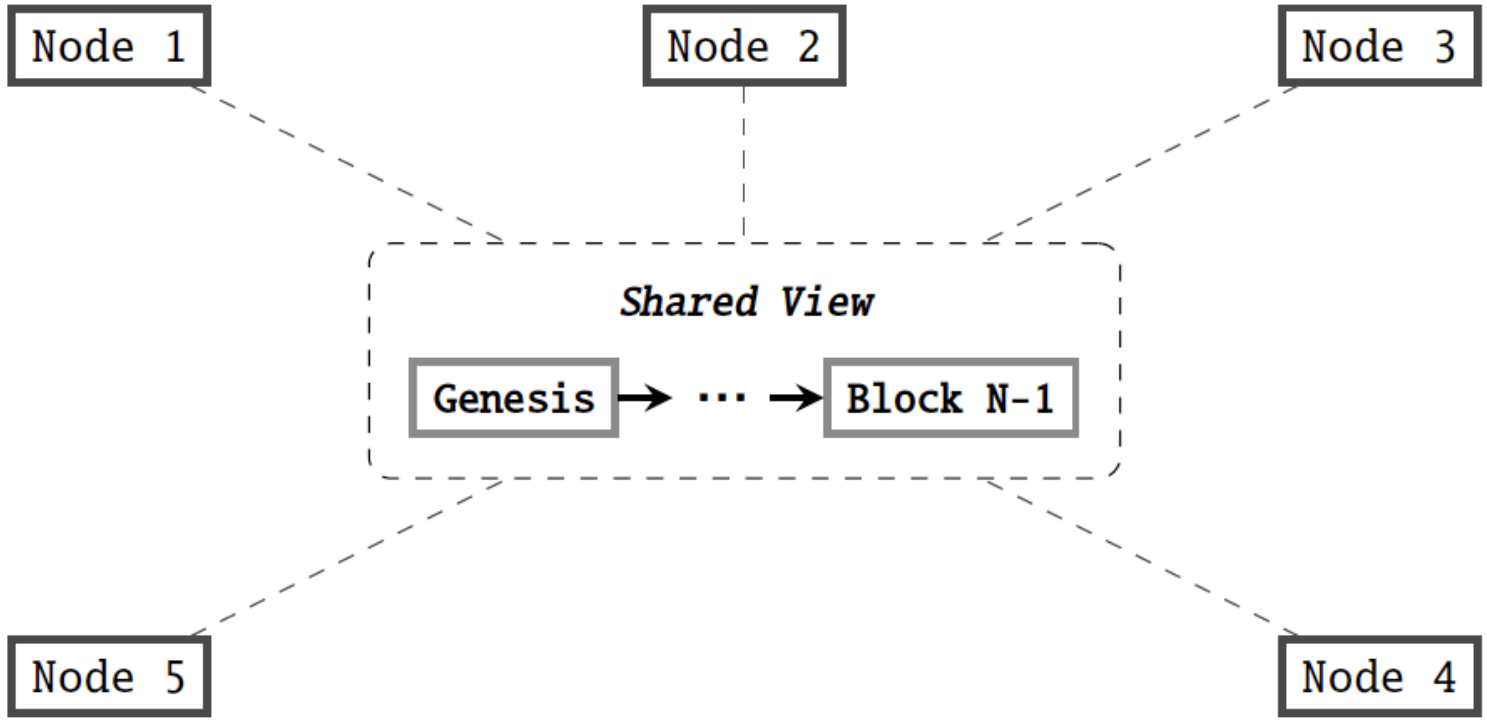
What goes into the payload?

- Anything depending on how you plan to use it:
  - Record vital information
    - **Example:** Chess Results
    - Alice vs Bob / 2025-02-19T14:30:00Z / Bob Won
  - Record Transactions
    - **Example:** Bitcoin
    - A sent TX to B
  - Record Transitions in a State Machine
    - **Example:** Ethereum SC
    - (var1 = “This” >>>> var1 =“That”)



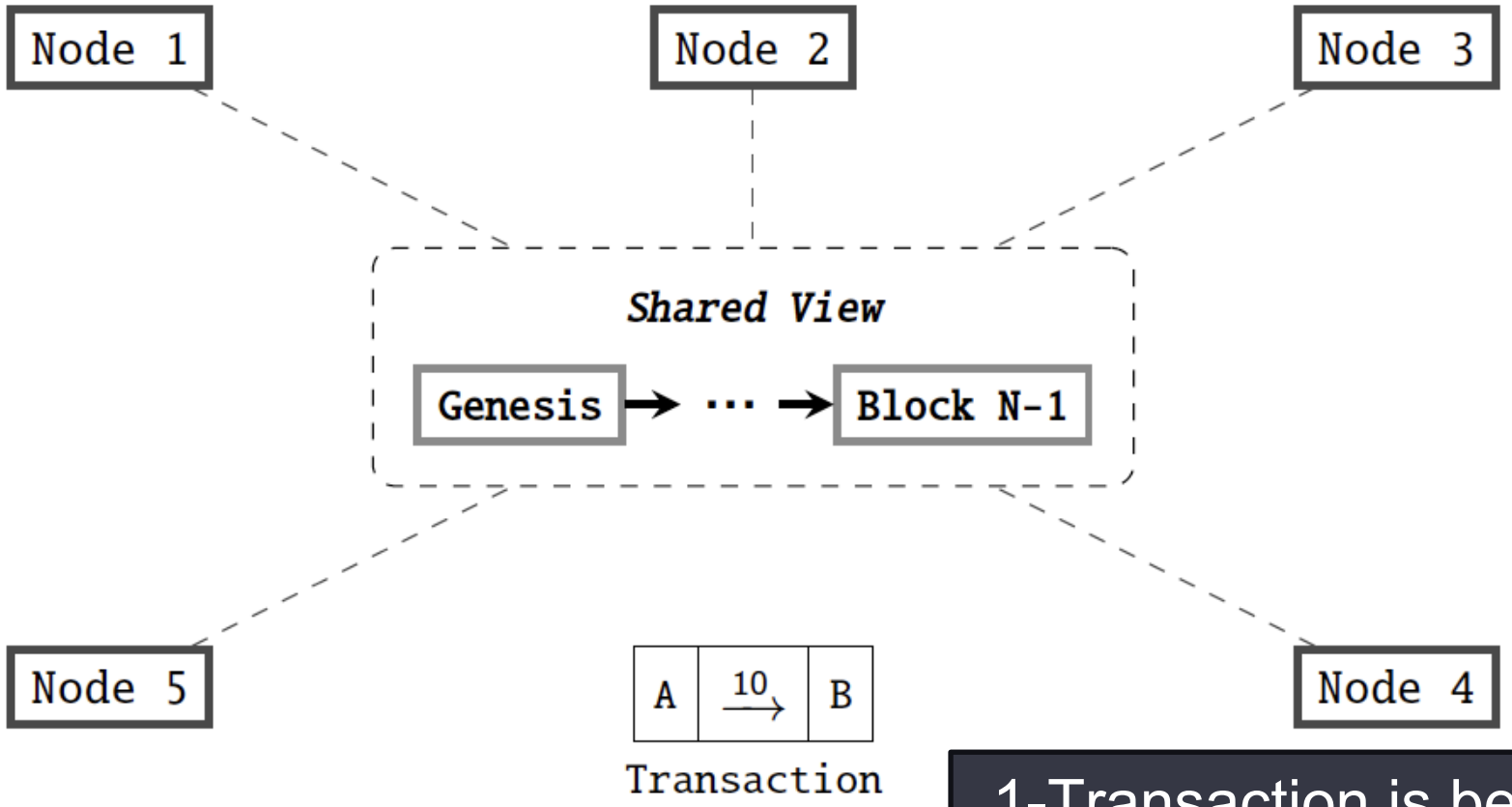
Consensus

How does data get into the block? Answer: Consensus



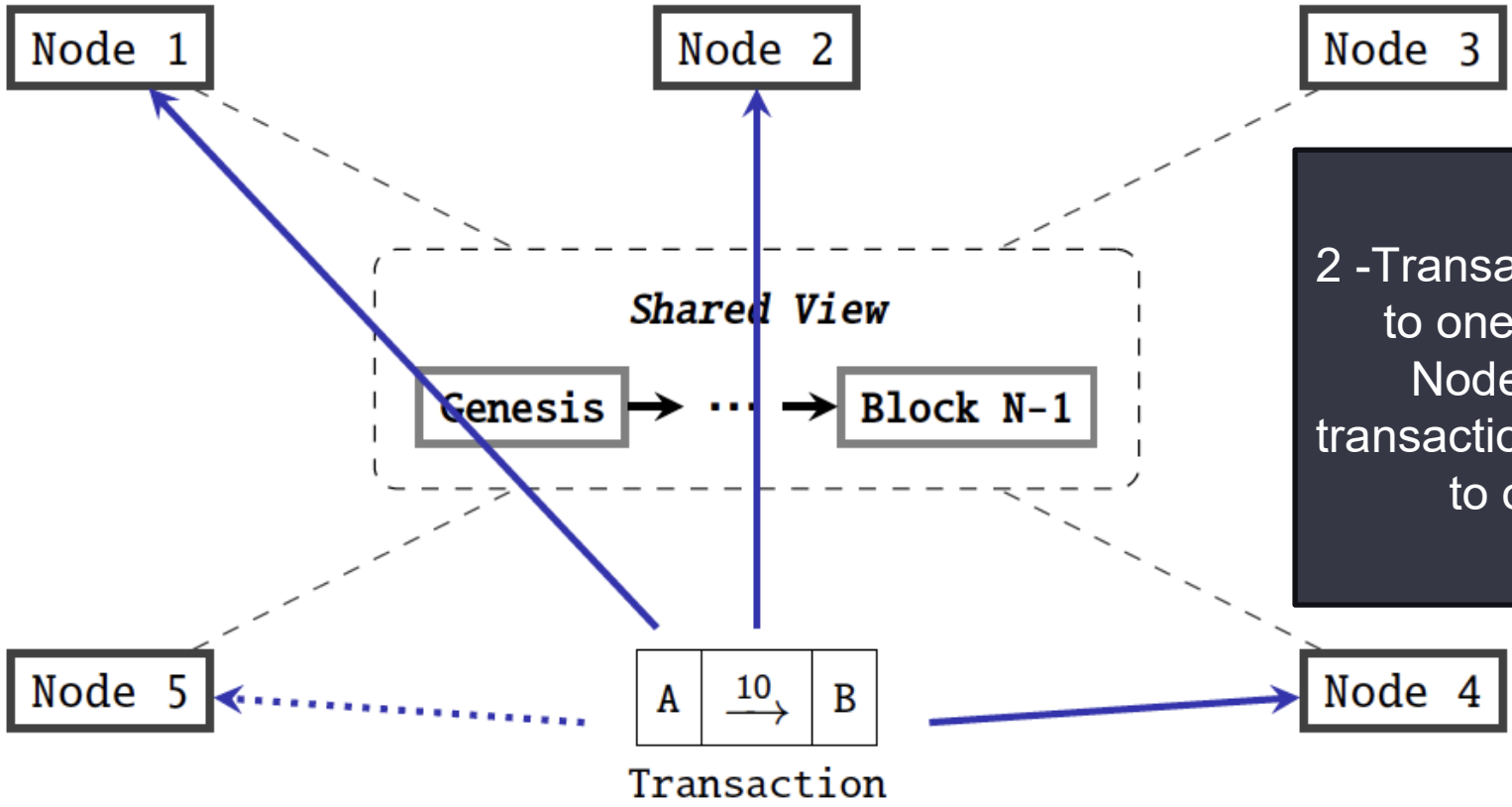
***Note:** Nodes are hosting the blockchain, clients (wallets) are not necessarily nodes.

How does data get into the block? Answer: Consensus



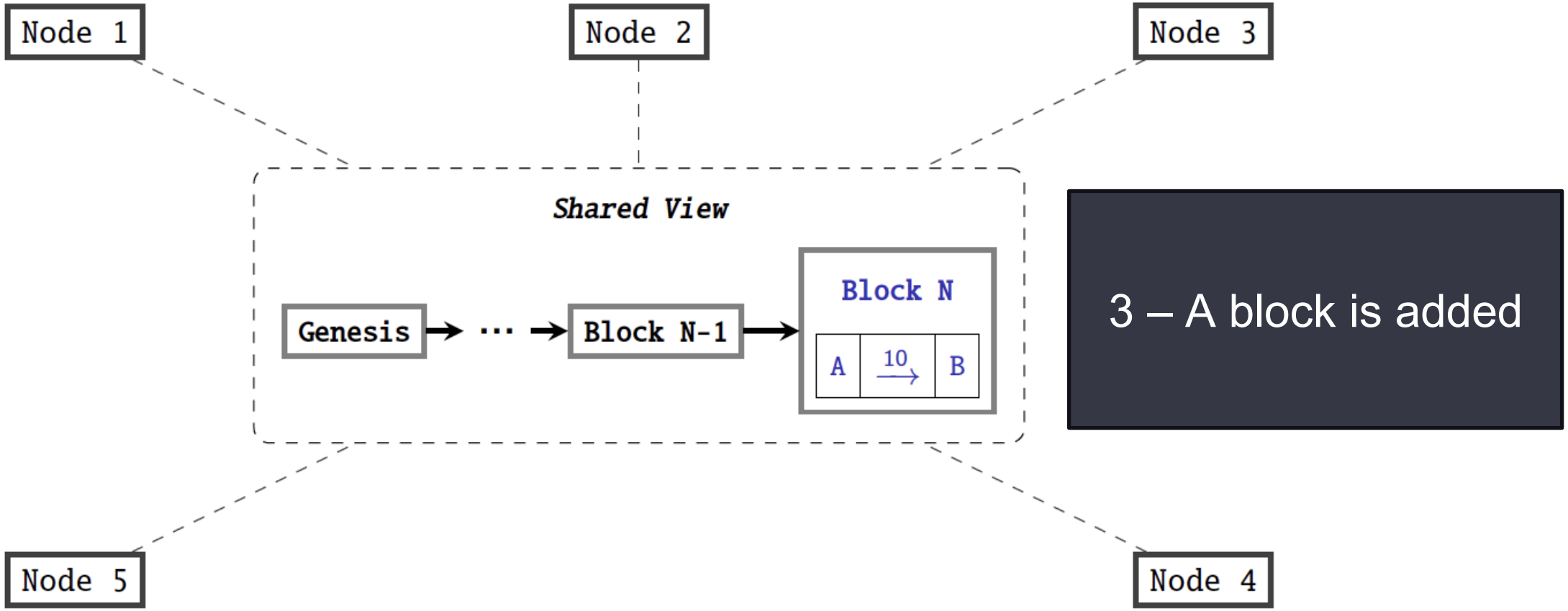
1-Transaction is being created and signed.

