CS489/698 Privacy, Cryptography, Network and Data Security

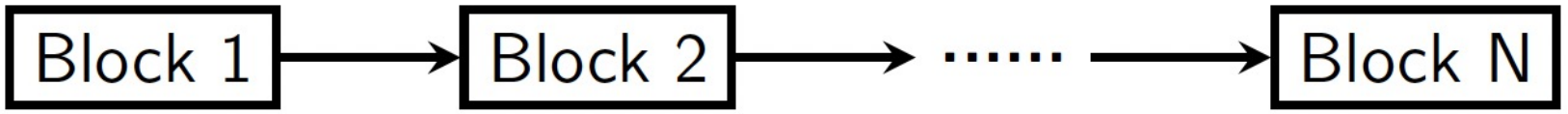
Blockchain

Spring 2024, Monday/Wednesday 11:30am-12:50pm

An overview of blockchain design

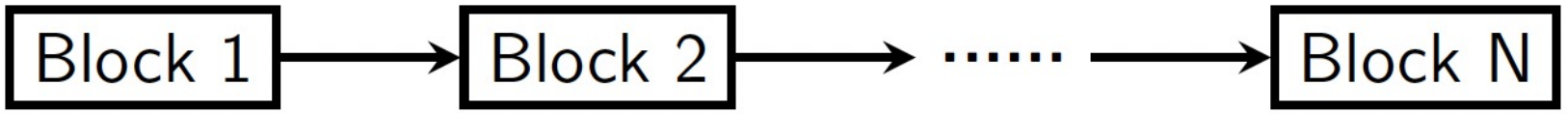
What is a blockchain?

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



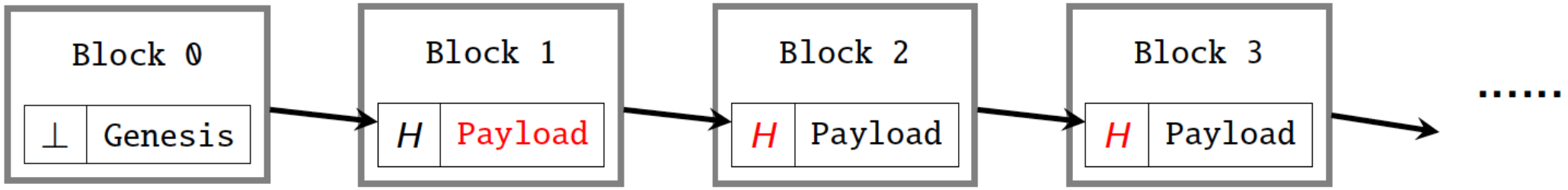
What is a blockchain?

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



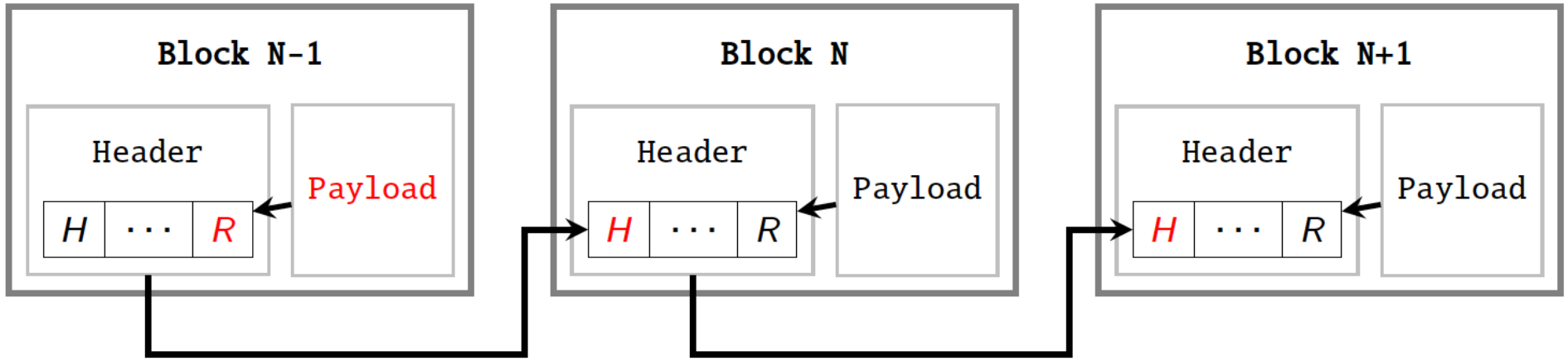
  - What does chaining mean here?
    - Linked list? Some cryptographic construct?
  - What goes into these blocks?
    - Anything? A fixed format? What makes a block valid?
  - Who can put up a block?
    - A single entity? A group of people? Anyone with Internet access?
  - How to ensure a same view of the chain?
    - Centralized? Distributed? How to resolve a dispute?

A basic chaining scheme



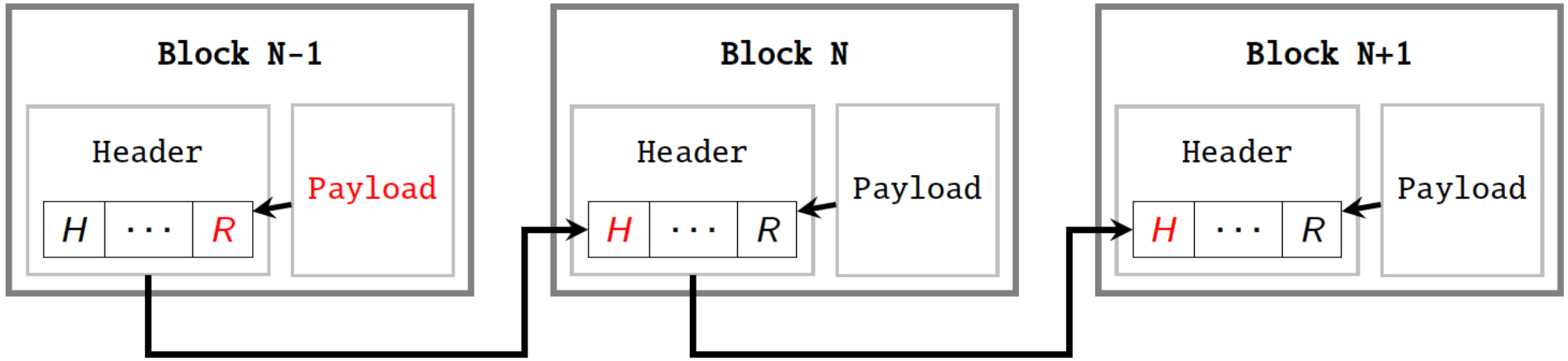
- Each block contains a cryptographic hash of the previous block
- Each block depends on the previous one

A basic chaining scheme



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

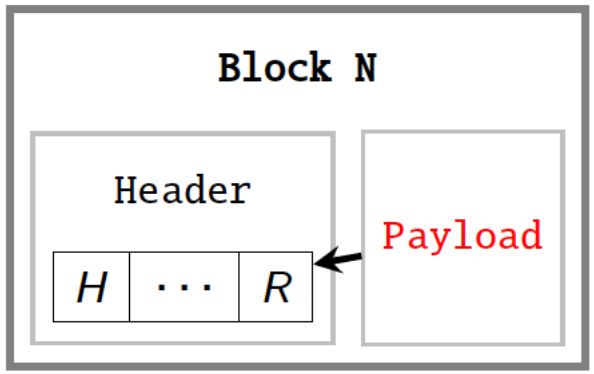
A basic chaining scheme



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

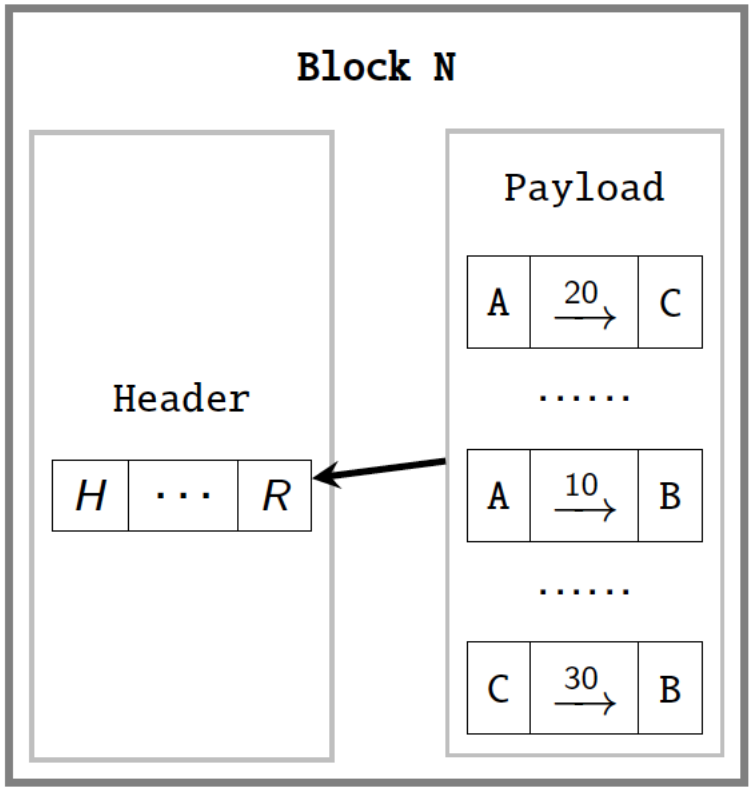
Q: Why is this a better chaining scheme?

What goes into the payload?

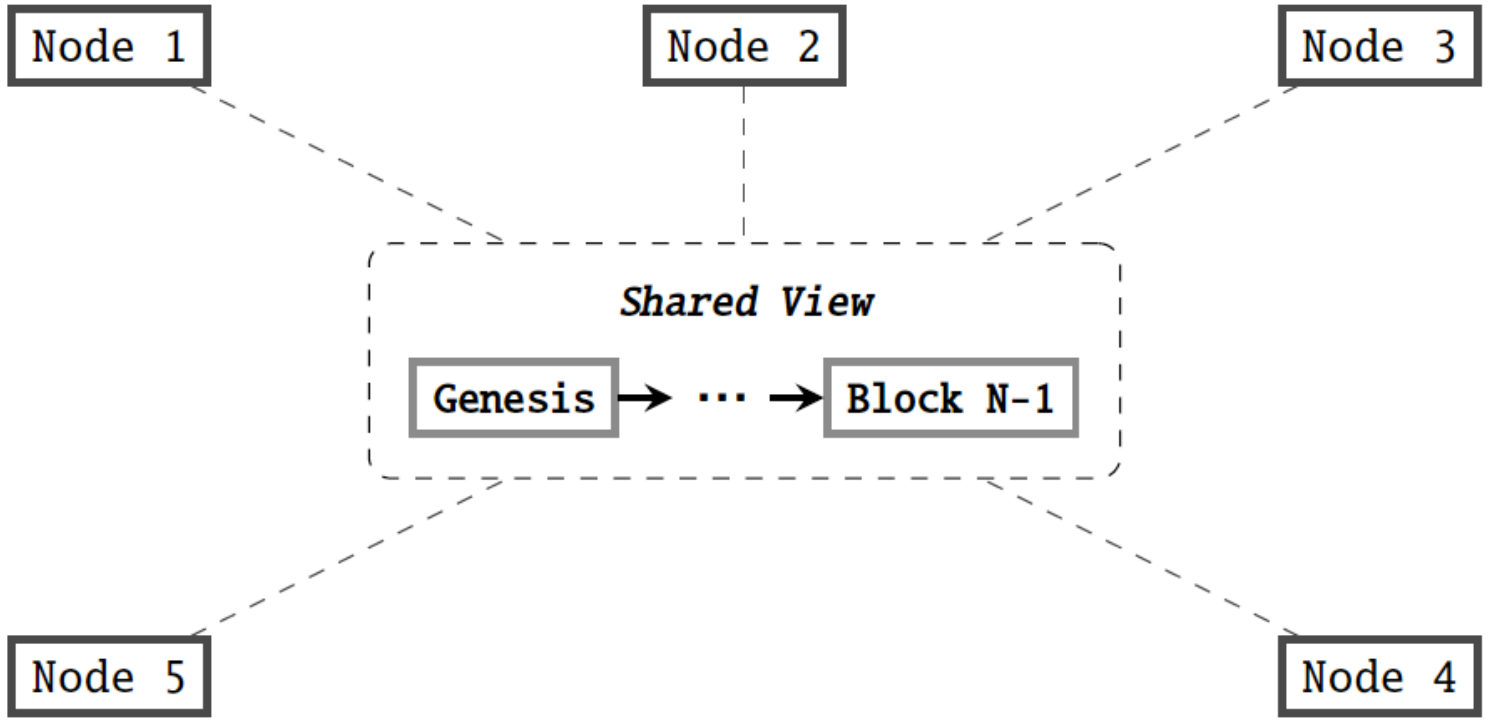


- Anything! Depending on how you plan to use this blockchain.
  - Bitcoin blockchain: ledger
  - Ethereum blockchain: state machine

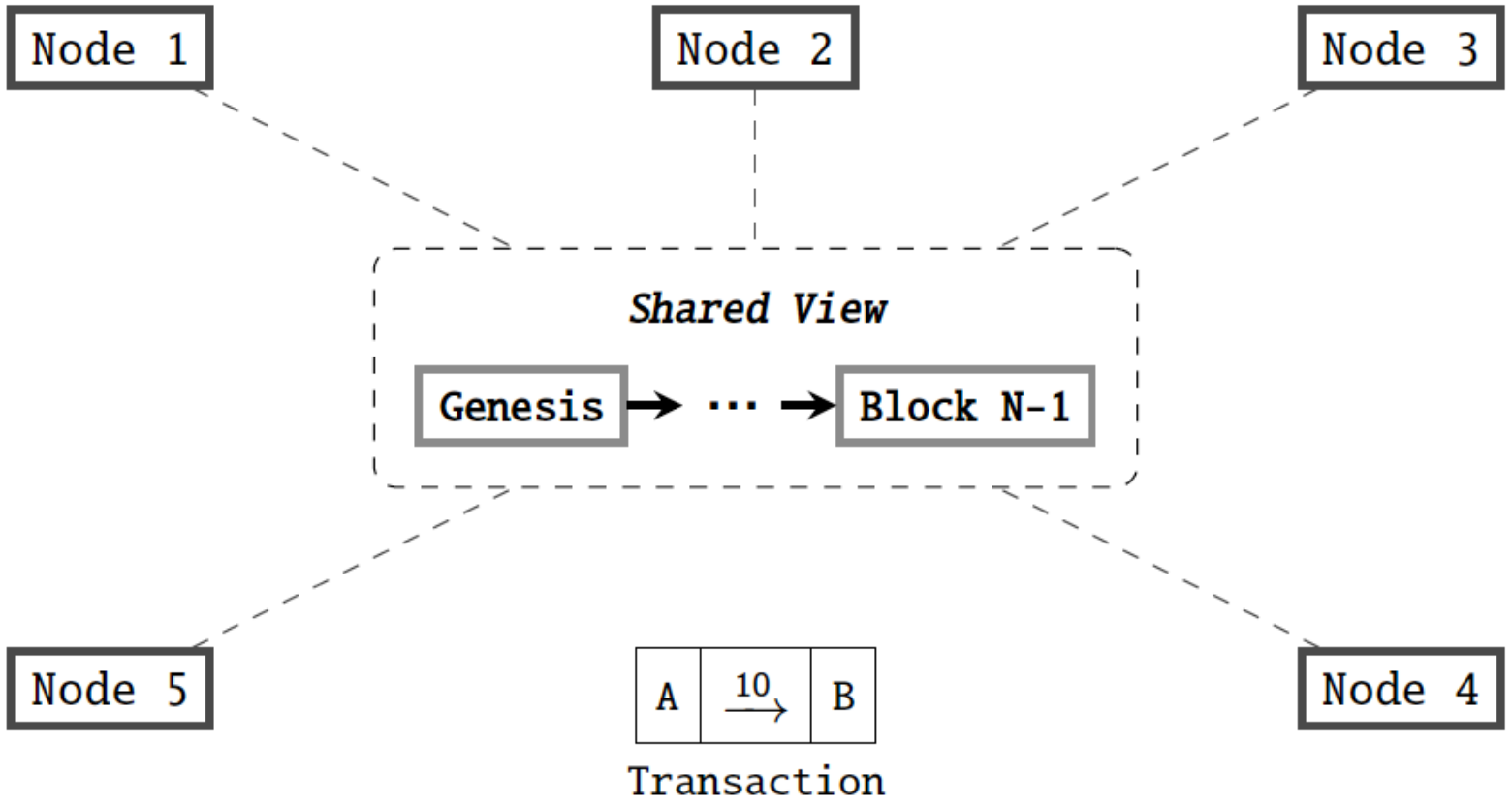
Payload example: a ledger



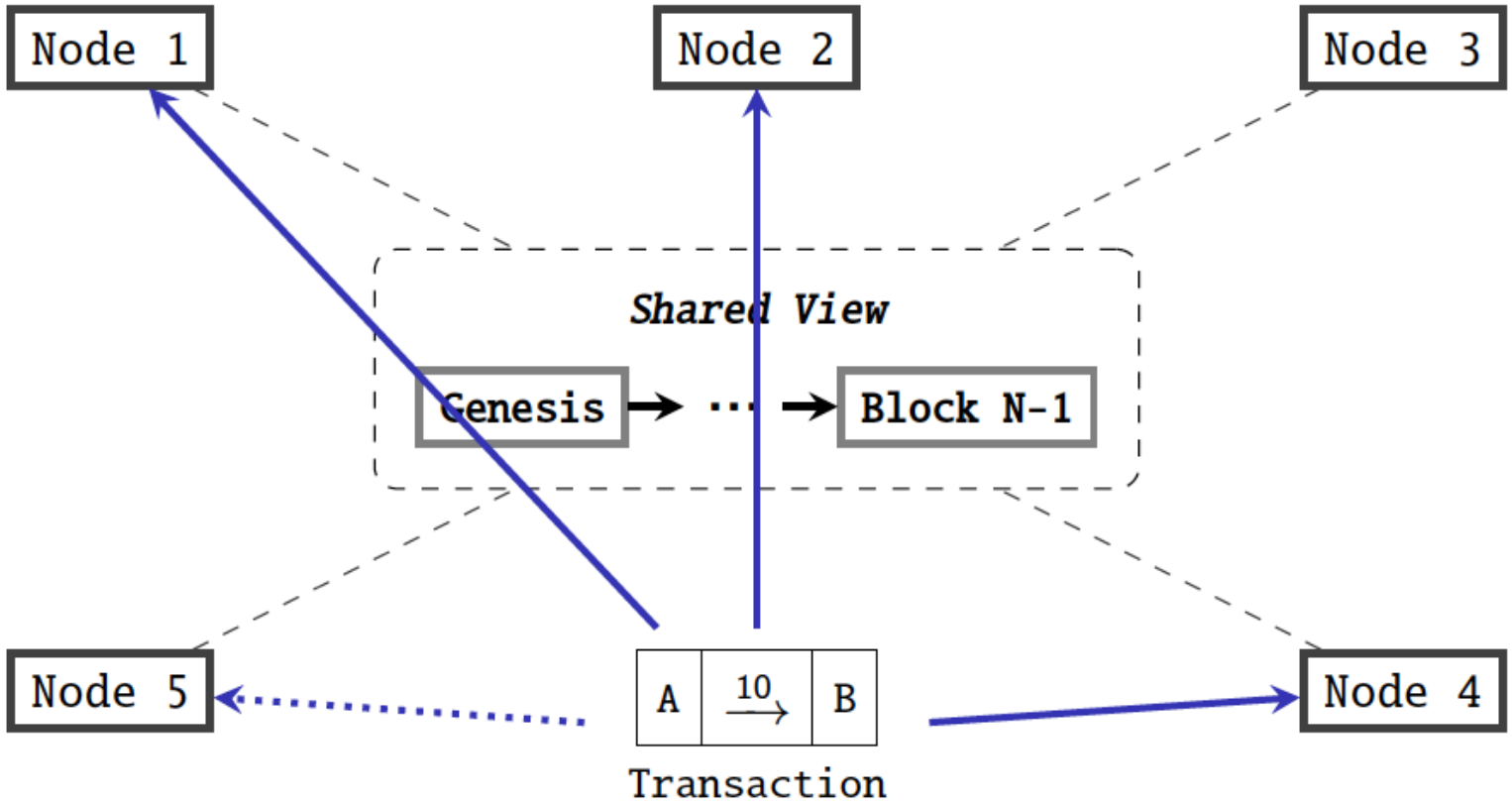
How does data get into the block?



