

CS489/698 Privacy, Cryptography, Network and Data Security


Blockchain

Fall 2024, Tuesday/Thursday 02:30pm-03:50pm

The problem of consensus

Bitcoin thieves got away with ATM double-spending spree across Canada

8:31 PM • May 07, 2021 — by Protos Staff

Share 




 **HoneyBadger** 
@badger_coin · Follow 


Your help in locating these criminals would be greatly appreciated.

 **Calgary Police**  @CalgaryPolice

Our Cybercrime Team has launched a joint investigation with @TorontoPolice @HamiltonPolice @wpgpolice @HaltonPolice into a national bitcoin fraud with losses estimated at \$195,000. Do you know these suspects? newsroom.calgary.ca/seeking-public...



NEED TO IDENTIFY **Cybercrime - Bitcoin Fraud**
It is alleged that between Sunday, Sept. 16, 2018, and Wednesday, Sept. 26, 2018, 112 fraudulent transactions had been made at bitcoin kiosks in seven cities across Canada. The total dollar loss as a result of these fraudulent transactions was \$195,000.
Case #: 18453695 / 5075 

Non-Emergency: 403.266.1234 | Emergency: 9.11 | calgarycrimestoppers.org | 1.800.222.TIPS (8477) 

6:02 PM · Mar 13, 2019

[View post](#)

 4  Reply  Copy link

The double spending problem



Alice has coins and gives them to Bob



The double spending problem



Alice has coins and gives them to Bob



Later..I wants to
give another set
of coins to Carol

The double spending problem



Alice has coins and gives them to Bob



Later..I wants to
give another set
of coins to Carol

How do I
know that
Alice still has
the coins?



The double spending problem



Alice has coins and gives them to Bob



Later..I wants to
give another set
of coins to Carol

**Local state at Alice is not
reliable, because Alice can
reset after transaction with
Bob**

How do I
know that
Alice still has
the coins?



The double spending problem



Alice has coins and give

Later..I wants

of coin

Thus: need a global state
E.g., Trusted party (bank
reconciliation),
Distributed (blockchain)



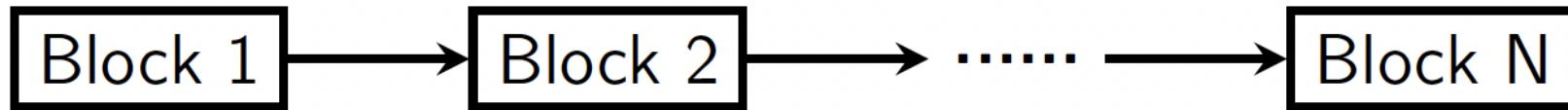
I know that Alice still has the coins?

Local state at Alice is not reliable, because it can be reset after transaction with Bob

An overview of blockchain design

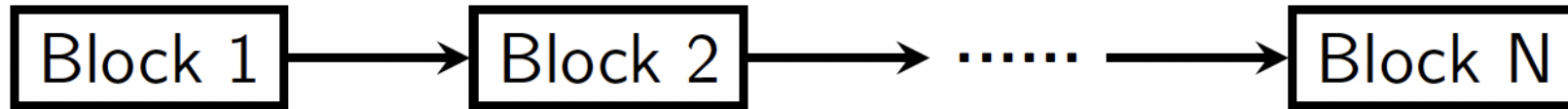
What is a blockchain?

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



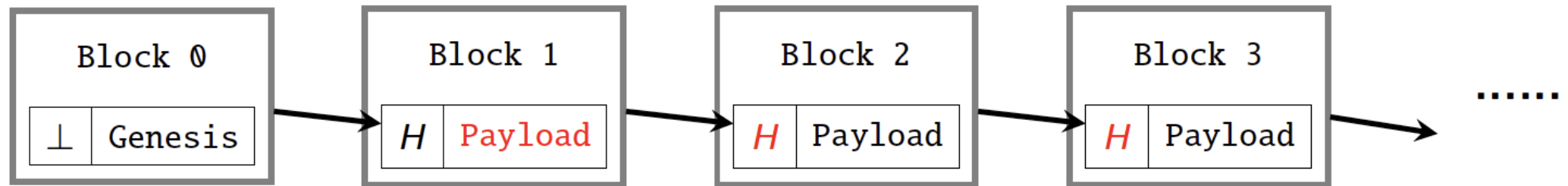
What is a blockchain?

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



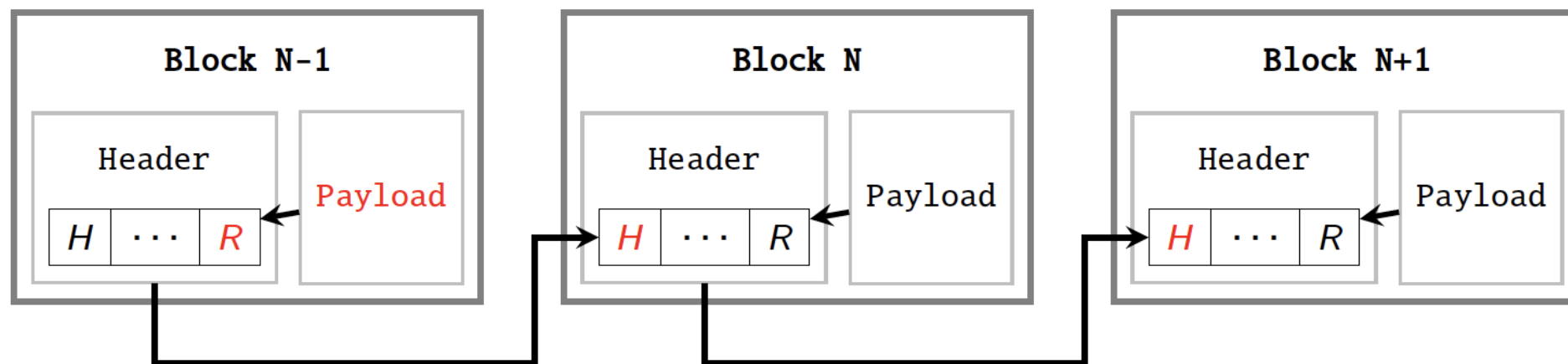
- What does chaining mean here?
 - Linked list? Some cryptographic construct?
- What goes into these blocks?
 - Anything? A fixed format? What makes a block valid?
- Who can put up a block?
 - A single entity? A group of people? Anyone with Internet access?
- How to ensure a same view of the chain?
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

A basic chaining scheme



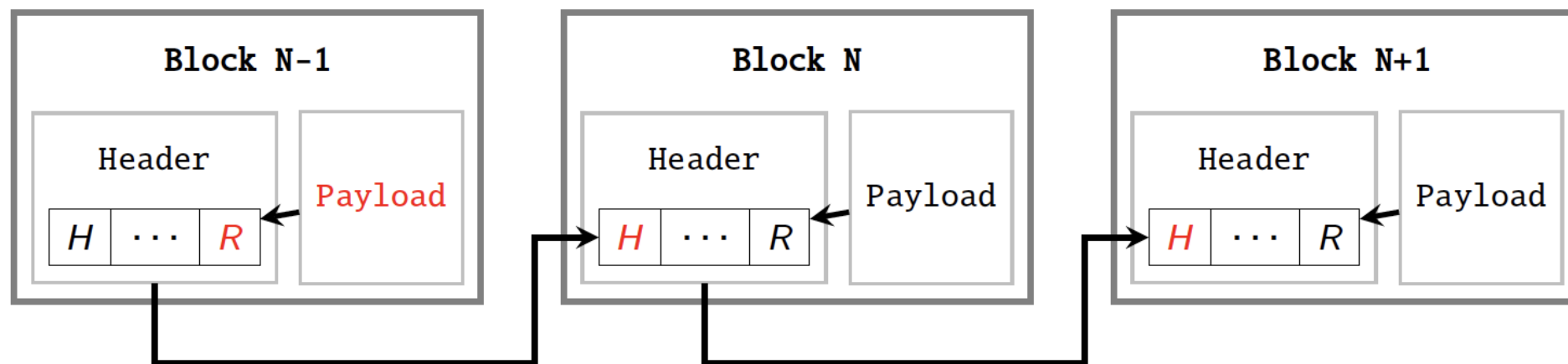
- Each block contains a cryptographic hash of the previous block
- Each block depends on the previous one

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

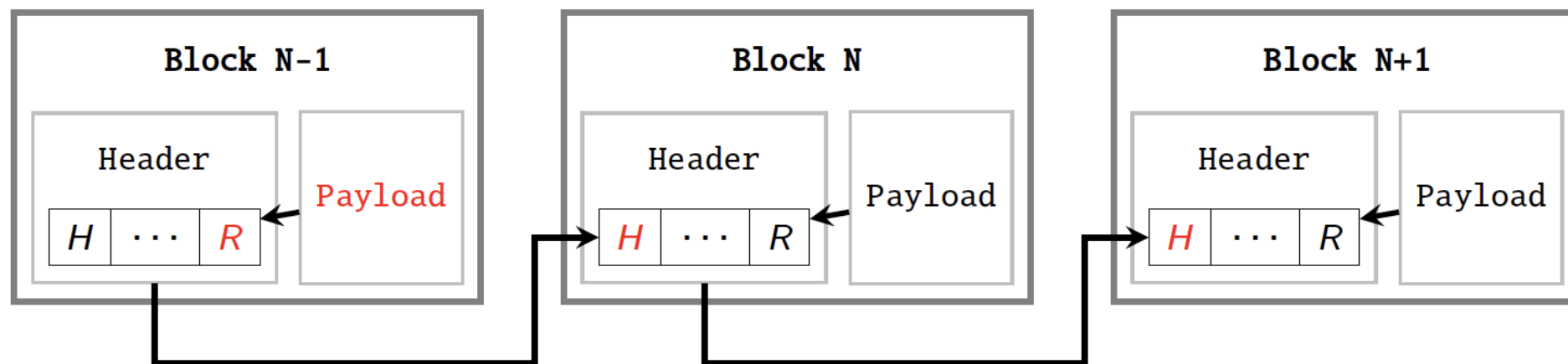
A basic chaining scheme



- 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?

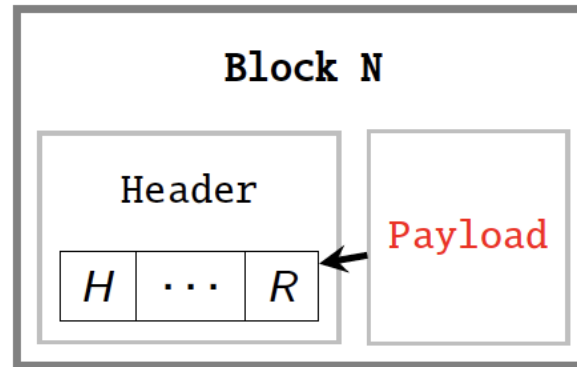
A basic chaining scheme



- 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

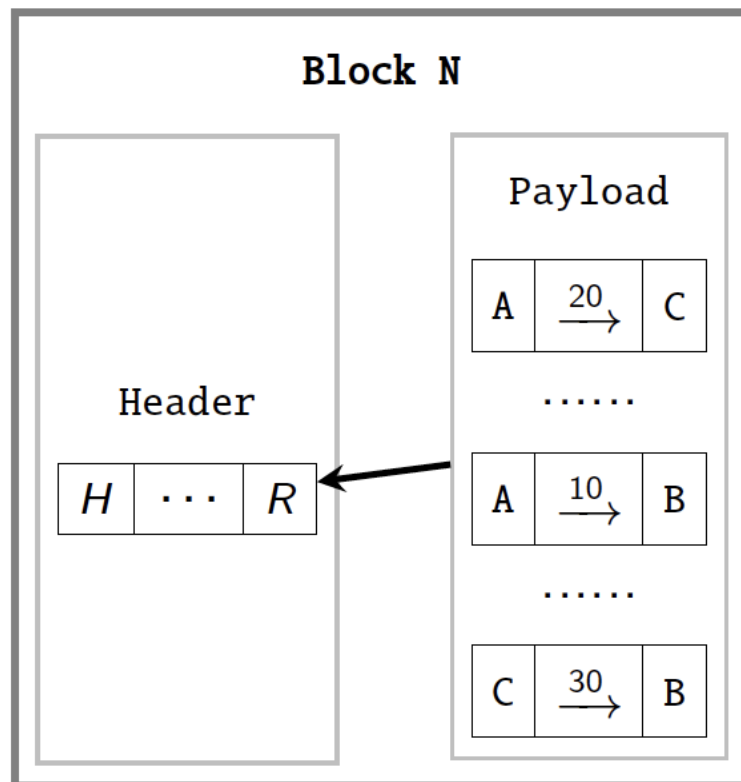
A: State is append only

What goes into the payload?

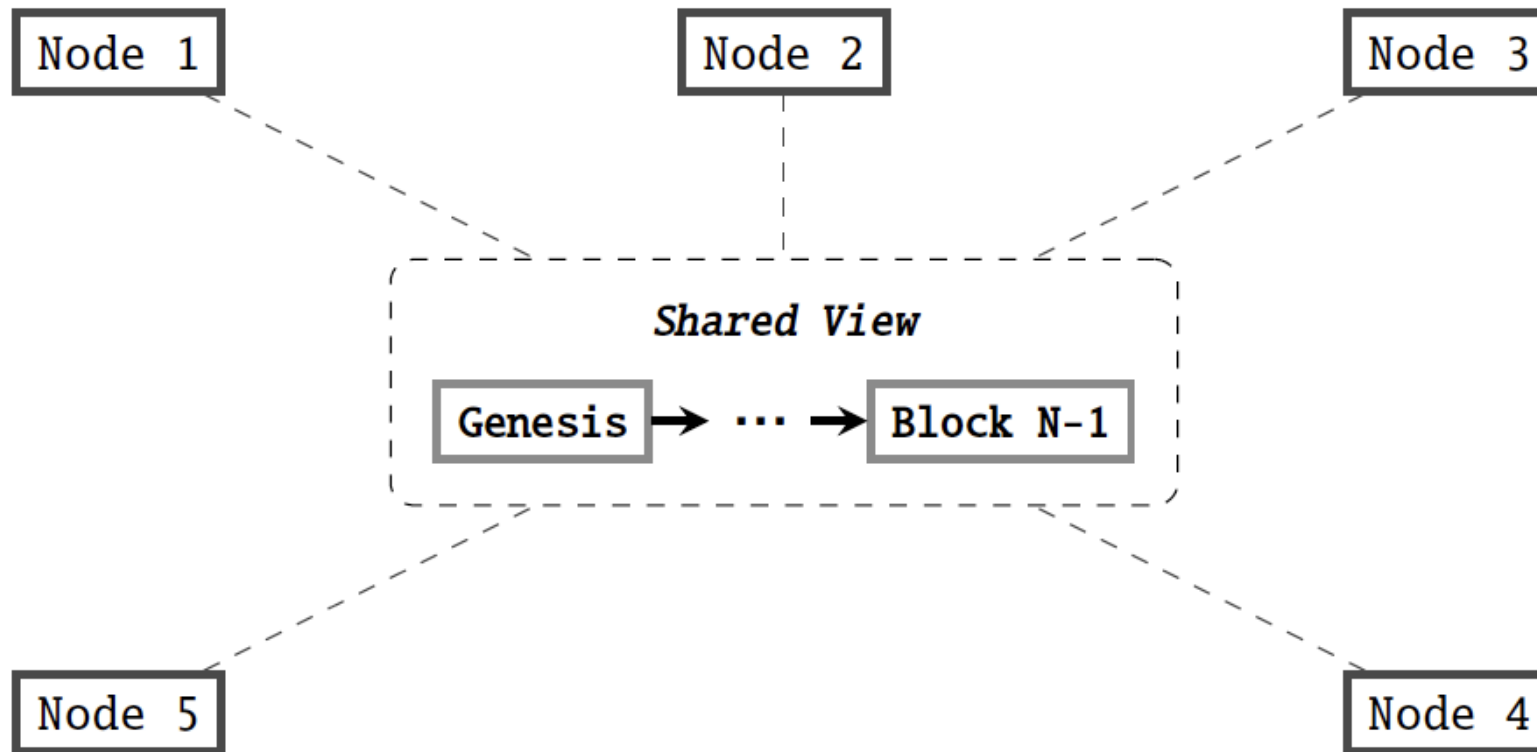


- Anything! Depending on how you plan to use this blockchain.
 - Bitcoin blockchain: **ledger**
 - Ethereum blockchain: **state machine**