Case 1:

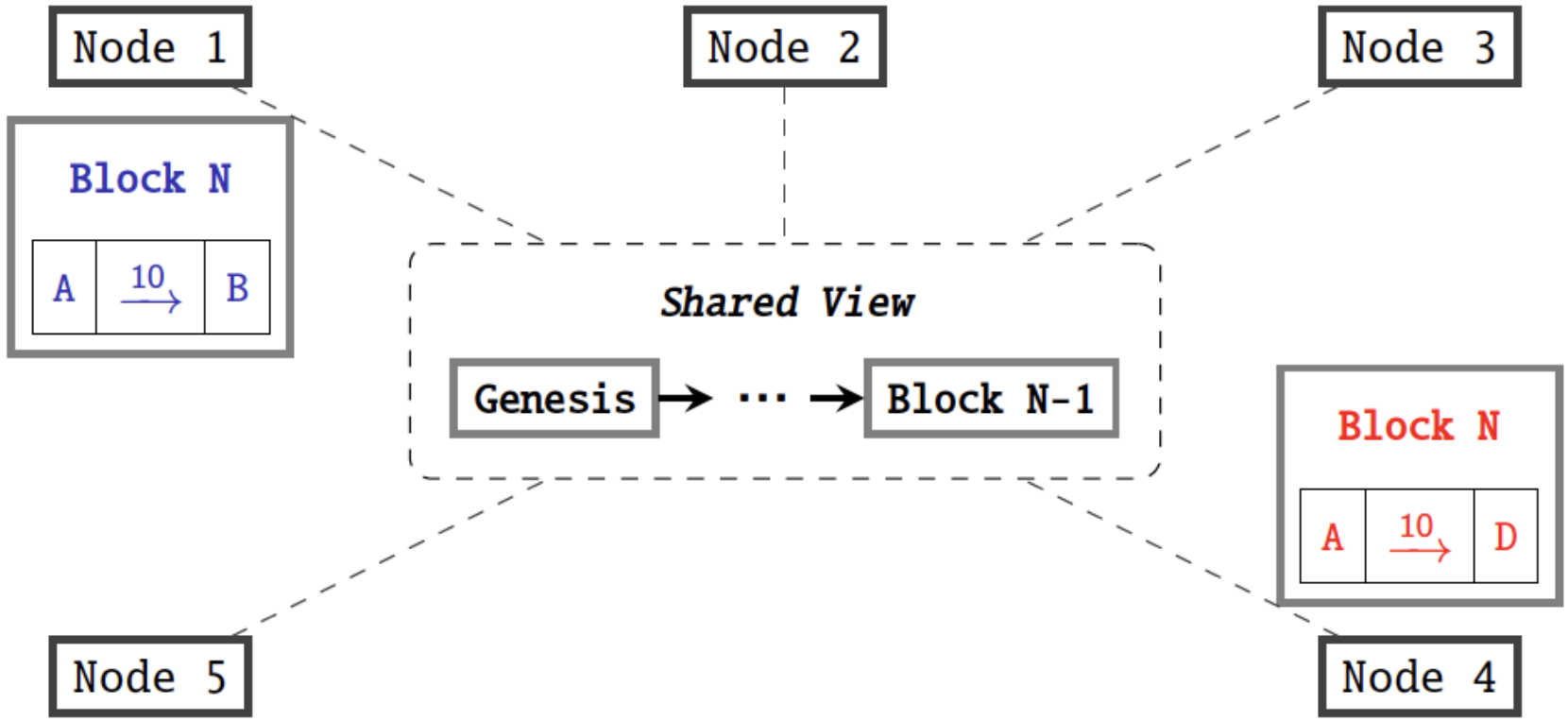


2 -Transaction is being sent to one or some node. Nodes validate the transaction and broadcast it to other nodes.

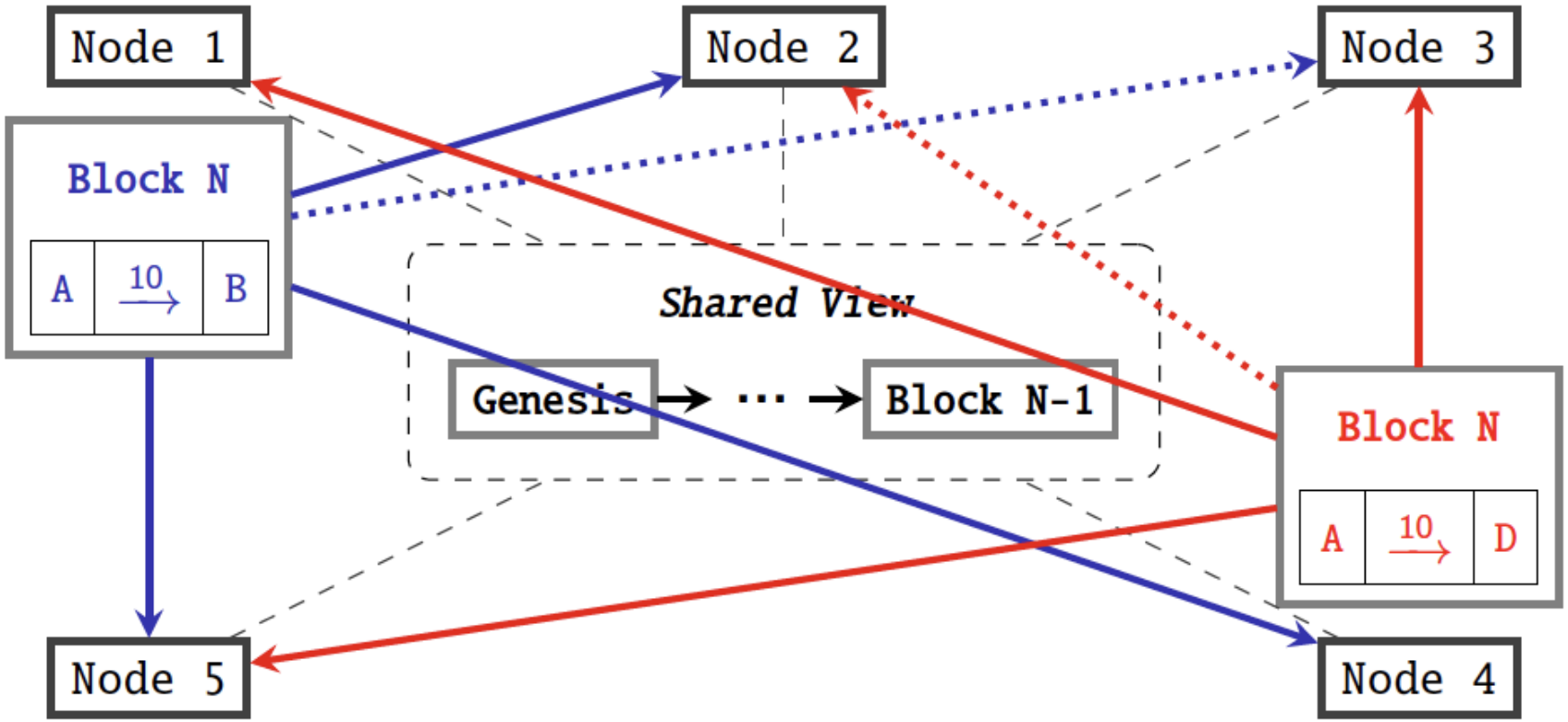
Case 1:


Node 1
Node 2
Node 3
Shared View
Genesis
Block N-1
Block N
A
10
B
3 – A block is added
Node 5
Node 4

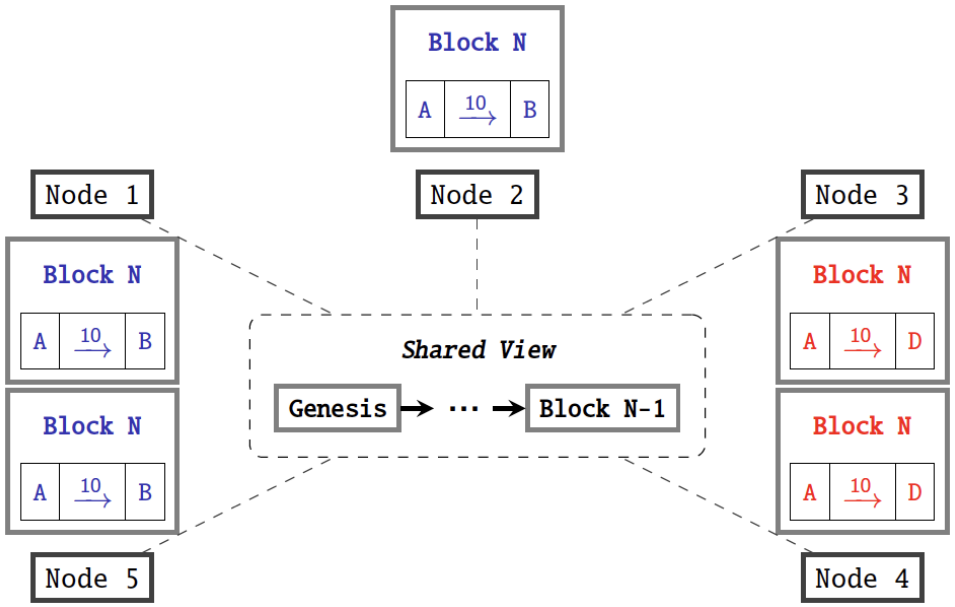
Case 2: Conflicting Transactions



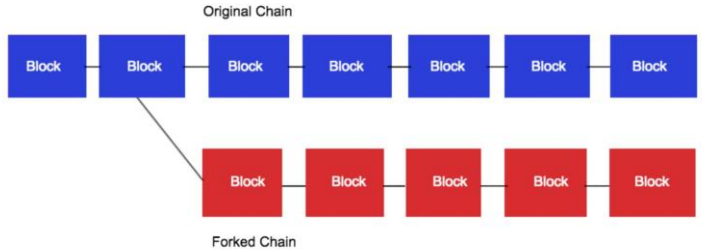
Case 2: Conflicting Transactions



Case 2: Conflicting Transactions



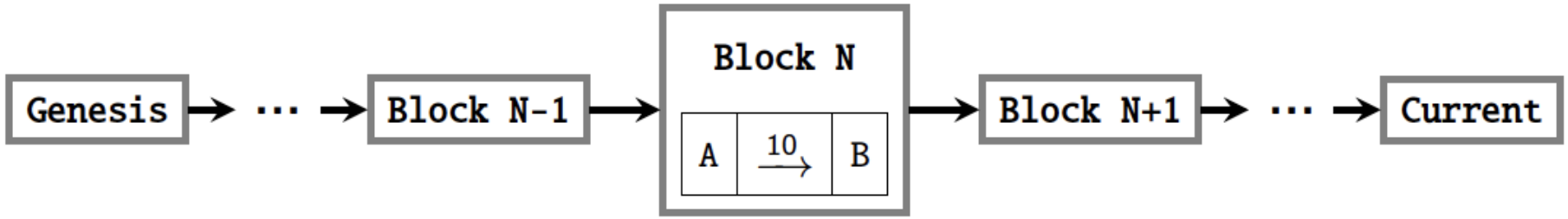
A fork happens, now they should choose wisely, regardless of their choice, the longer chain is valid. (*blocks are hard to create)



NOTE*: Blockchain nodes are incentivized, not obligated, to cooperate at any step.

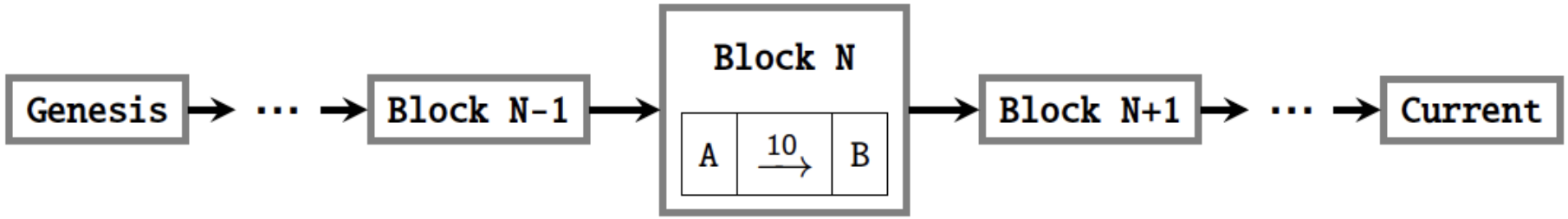
Why should we trust the blockchain?

- Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:
  - Cash, debit card, credit card, e-transfer (e.g., Interac®)
  - An entry in the blockchain-based ledger



Why should we trust the blockchain?

- Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:
  - Cash, debit card, credit card, e-transfer (e.g., Interac®)
  - An entry in the blockchain-based ledger

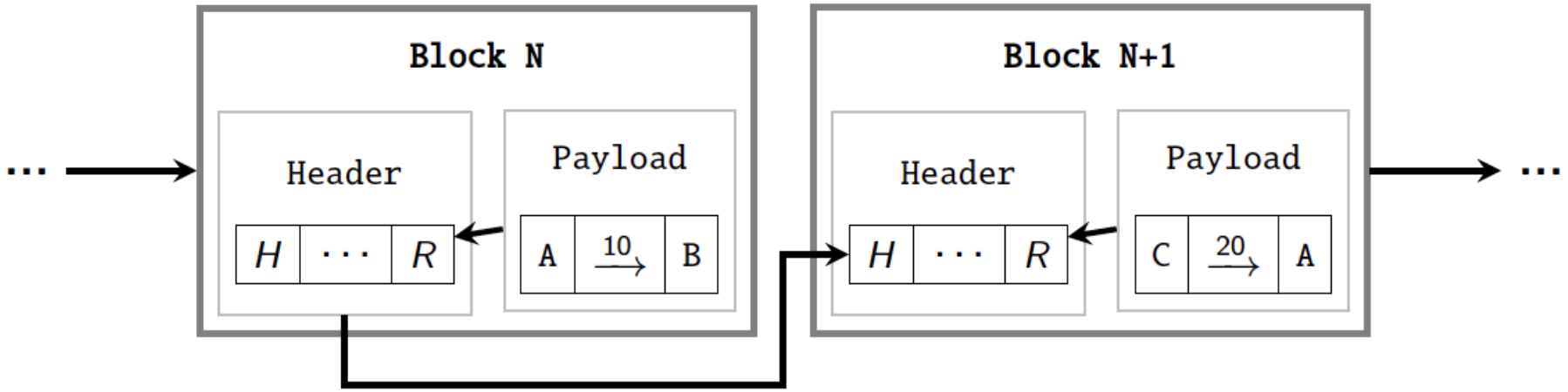


- To the best of everyone's knowledge:
  - It should be hard for Alice to produce such a chain of blocks.

But how? Let's see!