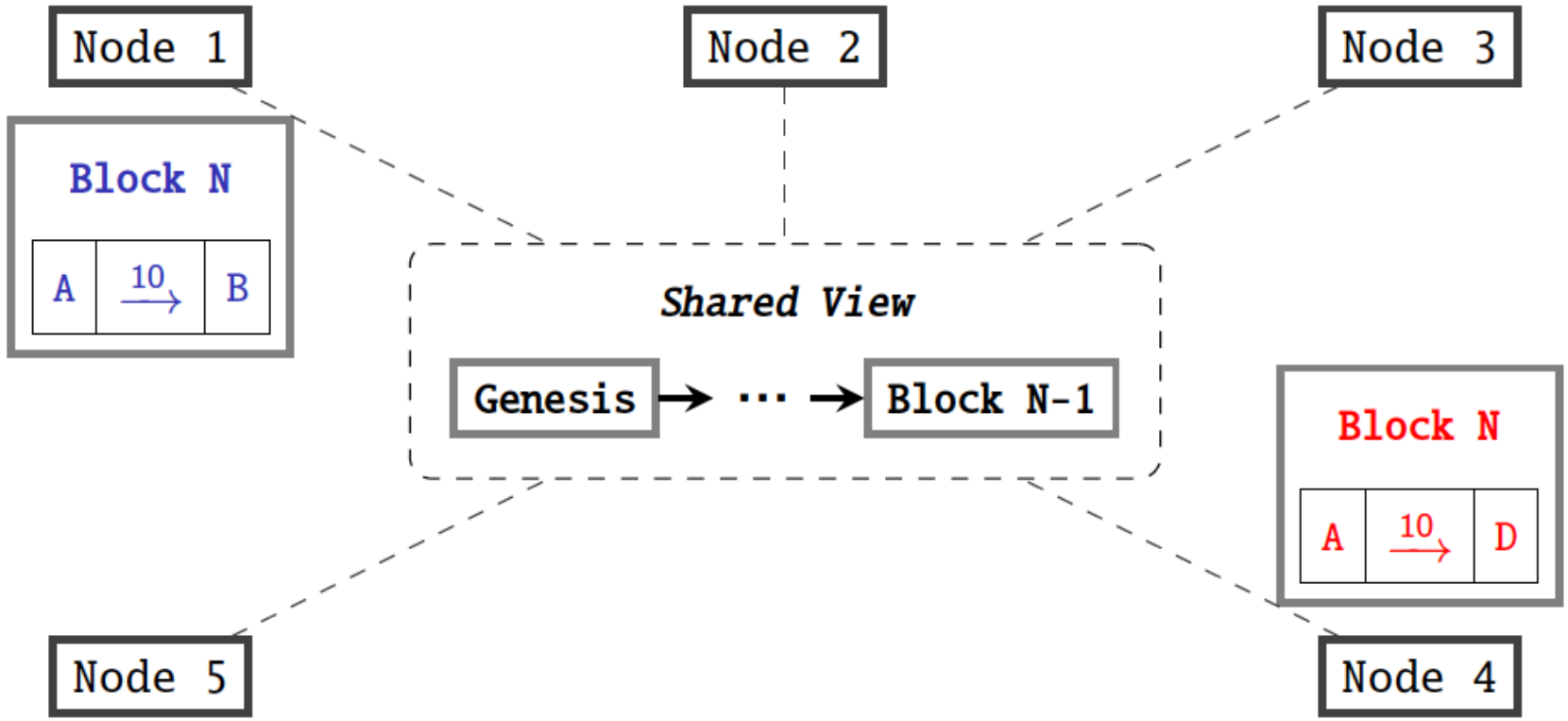
How does data get into the block?



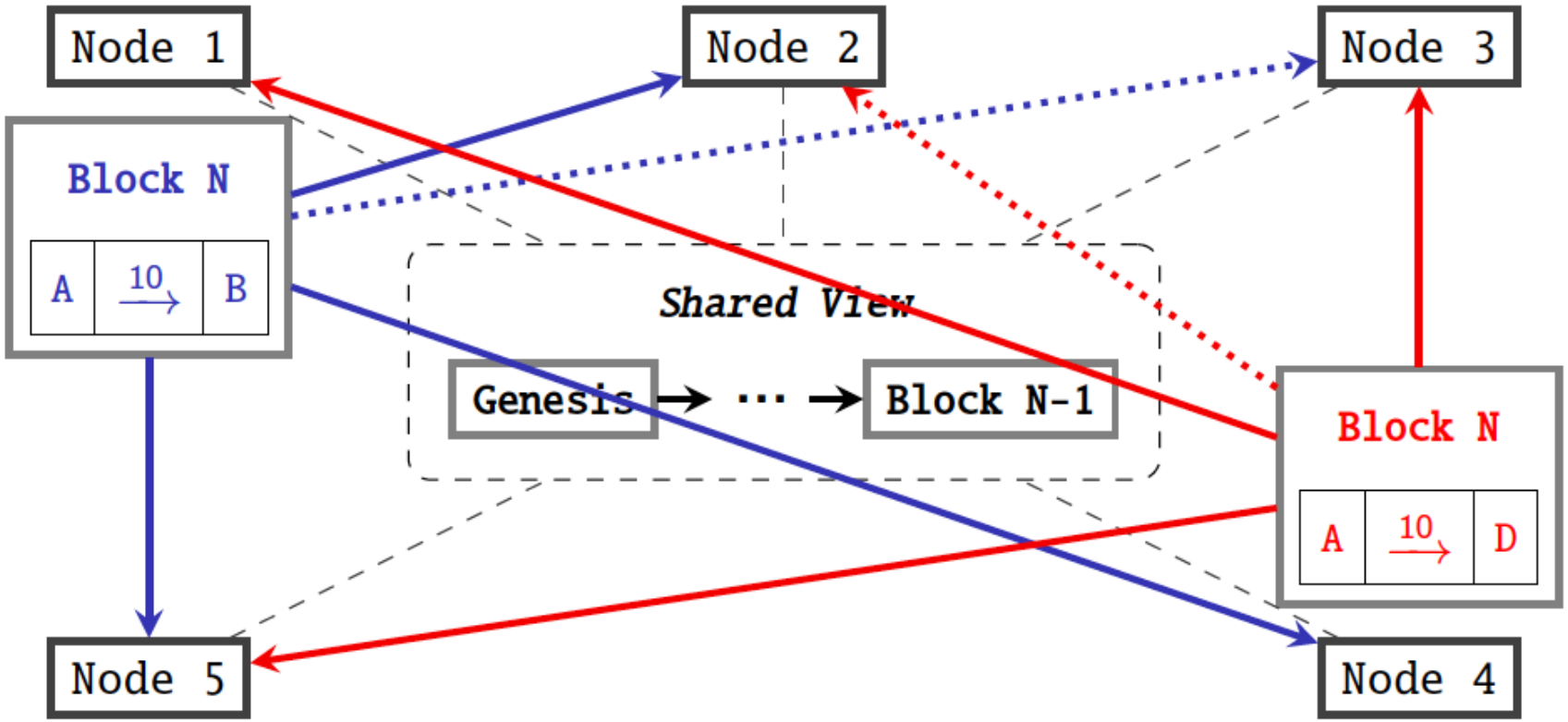
How does data get into the block?



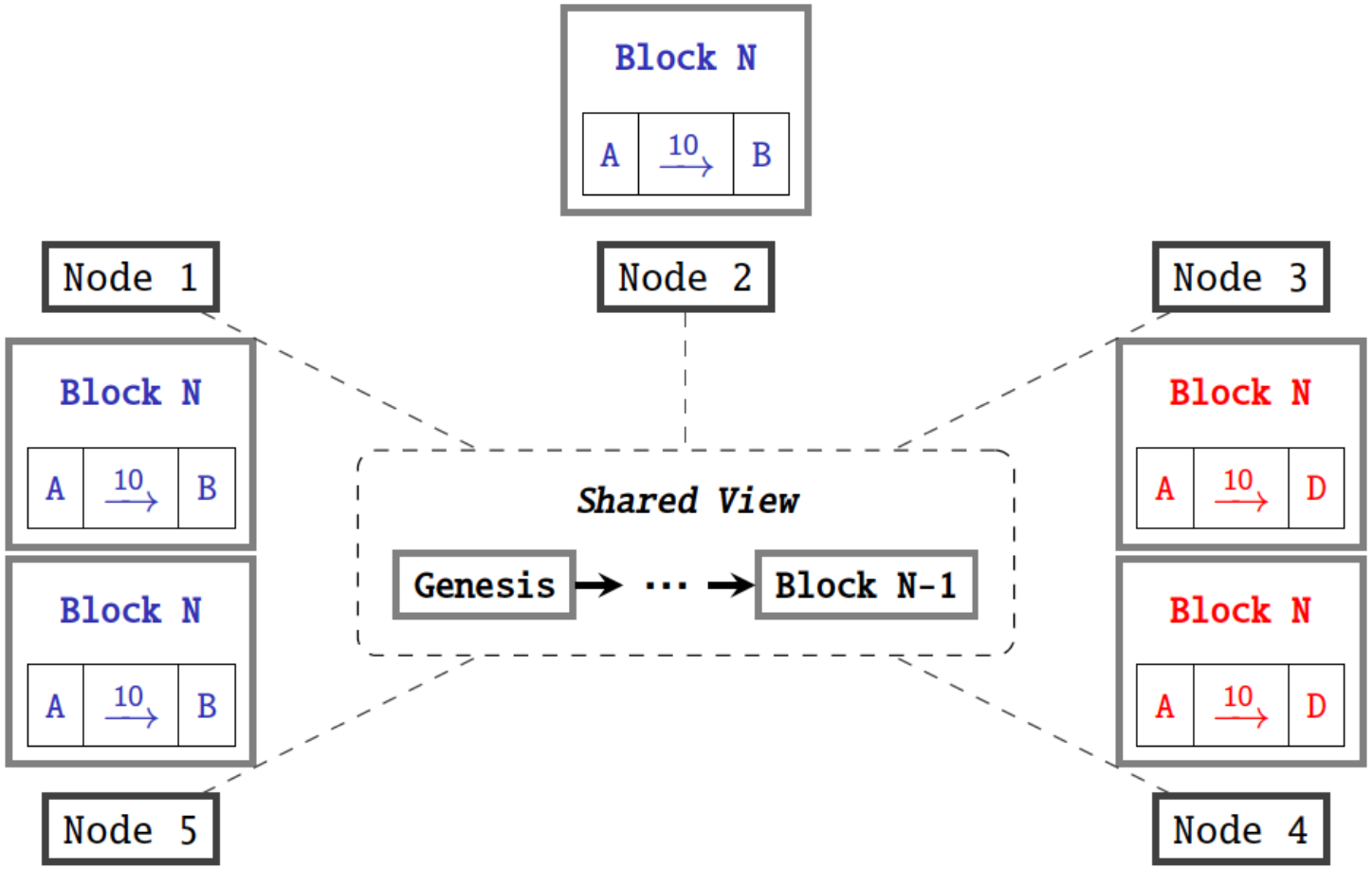
How does data get into the block?



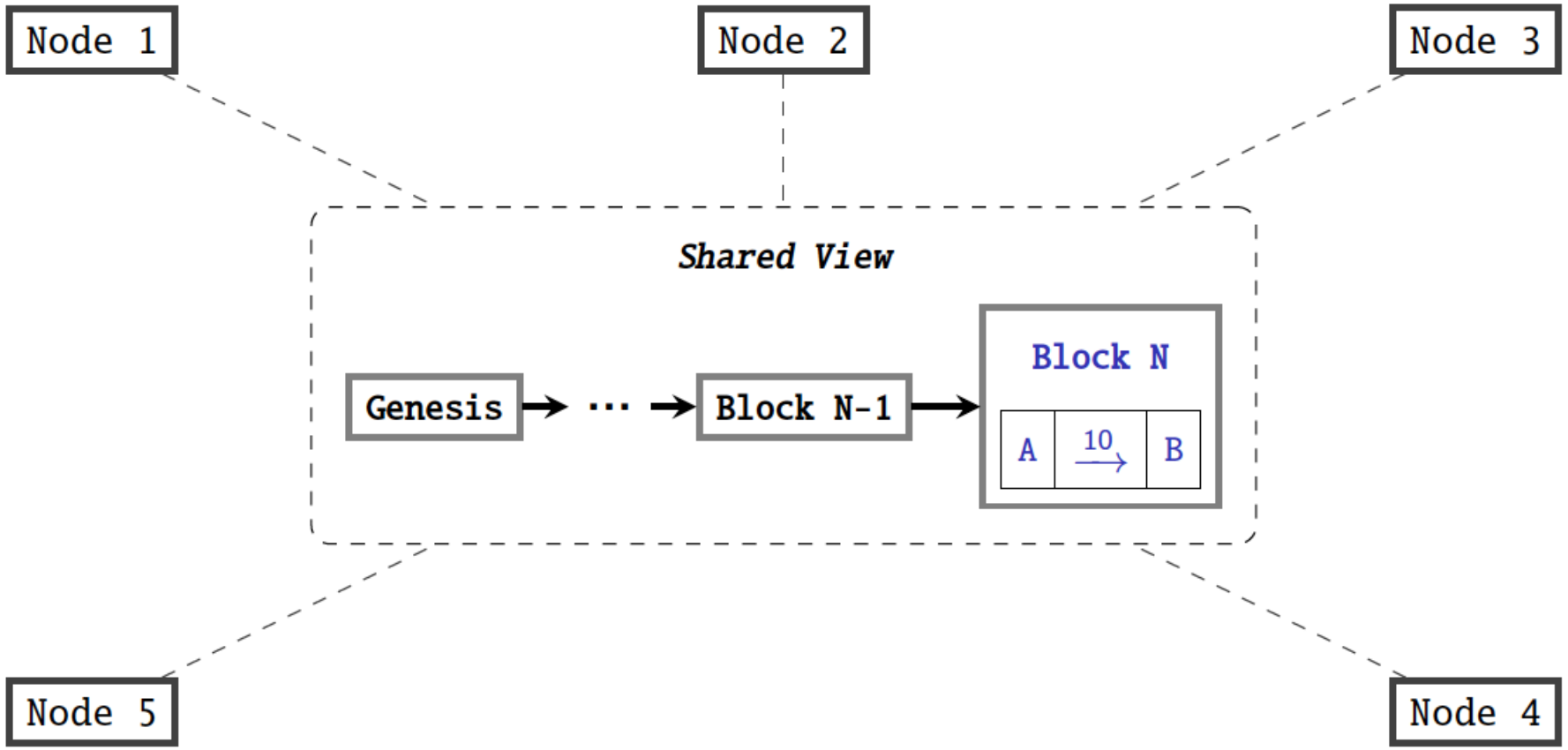
How does data get into the block?



How does data get into the block?

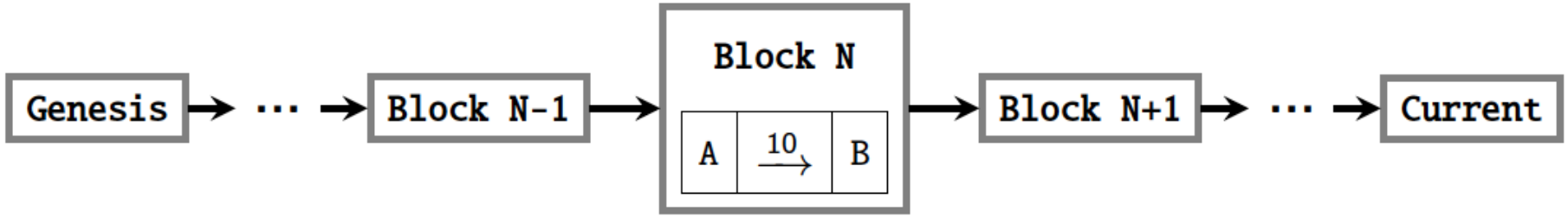


How does data get into the block?