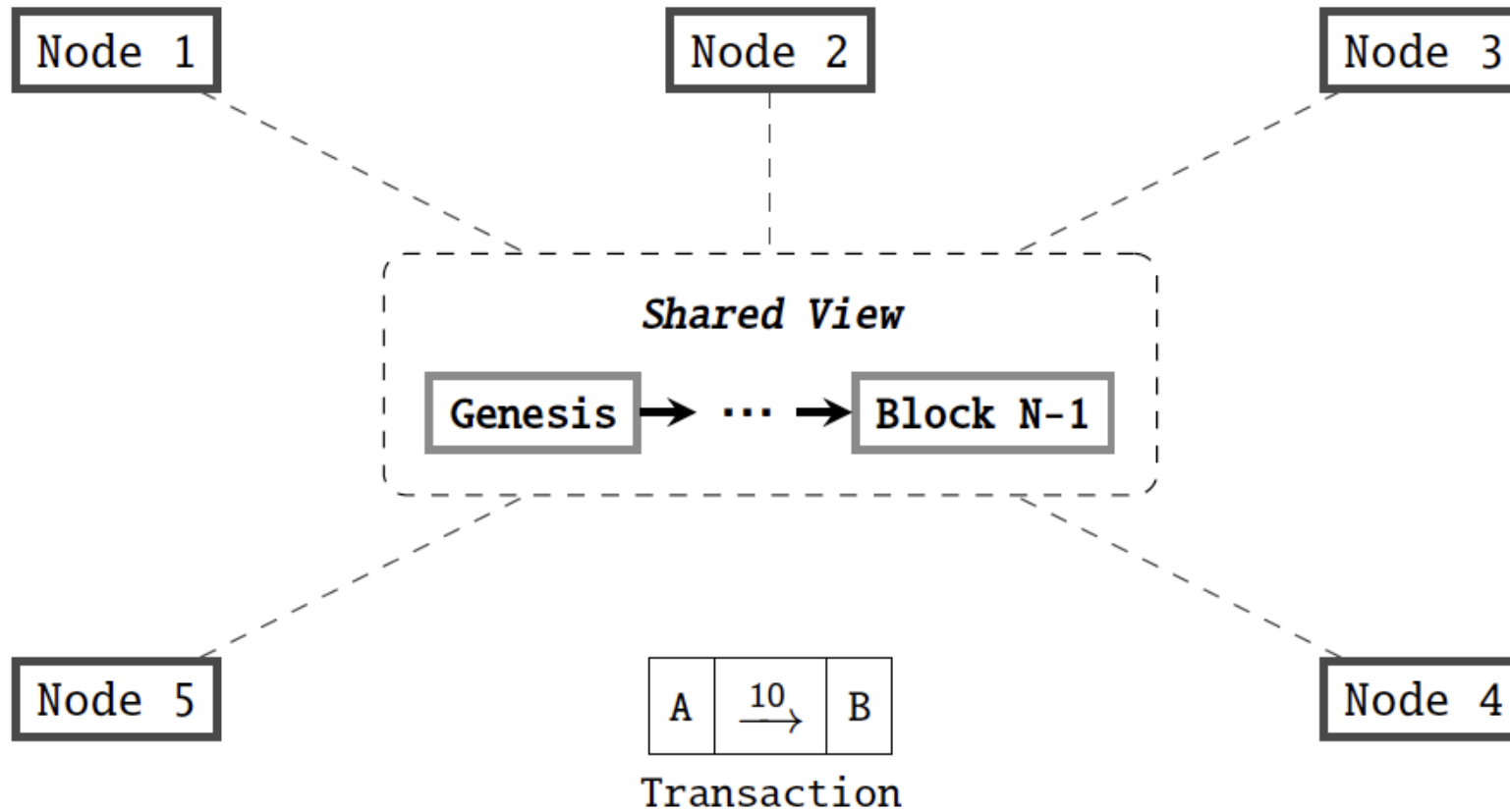
Payload example: a ledger



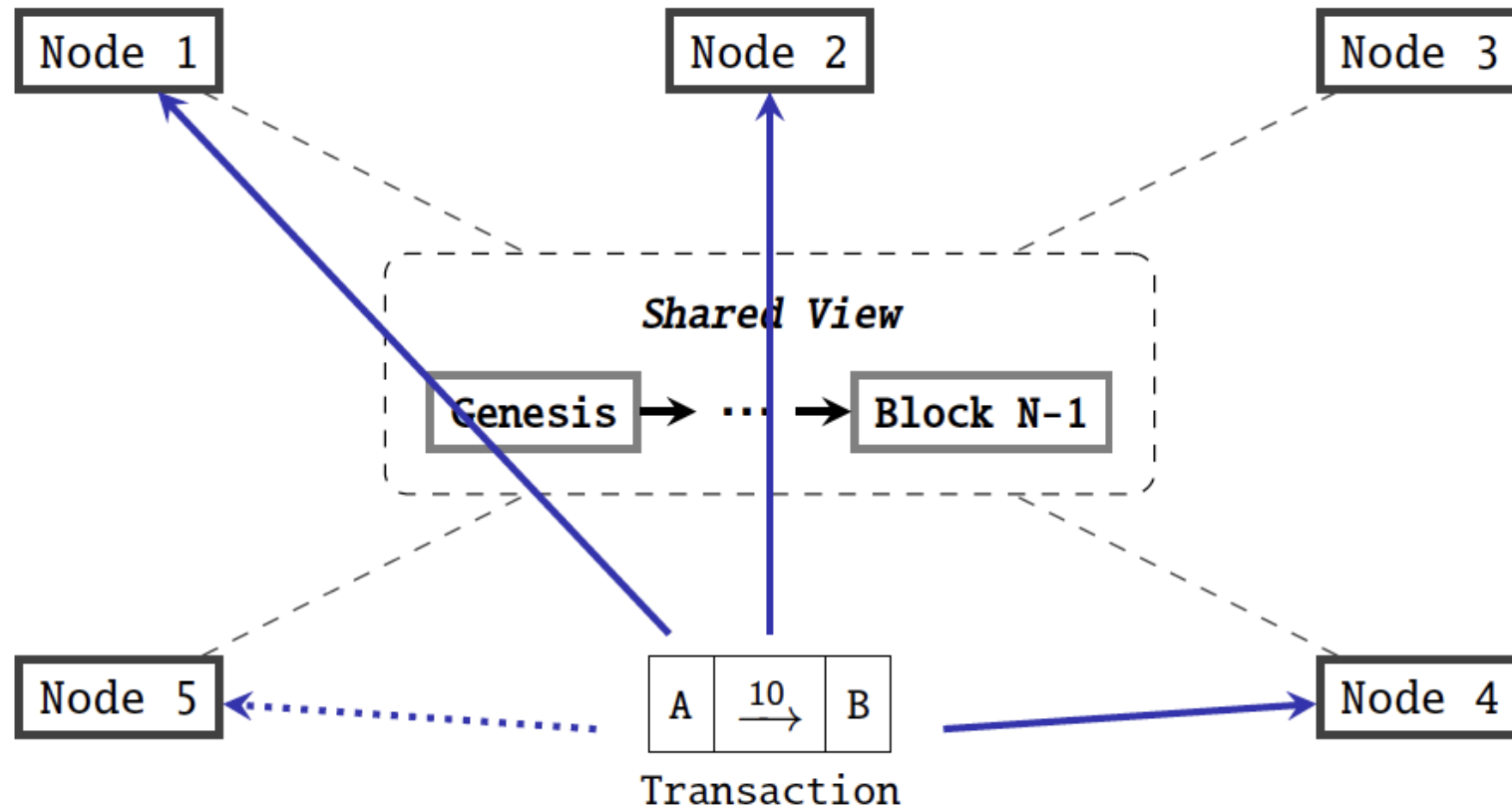
How does data get into the block?



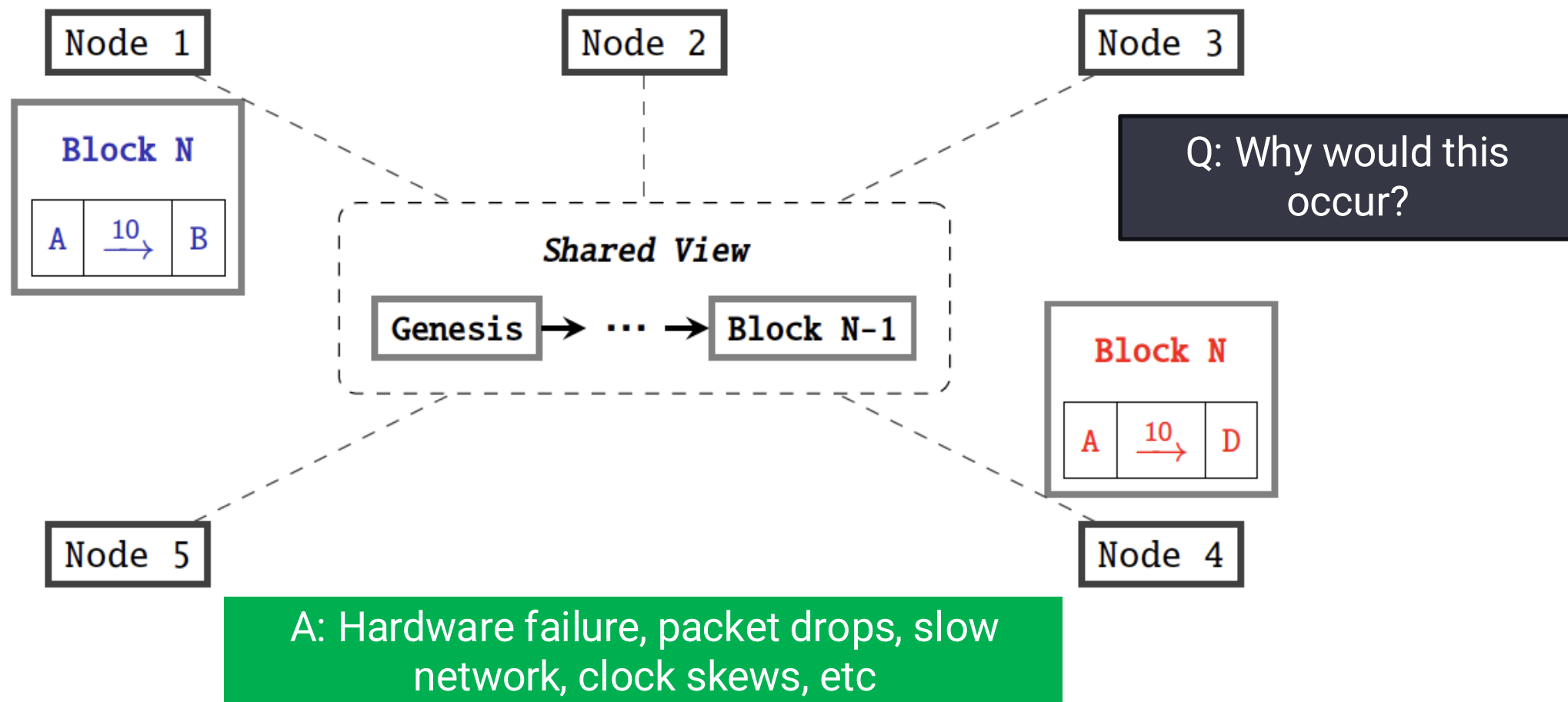
How does data get into the block?



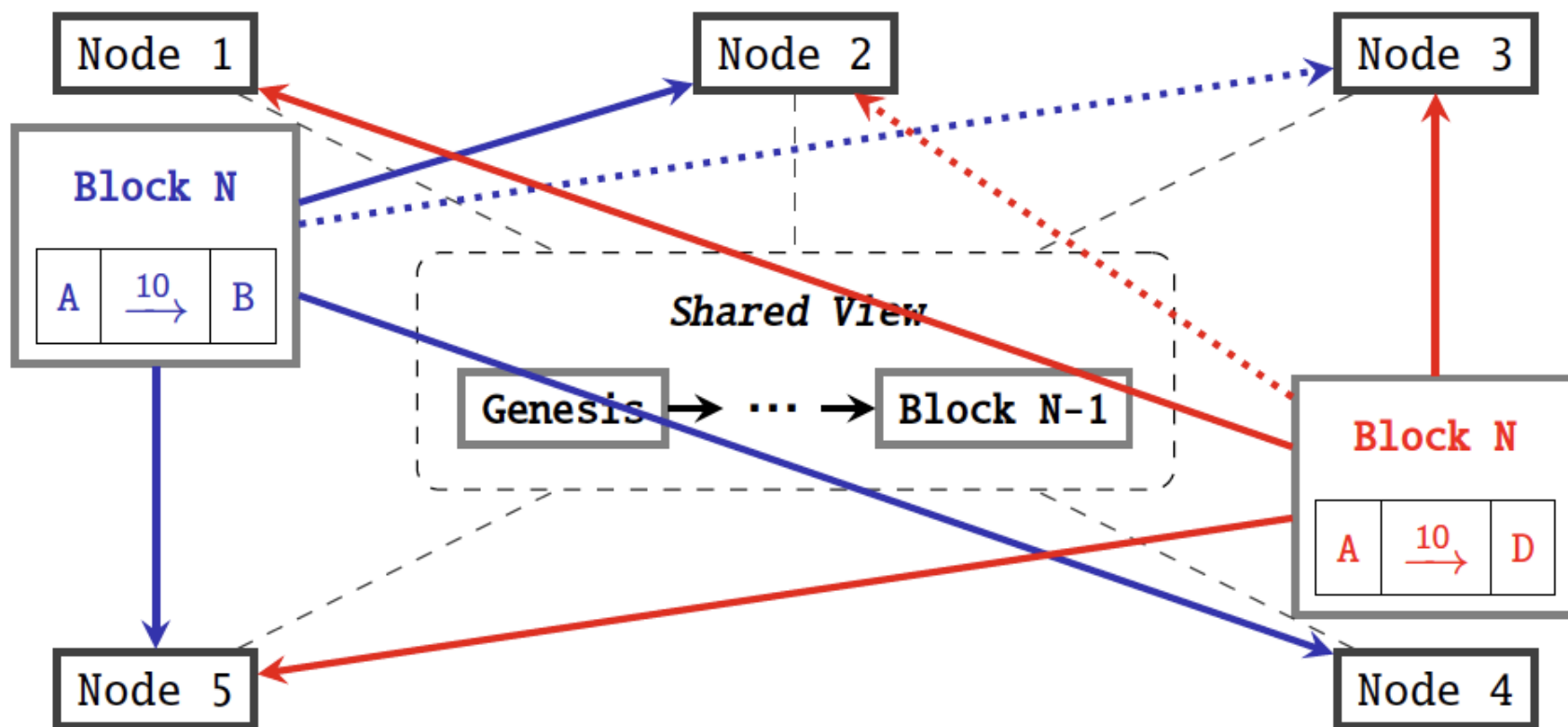
How does data get into the block?