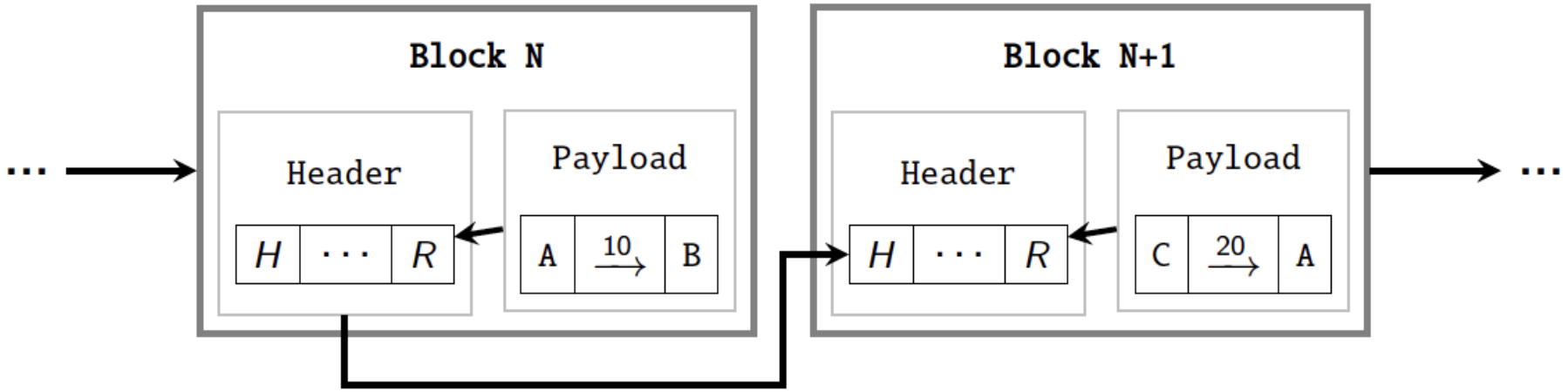
Proof-of-work

How hard is it to alter this chain?

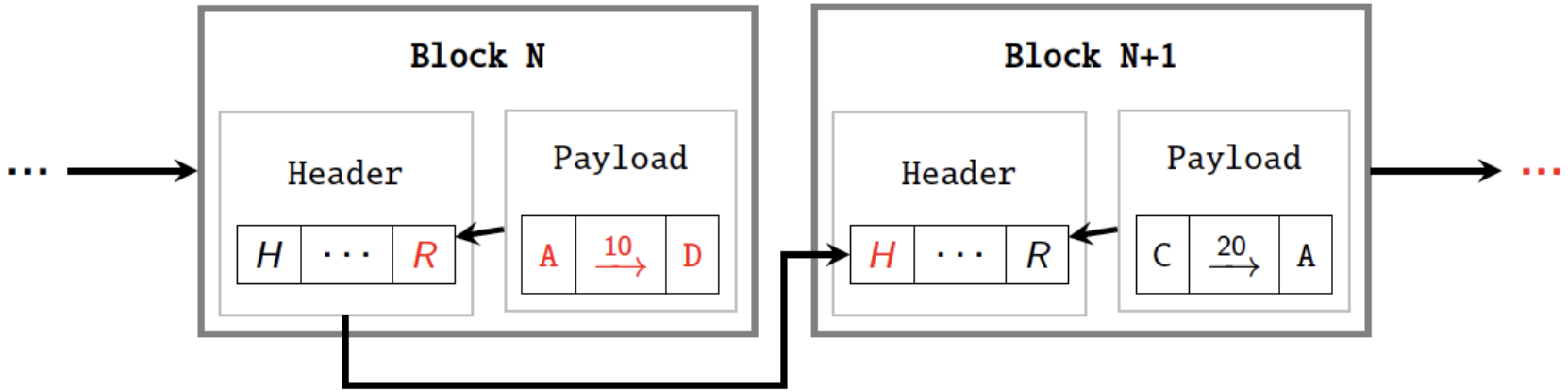


- This is the chain Alice shows Bob regarding her payment.

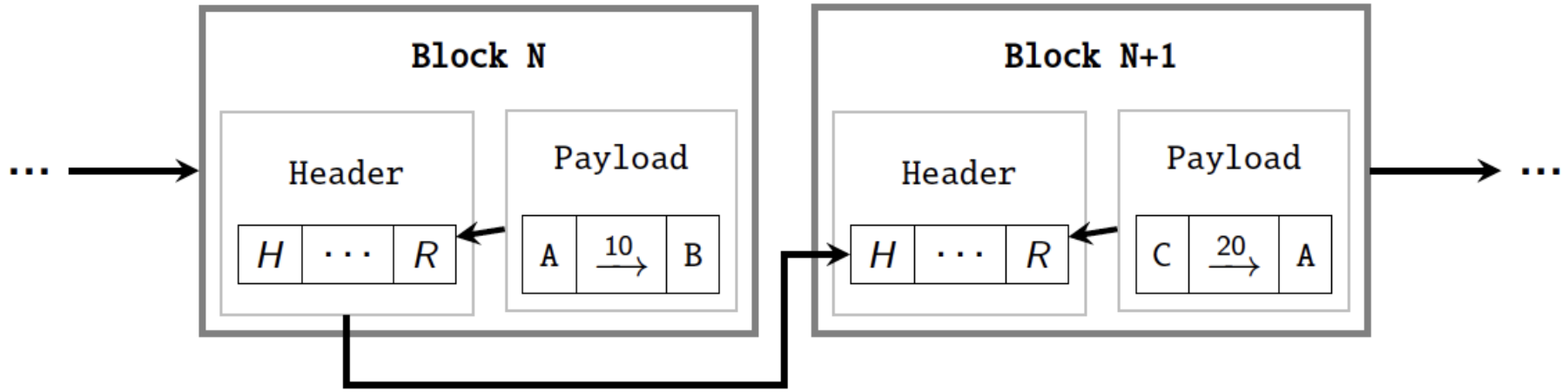
How hard is it to alter this chain?



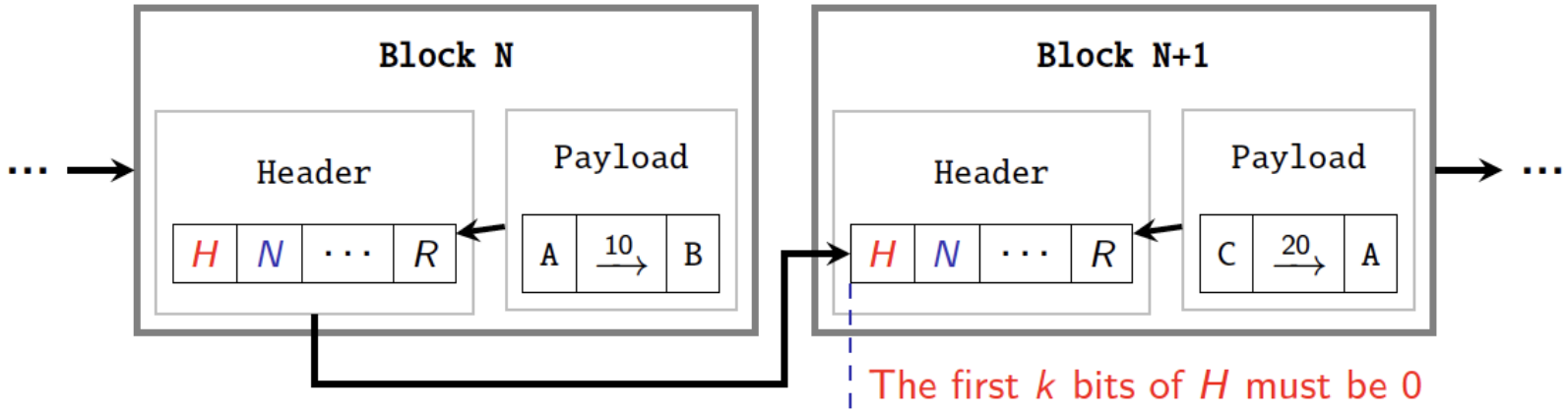
- It is not hard at all for Alice to revert this payment to Bob!



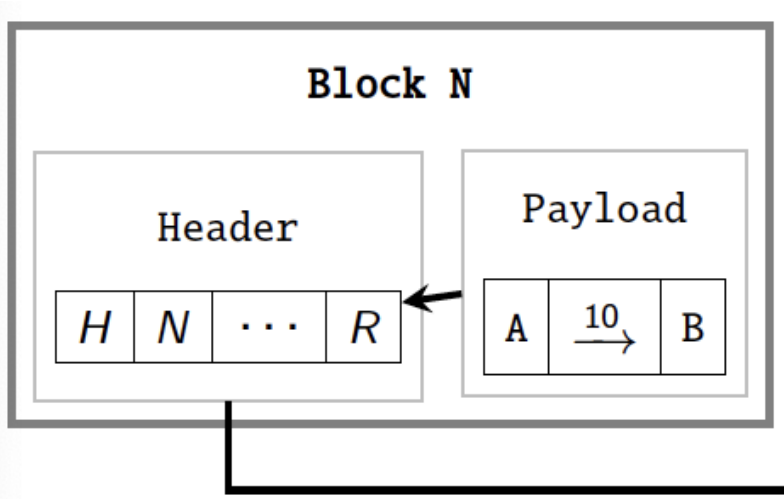
Let’s increase the difficulty



- Bob decides to make it harder for Alice to alter her payment



Mining



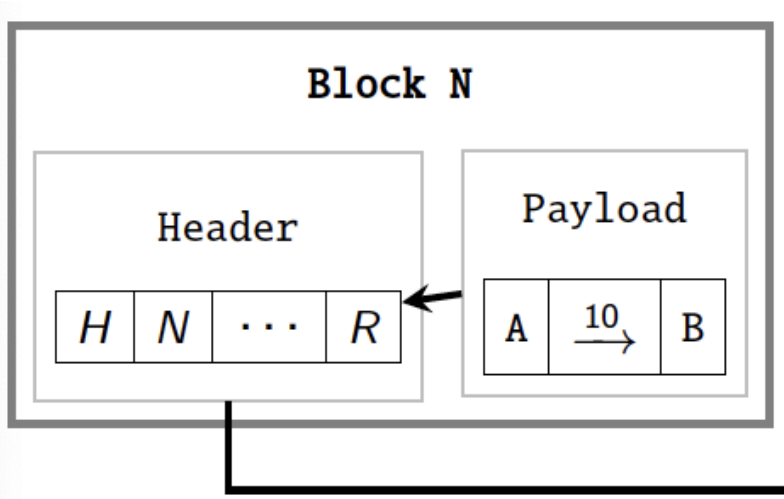
$N = 0 \implies Hash(H \,||\, N \,||\, \ldots \,||\, R) = \texttt{0x349c1a7e...}$ ×

$N = 1 \implies Hash(H \,||\, N \,||\, \ldots \,||\, R) = \texttt{0x6ffde7bf...}$ ×

.....

$N = x \implies Hash(H \,||\, N \,||\, \ldots \,||\, R) = \texttt{0x00.}_k\texttt{.004f7fed1a}$

Mining



$N = 0 \implies Hash(H \,||\, N \,||\, \ldots \,||\, R) =$ `0x349c1a7e...` ×

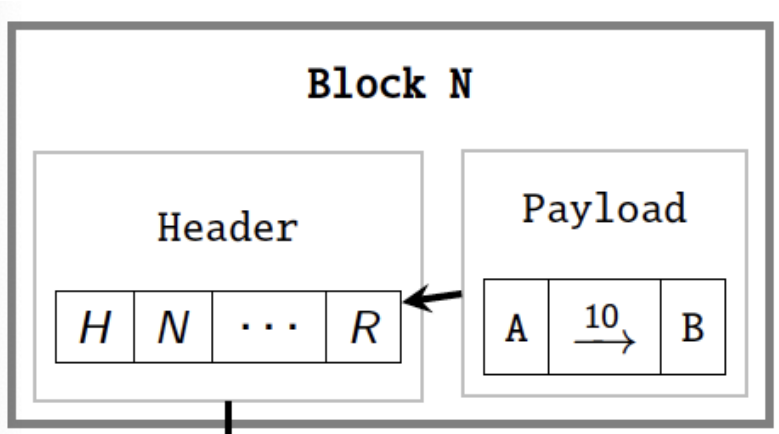
$N = 1 \implies Hash(H \,||\, N \,||\, \ldots \,||\, R) =$ `0x6ffde7bf...` ×

.....

$N = x \implies Hash(H \,||\, N \,||\, \ldots \,||\, R) =$ `0x00.`$_k$`.004f7fed1a`

Clearly, N is not required to increase monotonically—but can you predict the consequences if it did?

Mining



$N = 0 \implies Hash(H \,||\, N \,||\, \ldots \,||\, R) = \texttt{0x349c1a7e...}$ ×

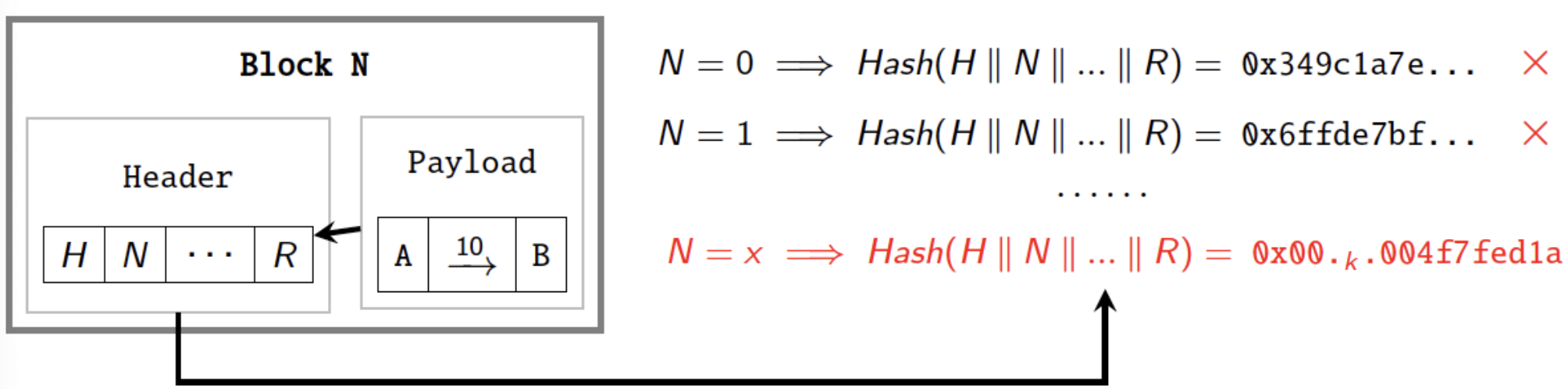
$N = 1 \implies Hash(H \,||\, N \,||\, \ldots \,||\, R) = \texttt{0x6ffde7bf...}$ ×

.....

$N = x \implies Hash(H \,||\, N \,||\, \ldots \,||\, R) = \texttt{0x00.}_k\texttt{.004f7fed1a}$

What is the chance of finding a valid N assuming an m-bit binary hash?