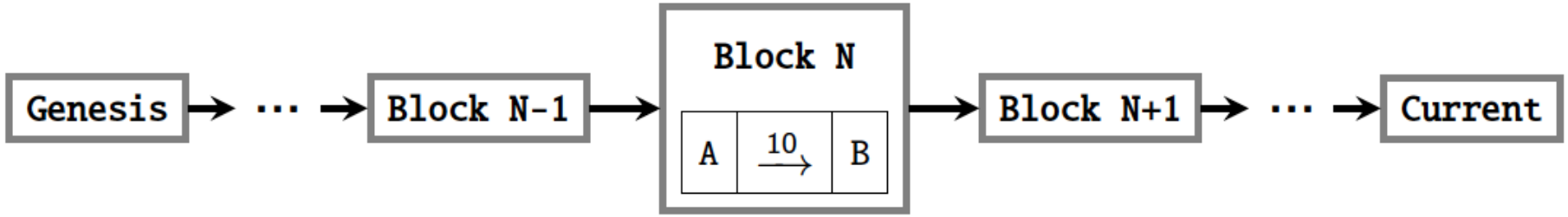
The power of consensus

- 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



The power of consensus

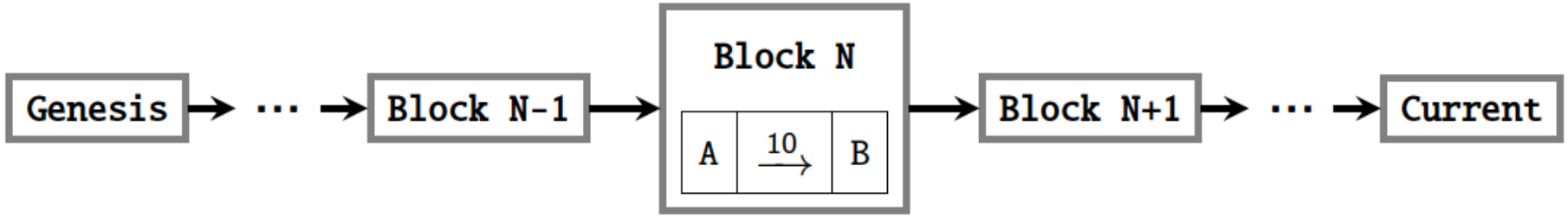
- Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:
  - Cash, debit card, credit card, e-transfer (e.g., Interac®)
  - An entry in the blockchain-based ledger



- To the best of Bob's knowledge:
  - It is hard for Alice to produce such a chain of blocks

The power of consensus

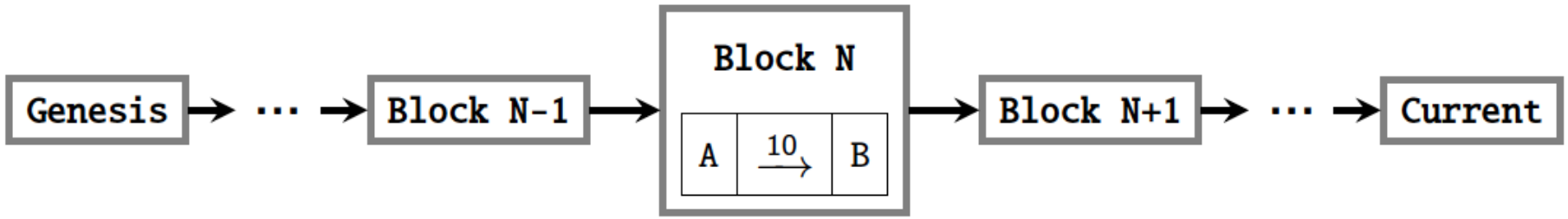
- Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:
  - Cash, debit card, credit card, e-transfer (e.g., Interac®)
  - An entry in the blockchain-based ledger



- To the best of Bob's knowledge:
  - It is hard for Alice to produce such a chain of blocks
  - There does not exist a better chain of blocks as of now

The power of consensus

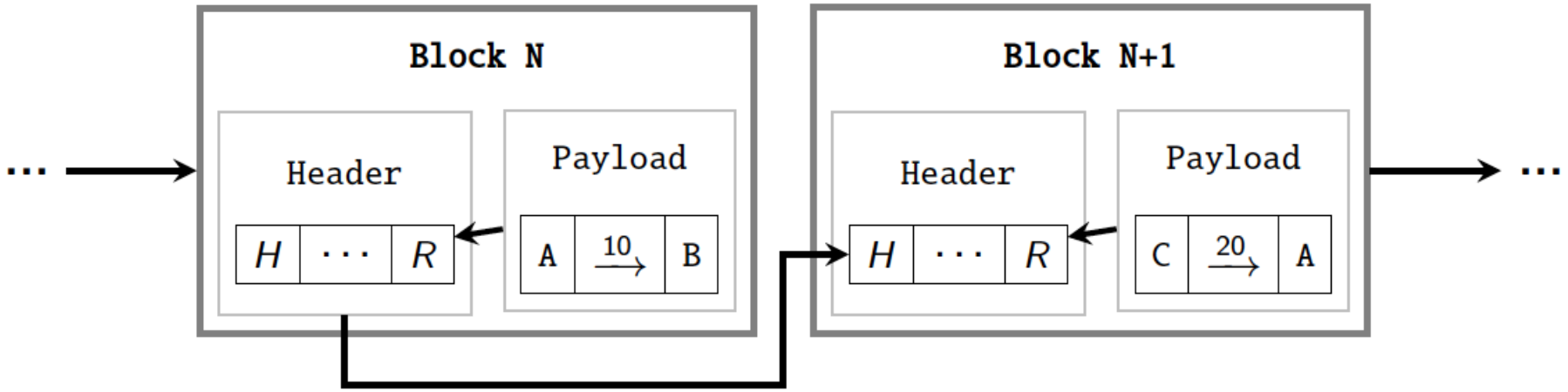
- Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:
  - Cash, debit card, credit card, e-transfer (e.g., Interac®)
  - An entry in the blockchain-based ledger



- To the best of ~~Bob's~~ everyone's knowledge:
  - It is hard for Alice to produce such a chain of blocks
  - There does not exist a better chain of blocks as of now

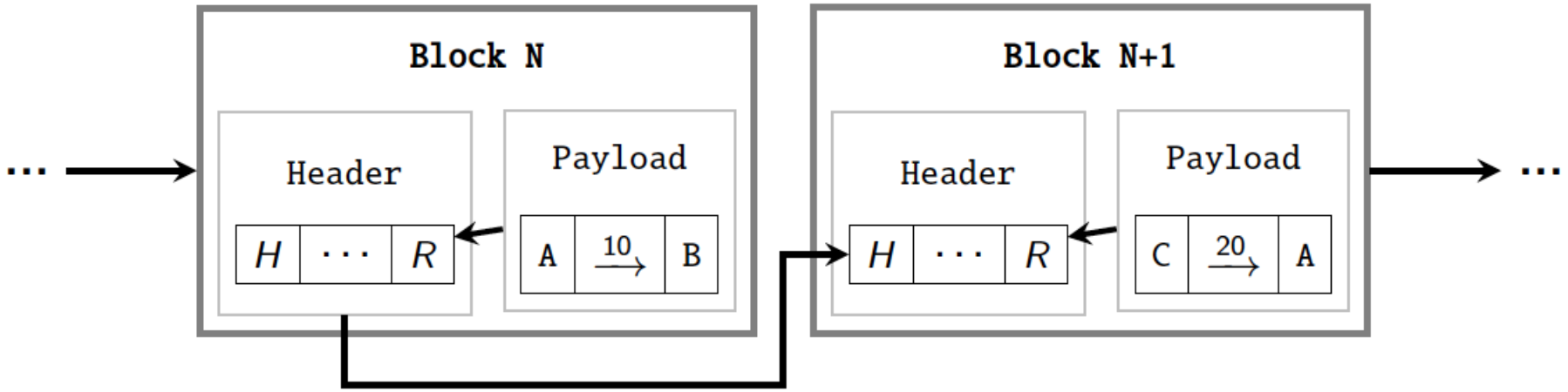
Consensus: Proof-of-work

How hard is it to alter this chain?

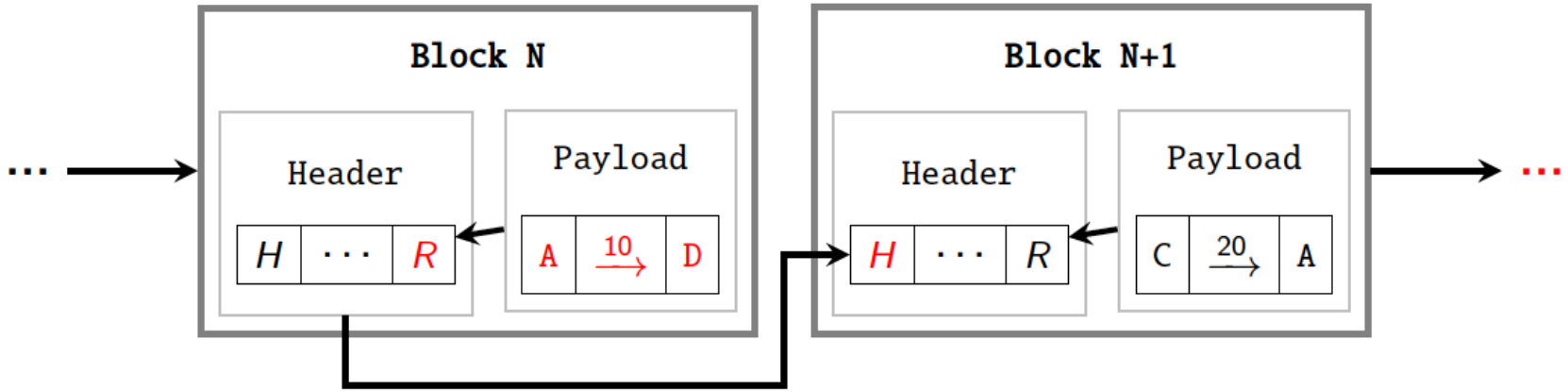


- This is the chain Alice shows Bob w.r.t her payment to Bob.

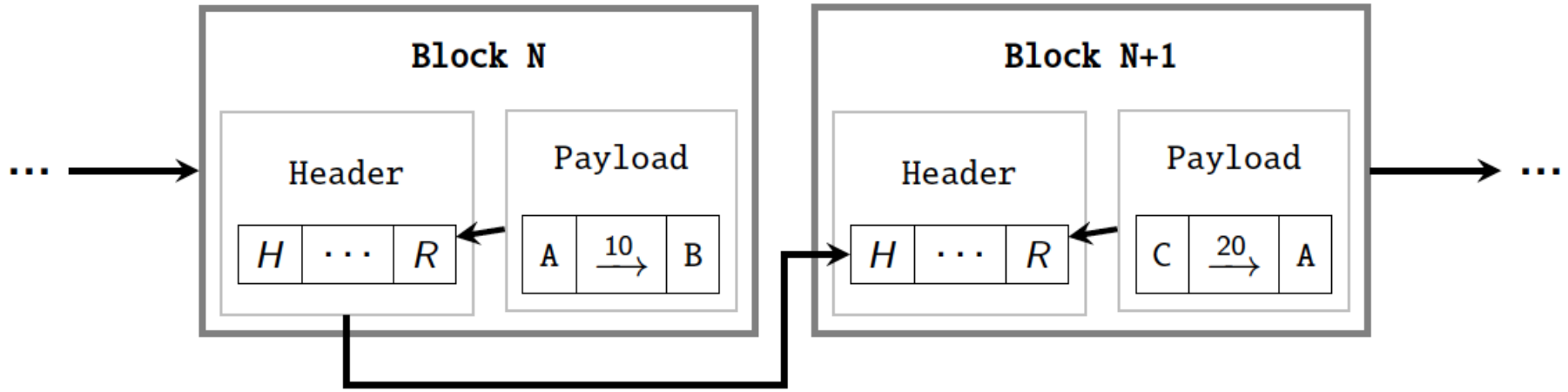
How hard is it to alter this chain?



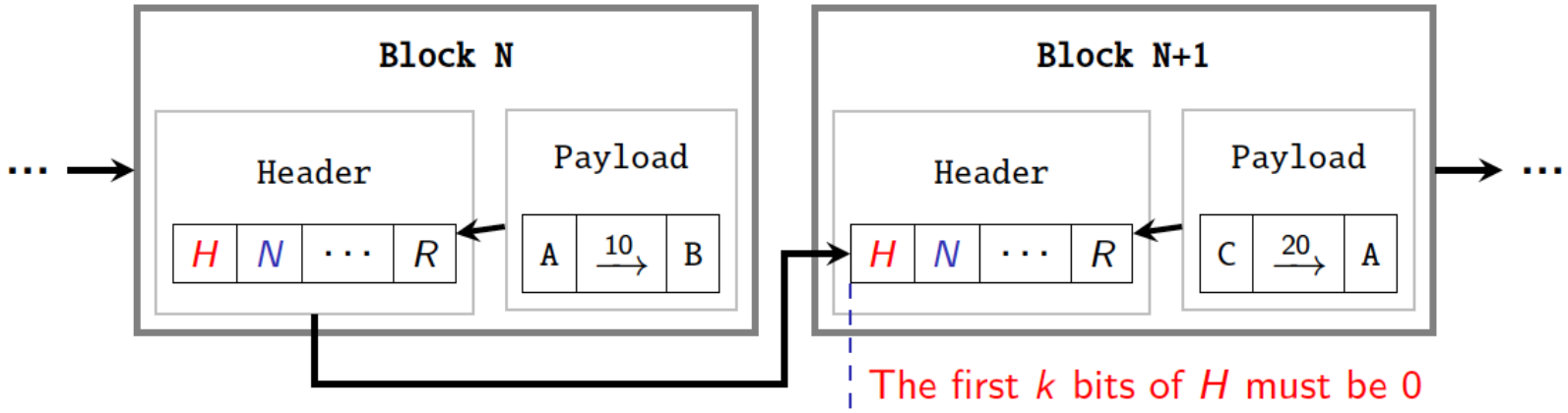
- It is not hard at all for Alice to revert this payment to Bob!



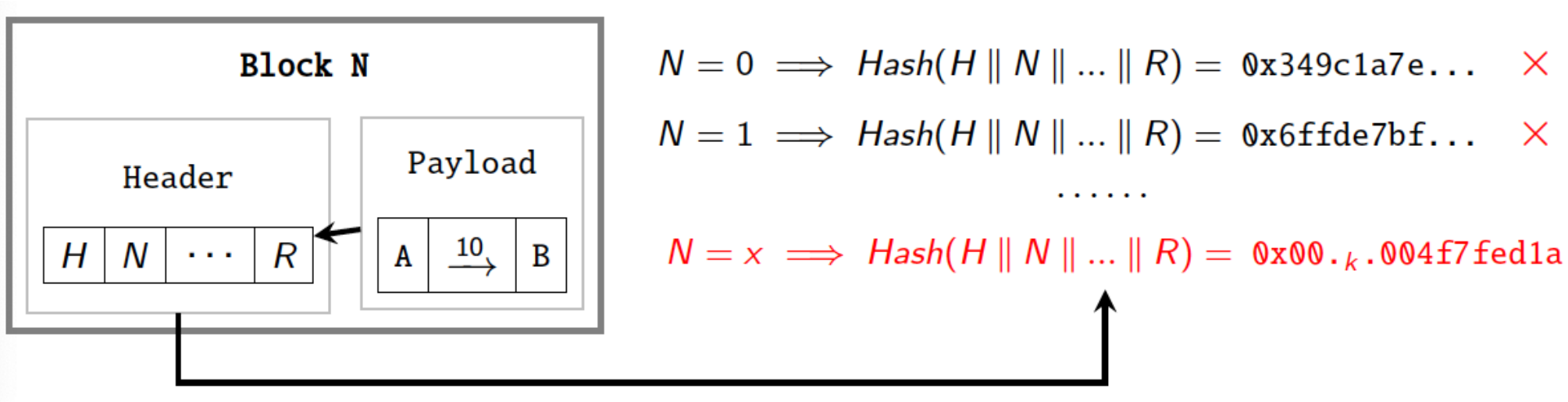
Let’s increase the difficulty



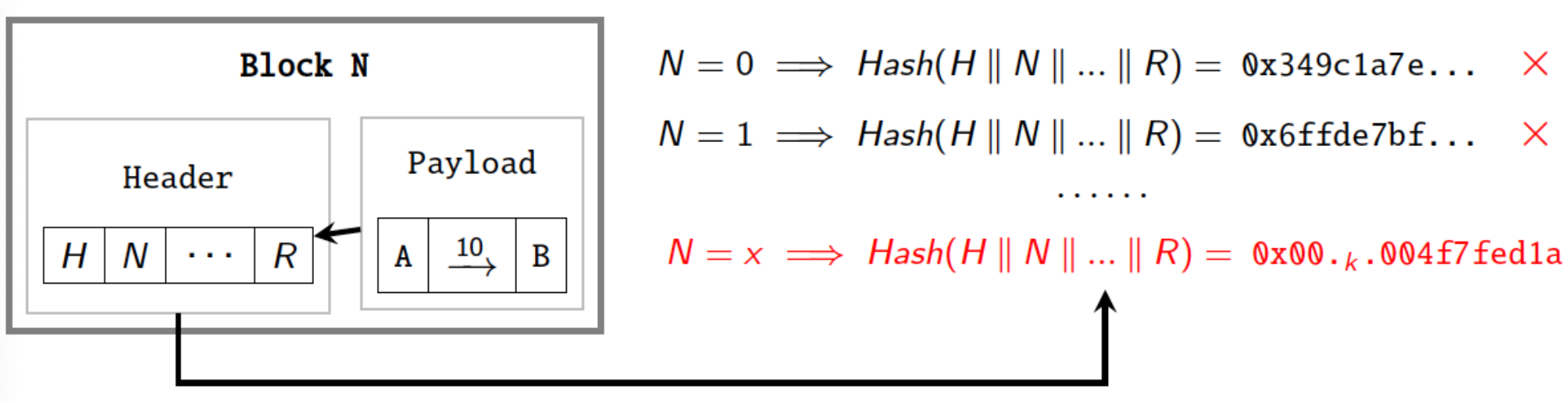
- Bob decides to make it harder for Alice to alter her payment



Mining for a valid hash

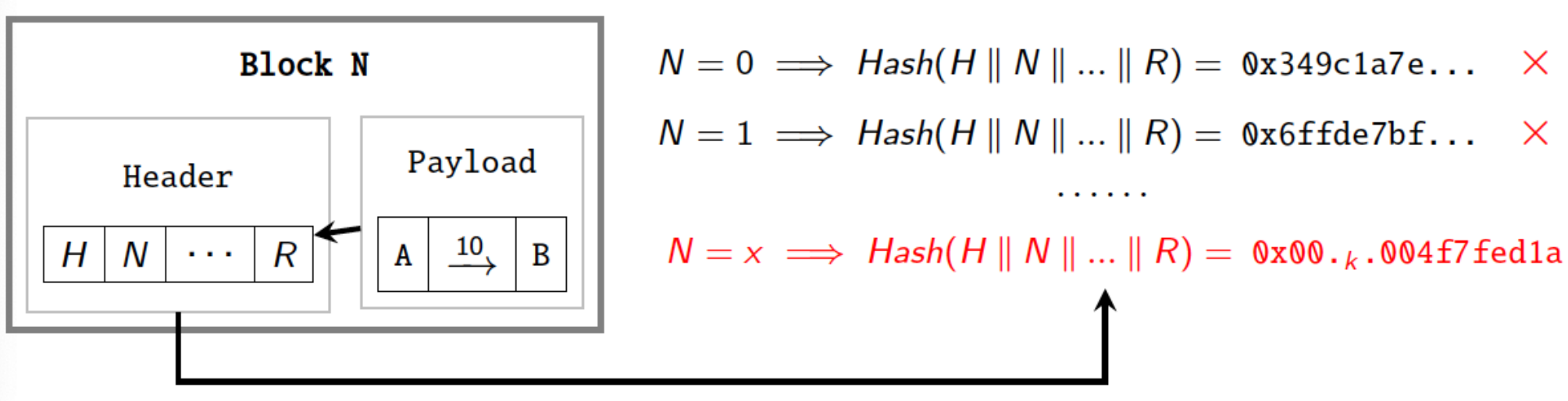


Mining for a valid hash



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

Mining for a valid hash

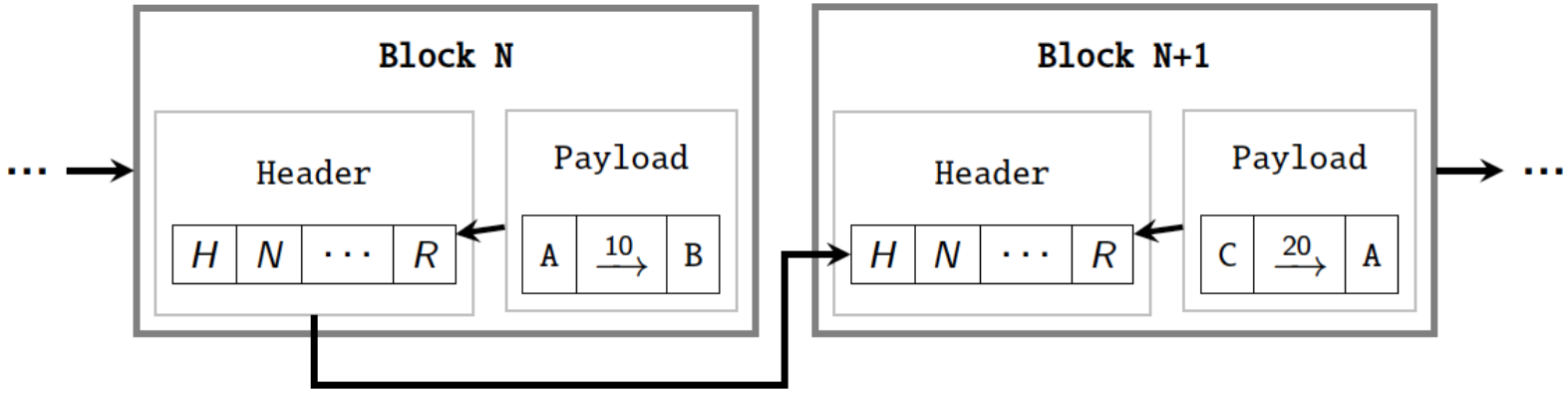


Q: What is the chance of finding a valid N assuming an m-bit hash?