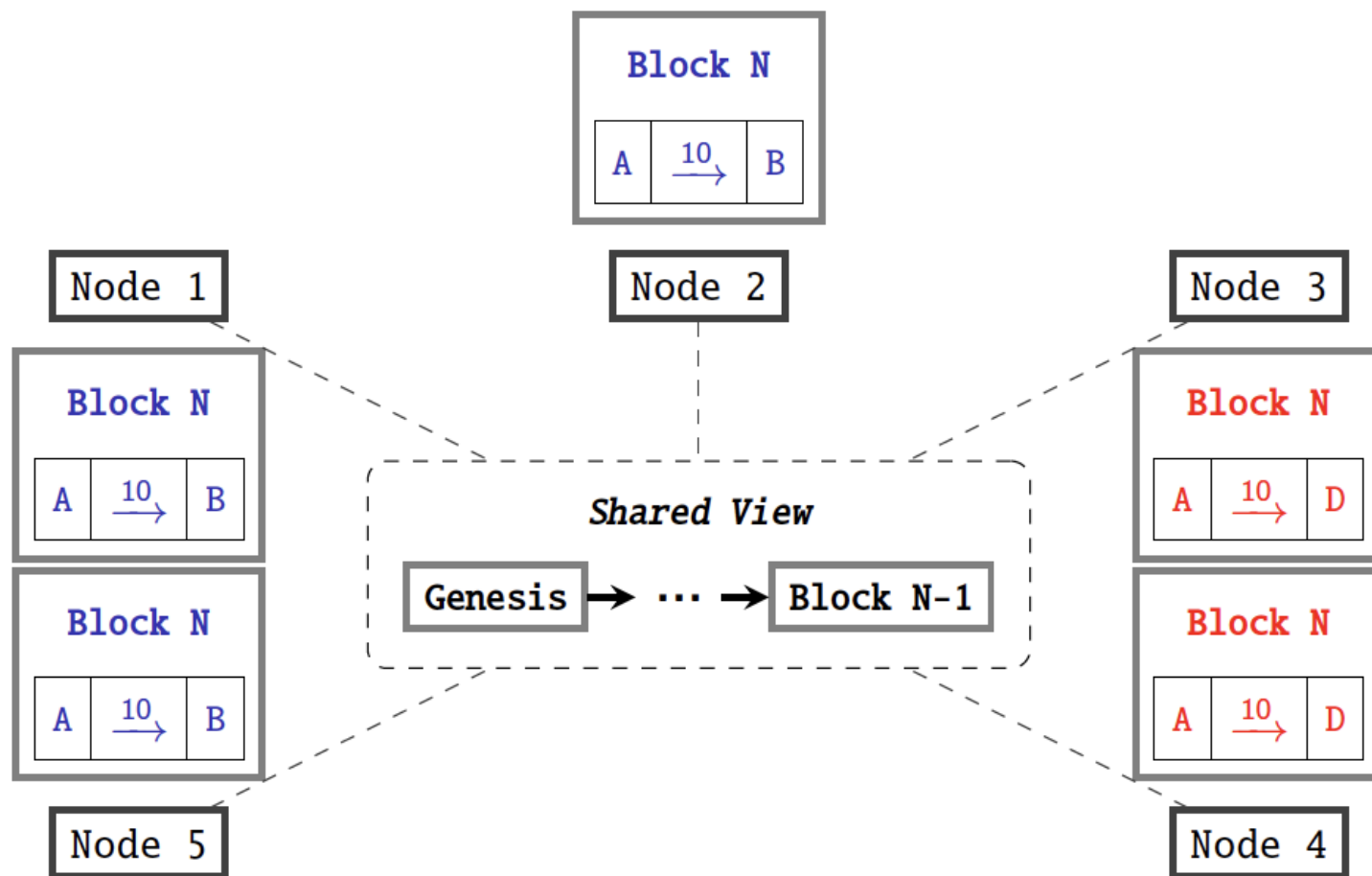
How does data get into the block?



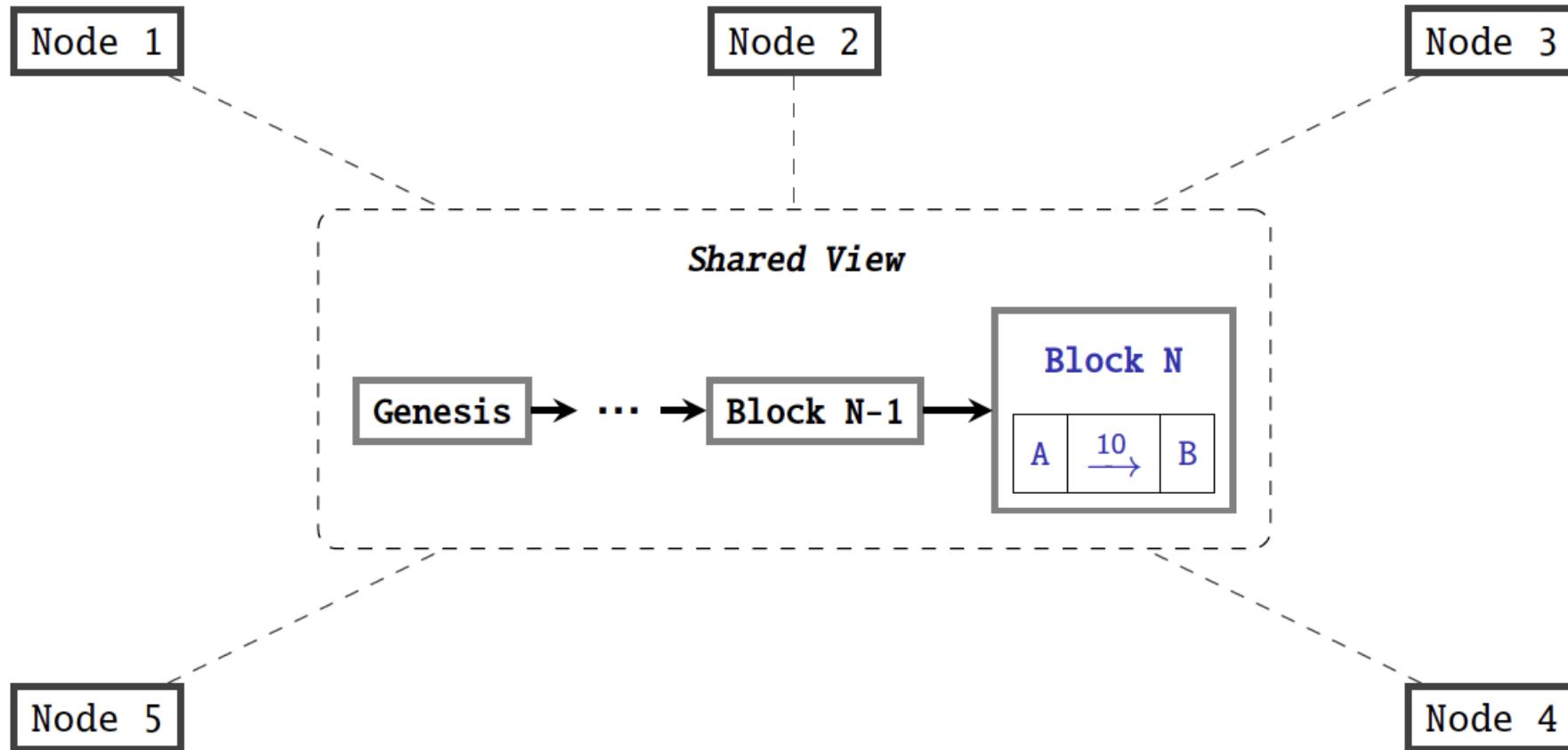
How does data get into the block?



How does data get into the block?

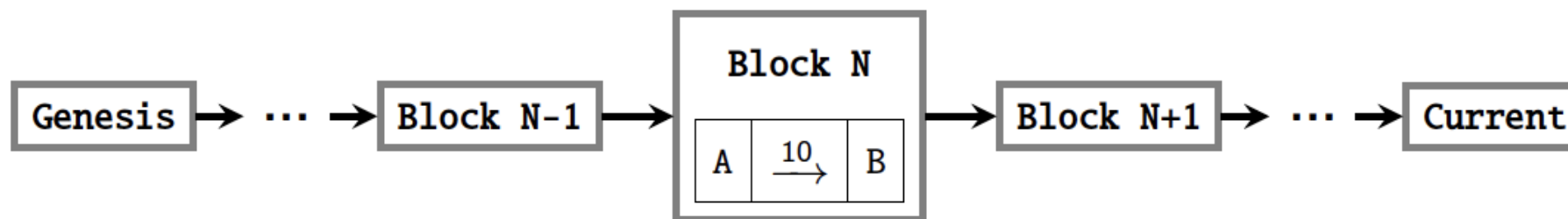


How does data get into the block?



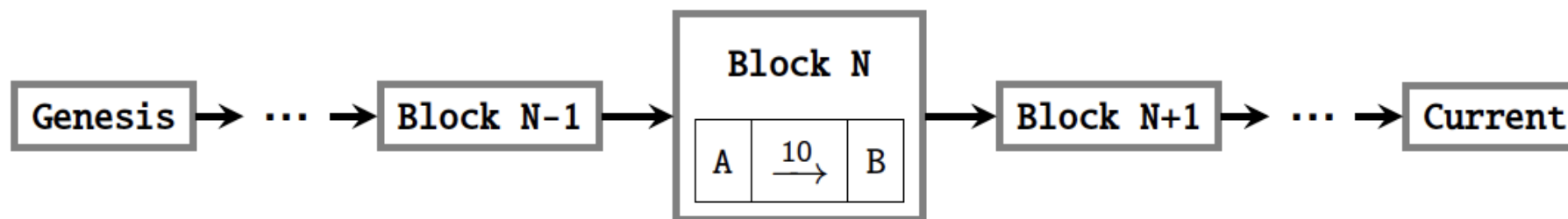
The power of consensus

- 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

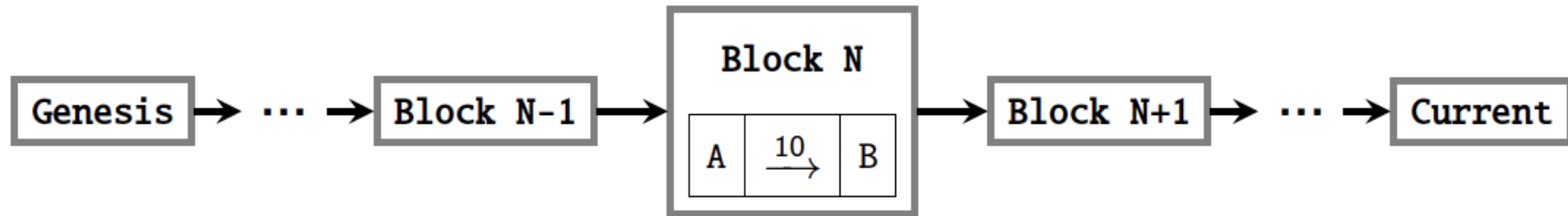
- Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:
 - Cash, debit card, credit card, e-transfer (e.g., Interac®)
 - An entry in the blockchain-based ledger



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

The power of consensus

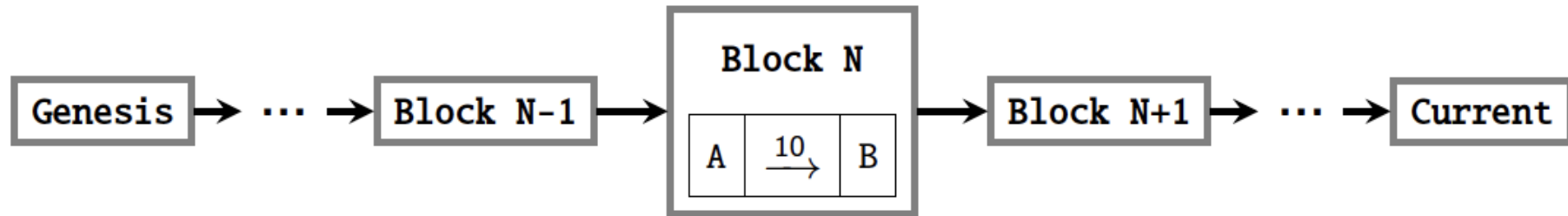
- Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:
 - Cash, debit card, credit card, e-transfer (e.g., Interac®)
 - An entry in the blockchain-based ledger



- 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

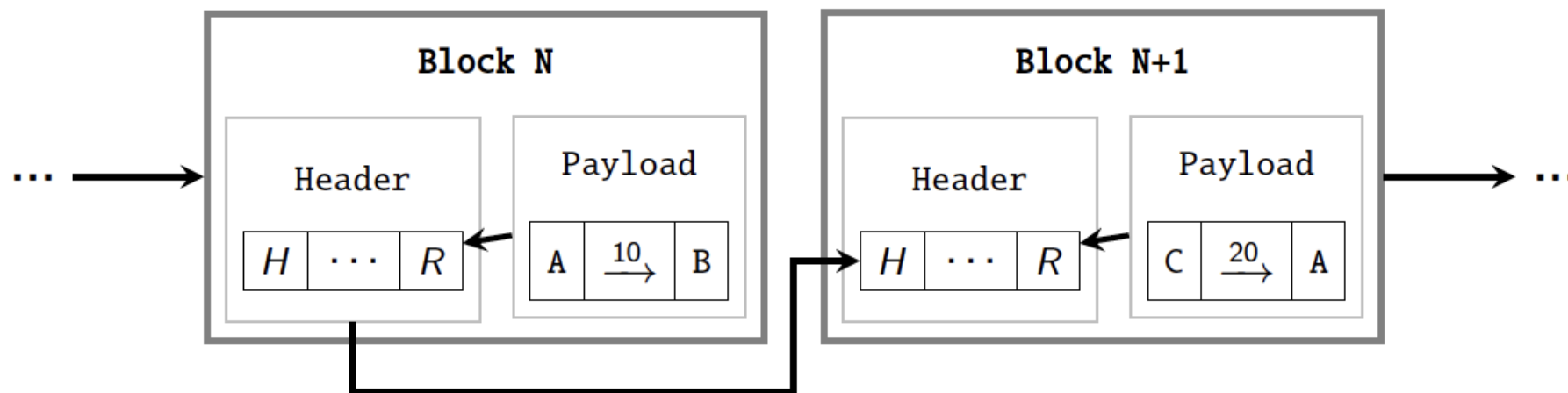
- Imagine Alice goes to Bob's Pizzeria and orders a pizza, she has the following payment options:
 - Cash, debit card, credit card, e-transfer (e.g., Interac®)
 - An entry in the blockchain-based ledger



- To the best of ~~Bob's~~ **everyone's** knowledge:
 - It is **hard** for Alice to produce such a chain of blocks (e.g., 51% Attack)
 - There does not exist a **better** chain of blocks as of now (e.g., Forks)

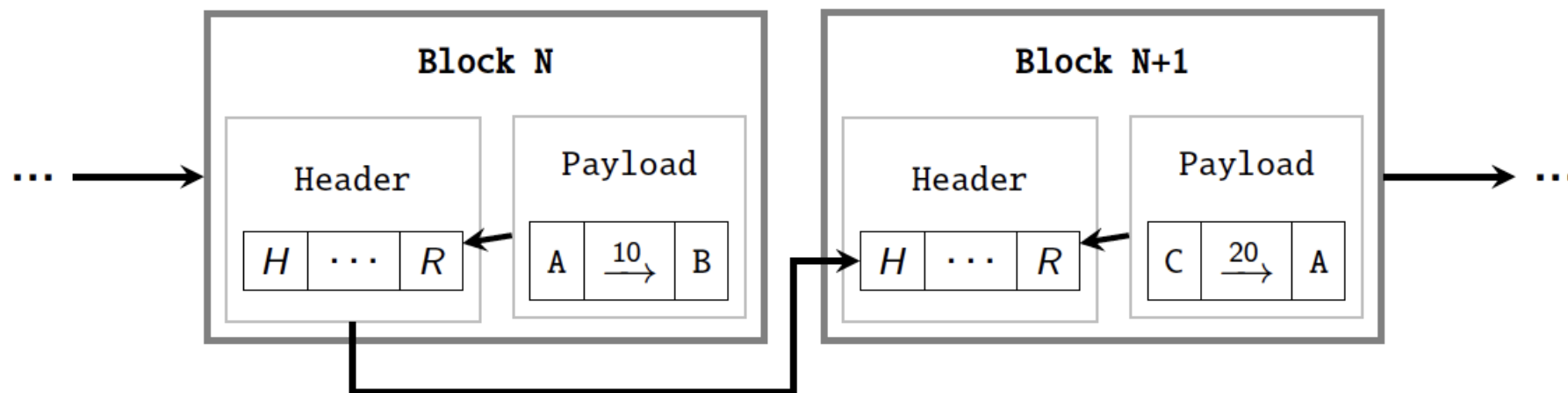
Consensus: Proof-of-work

How hard is it to alter this chain?