Mining for a valid hash

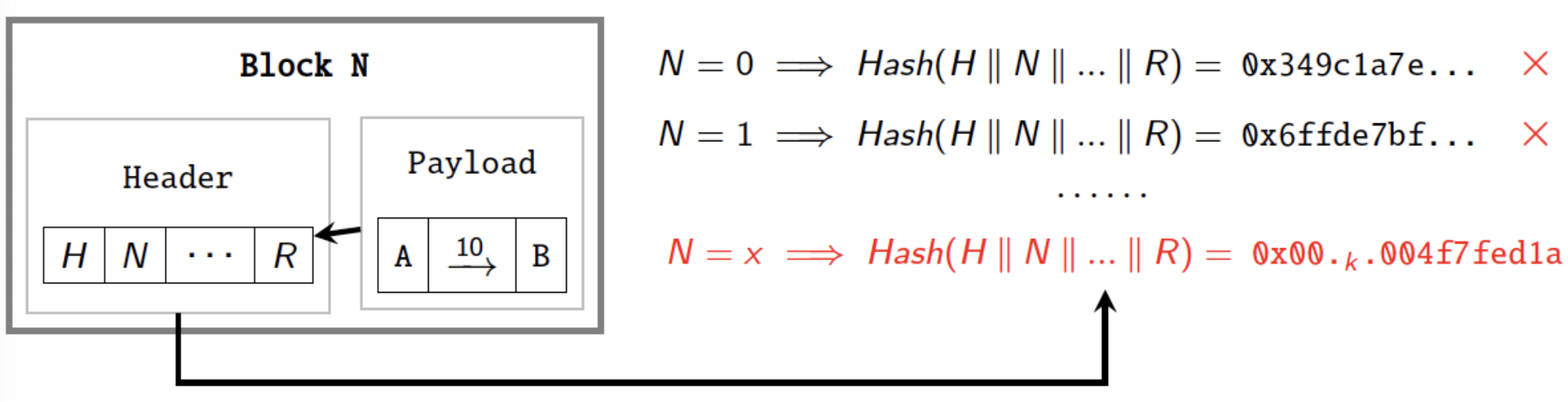


What is the chance of finding a valid N assuming an m-bit binary hash?

A: $\frac{2^{m-k}}{2^m}$, a larger k ⇒ a higher difficulty of finding N

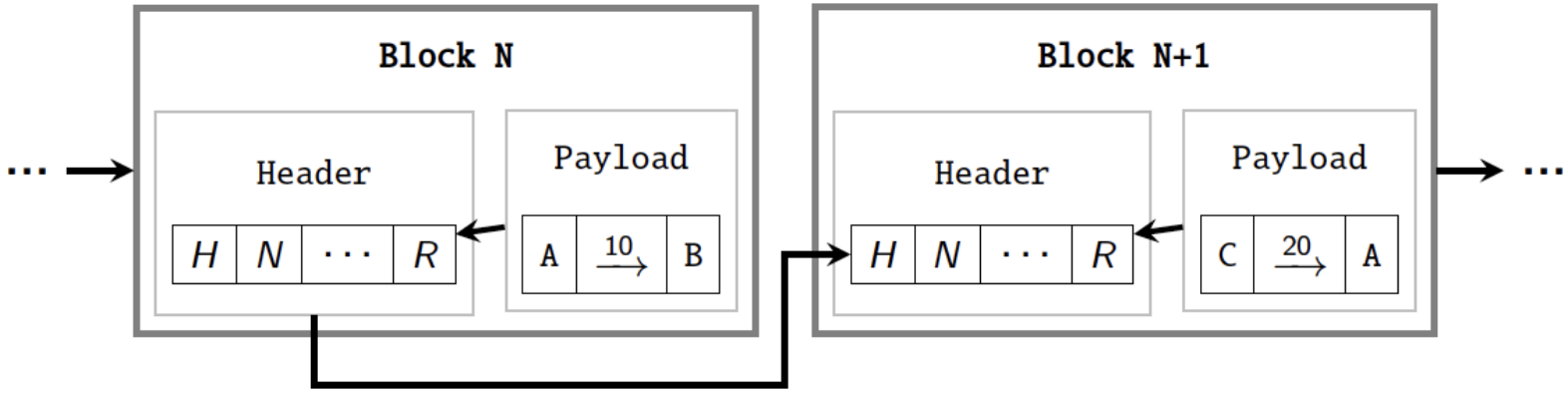
Expect 2^k hash operations to find a valid N

Mining

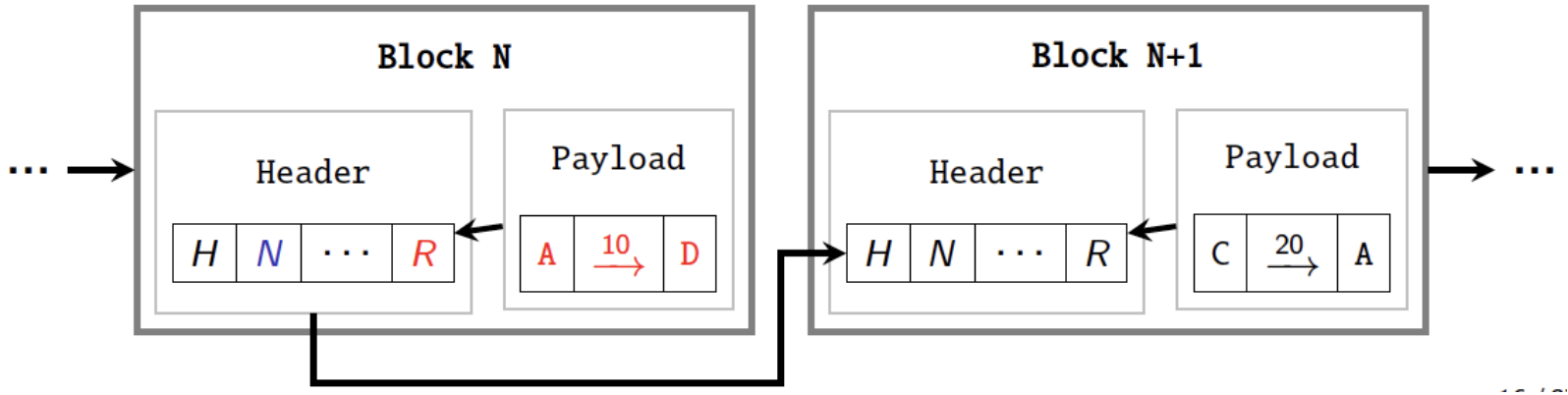


- Example: Bitcoin uses double **SHA-256** hashing of block header**:** SHA-256(SHA-256(header))
- The network adjusts the mining difficulty approximately every 2016 blocks to maintain an average block time of 10 minutes.
- Incentive: Block Reward and Transaction fees.

Hack The Mining– Case 1

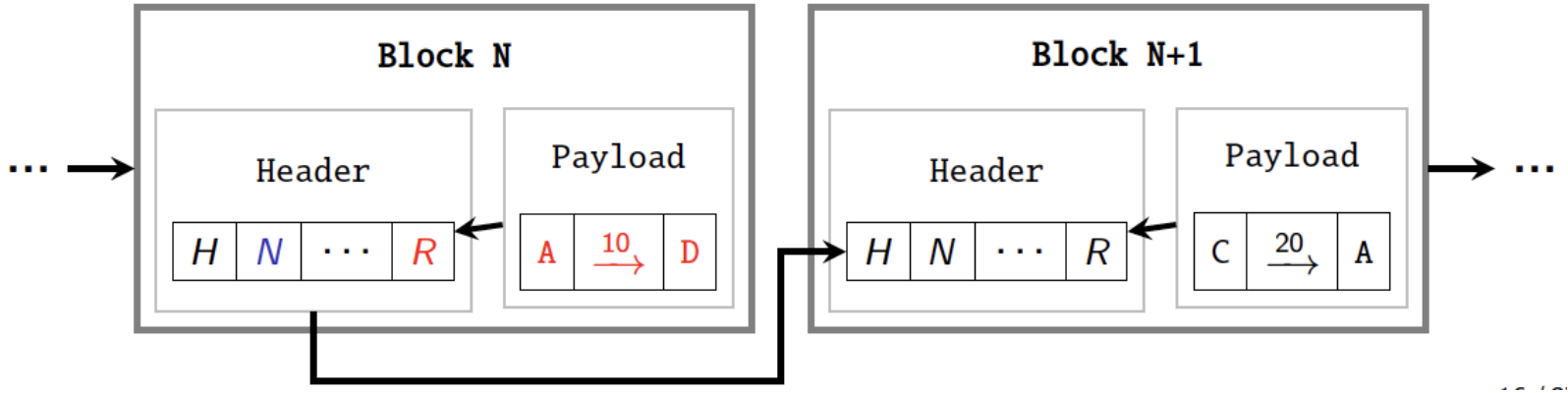


- Surgical change: Alice re-mines block *N* and finds a new nonce such that the block header hash in block *N+1* remains unchanged



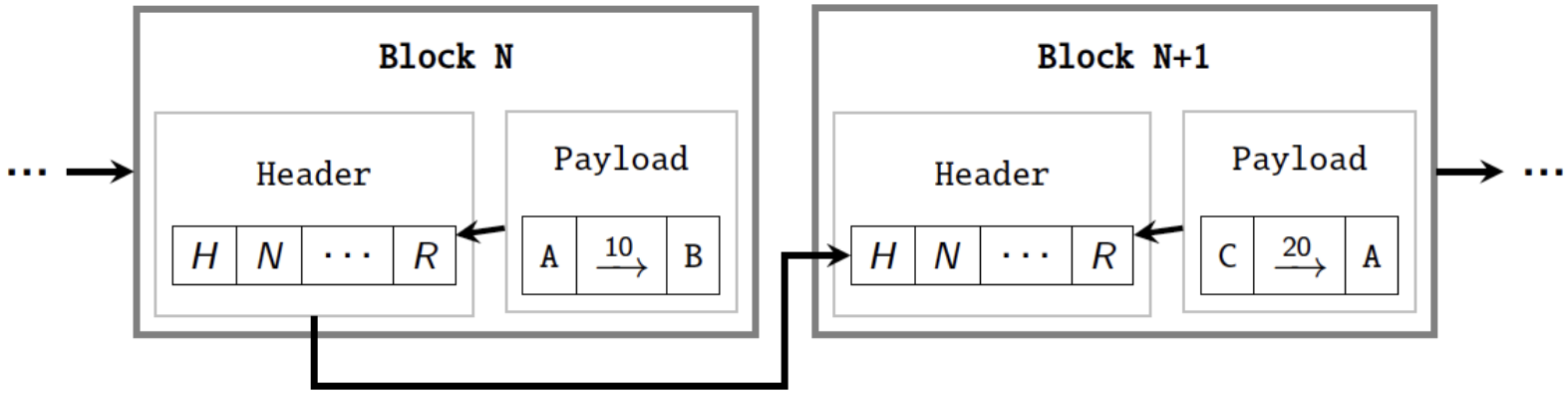
Hack The Mining– Case 1

- Surgical change: Alice re-mines block *N* and finds a new nonce such that the block header hash in block *N+1* remains unchanged

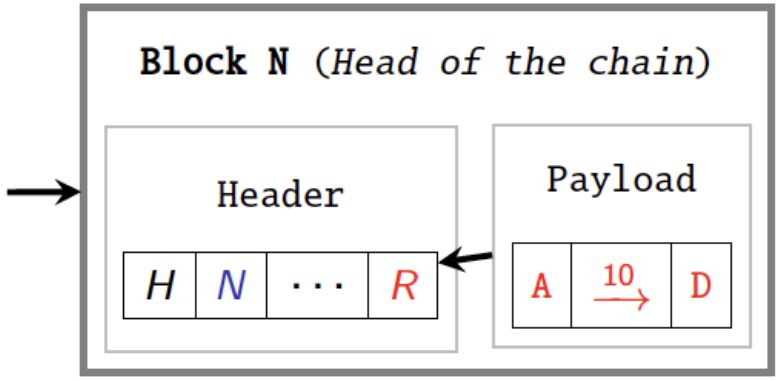


- Deterrent: This is extremely hard for a cryptographic hash function that has preimage resistance and second-preimage resistance.

Hack The Mining– Case 2

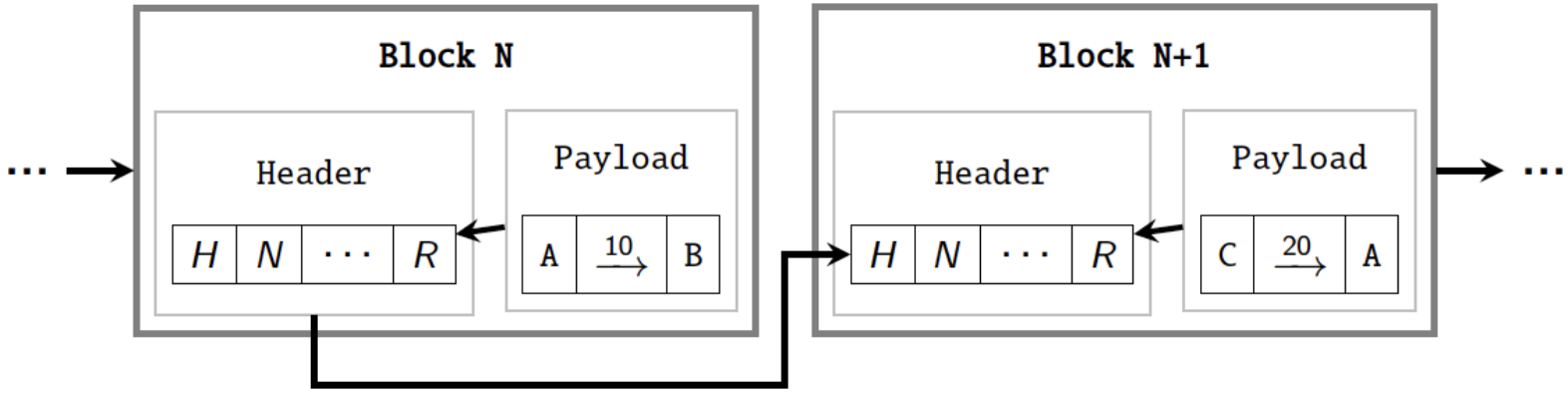


- Change-and-cut: Alice re-mines the nonce for block *N* and stops

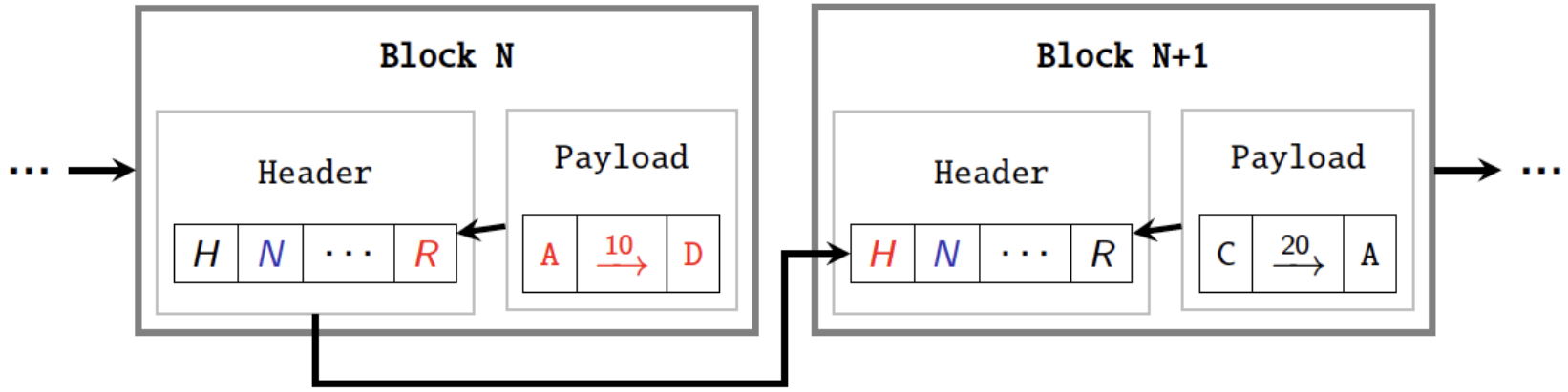


- Deterrent: Longer chains are preferred over shorter chains.