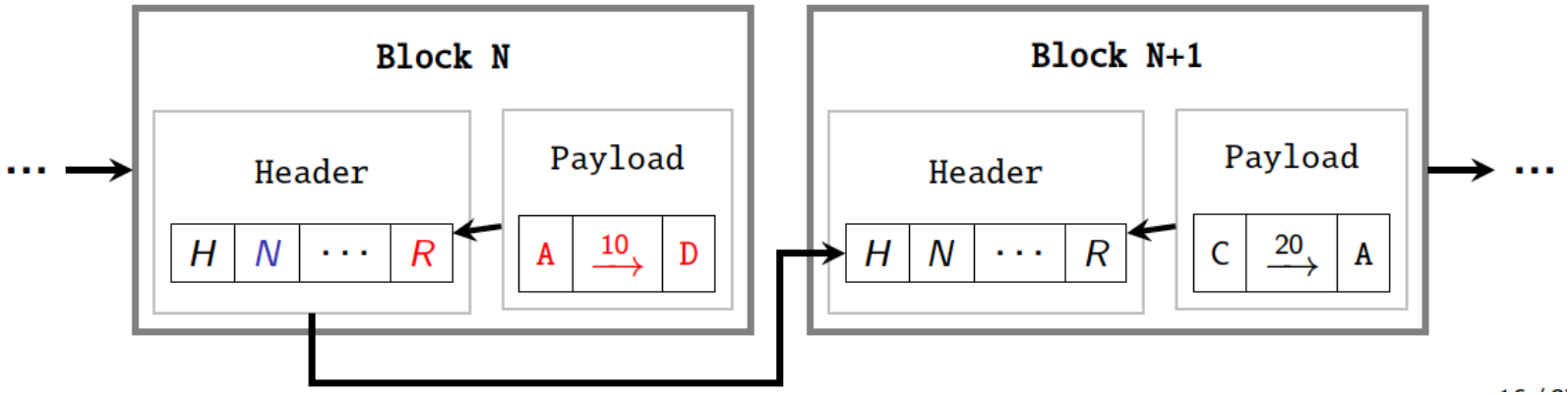
A: $\frac{2^{m-k}}{2^m}$, a larger k ⇒ a higher difficulty of finding N

Expect 2^k hash operations to find a valid N

How does mining deter alteration? – Case 1

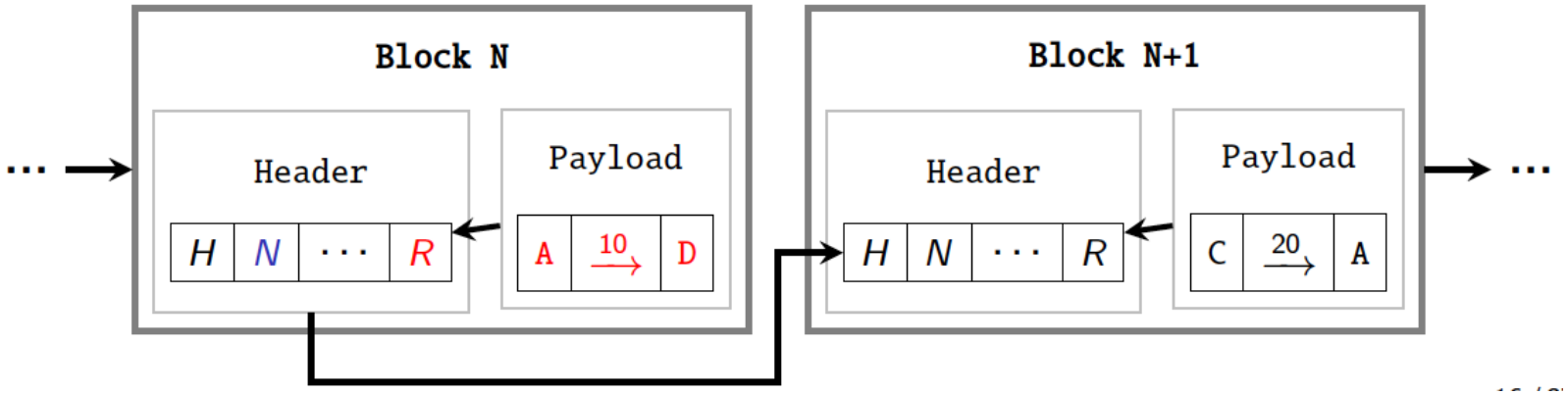


- Surgical change: Alice re-mines block *N* and finds a new nonce such that the block header hash remains unchanged



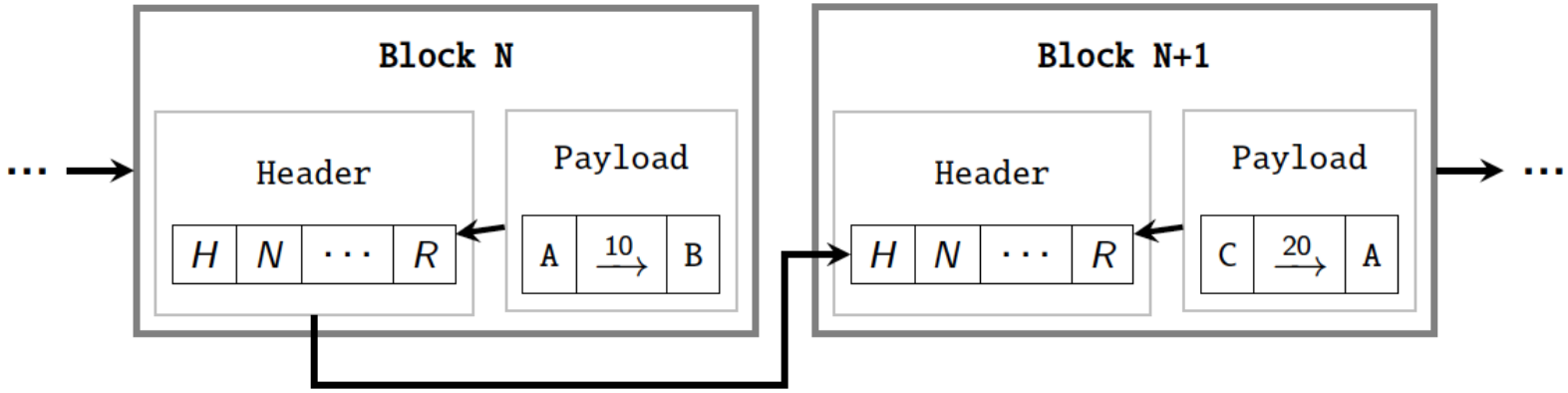
How does mining deter alteration? – Case 1

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

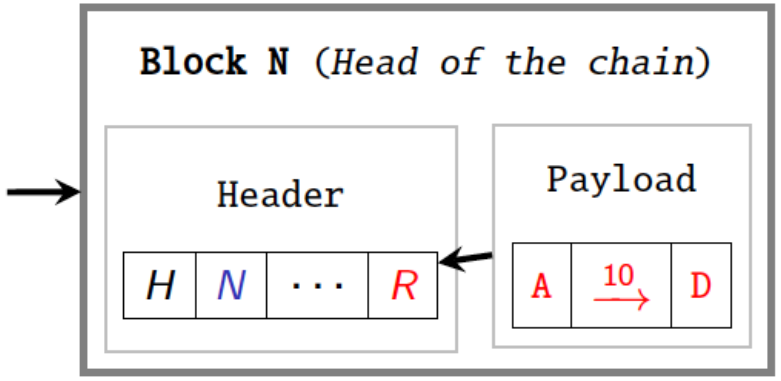


- Deterrent: This is extremely hard for a cryptographic hash function that has preimage resistance and second-preimage resistance.

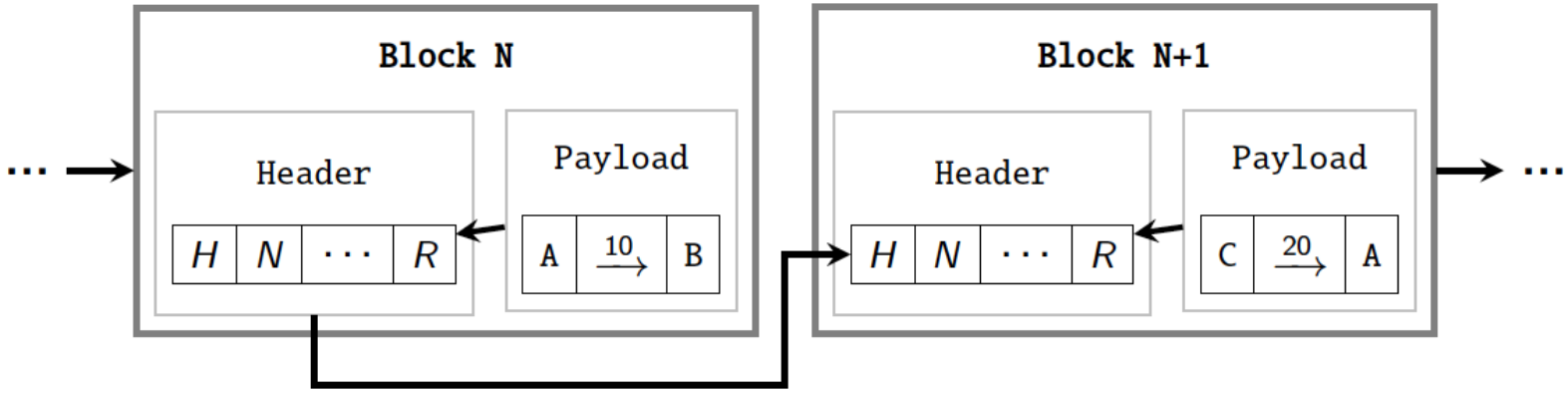
How does mining deter alteration? – Case 2



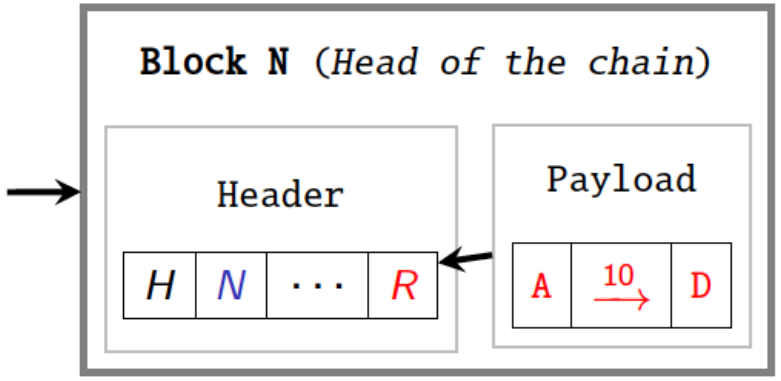
- Change-and-cut: Alice re-mines the nonce for block *N* and stops



How does mining deter alteration? – Case 2

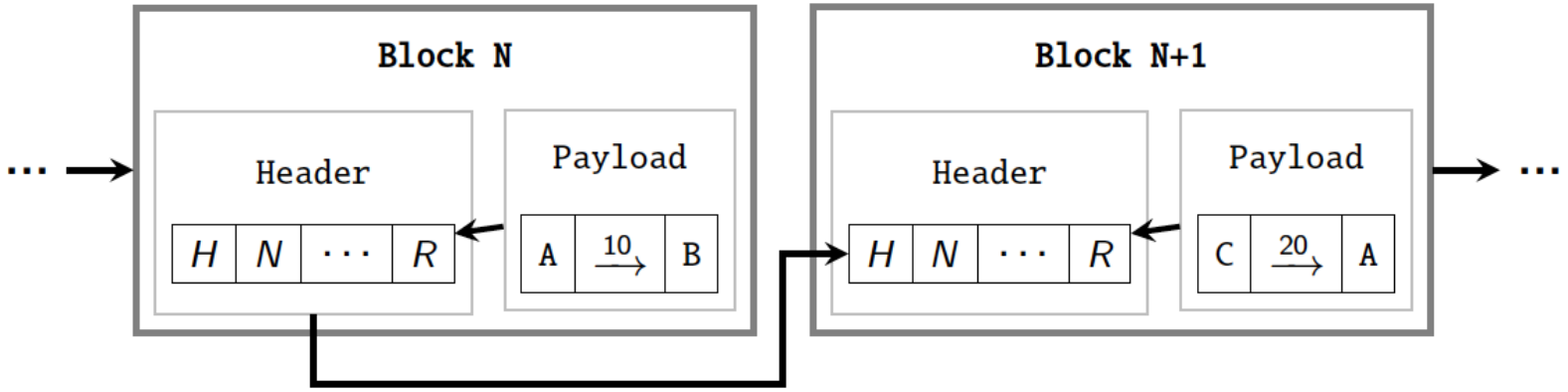


- Change-and-cut: Alice re-mines the nonce for block *N* and stops

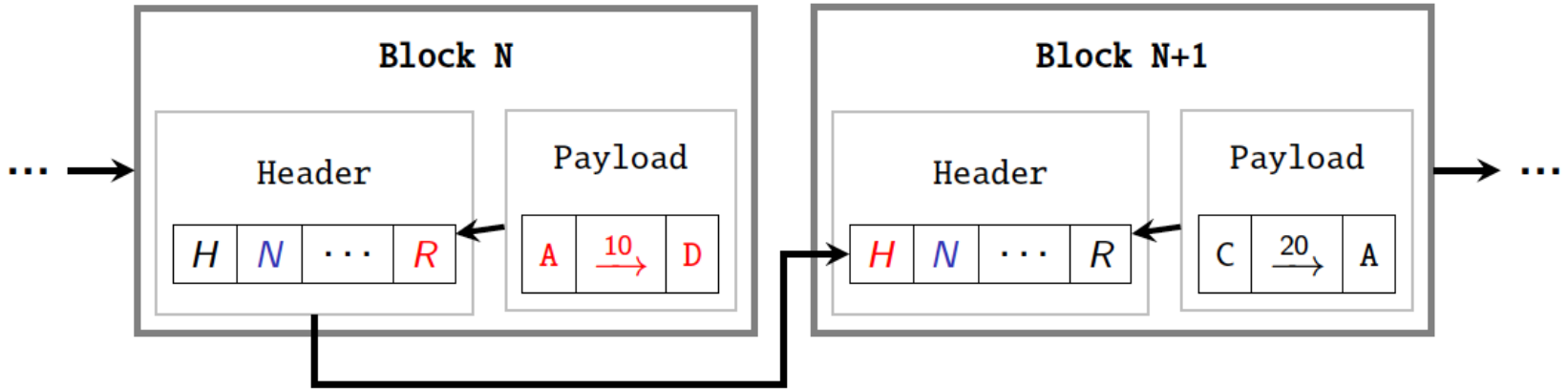


- Deterrent: Longer chains are preferred over shorter chains.

How does mining deter alteration? – Case 3

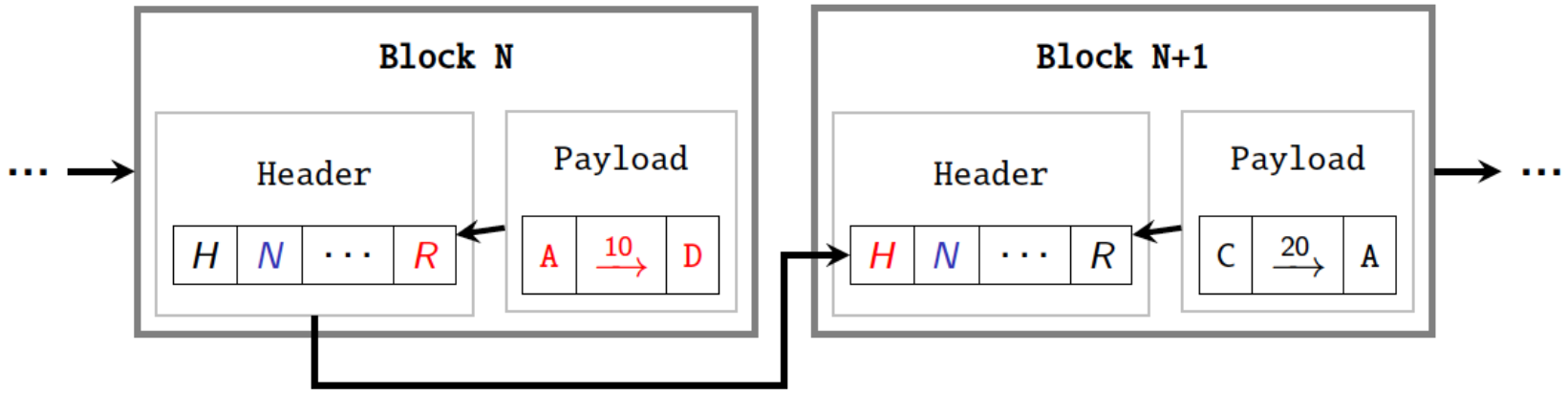


- Partial chain re-mining: Alice re-mines all the nonces since block *N*



How does mining deter alteration? – Case 3

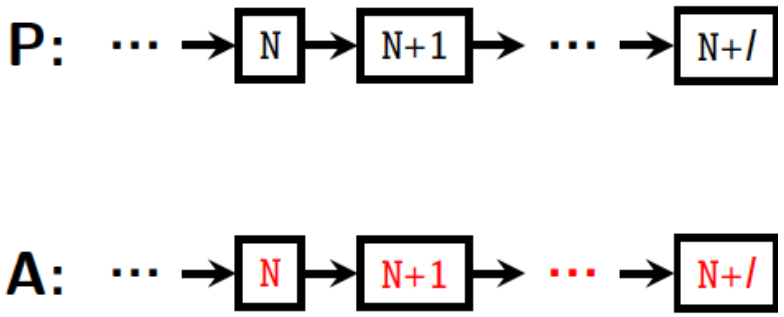
- Partial chain re-mining: Alice re-mines all the nonces since block *N*



- Deterrent: If there are *L* blocks between and including block *N* and the chain head, Alice is expected to perform $L \times 2^k$ hash operations to build-up an equally competitive chain assuming the difficulty level *k* does not change.