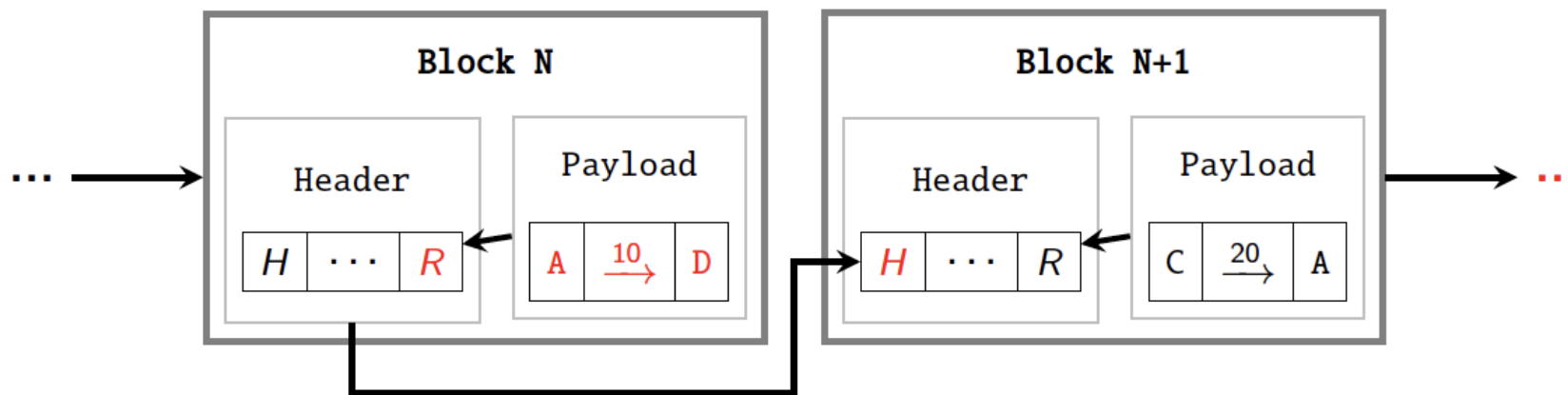


- This is the chain Alice shows Bob w.r.t her payment to Bob.

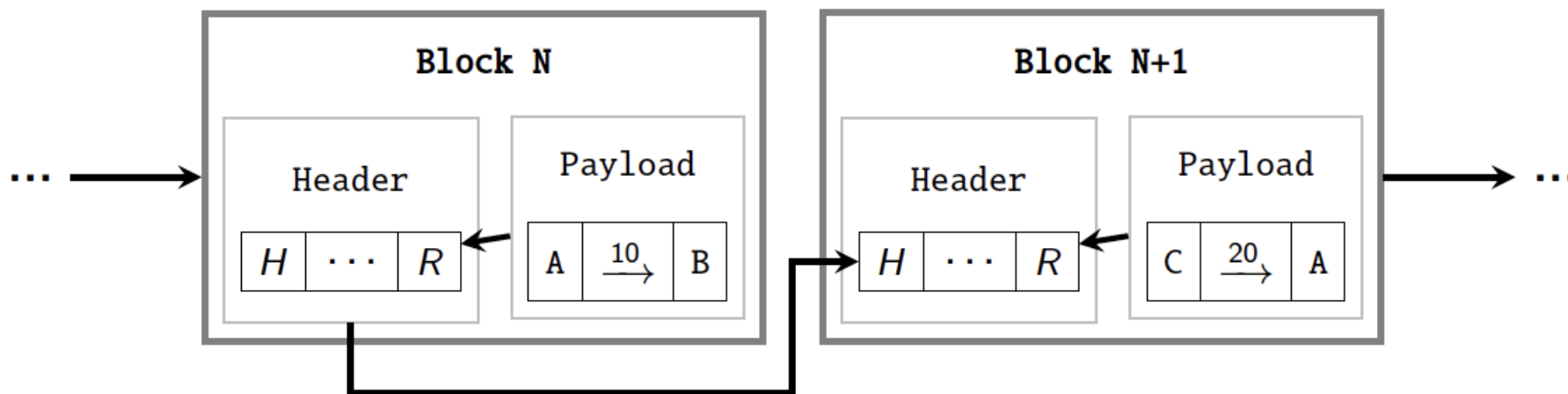
How hard is it to alter this chain?



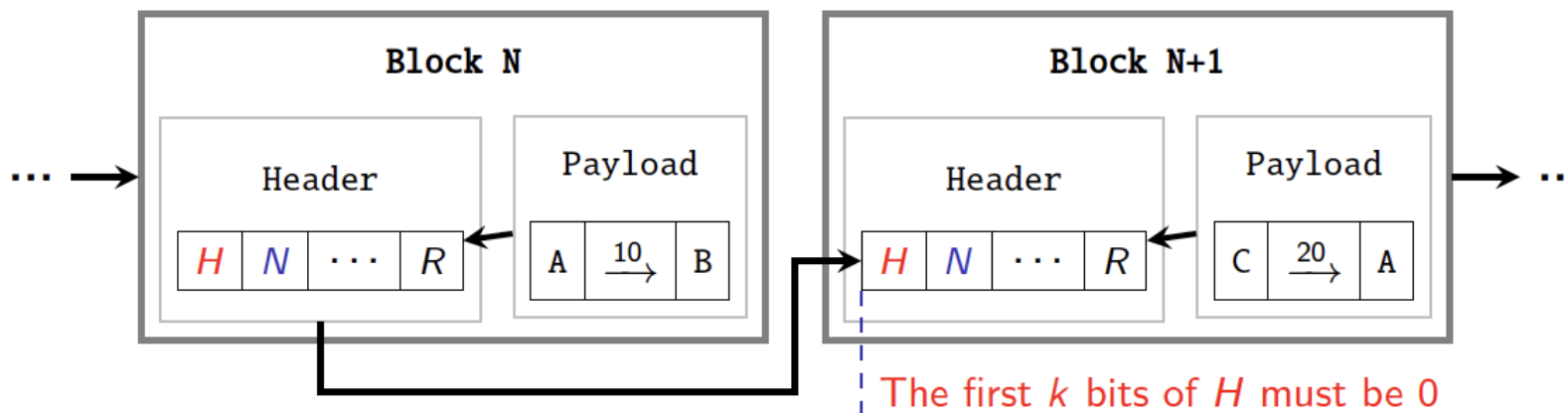
- It is **not hard** at all for Alice to revert this payment to Bob!



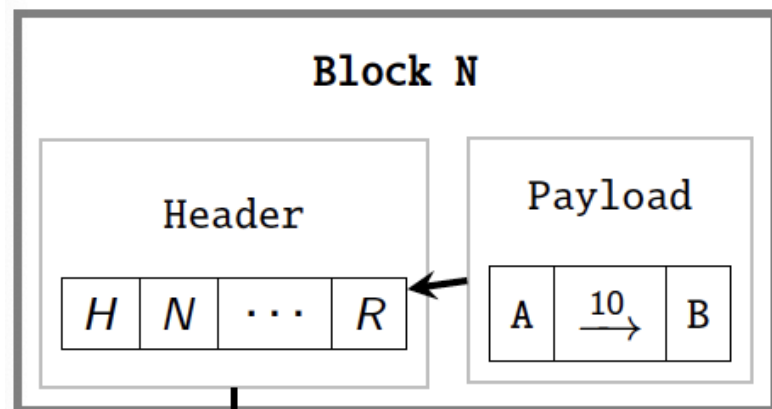
Let's increase the difficulty



- Bob decides to make it harder for Alice to alter her payment by adding a **Nonce**



Mining for a valid hash



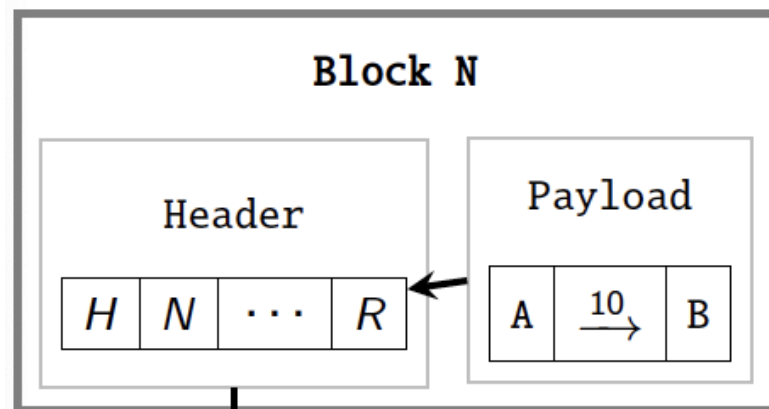
$N = 0 \implies Hash(H \parallel N \parallel \dots \parallel R) = 0x349c1a7e\dots \quad \times$

$N = 1 \implies Hash(H \parallel N \parallel \dots \parallel R) = 0x6ffde7bf\dots \quad \times$

.....

$N = x \implies Hash(H \parallel N \parallel \dots \parallel R) = 0x00.k.004f7fed1a$

Mining for a valid hash



$$N = 0 \implies \text{Hash}(H \parallel N \parallel \dots \parallel R) = 0x349c1a7e\dots \quad \times$$

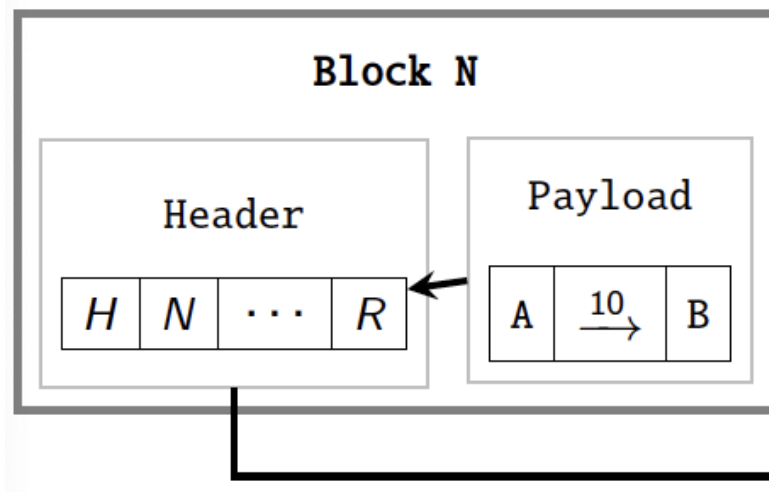
$$N = 1 \implies \text{Hash}(H \parallel N \parallel \dots \parallel R) = 0x6ffde7bf\dots \quad \times$$

.....

$$N = x \implies \text{Hash}(H \parallel N \parallel \dots \parallel R) = 0x00.k.004f7fed1a$$

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

Mining for a valid hash



$$N = 0 \implies \text{Hash}(H \parallel N \parallel \dots \parallel R) = 0x349c1a7e\dots \quad \times$$

$$N = 1 \implies \text{Hash}(H \parallel N \parallel \dots \parallel R) = 0x6ffde7bf\dots \quad \times$$

.....

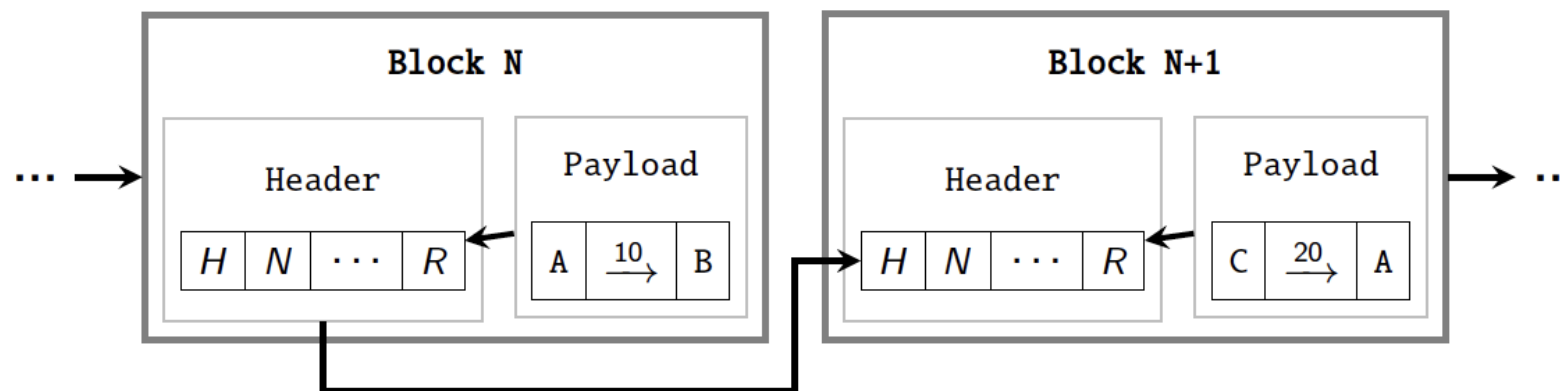
$$N = x \implies \text{Hash}(H \parallel N \parallel \dots \parallel R) = 0x00.k.004f7fed1a$$

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

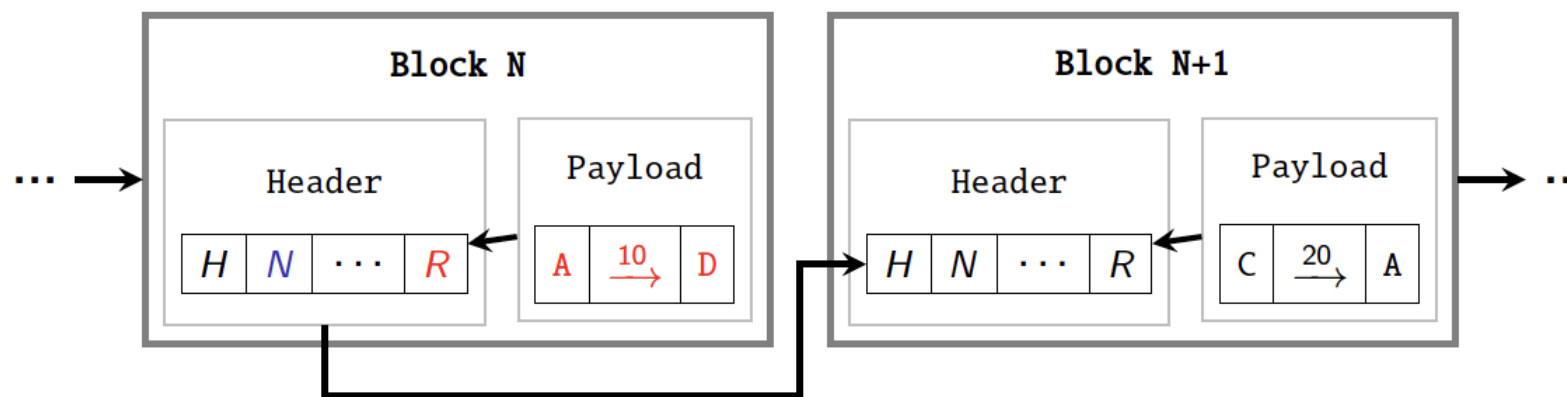
A: $\frac{2^{m-k}}{2^m}$, a larger $k \rightarrow$ a **higher difficulty** of finding N