Hack The Mining– Case 3

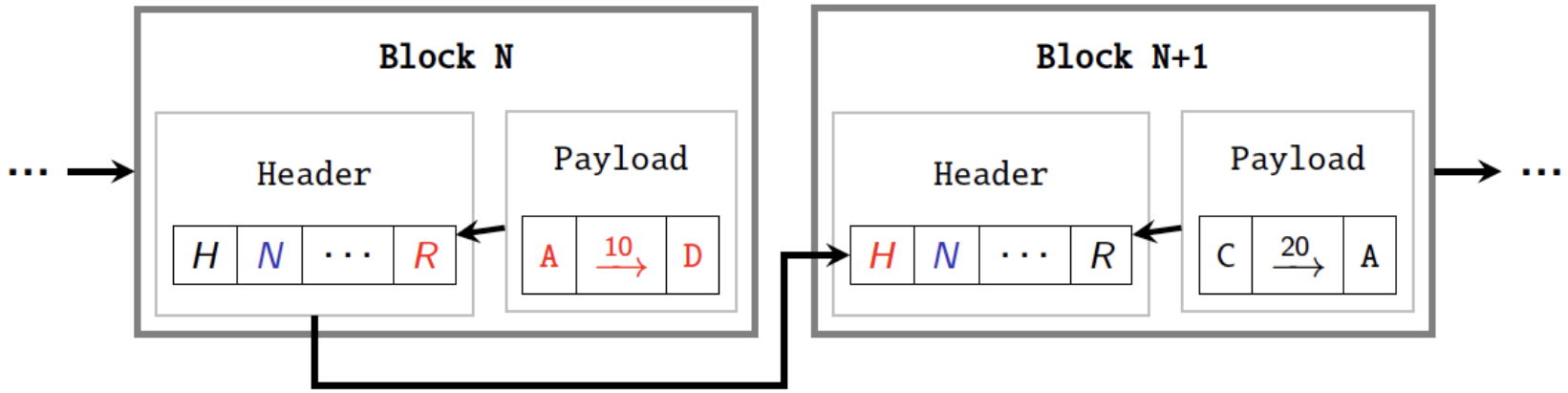


- Partial chain re-mining: Alice re-mines all the nonces *to the current head.*



Hack The Mining– Case 3

- Partial chain re-mining: Alice re-mines all the nonces since block *N*

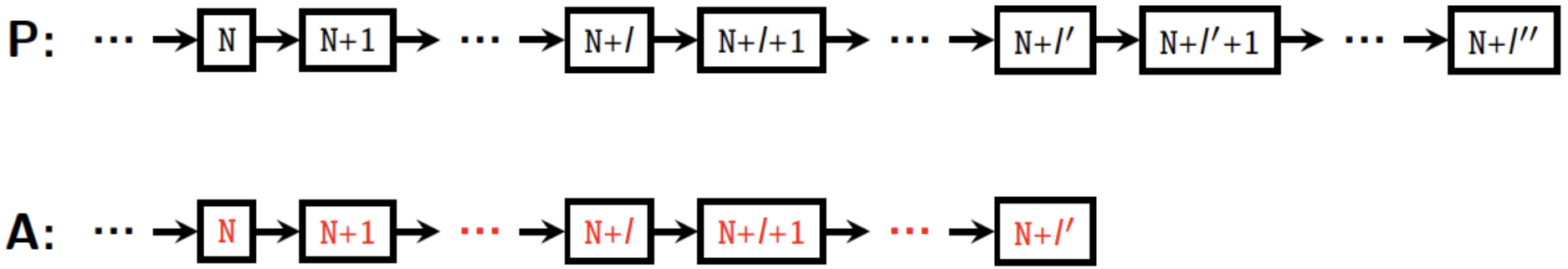


- Deterrent: If there are *L* blocks after N to the head, Alice is expected to perform $L \times 2^k$ hash operations to build up an equally competitive chain. *assuming a fixed k.

This is extremely difficult, yet still possible.

The 51% attack

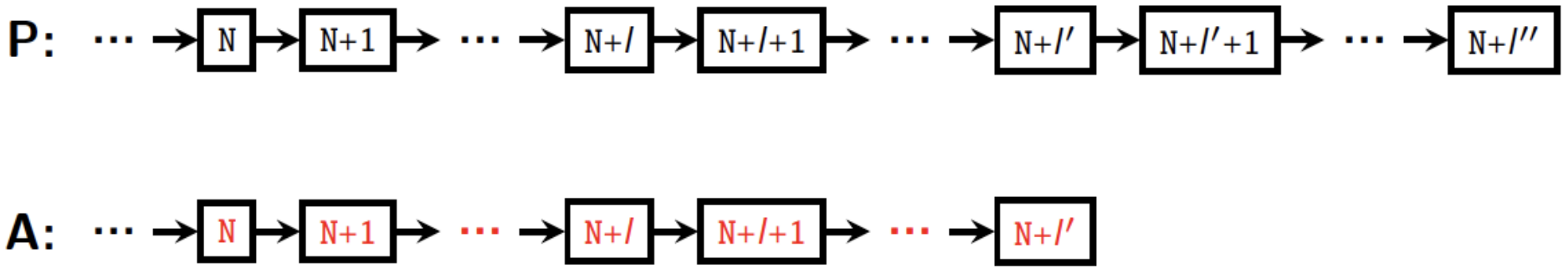
- To prevent Alice from mining all the blocks:
  - Alice needs to mine slower than the rest of the participants combined.



→ If Alice controls the majority of computational power, she can ultimately rewrite the history.

The 51% attack

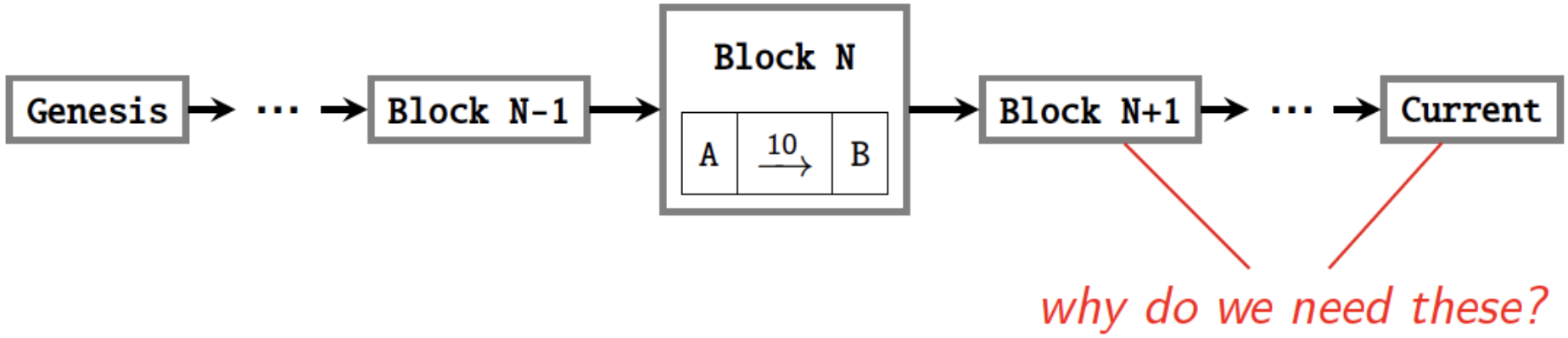
- To prevent Alice from mining all the blocks:
  - Alice needs to mine slower than the rest of the participants combined.



→ If Alice controls the majority of computational power, she can ultimately rewrite the history.

Does Alice necessarily need 51% to rewrite history? Let's see!

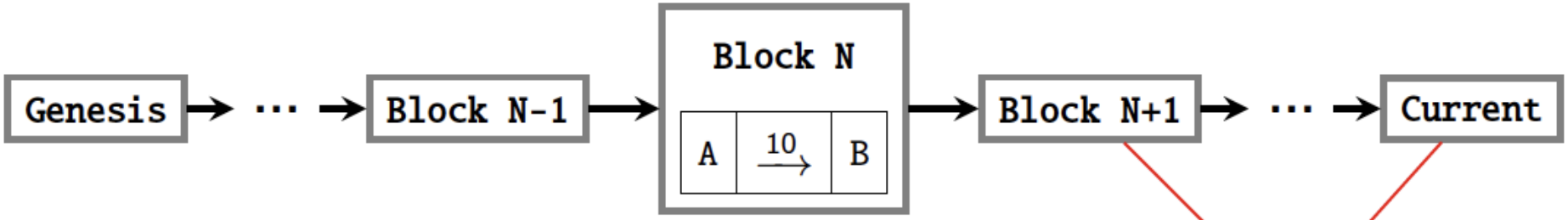
Confirmation level



It is always advisable to consider only transactions from blocks that reside on the longest branch of a fork and have been further confirmed by additional blocks.
Even when nobody has 51% of the power!
But why?

Confirmation level

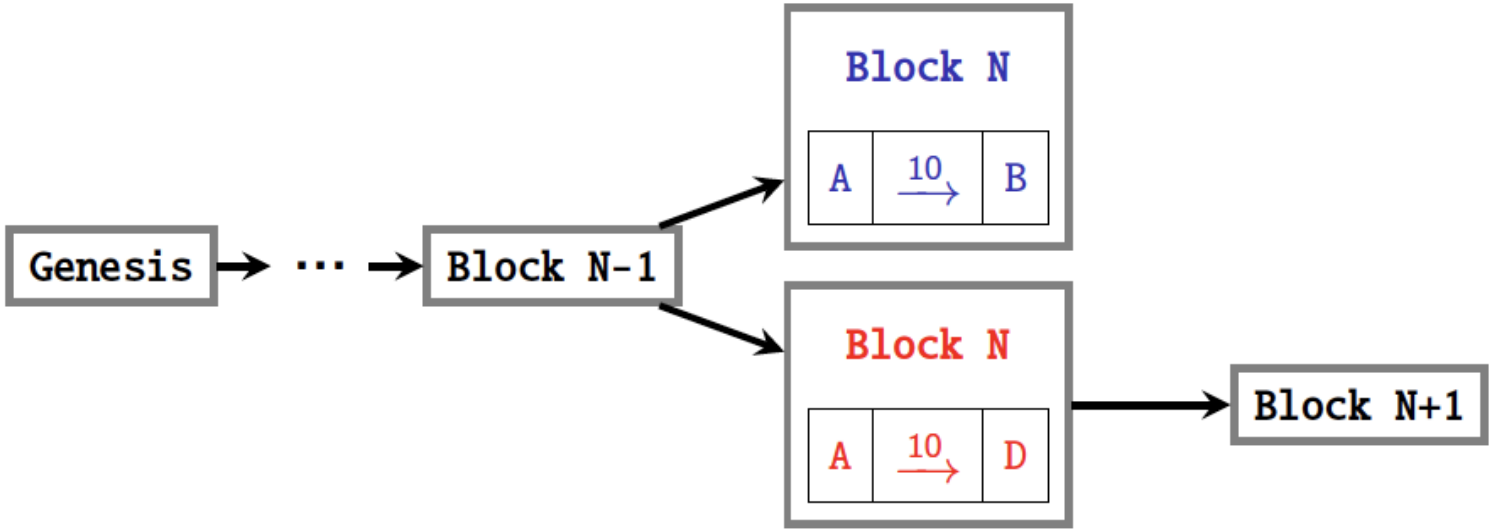
- Recall that when we show a proof of payment, we need a few extra blocks after the block that hosts the ledger entry.



1. Mining is non-deterministic. Anybody, regardless of power, can (probably) get lucky enough to mine a block before others at some point in history.

2. The network is fragmented, with miners competing rather than uniting. Someone with 10% of power might be the most powerful node.

How to trigger a fork?



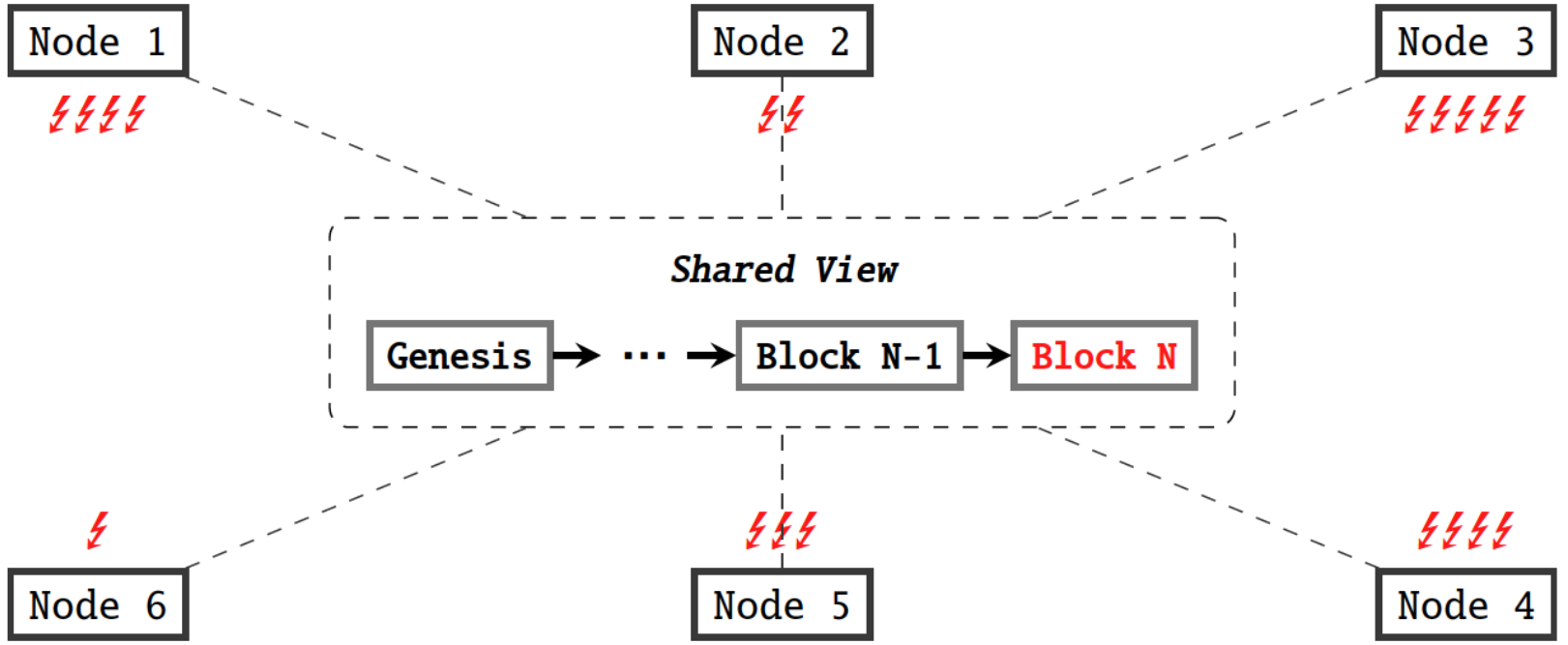
- To trigger a fork, Alice could:
  - Send two transactions to two different nodes in a short time window.
  - Send two transactions to two nodes located in separate sectors of the network.
  - Selfish Mining: If mined a block faster than others, keep it and build on top of it and only publish it after another block is published by someone else.