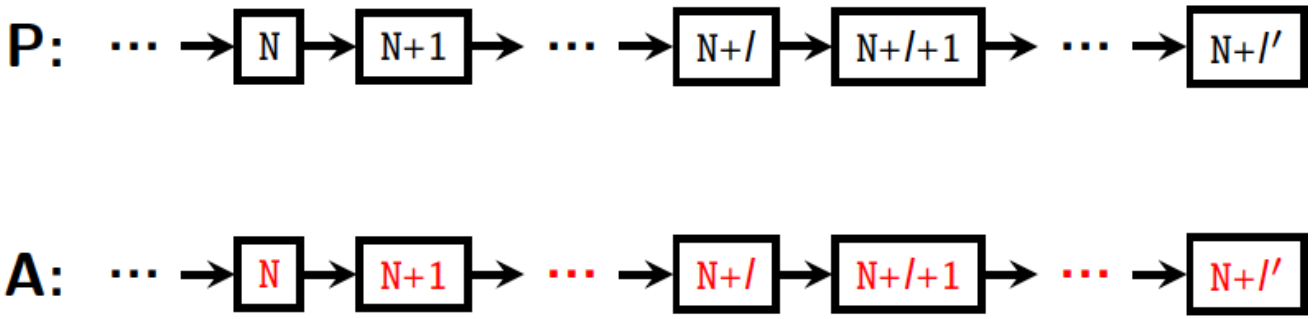
The 51% attack

- There is a catch in the deterrent:
  - Alice needs to mine slower than the rest of the participants combined



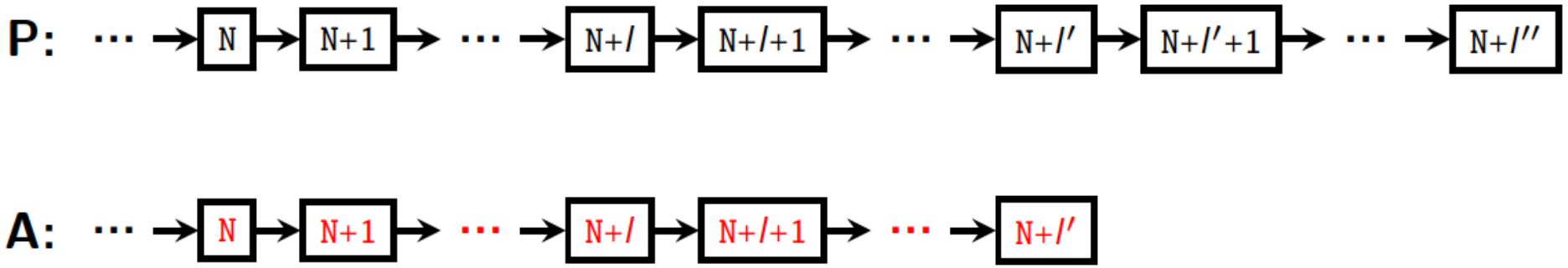
The 51% attack

- There is a catch in the deterrent:
  - Alice needs to mine slower than the rest of the participants combined



The 51% attack

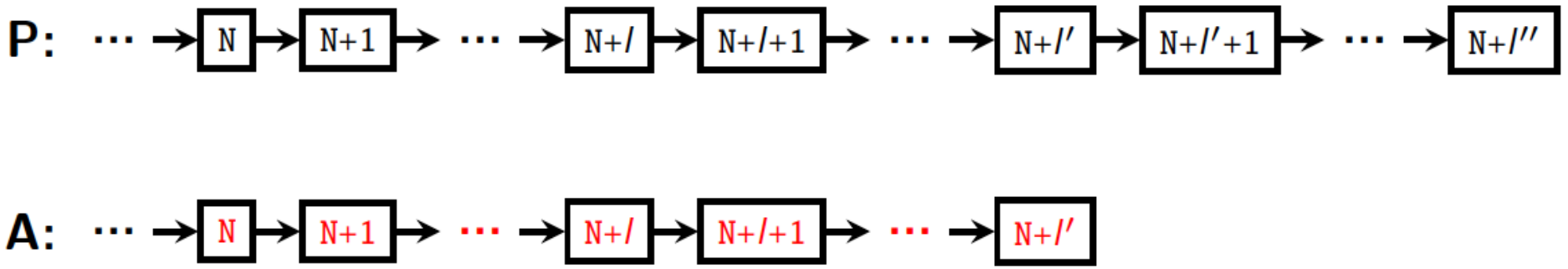
- There is a catch in the deterrent:
  - Alice needs to mine slower than the rest of the participants combined



→ i.e., the public chain needs to grow faster than Alice's chain

The 51% attack

- There is a catch in the deterrent:
  - Alice needs to mine slower than the rest of the participants combined

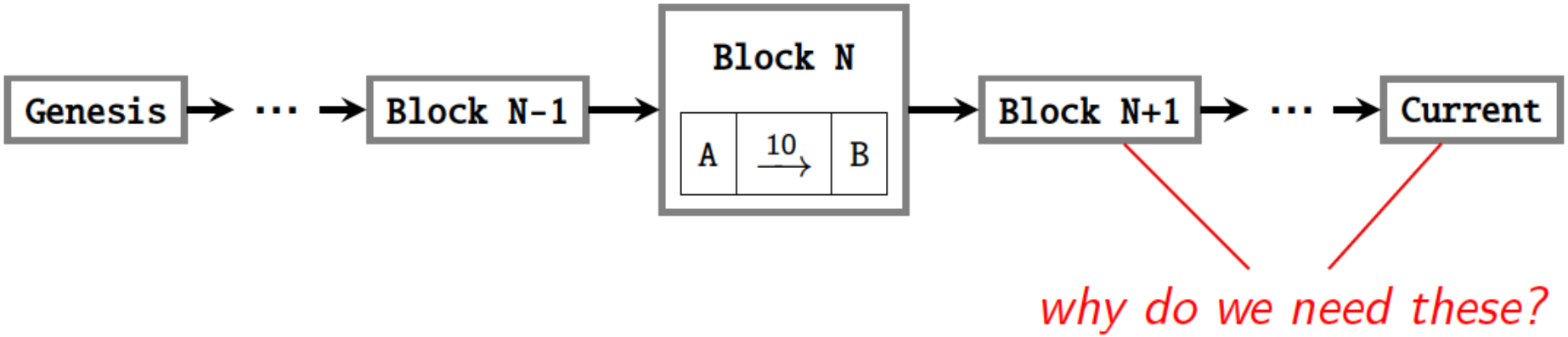


→ i.e., the public chain needs to grow faster than Alice's chain

→ If Alice mines faster, she eventually gets to rewrite history

Confirmation level

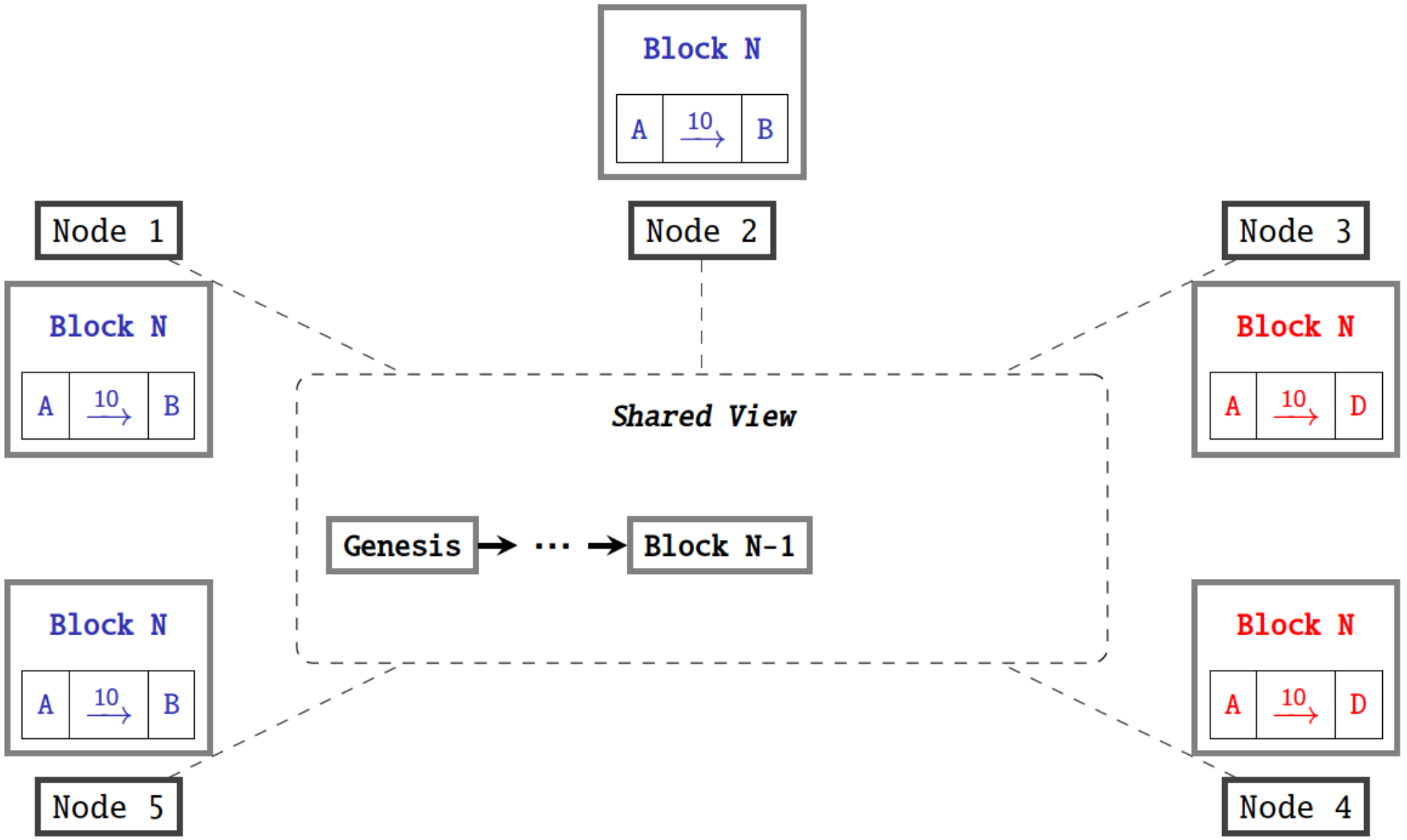
- Recall that when we show a proof of payment, we need a few extra blocks after the block that hosts the ledger entry.



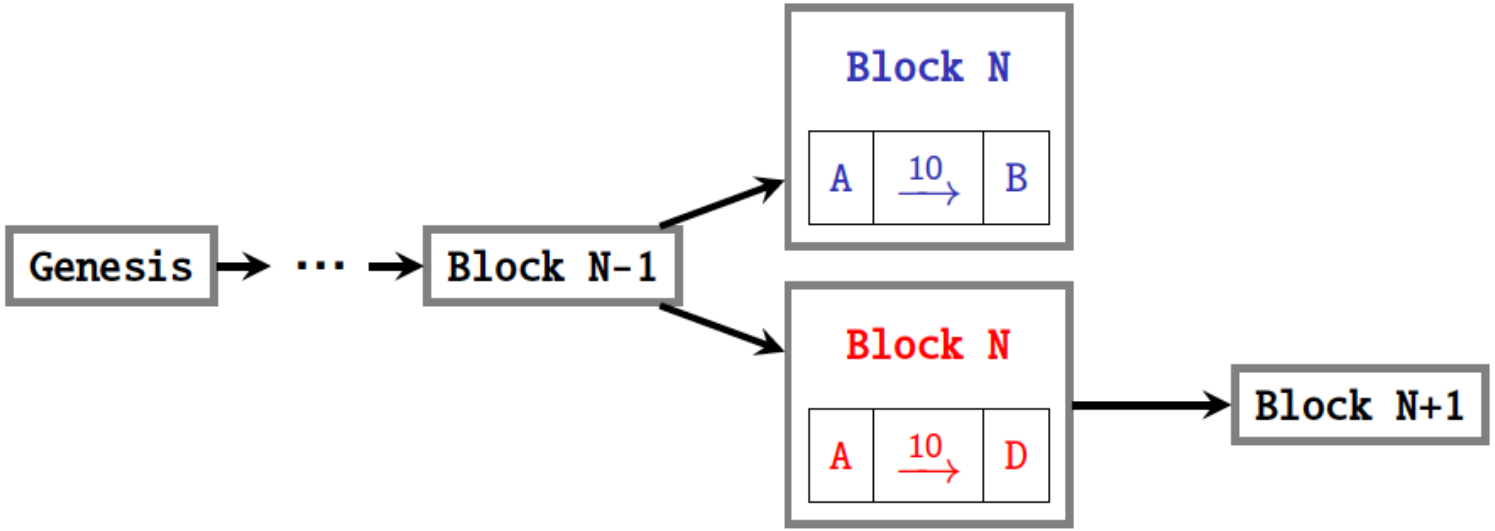
Q: Why do we need these extra blocks, even when:

1. Alice does not control over 50% computational power?
2. Everyone else is honest and cooperative?

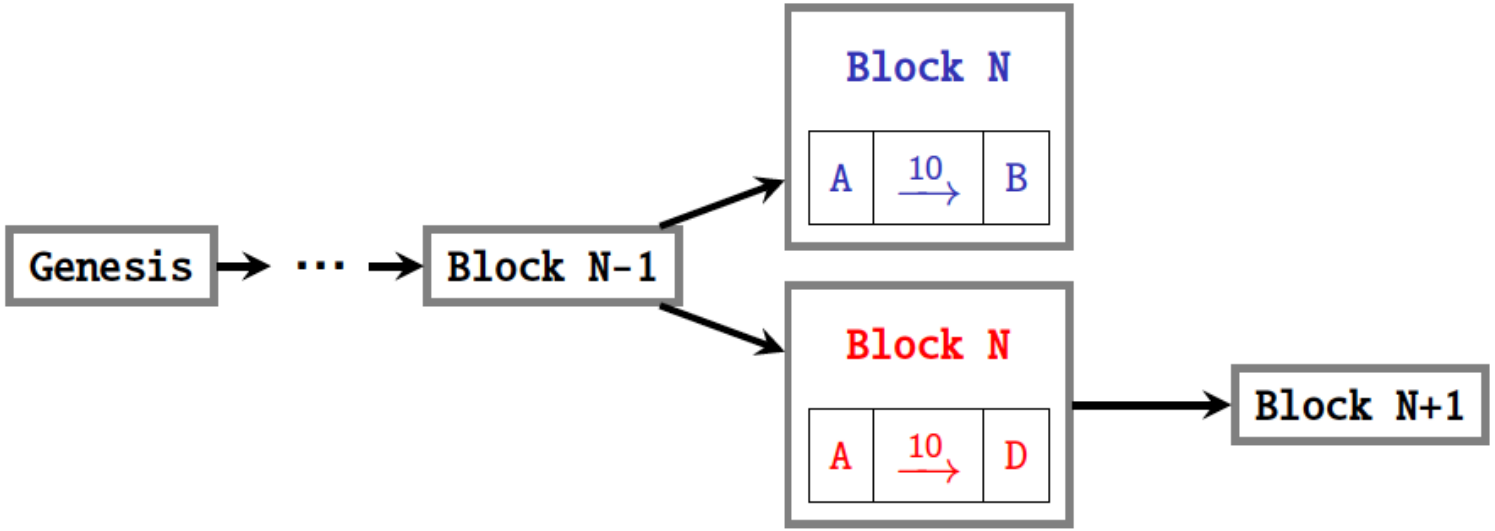
How does data get into the block?



Back to confirmation level



Back to confirmation level



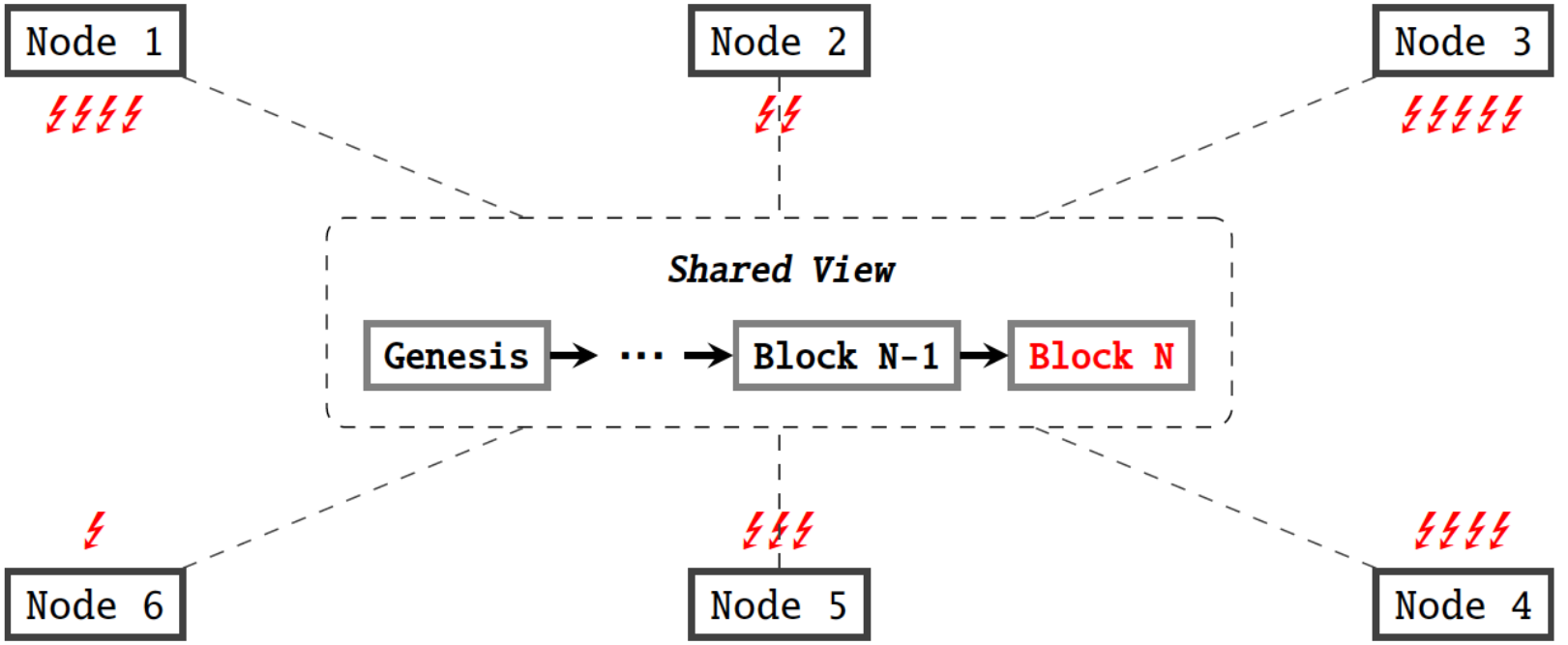
- To trigger a fork, Alice could:
  - Send two transactions in a short time window
  - Send two transactions to separate halves of the network
  - Pre-mine one block and only reveal it after the first transaction is sent to the network

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
  - Hashing itself is not useful
  - These operations are repeated across the fleet of nodes