Expect 2^k hash operations to find a valid N

How does mining deter alteration? – Case 1

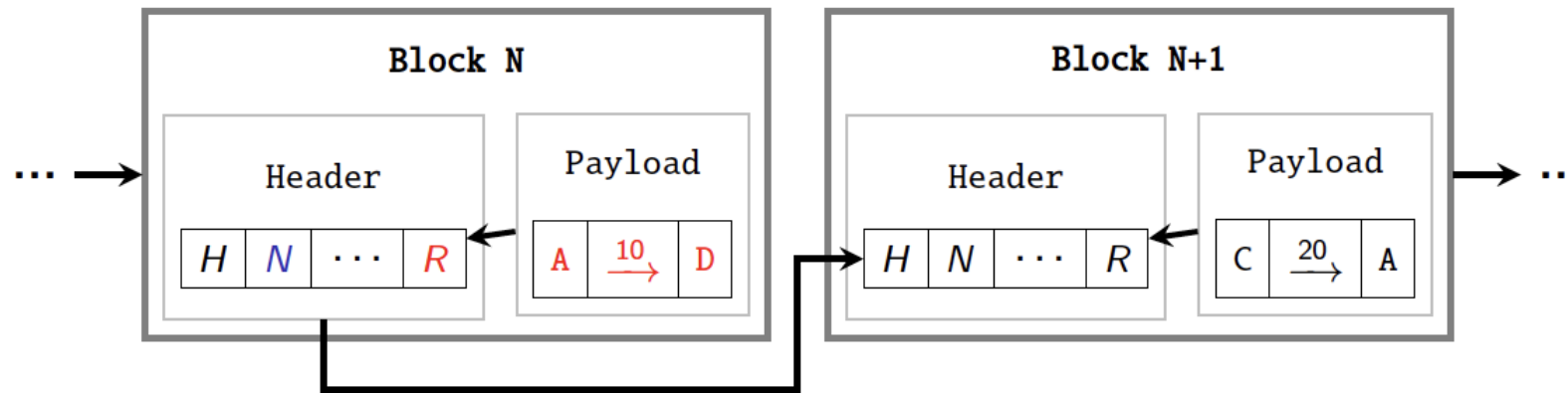


- **Surgical change:** Alice re-mines block N and finds a new **Nonce** such that the block header hash remains unchanged



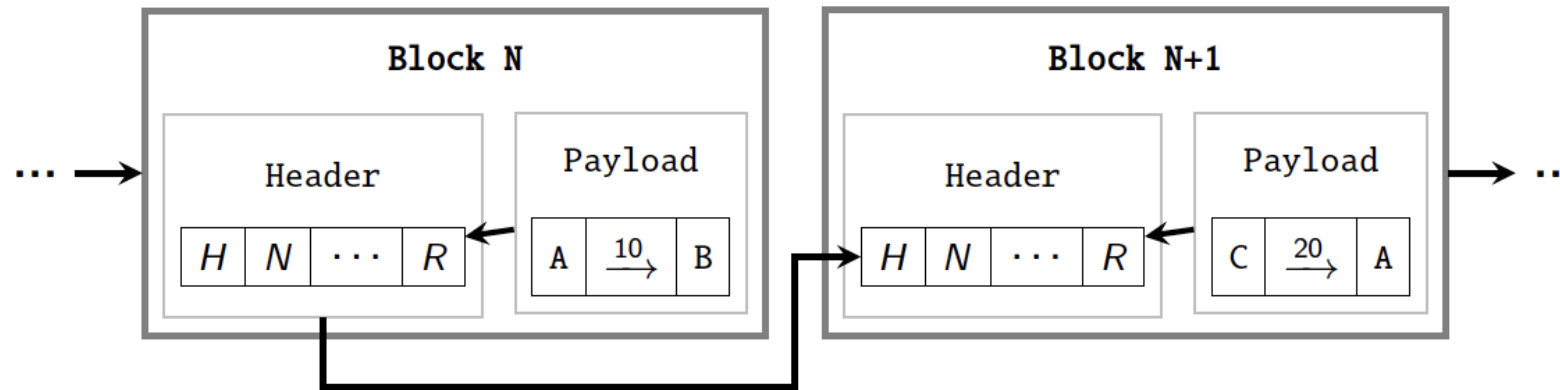
How does mining deter alteration? – Case 1

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

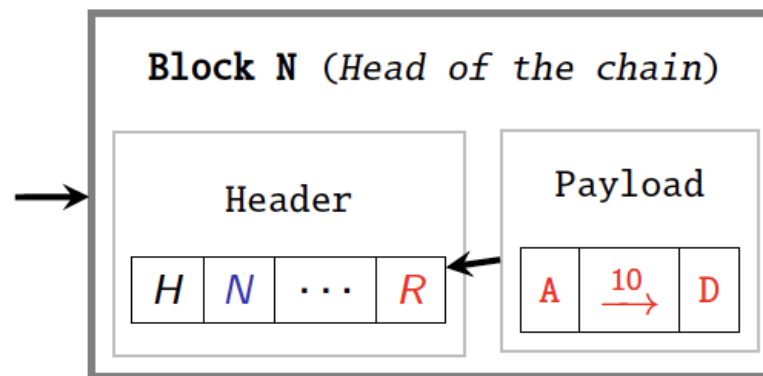


- **Deterrent:** This is extremely hard for a cryptographic hash function that has **preimage resistance** and **second-preimage resistance**.

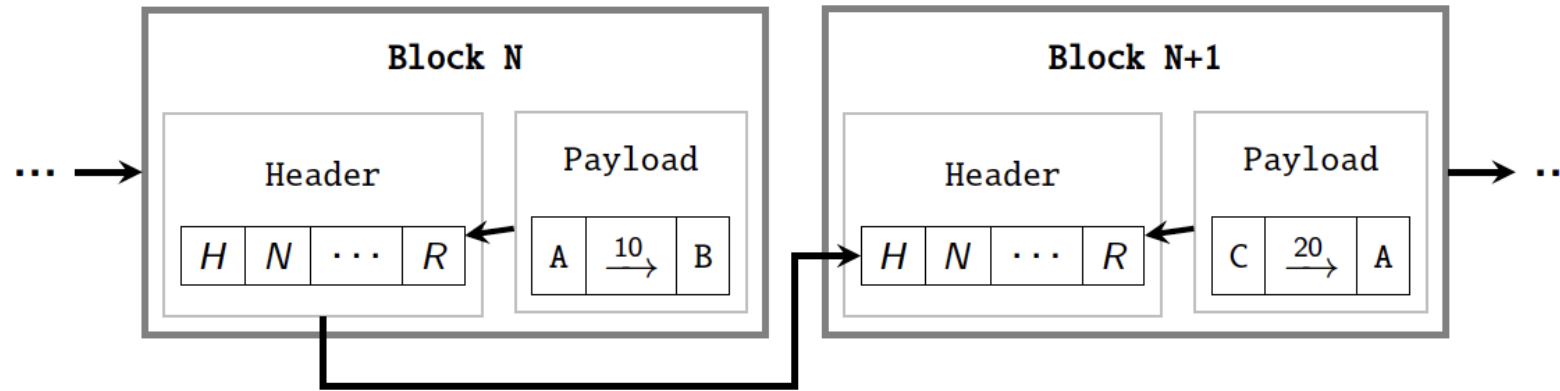
How does mining deter alteration? – Case 2



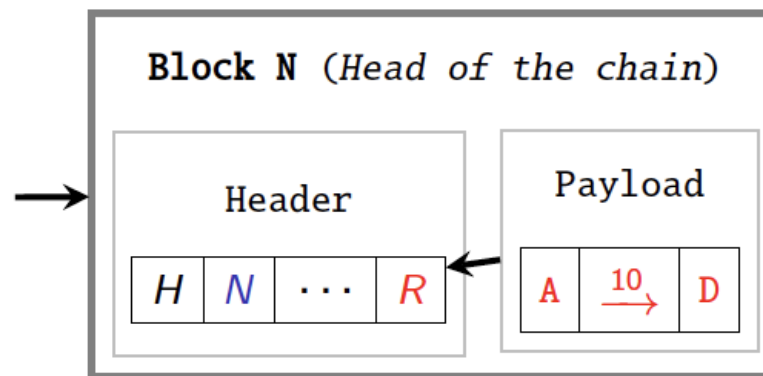
- **Change-and-cut:** Alice re-mines the **Nonce** for block *N* and stops



How does mining deter alteration? – Case 2

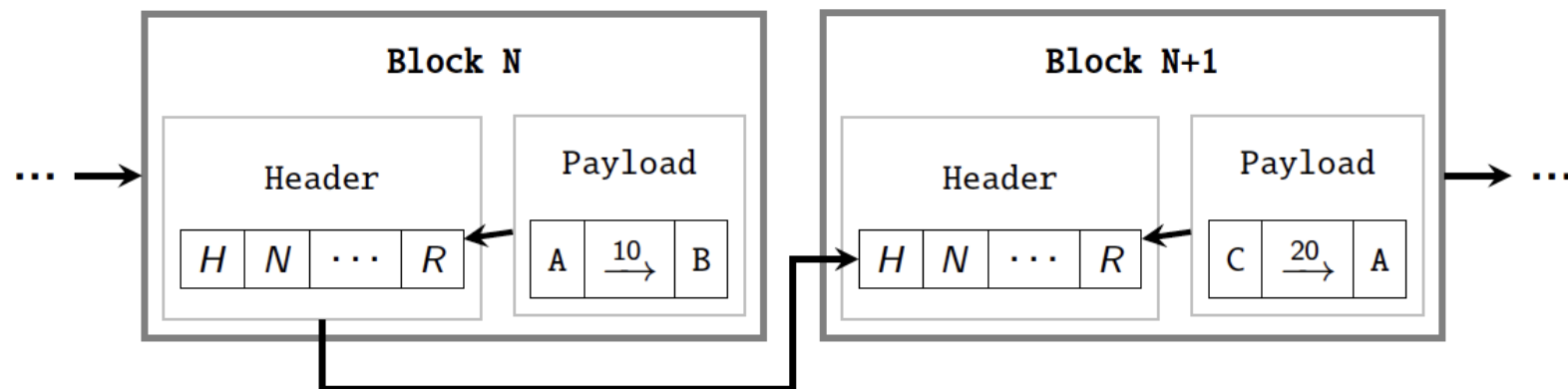


- **Change-and-cut:** Alice re-mines the **Nonce** for block N and stops

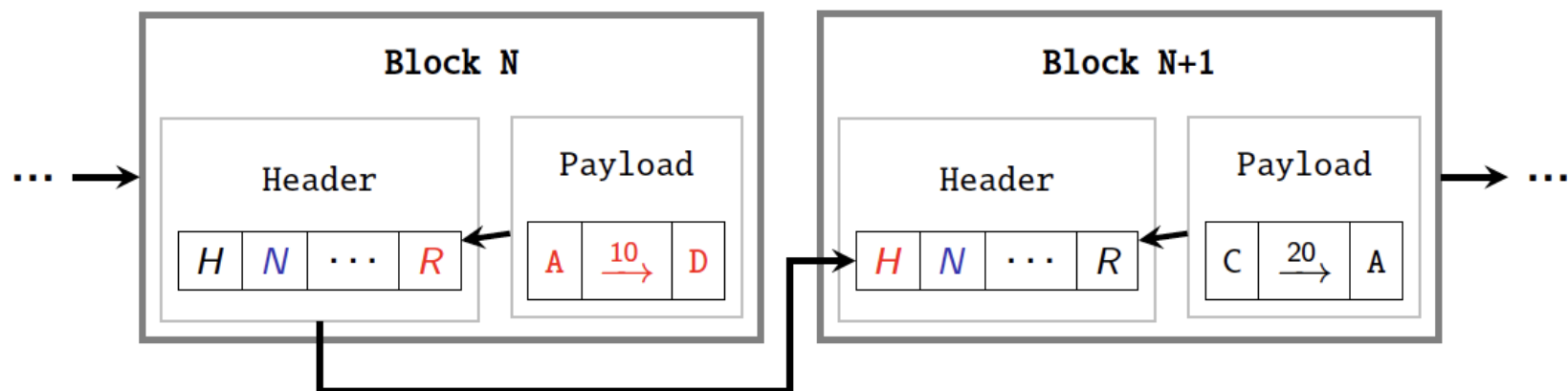


- **Deterrent:** Longer chains are preferred over shorter chains.

How does mining deter alteration? – Case 3

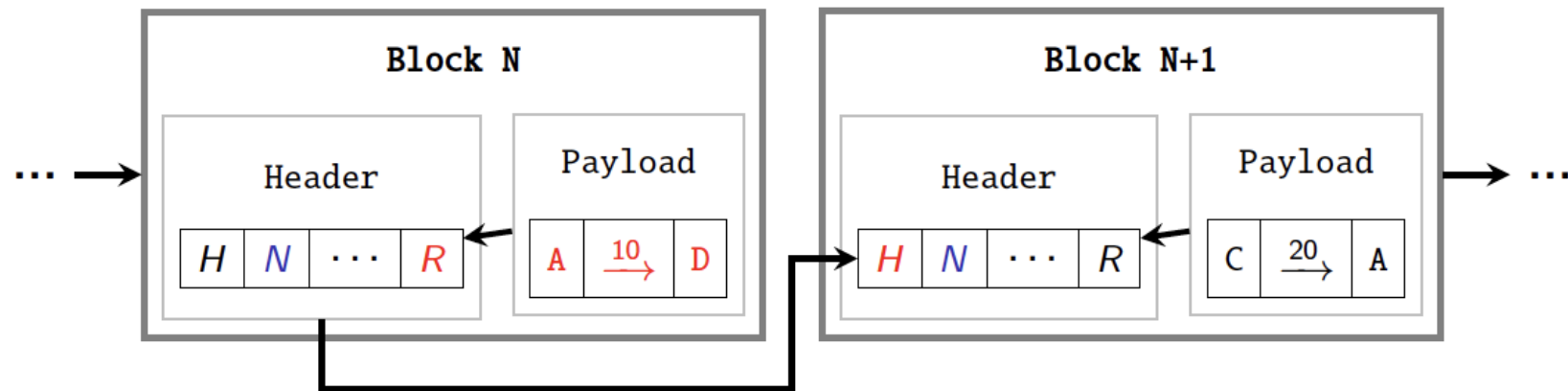


- **Partial chain re-mining:** Alice re-mines **all** the **Nonces** since block *N*



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 block N (included) 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

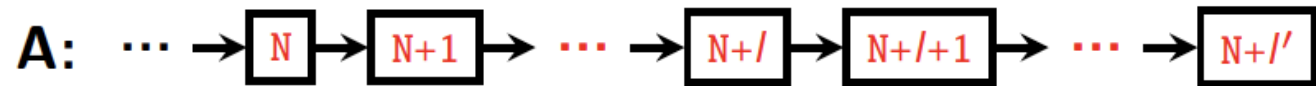
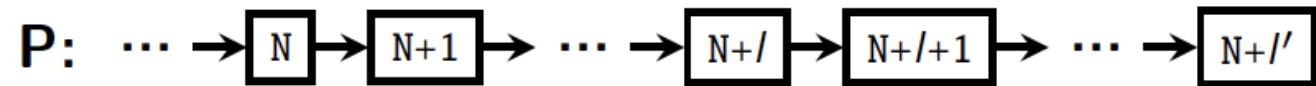
- There is a catch in the deterrent:
 - Alice needs to mine slower than the rest of the participants combined

P: ... → N → N+1 → ... → N+I

A: ... → N → N+1 → ... → N+I

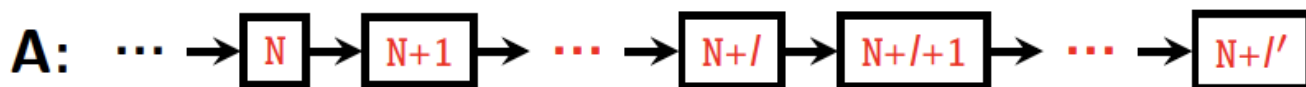
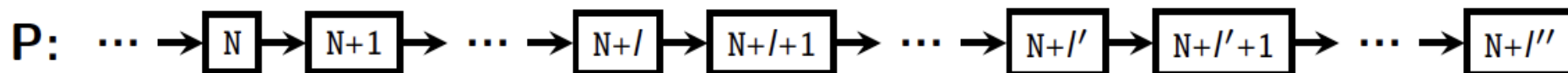
The 51% attack