Drawbacks of Proof-of-work consensus

- Speed of confirmation
  - E.g., a Bitcoin transaction takes on average 10 minutes to confirm.
  - Even worse, it is advised to wait for 6 confirmations, i.e., an hour.
- Vulnerable to 51% attacks
  - In 2014, mining pool Ghash.io obtained 51% hash rate in Bitcoin.
  - Bitcoin Gold was hit by such attacks twice in 2018 and 2020.
- Energy consumption
  - PoW is a race of power, relying on the assumption that the collective power of the honest network surpasses a malicious node.
  - PoW burns energy and negatively impacts the environment.

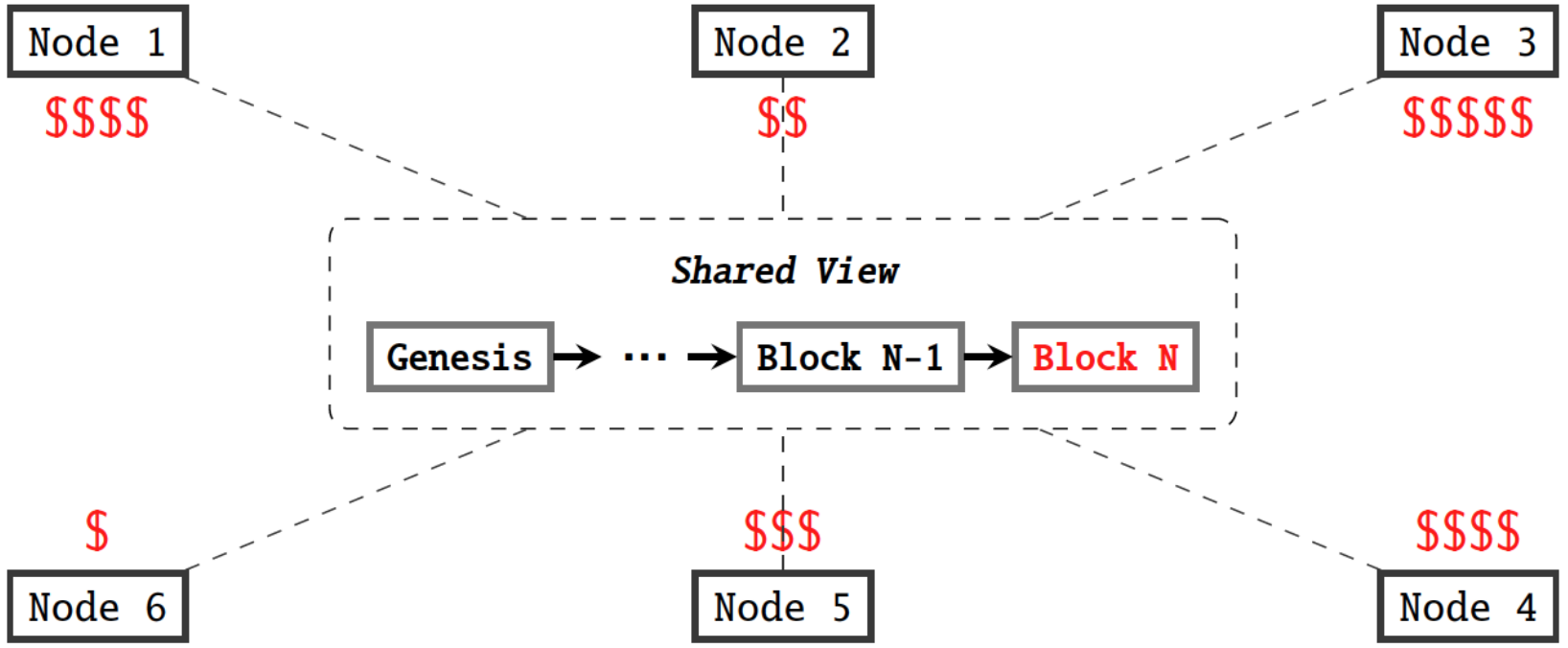
Proof-of-stake

PoW: Hash Power ~ Chance to be the next leader



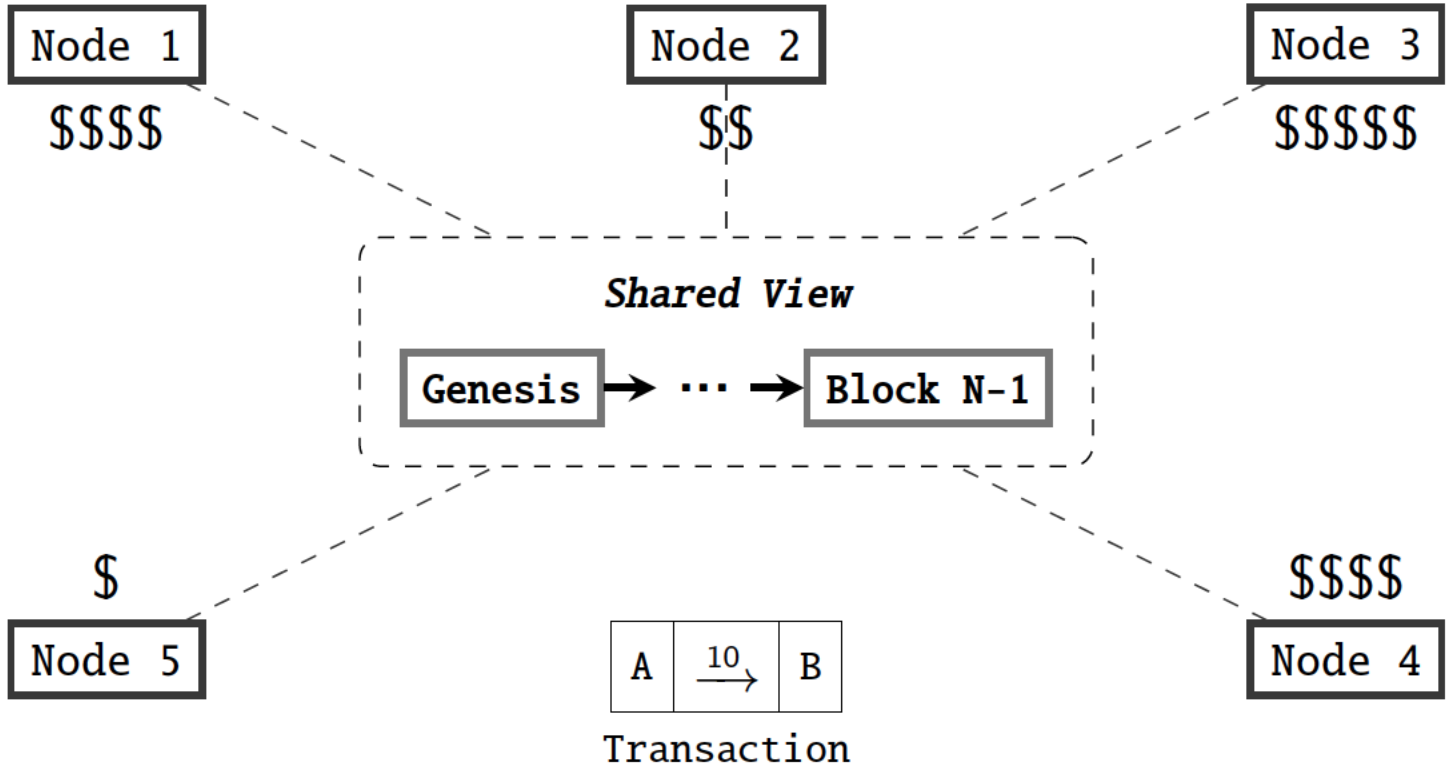
- In a proof-of-work scheme:
  - Collisions are allowed. Every node can publish a mined block, and collision is resolved by the longest chain rule.

PoS: Staked Value ~ Chance to be the next leader

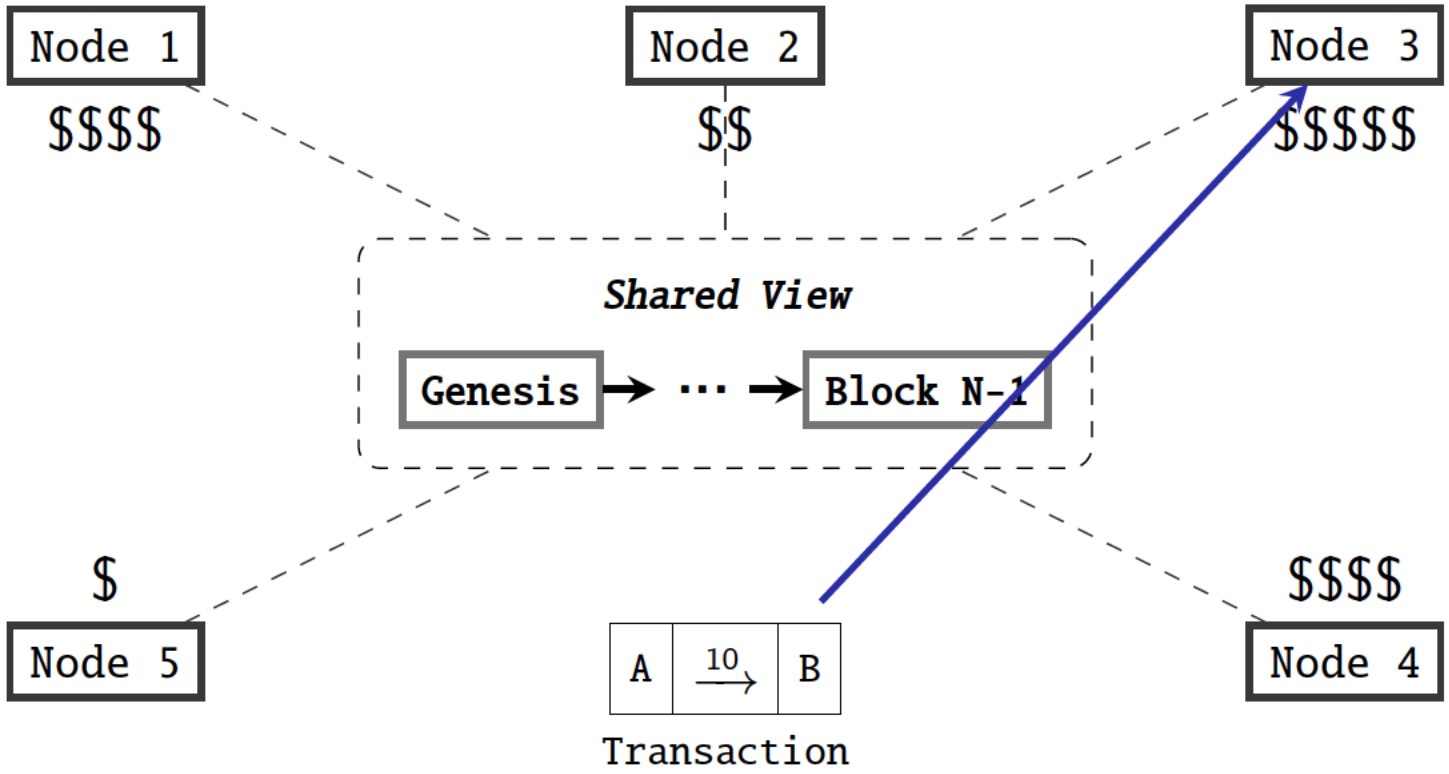


- In a proof-of-stake scheme:
  - Collisions are not allowed by design, only the elected leader proposes a block.