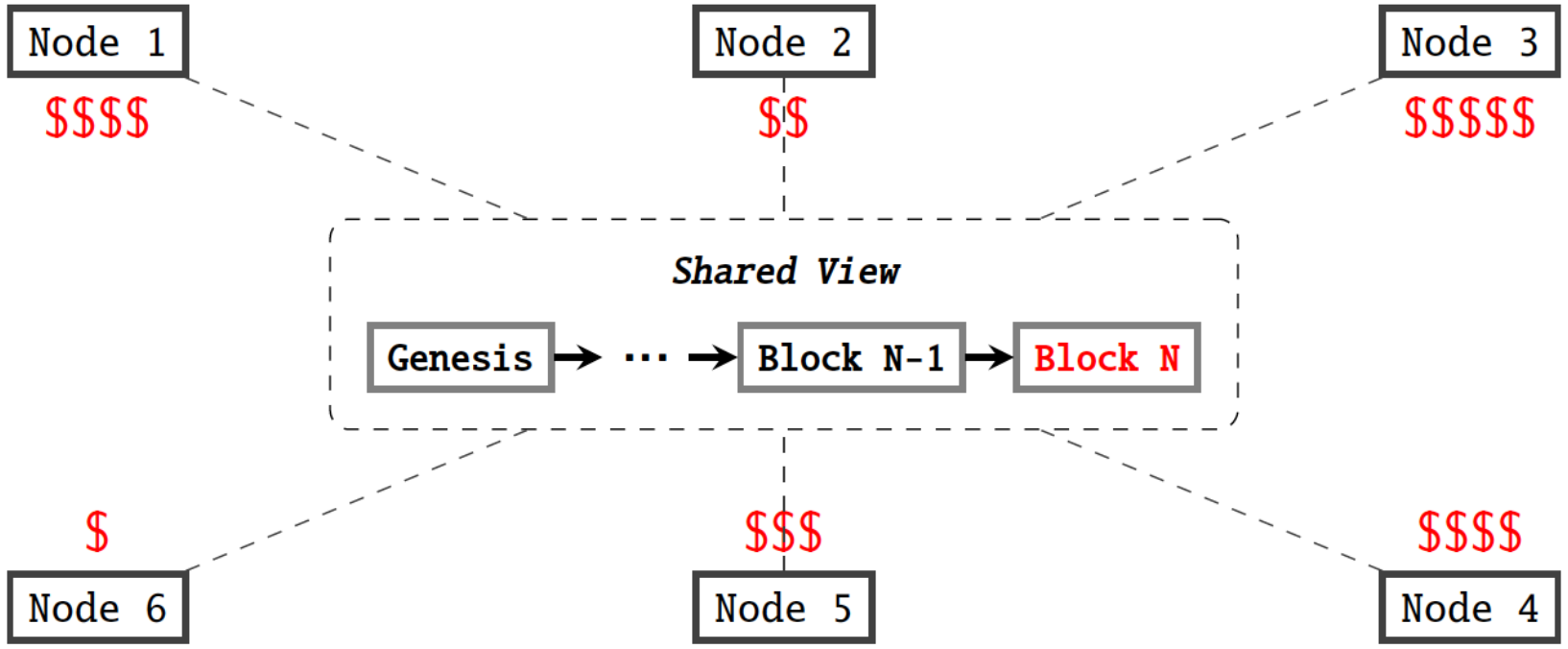
Consensus: Proof-of-stake

Block production as election



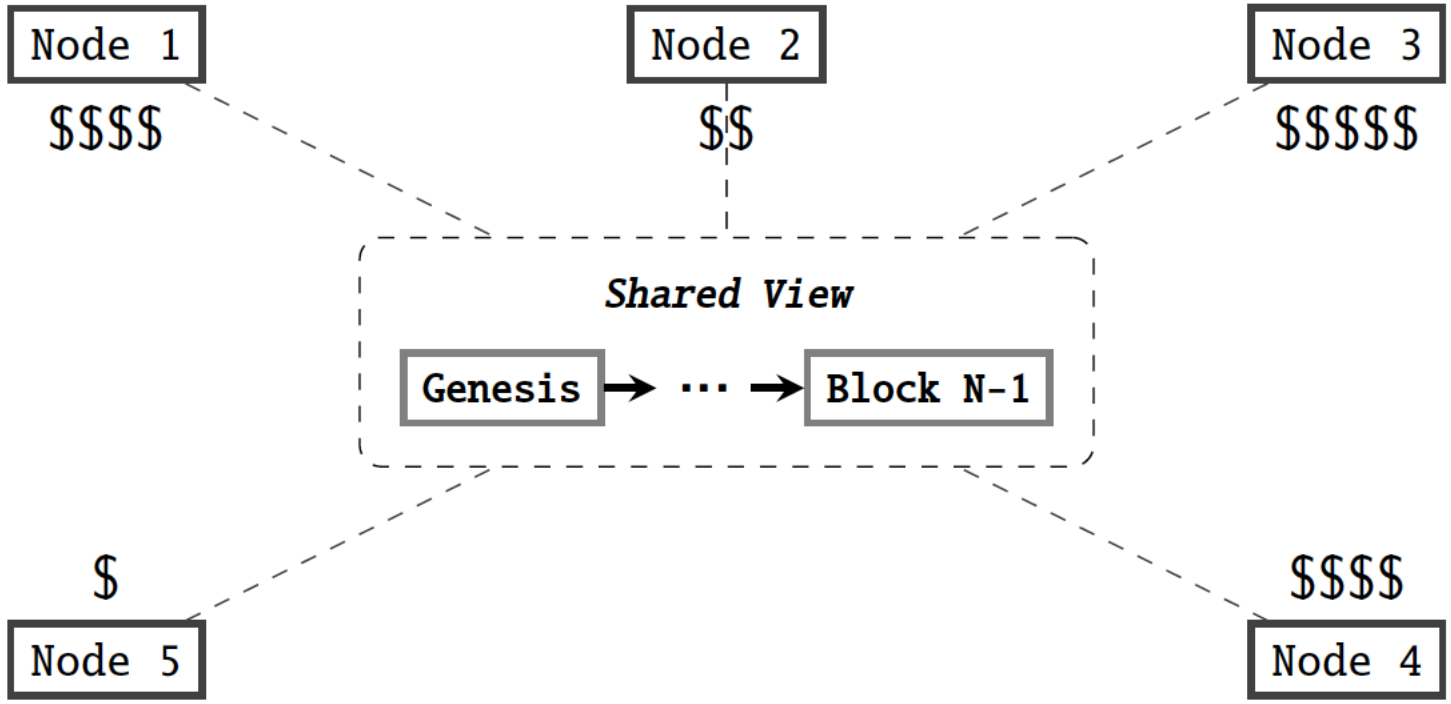
- In a proof-of-work scheme:
  - The chance of which node is elected to propose a new block is proportional to its hashing power
  - Collisions are allowed and are resolved by the longest chain rule

Block production as election

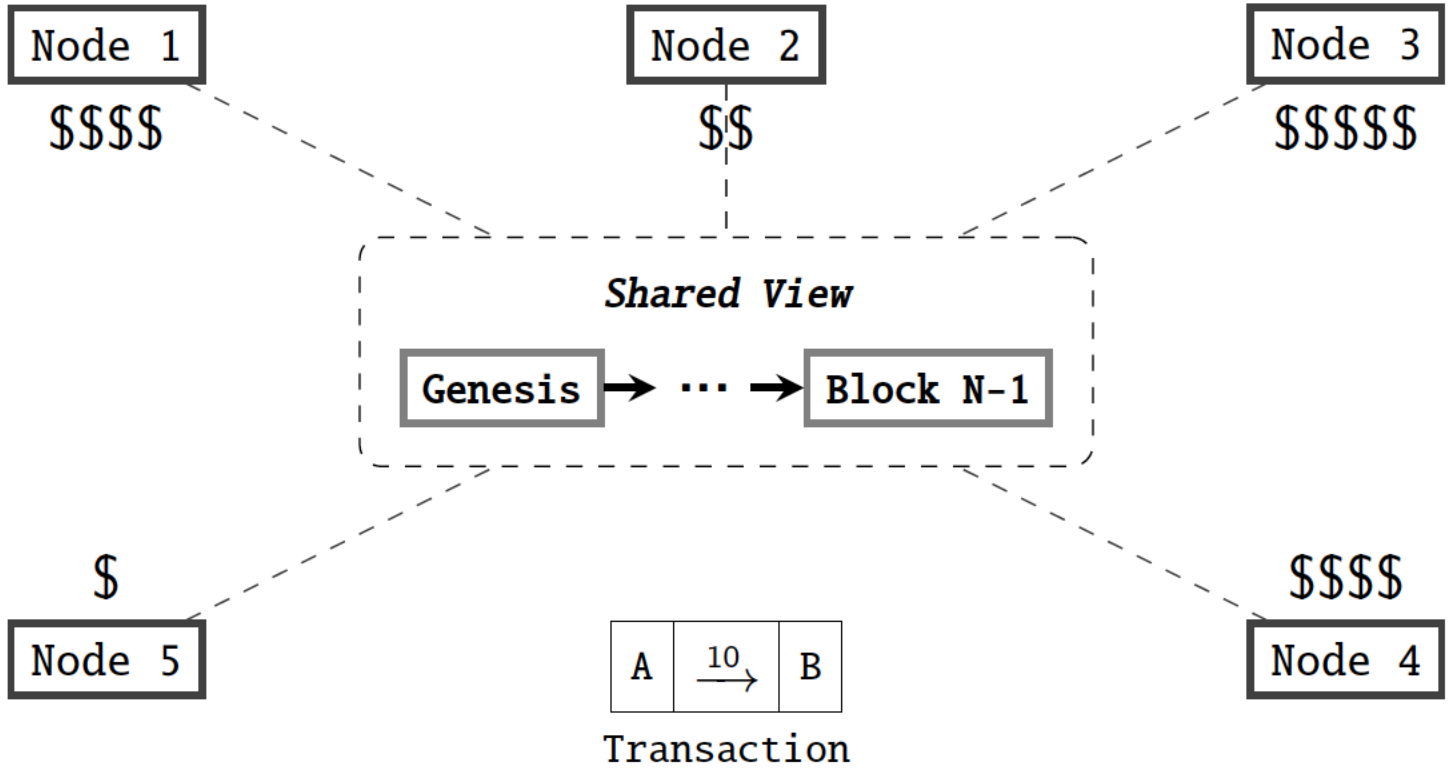


- In a proof-of-stake scheme:
  - The chance of which node is elected to propose a new block is proportional to its staked value
  - Collisions are not allowed by design, only the leader creates a block