- 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

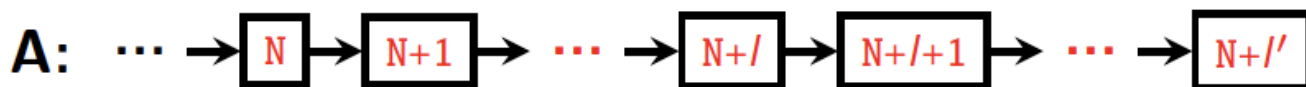
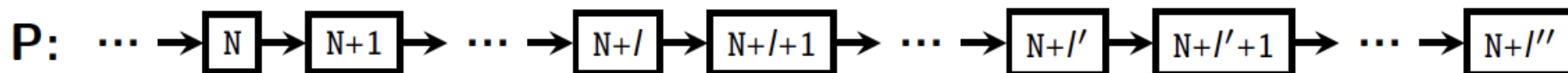


To avoid detection and to maintain the integrity of her attack strategy

→ 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

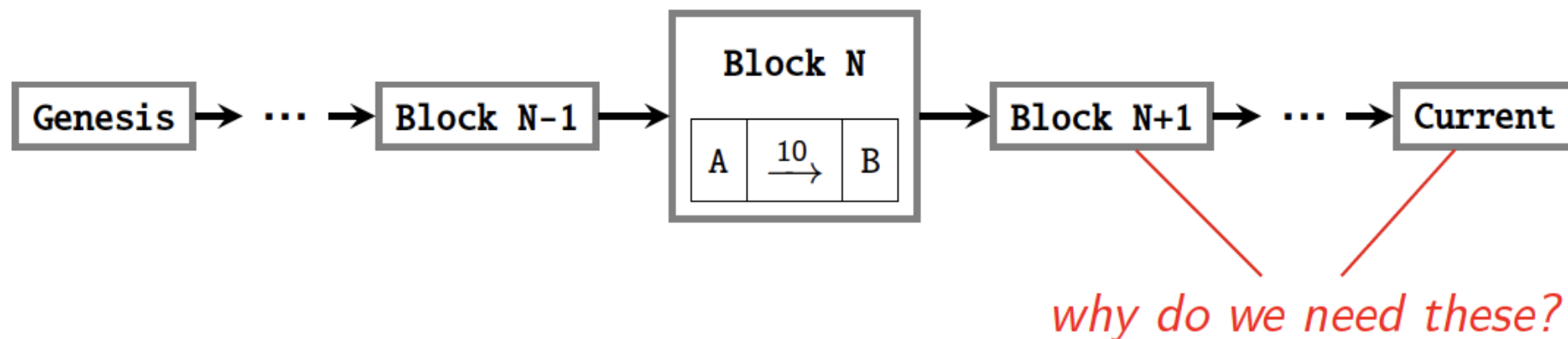


To avoid detection and to maintain the integrity of her attack strategy

- The public chain needs to **grow faster** than Alice's chain
- If Alice mines **too quickly**, the network would notice an increase in **hashing power** and adjust the difficulty upward → harder for her to continue the attack (**re-write 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.

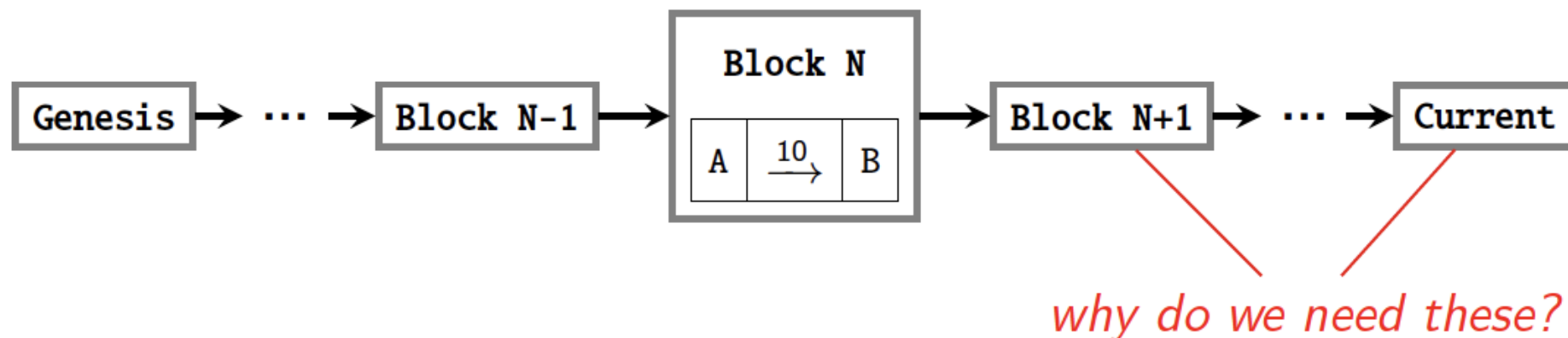


Q: Why do we need these extra blocks, even when:

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

Confirmation level

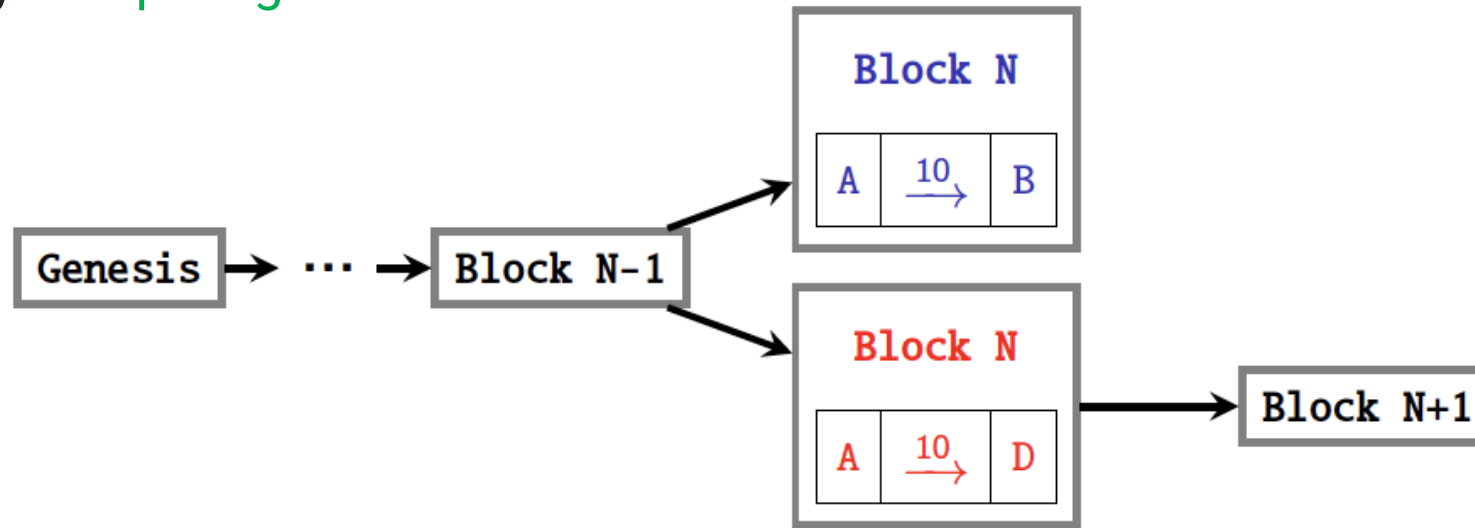
- Recall that when we show a proof of payment, we need a few extra blocks after the block that hosts the ledger entry.



A: Extra blocks act as a safety net, enhancing the security, trustworthiness, and overall stability of the blockchain

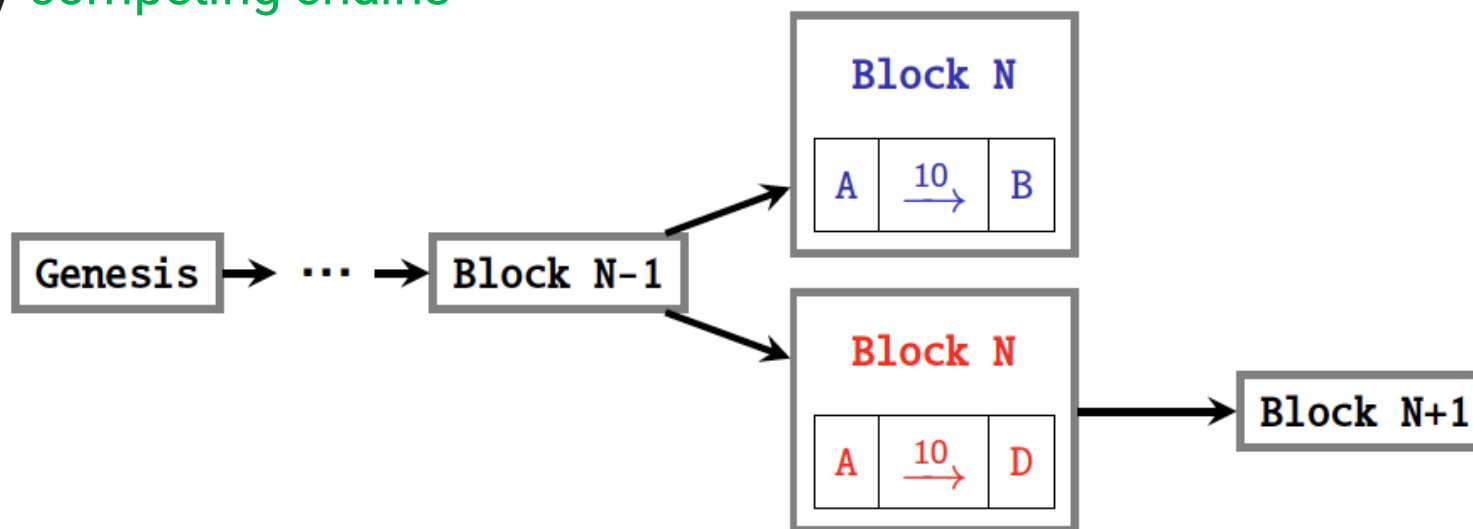
Reducing the risk of Forks

Confirmation blocks help ensure that a single chain becomes **dominant**, reducing the risk of issues caused by **competing chains**



Reducing the risk of Forks

Confirmation blocks help ensure that a single chain becomes **dominant**, reducing the risk of issues caused by **competing chains**



- 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

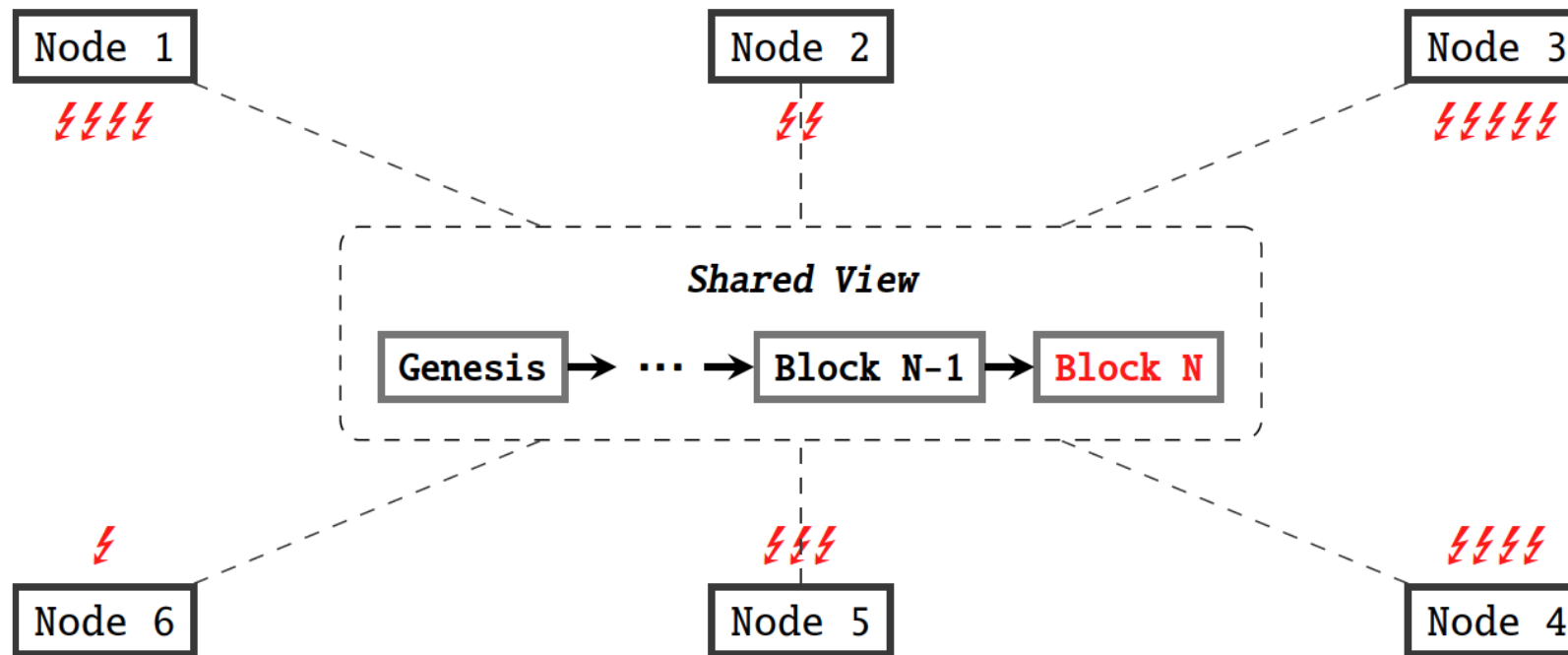
Drawbacks of Proof-of-work consensus

- Speed of confirmation
 - A single Bitcoin block can hold around 1000 to 2000 transactions($\approx 10min$)
 - It's common for users to wait for 6 confirmations to consider a Bitcoin transaction secure
- 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 (<https://dci.mit.edu/51-attacks>)
- Energy consumption (As much as Argentina in 2022)*
 - Hashing itself is not useful (Provide incentive by offering a transaction fee)
 - These operations are repeated across the fleet of nodes

[*https://ccaf.io/cbeci/index](https://ccaf.io/cbeci/index)