Transaction lifecycle in PoS

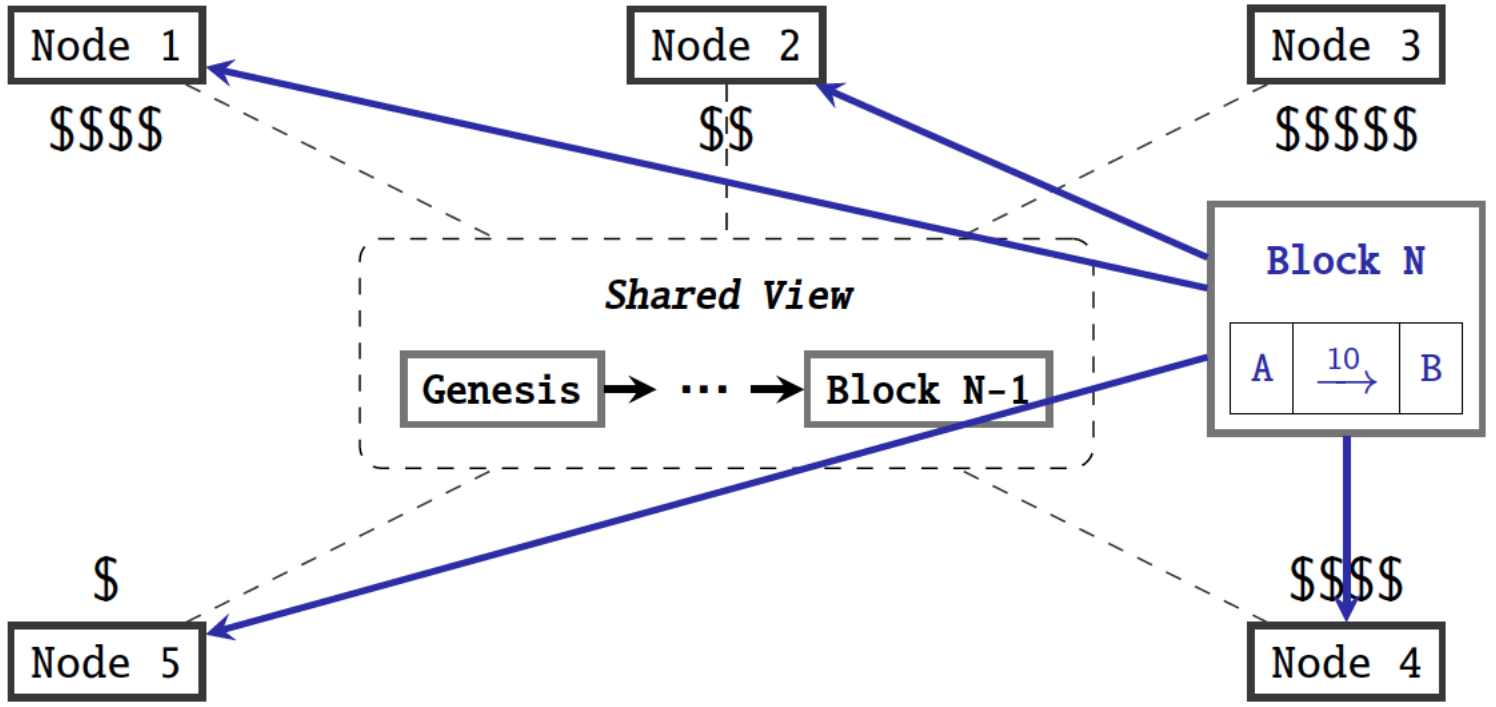


Transaction lifecycle in PoS



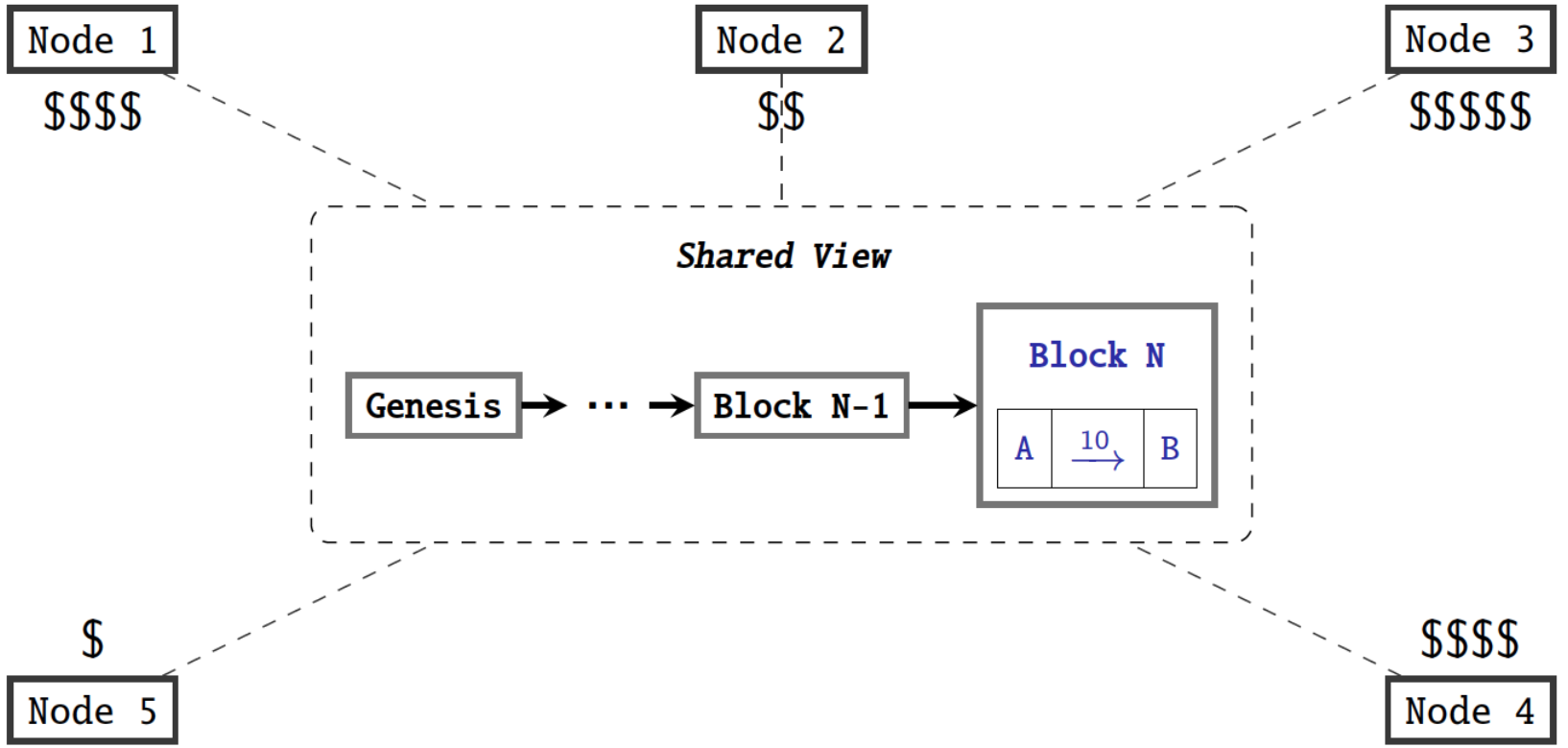
Every node sends transactions to the elected leader so it can publish the block.

Transaction lifecycle in PoS

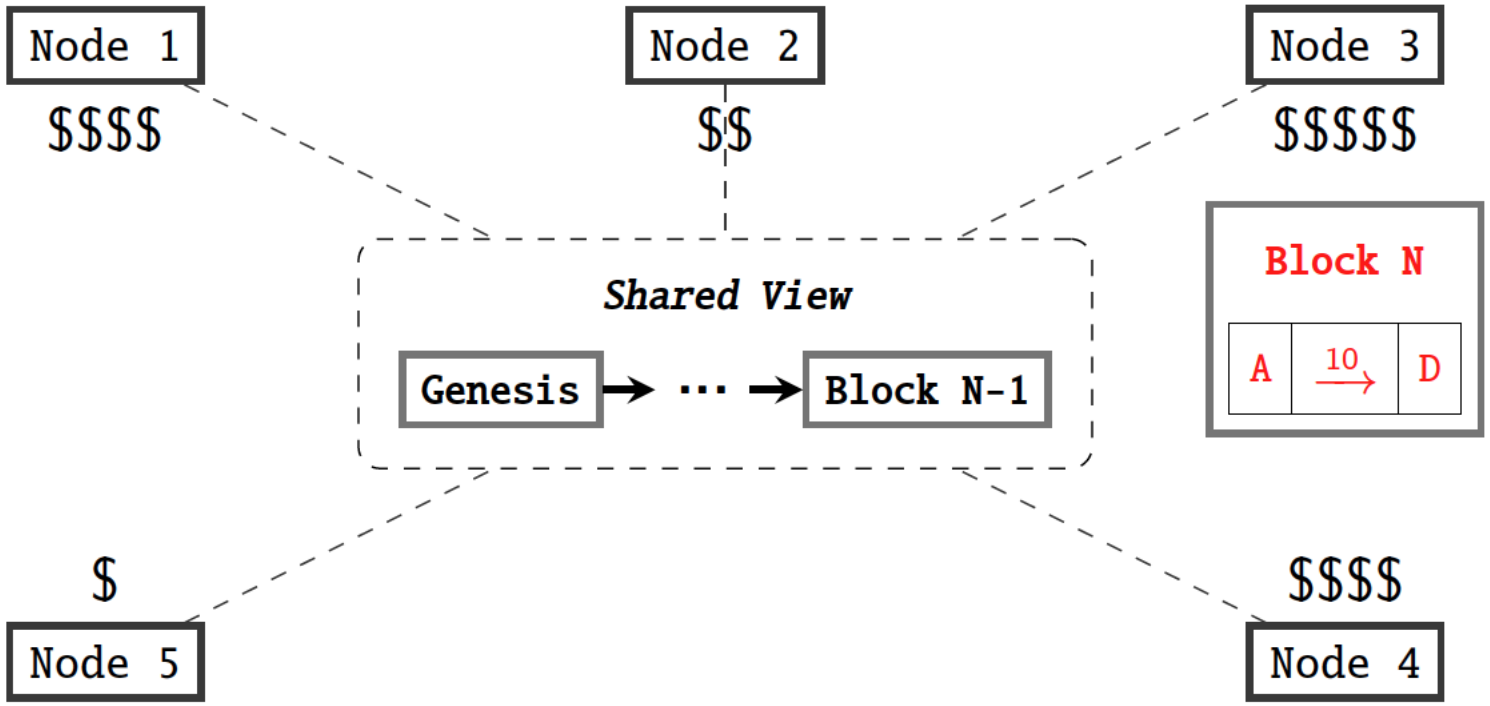


Elected leader publishes the block.

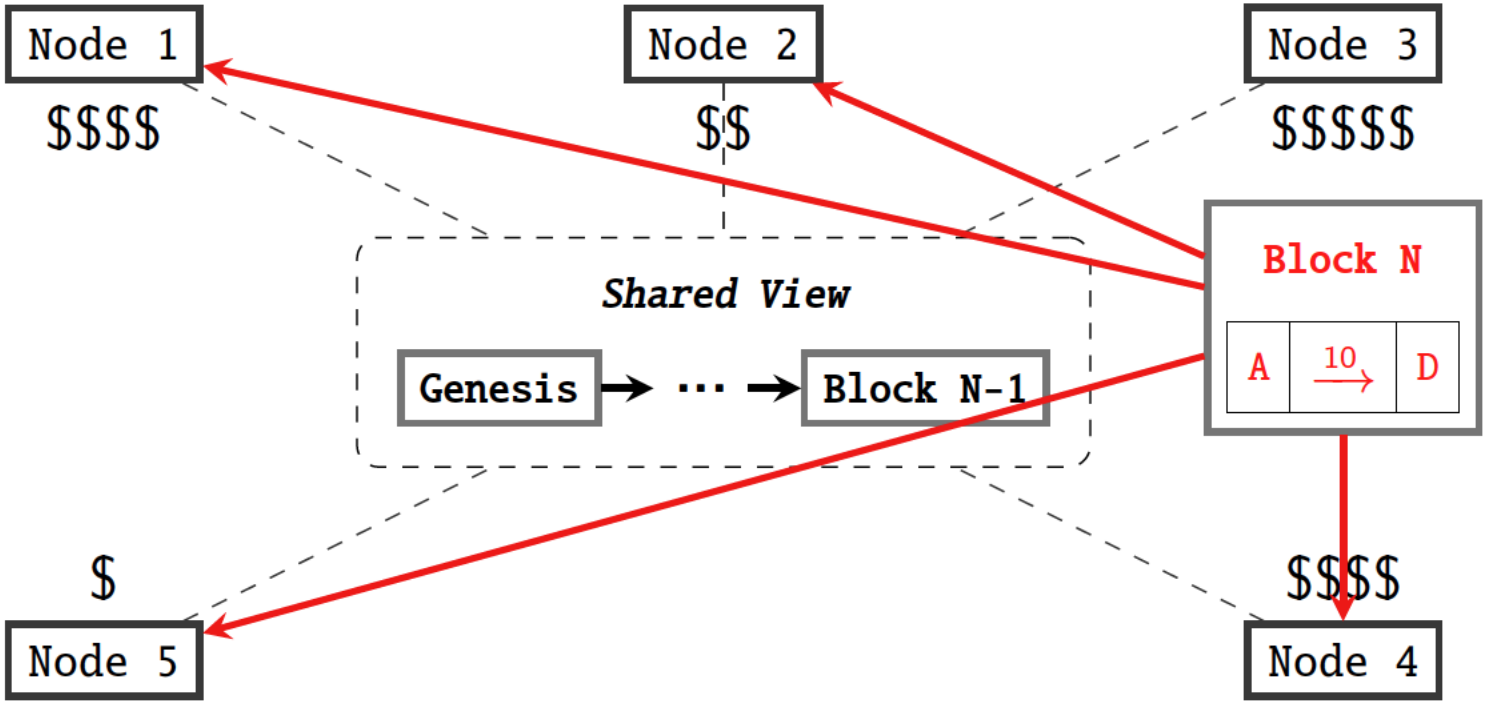
Transaction lifecycle in PoS



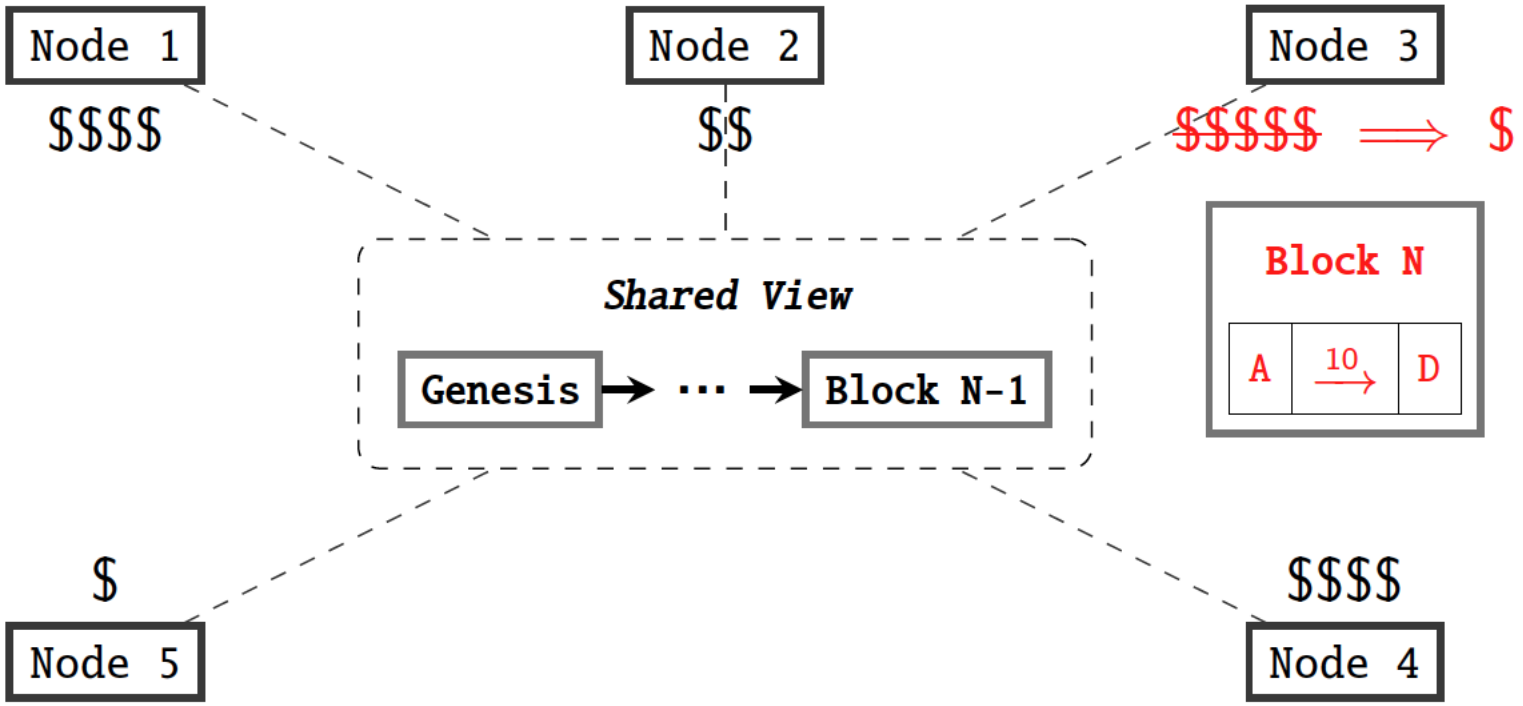
Transaction lifecycle in PoS: Misbehaviour



Transaction lifecycle in PoS: Misbehaviour



Transaction lifecycle in PoS: Misbehaviour



A node can be punished for misbehavior! Its stake can get slashed (burned, redistributed or confiscated).