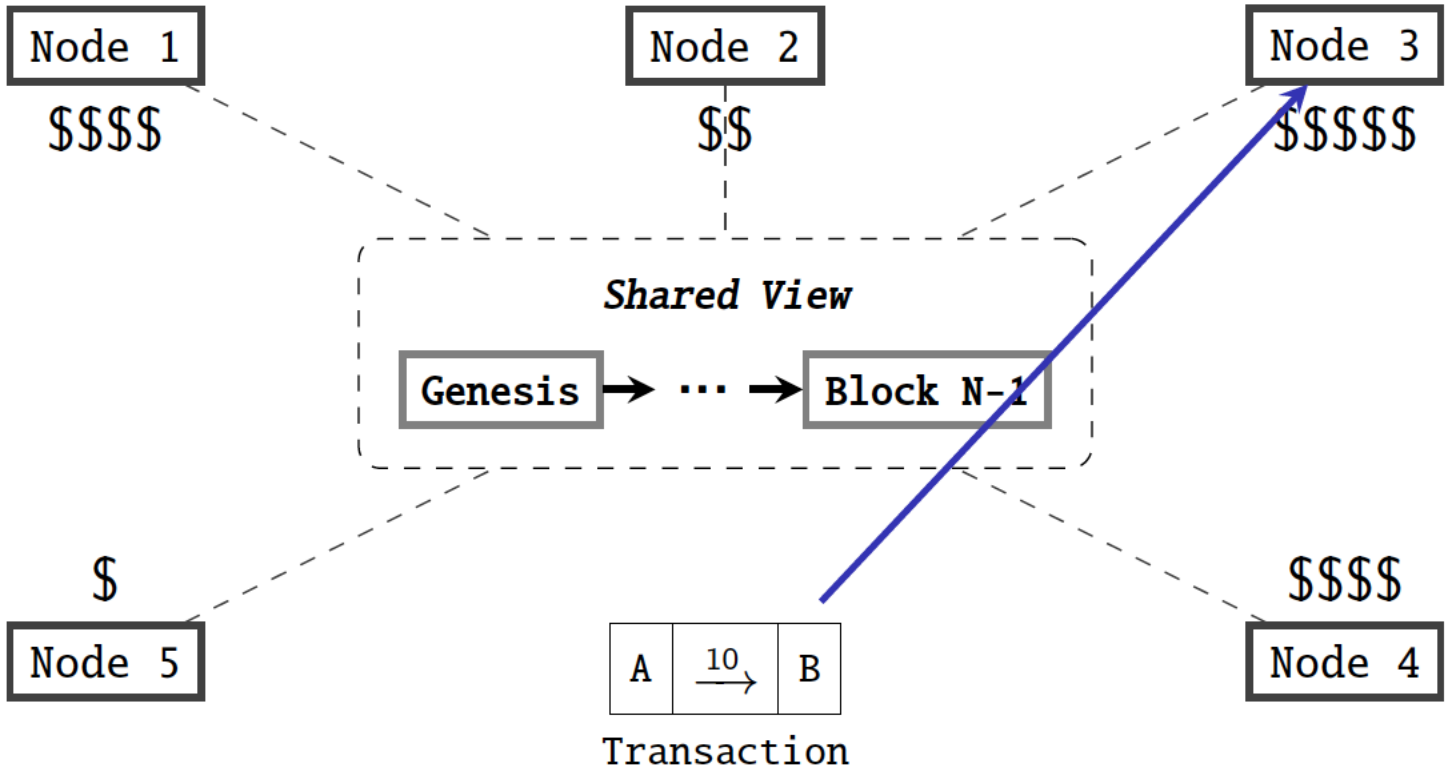
Transaction lifecycle in PoS



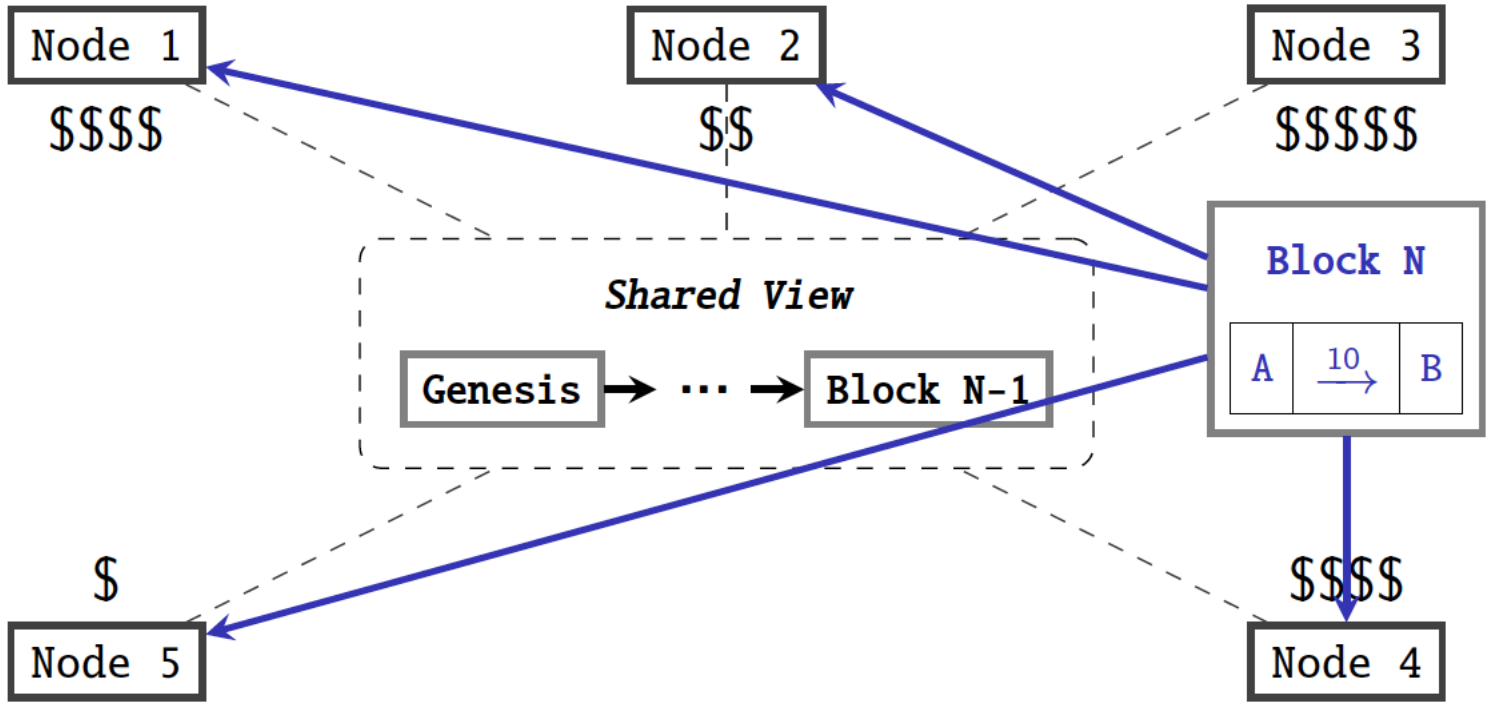
Transaction lifecycle in PoS



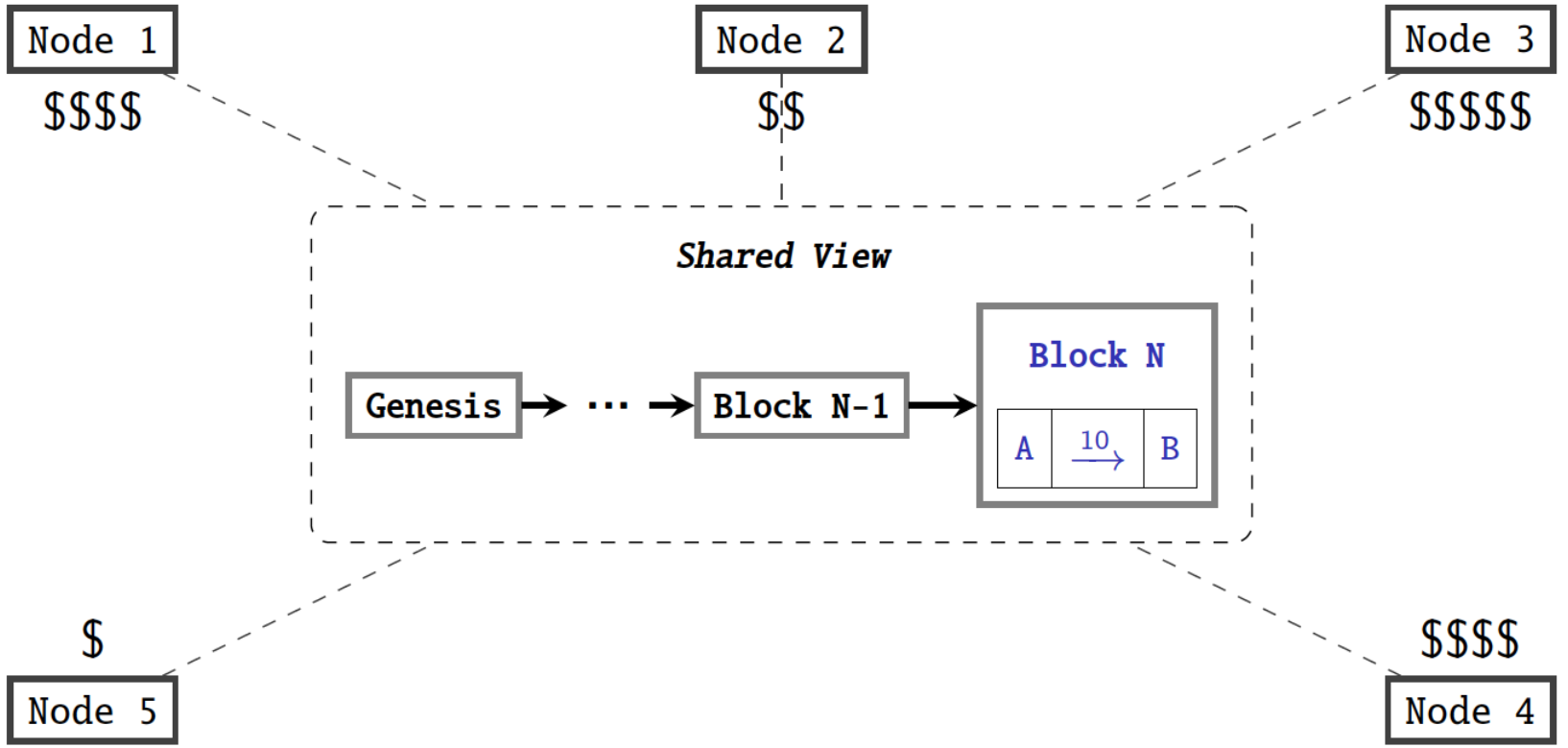
Transaction lifecycle in PoS



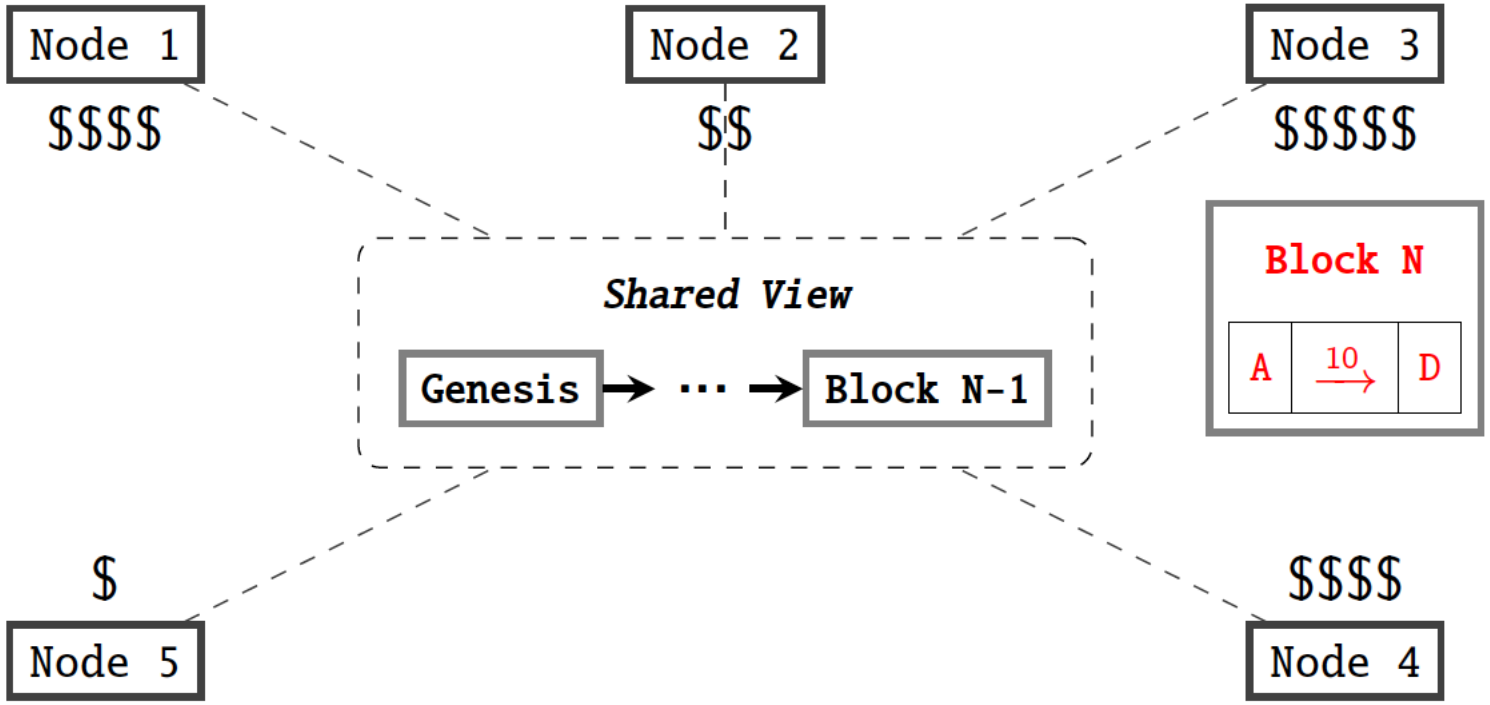
Transaction lifecycle in PoS



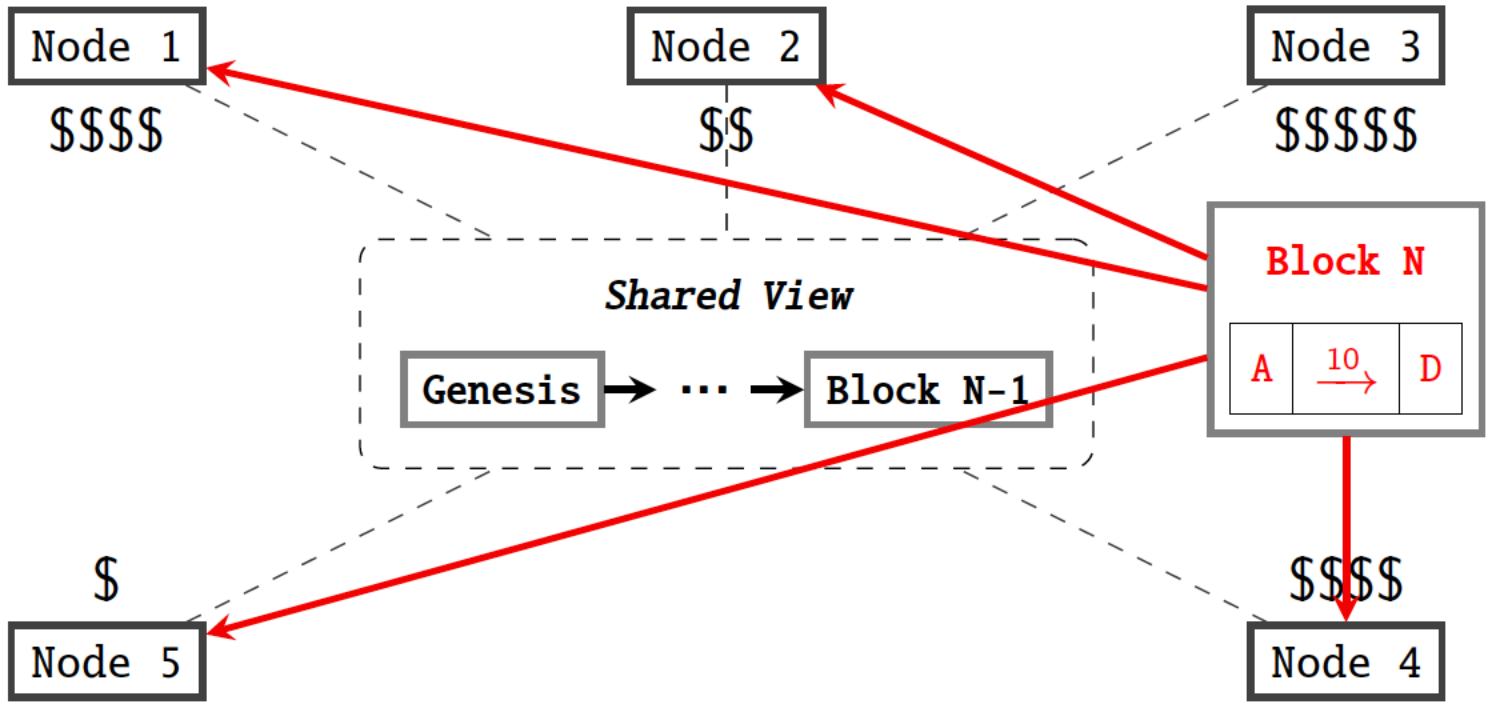
Transaction lifecycle in PoS



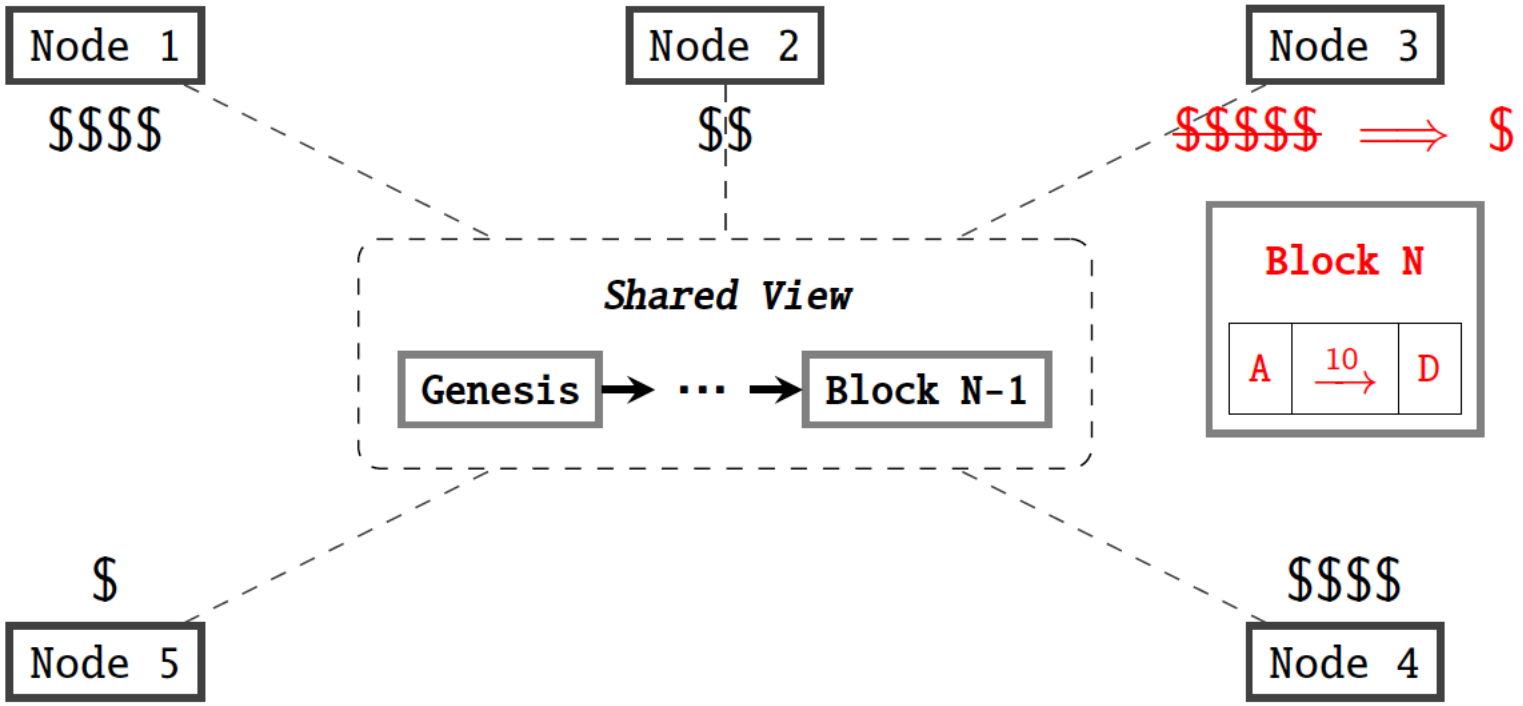
Transaction lifecycle in PoS



Transaction lifecycle in PoS



Transaction lifecycle in PoS



Catching lies

- If a validator node gets caught lying, its stake is burned!
- Other nodes may catch a fraudulent block by comparing it with the transaction that Alice intended to perform
  - e.g., by checking Ethereum's "mempool"
- This works as long as the attacker does not control a majority of stake in the system

The 51% attack on PoS

- **Q:** What if the attacker controls ≥ 50% of staked resources?
- **A:** The attacker can prove fraudulent transactions.

The 51% attack on PoS

- **Q:** What if the attacker controls ≥ 50% of staked resources?
- **A:** The attacker can prove fraudulent transactions.

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

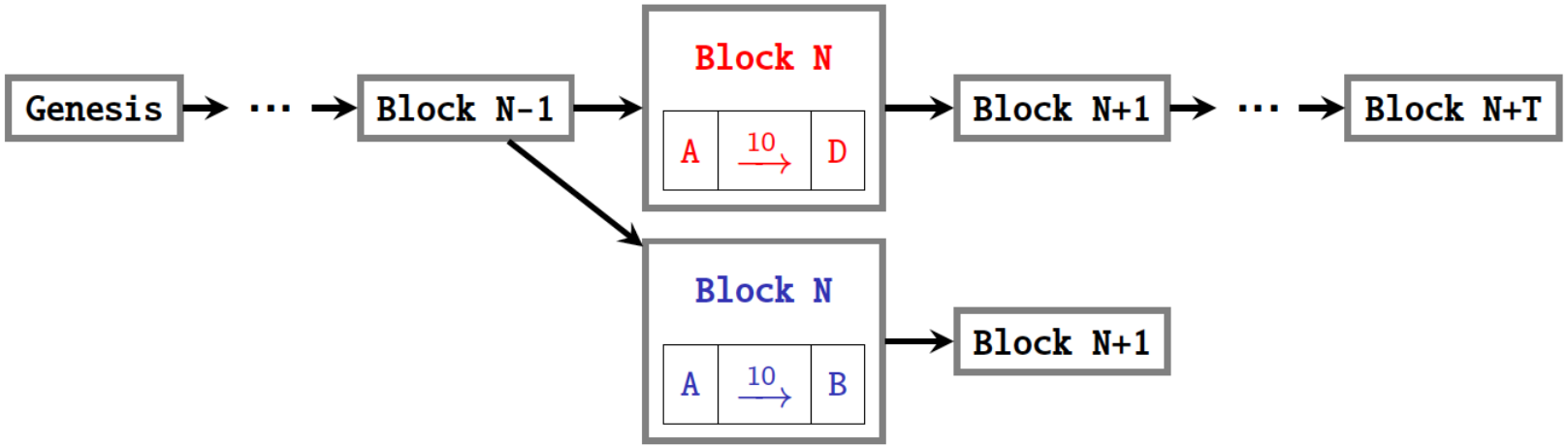
The 51% attack on PoS

- **Q:** What if the attacker controls ≥ 50% of staked resources?
- **A:** The attacker can prove fraudulent transactions.

- **Q:** Is the 51% attack less likely in PoS compared with PoW?
- **A:** Yes, because in PoS, the attacker loses the weapon to future attacks, i.e., all the stake are gone, and is not easily recoverable!

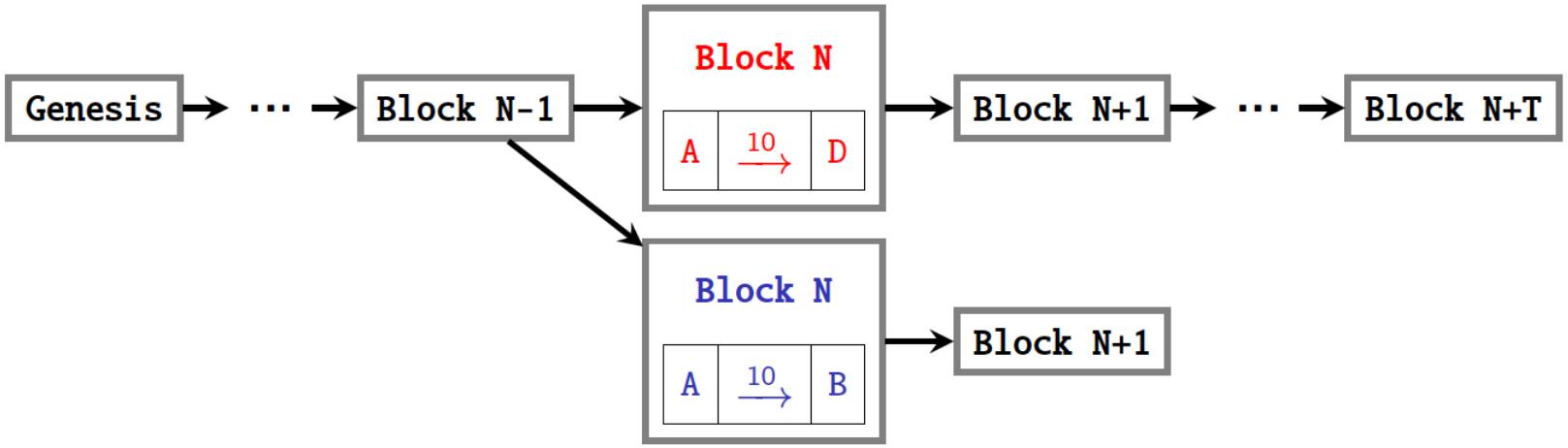
Hard fork as a recovery of a 51% attack

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



Hard fork as a recovery of a 51% attack

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



- NOTE: The forked chain can be shorter than the previous chain!
  - A higher level of social coordination is required

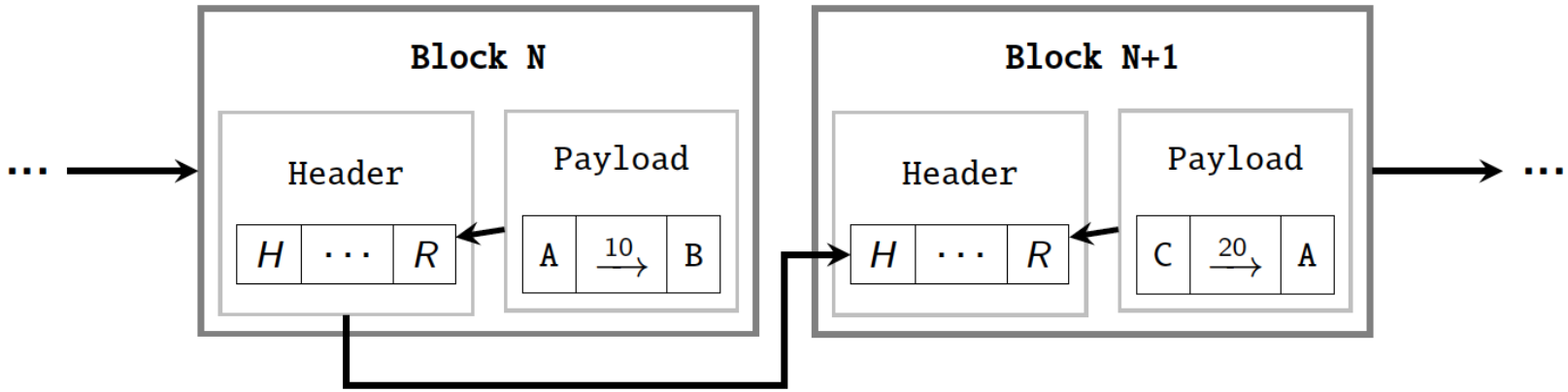
Hard fork as a recovery of a 51% attack

- In **PoS**, we do a hard fork to invalidate fraudulent transactions AND wipe out the attacker who controls ≥ 50% of the staked resources.