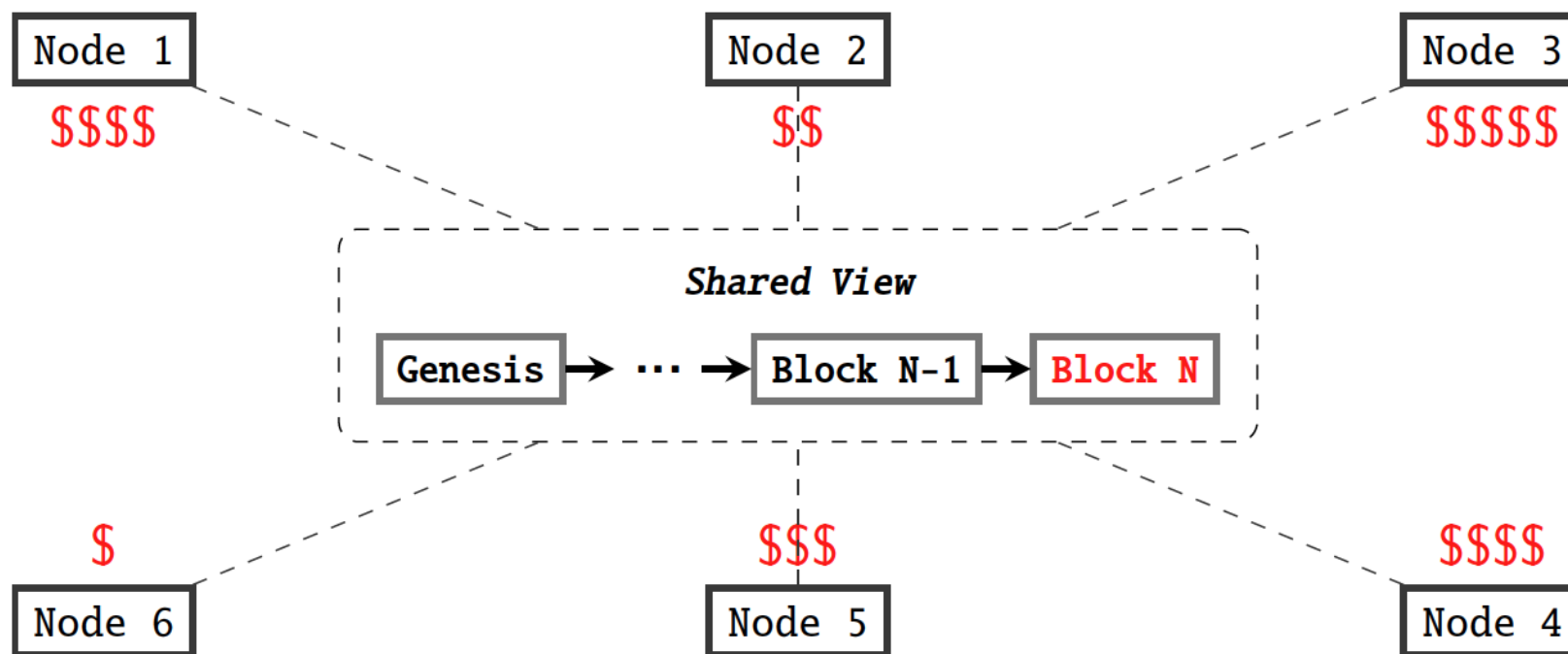
Consensus: Proof-of-stake

Block production as election



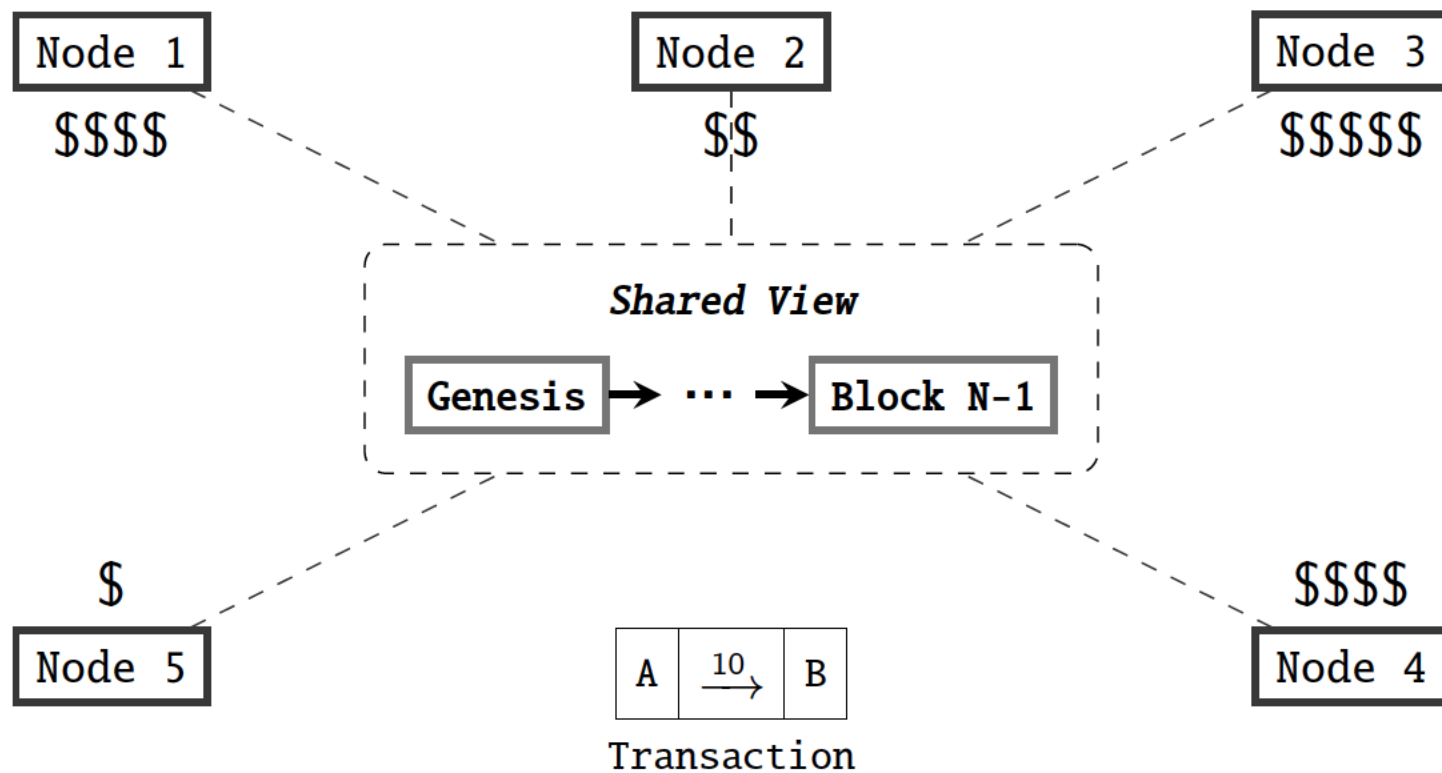
- In a proof-of-work scheme:
 - The chance of which node is elected to propose a new block is proportional to its **hashing power**
 - **Collisions** are allowed and are resolved by the longest chain rule

Block production as election

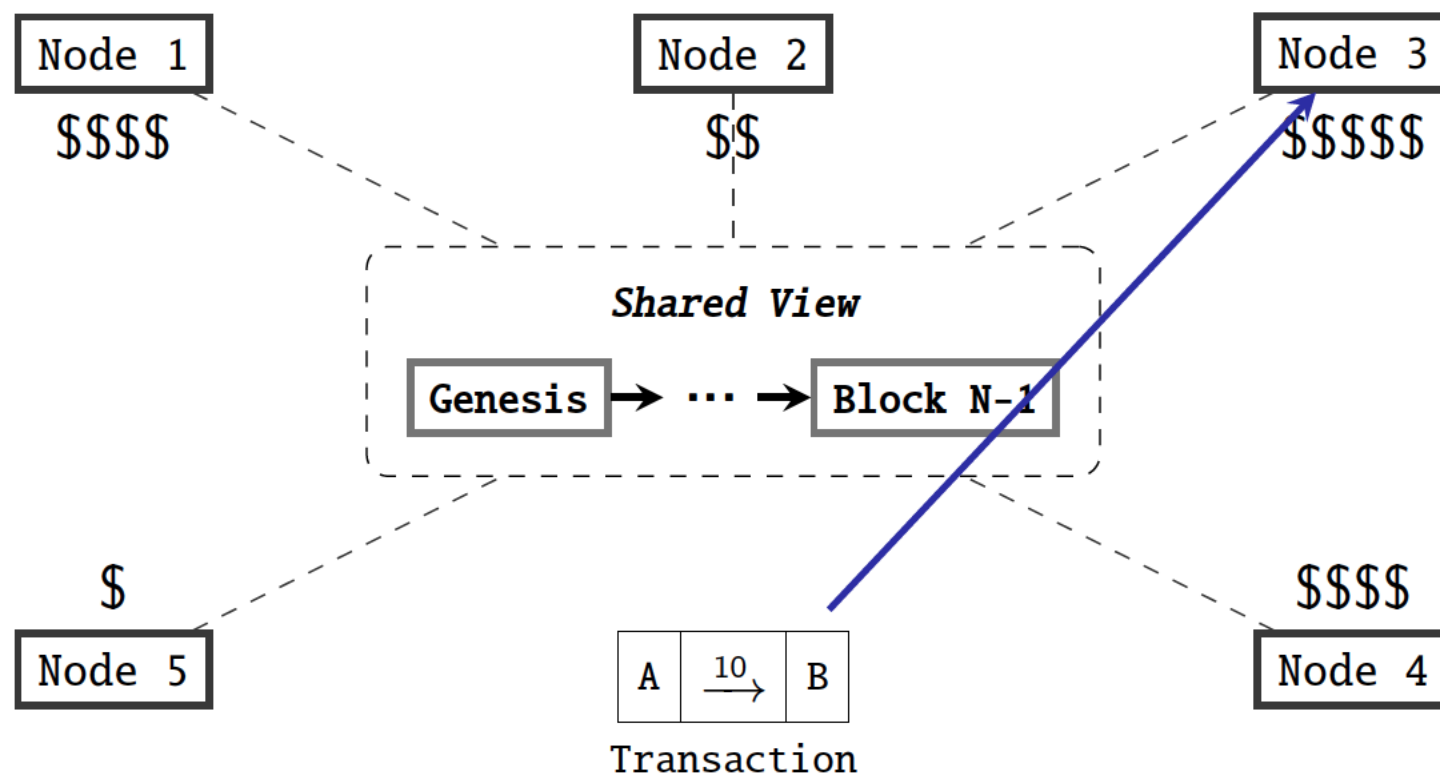


- In a **proof-of-stake** scheme:
 - The chance of which node is elected to propose a new block is proportional to its **staked value**
 - **Collisions** are not allowed by design, only the leader creates a block