Catching lies

- How?
  - Mostly automatically detected and enforced by the blockchain.
  - In some PoS blockchains (e.g., Polkadot, Cosmos), specialized nodes (sometimes called “fishermen”) can report bad behavior.
  - In rare cases, some blockchains with on-chain governance (e.g., Polkadot), can make governance-driven slashing where stakeholders (often via token-weighted voting) decide if a node should be punished for behavior that is harder to detect automatically.

Fun fact: Burning stake usually affects the price of the gas coin positively and makes every other stake holder happy!

The 51% attack on PoS

- **Q:** What if the attacker controls ≥ 50% of staked resources?

The 51% attack on PoS

- **Q:** What if the attacker controls ≥ 50% of staked resources?
- **A:**
  - **1-** Attacker doesn’t need to create an obvious fork but just censor certain transactions by excluding them from blocks.
  - **2-** The attacker can sometimes reorg without being slashed by carefully following the protocol rules.

The 51% attack on PoS

- **Q:** Is the 51% attack less likely in PoS compared with PoW?

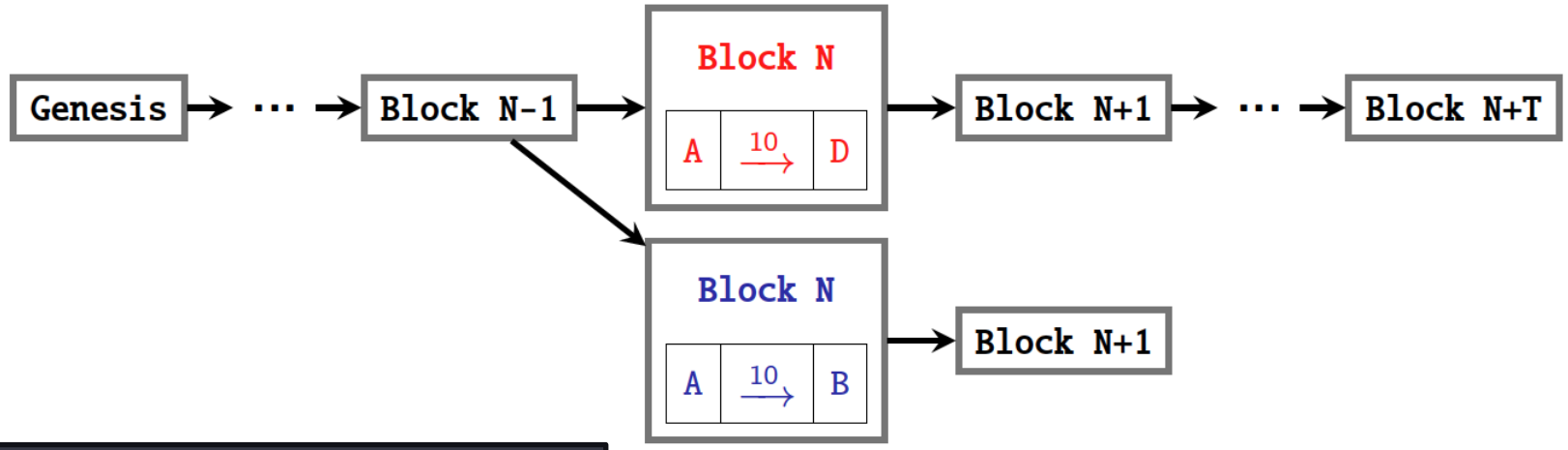
The 51% attack on PoS

- **Q:** Is the 51% attack less likely in PoS compared with PoW?
- **A:** Yes, because in PoS not obeying some rules causes slashing. If that happens the attacker loses the weapon to future attacks and assets are not easily recoverable!

Good to mention that a similar scenario sometimes happens for PoW. When mining pools misbehave, they might lose their reputation in the community and miners will migrate.

51% attack: Recovery with Hard Fork

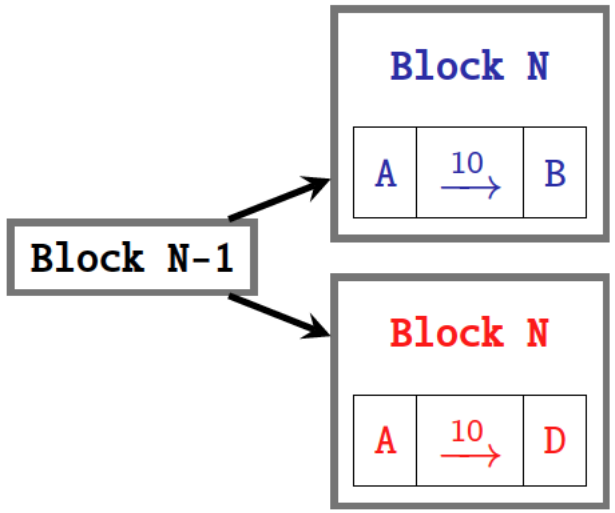
- To recover from a 51% attack, the only solution is to hard fork the blockchain in order to roll-back the fraudulent transactions added by the attackers.



Q: What are the challenges of the community to make a hard fork?
A: Social coordination

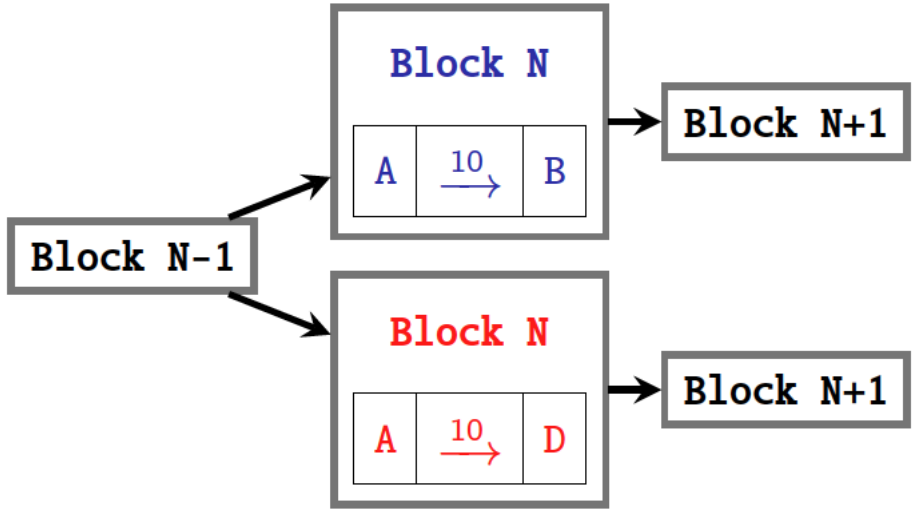
PoS Problems: Nothing-at-Stake

- Alice has some small stake (e.g., 1%) and can be elected as a leader:



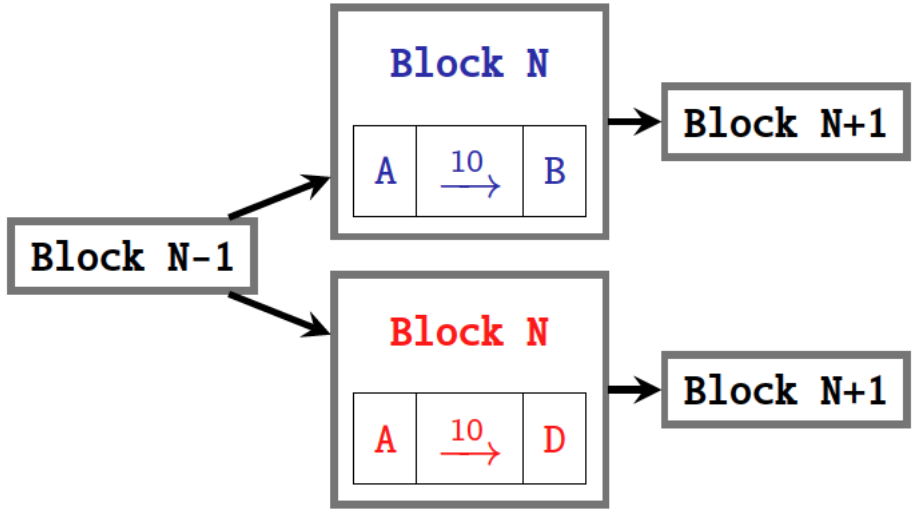
- As a leader, Alice triggers a fork and signs two conflicting blocks (double-signing).

PoS Problems: Nothing-at-Stake



- The next honest leader has no incentive to select which chain to converge on. The proposer has no idea which chain will survive in the future; the logical choice would be to mine on both.

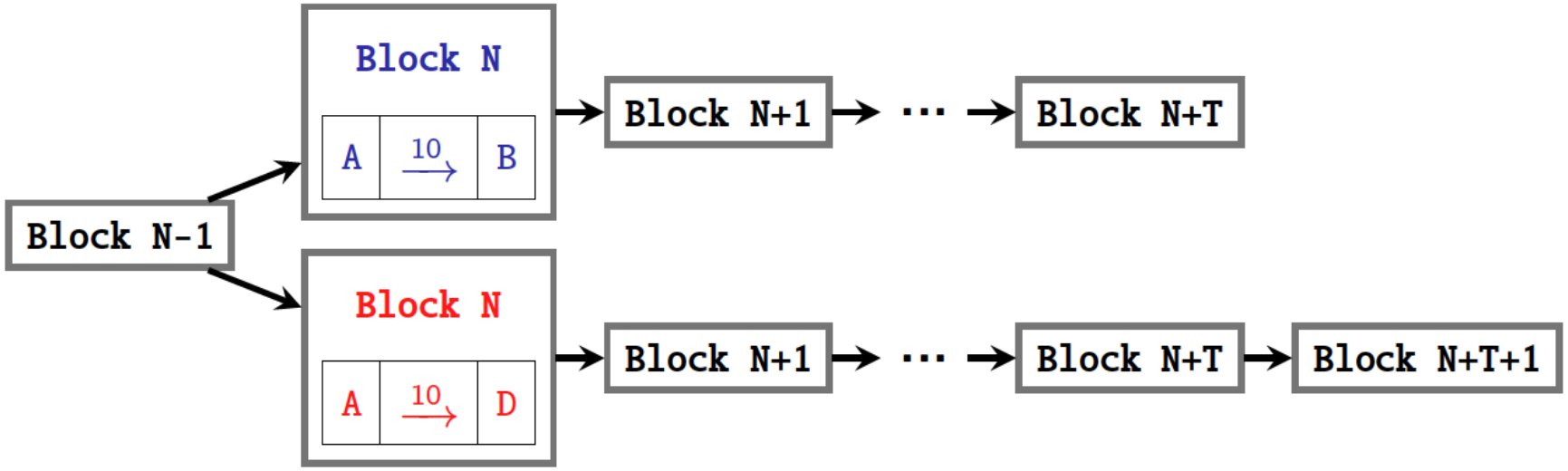
PoS Problems: Nothing-at-Stake



- The next honest leader has no incentive to select which chain to converge on. The proposer has no idea which chain will survive in the future; the logical choice would be to mine on both.

Q: Why does this dilemma not exist in PoW?

PoS Problems: Nothing-at-Stake



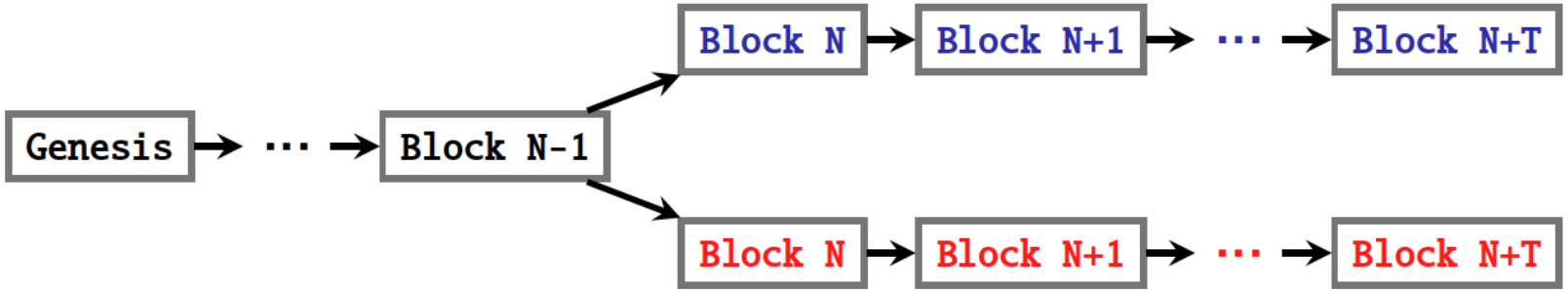
- When its Alice's turn again, she only appends a block to the chain that is more favorable to her. The other chain dies as a result.
- This is sometimes called the 1% attack.

PoS Problems: Nothing-at-Stake

- Solution? There is no common solution. Different PoS chains adopt different mechanisms.
- The Slash protocol in most PoS protocols has three rules:
  - Double Signing (Equivocation) → Major Slashing [tries to handle Nothing-at-Stake]
  - Downtime / Liveness Failure → Minor Slashing (or Just Penalties)
  - Surround Voting (In Some PoS Networks) → Major Slashing
    - A validator **votes for an older block while pretending it's current**, trying to **rewrite history**.

PoS Problems: bootstrapping

- A node could forge an entire chain by itself (called a long-range attack).
- If Bob, a new user, joins the network, which chain should he accept?



Q: Why is this not a problem in PoW?

PoS Problems: bootstrapping

- Solution? In short, there are no simple solutions.
  - Casper (Ethereum's PoS protocol) depends on trusted nodes to broadcast the correct block hash.
  - Peercoin broadcasts the hash of the "legitimate" chain on a daily basis.

a centralized point of trust, is betraying the very principles of decentralization!

  - Ouroboros Genesis: a solution by "Cardano" which has no centralized point of trust.

Questions?