- In **PoW**, the hard fork can only invalidate transaction WHILE the ≥ 50% computational power is still controlled by the attacker

Chain validation

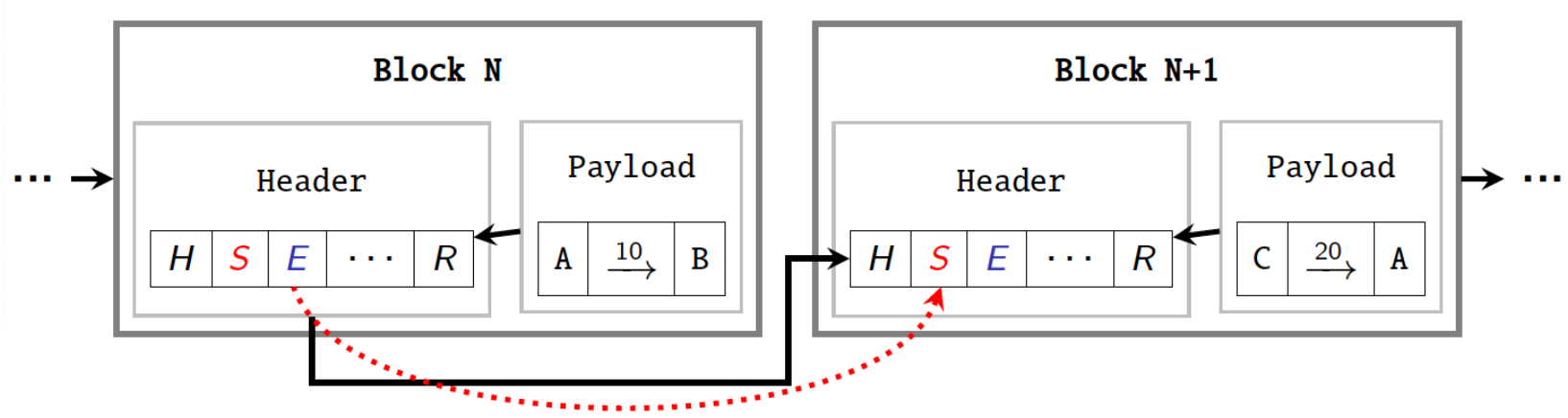
- If Alice shows Bob, the Pizzeria owner, the following blockchain, why would Bob accept it? Why would Bob believe that:
  - It is hard for Alice to produce such a chain of blocks
  - There does not exist a better chain of blocks as of now



- With PoS, forging a blockchain would be easy!

Chain validation

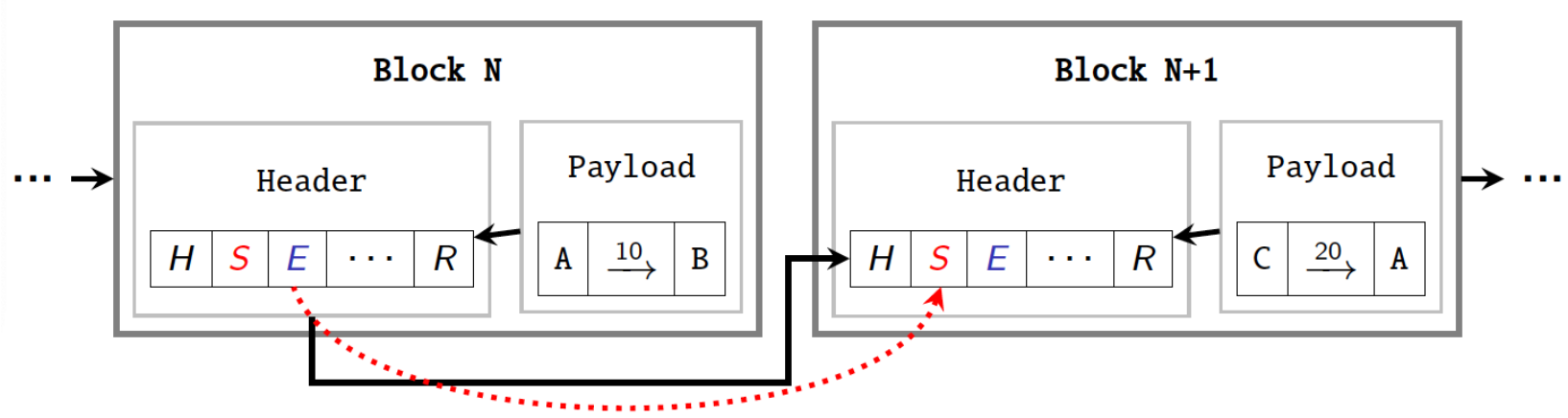
- This turns out to be an extremely complicated problem!



- *S* - Signature of the proposer of this block
- *E* - Election packet that records how this proposer is elected

Chain validation

- This turns out to be an extremely complicated problem!

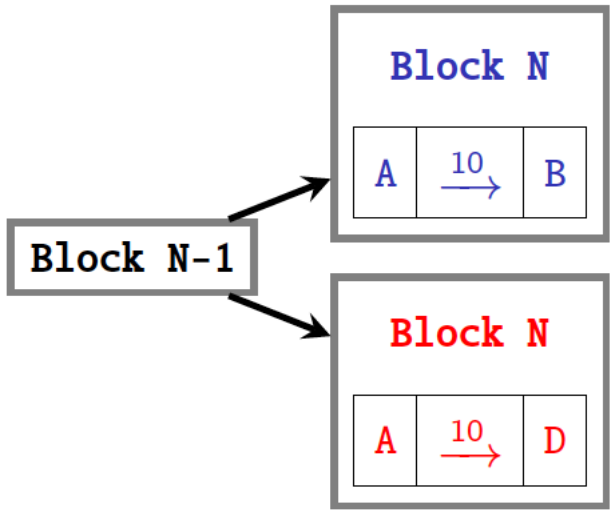


- *S* - Signature of the proposer of this block
- *E* - Election packet that records how this proposer is elected

Q: What are the issues with this scheme?

The Nothing-at-Stake problem

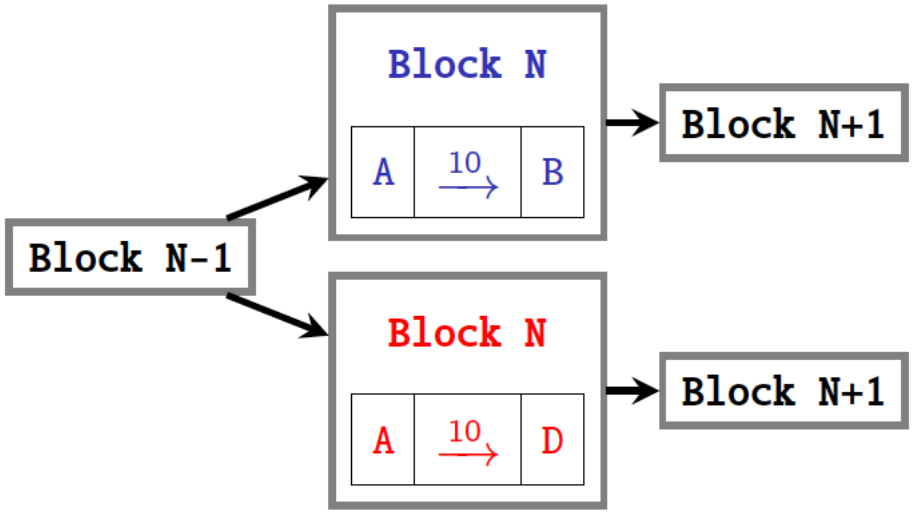
- Alice has some small stake (e.g., 1%) and can be elected as a block proposer:



- In one of her turn as a block proposer, Alice triggers a fork in the chain with an attempt to double-spend.

The Nothing-at-Stake problem

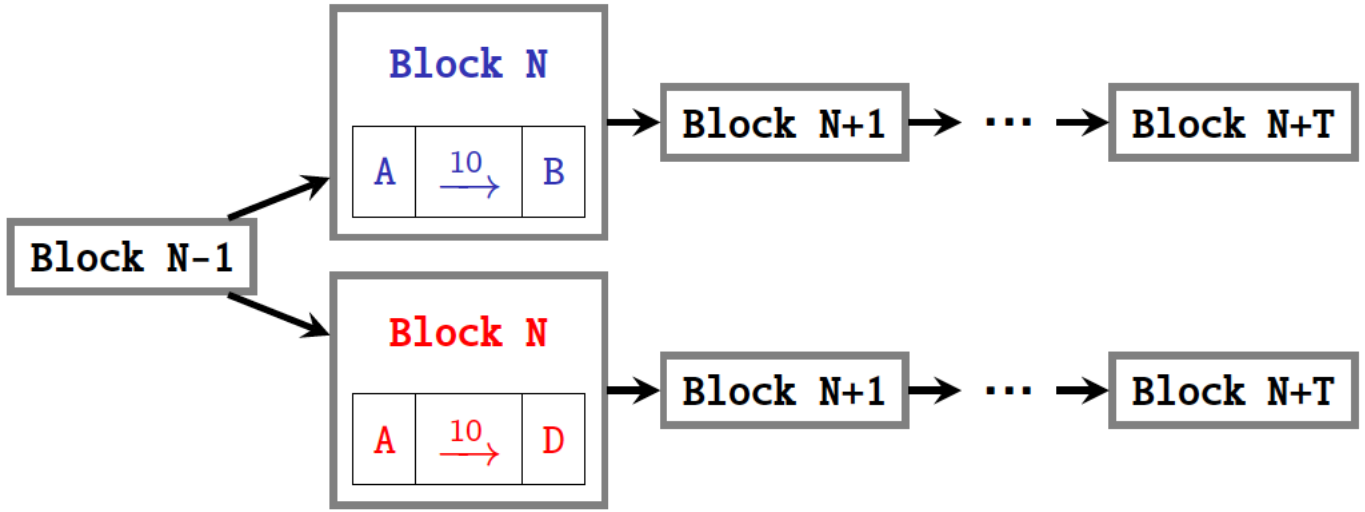
- Alice has some small stake (e.g., 1%) and can be elected as a block proposer:



- The next block proposer, even honest, has no incentive to select which chain to converge on. The proposer has no idea which chain will survive in the future, the logical thing to do is to mine on both

The Nothing-at-Stake problem

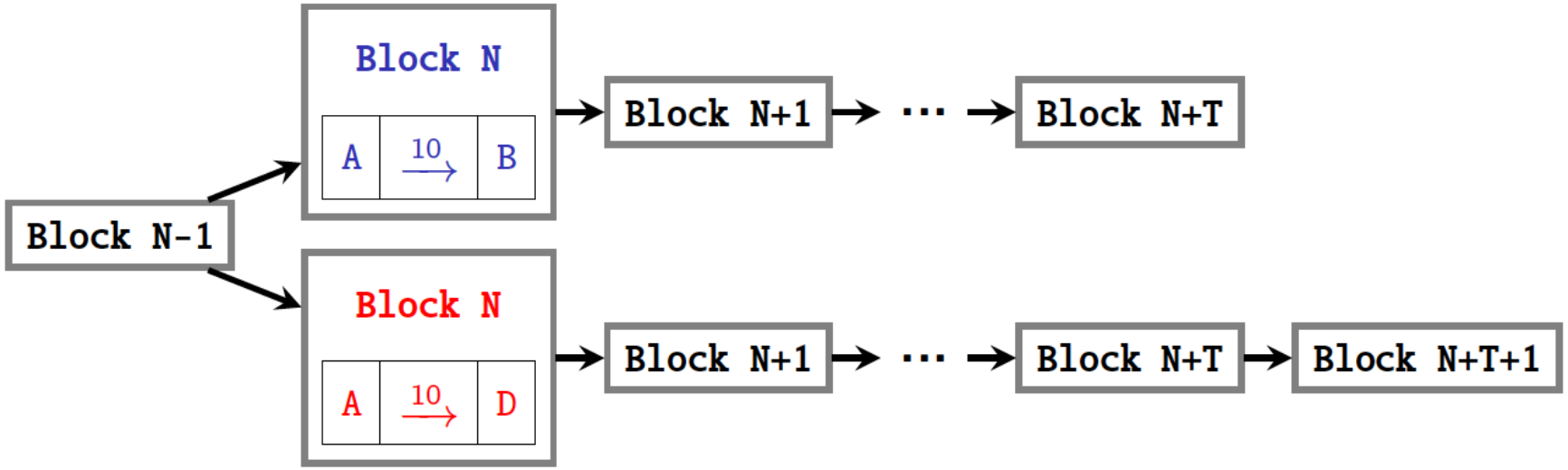
- Alice has some small stake (e.g., 1%) and can be elected as a block proposer:



- The next block proposer, even honest, has no incentive to select which chain to converge on. The proposer has no idea which chain will survive in the future, the logical thing to do is to mine on both

The Nothing-at-Stake problem

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



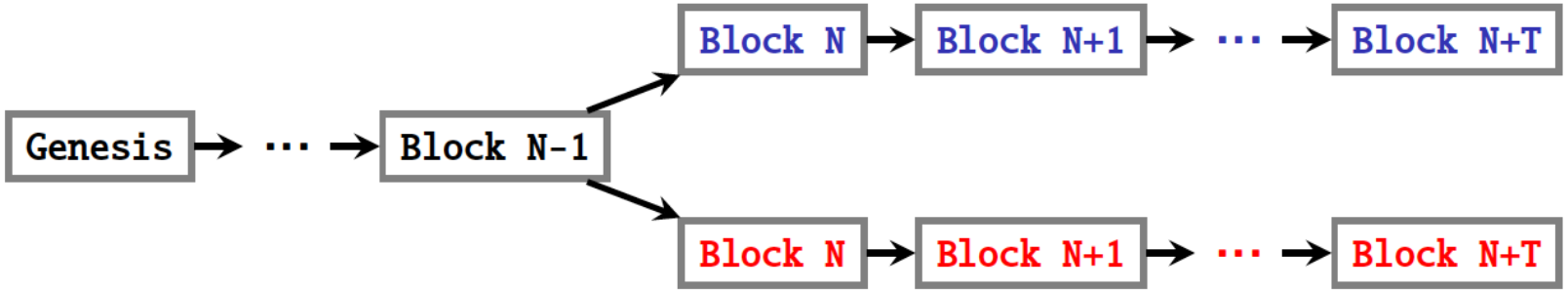
- When its Alice's turn again, she only append a block to the chain that is more favorable to her. The other chain dies as a result.
- This is sometimes called the 1% attack.

The Nothing-at-Stake problem

- Solution? There is no common solution. Different PoS chains adopt different mechanisms.
- The Slash protocol (Ethereum PoS candidate) has two rules:
  - Penalize those who “equivocated” on a given block, i.e., voted on two different versions of it.
  - Penalize those who voted on the wrong block, regardless of whether they double-voted.

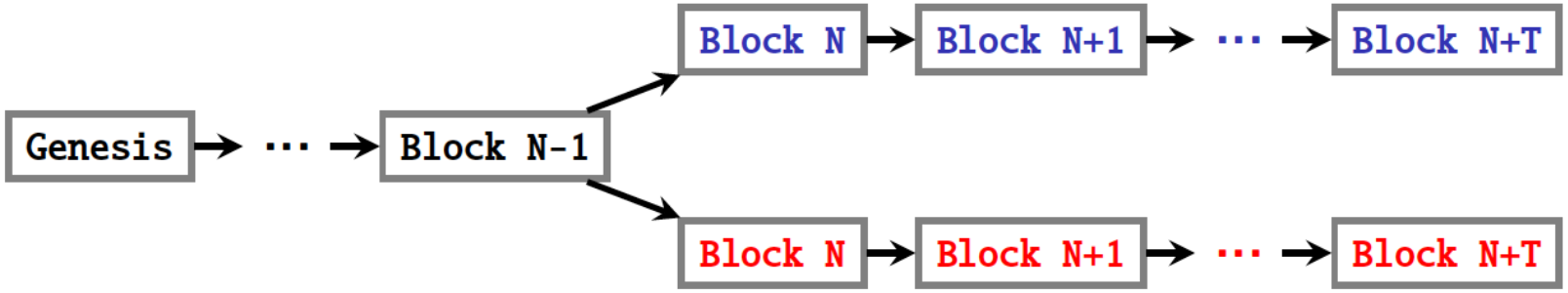
Long-range attacks (the bootstraping problem)

- A validator node could forge an entire chain by itself
- If Bob, a new user, joins the network, which chain should he accept?



Long-range attacks (the bootstraping problem)

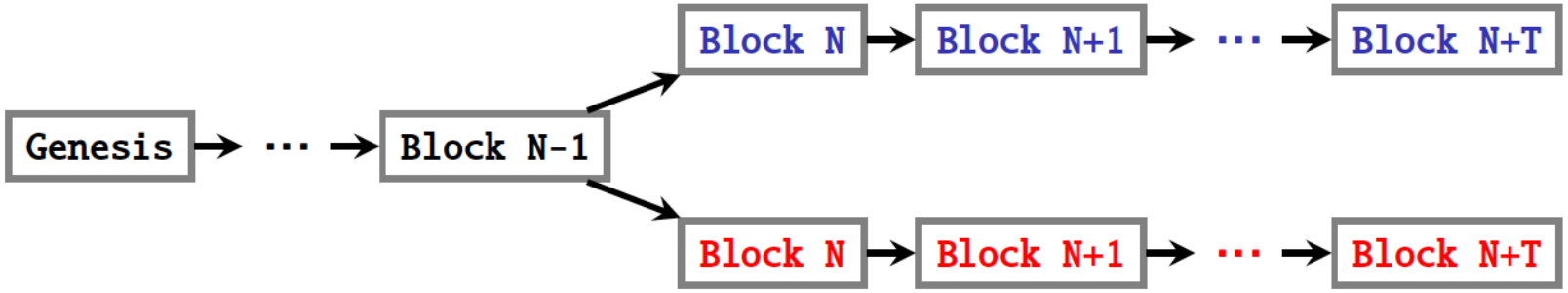
- A validator node could forge an entire chain by itself
- If Bob, a new user, joins the network, which chain should he accept?



Q: Why is this not a problem in PoW?

Long-range attacks (the bootstraping problem)

- A validator node could forge an entire chain by itself
- If Bob, a new user, joins the network, which chain should he accept?



Q: Why is this not a problem in PoW?

A: Because it is computationally expensive to create a counterfeit chain in PoW. But it is easy (almost no cost) in the PoS case

Long-range attacks (the bootstraping problem)

- 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.
  - Extremely complicated solutions have been proposed e.g., Ouroboros Genesis.