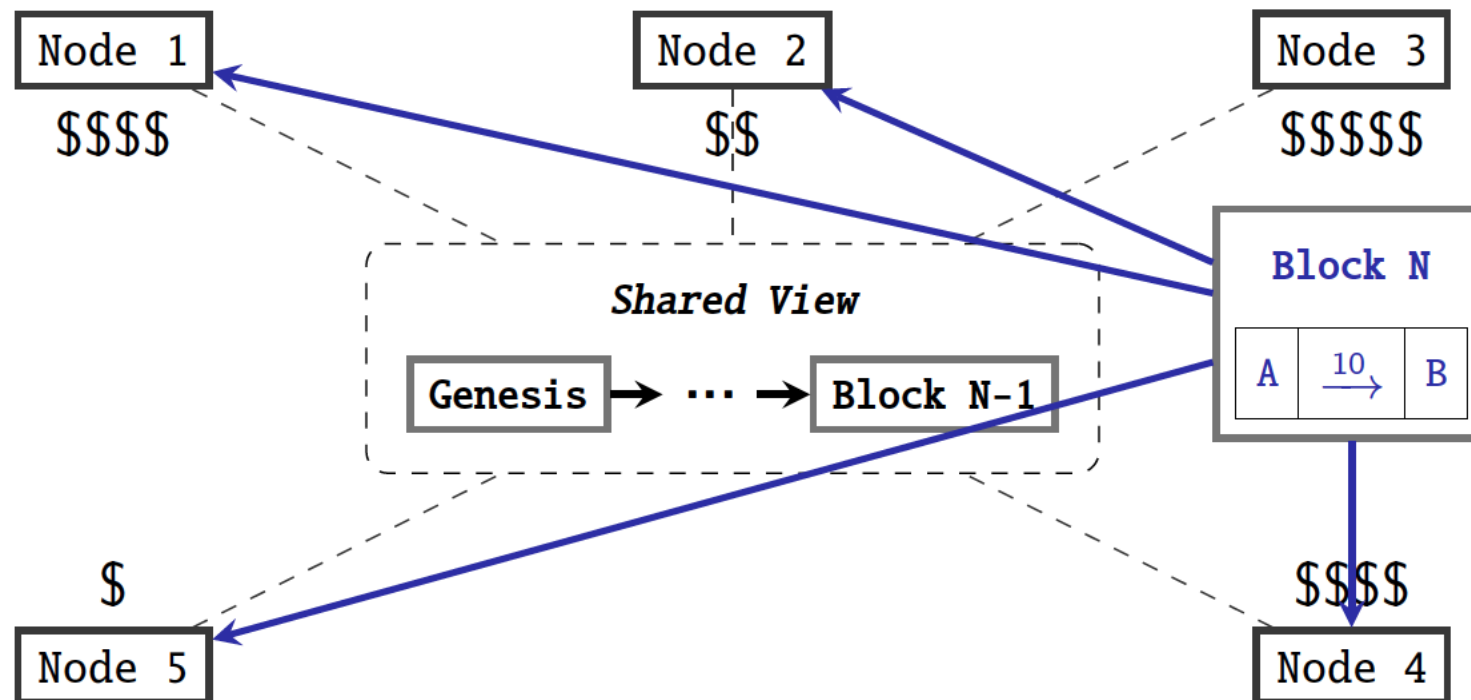
Transaction lifecycle in PoS



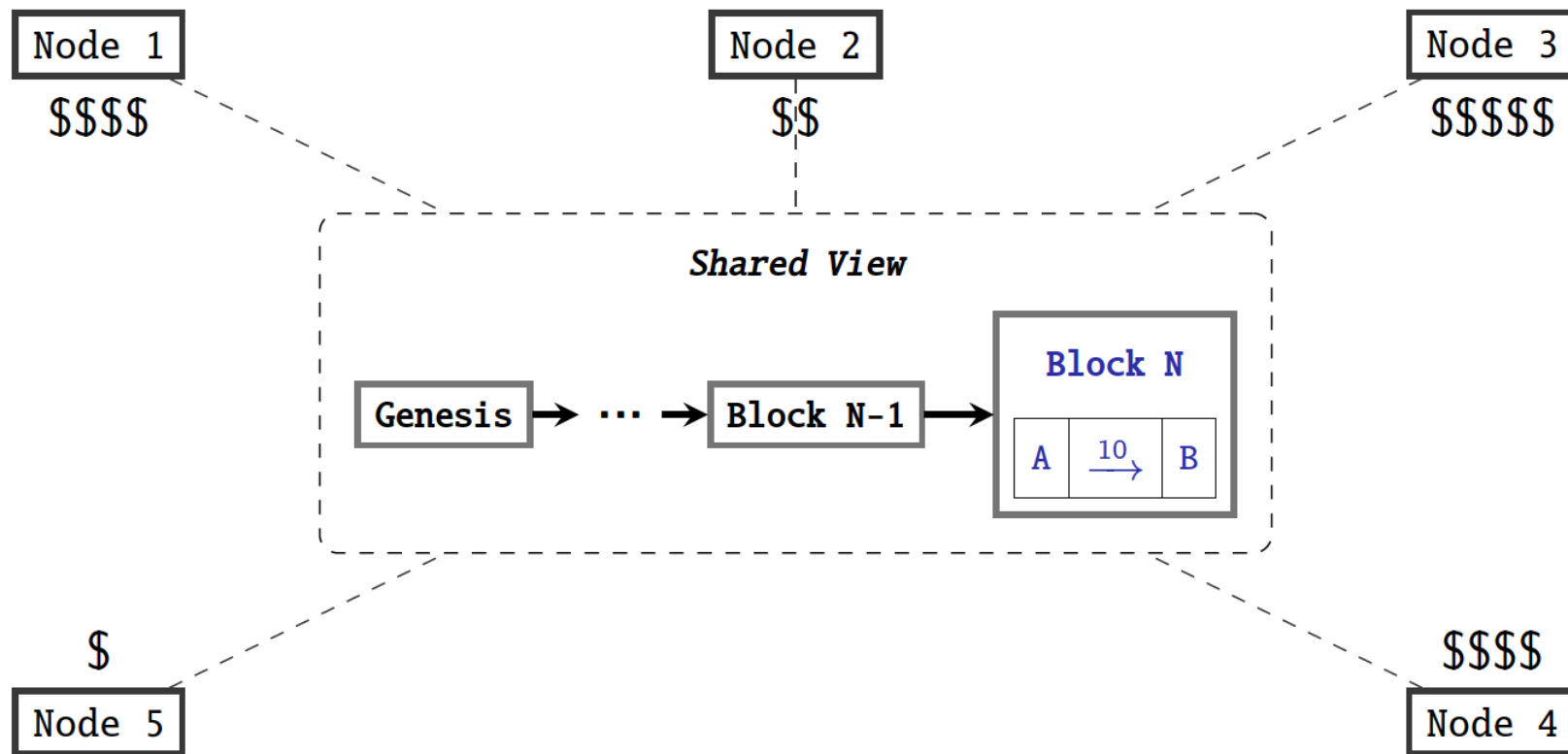
Transaction lifecycle in PoS



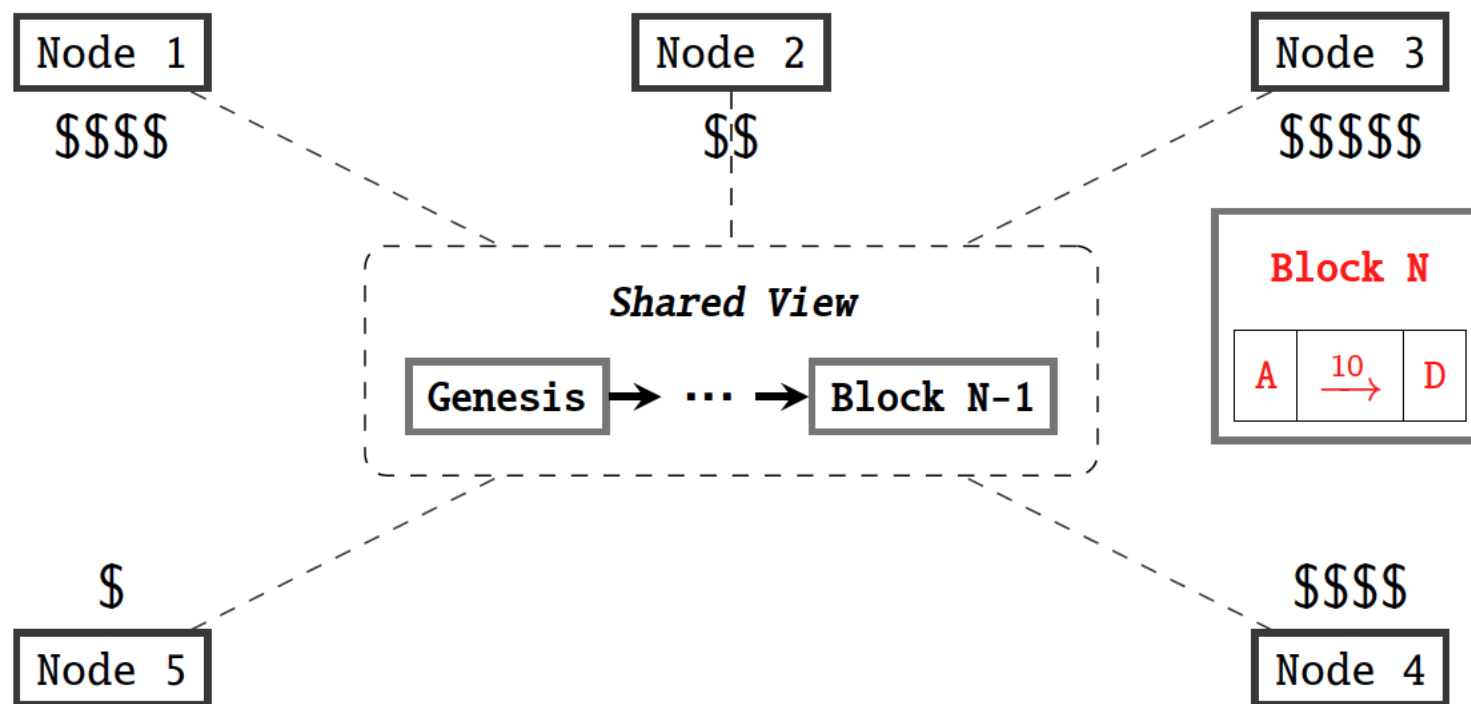
Transaction lifecycle in PoS



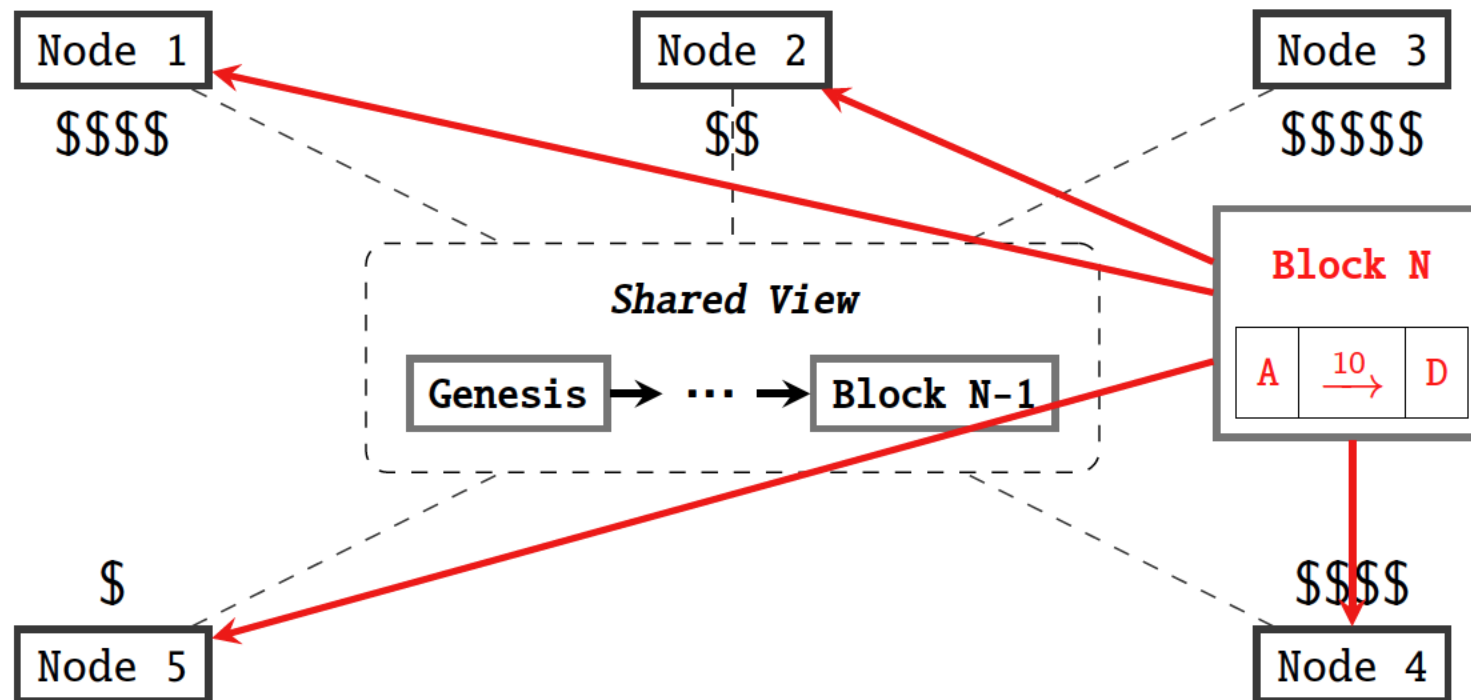
Transaction lifecycle in PoS



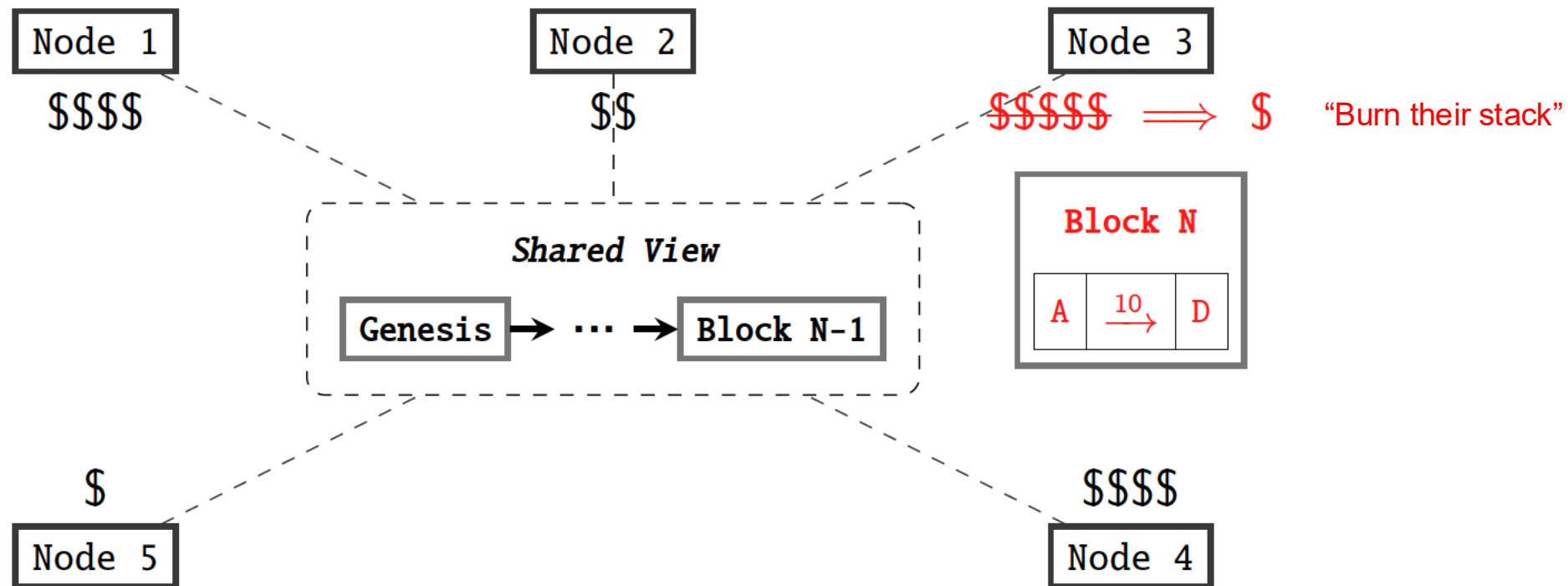
Transaction lifecycle in PoS



Transaction lifecycle in PoS



Transaction lifecycle in PoS



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

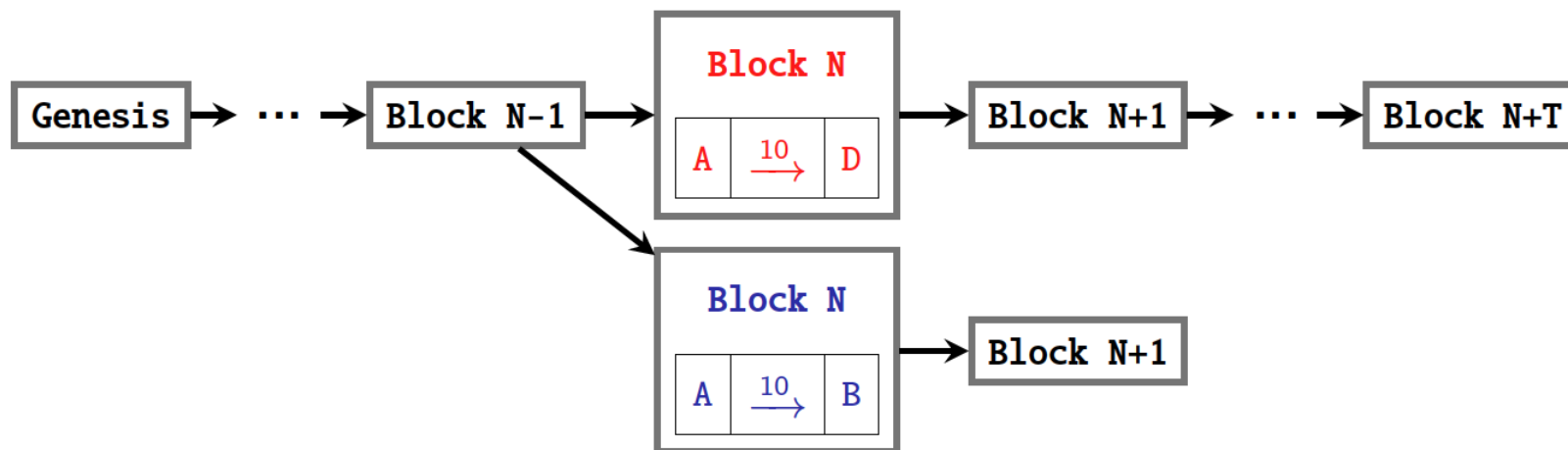
The 51% attack on PoS

- **Q:** What if the attacker controls $\geq 50\%$ 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!**.

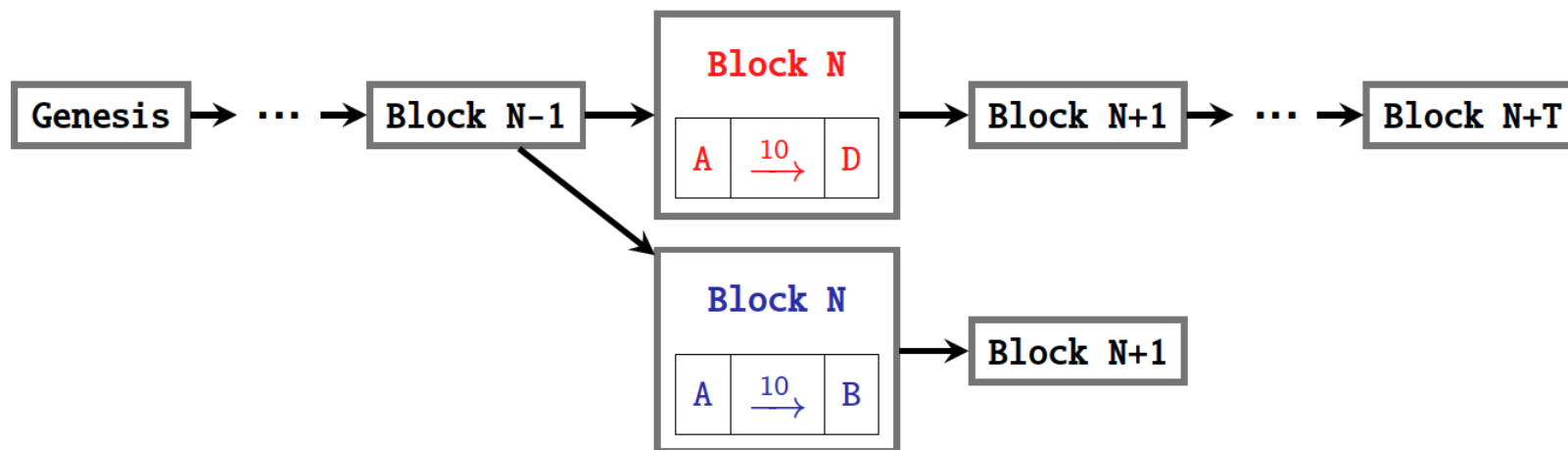
Hard fork as a recovery of a 51% attack

- 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.



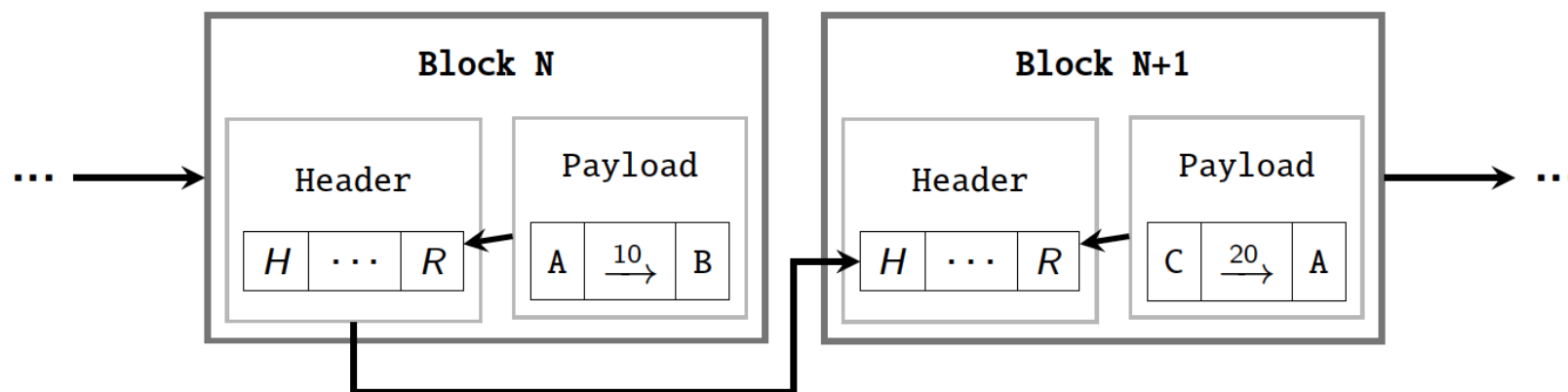
- NOTE: The forked chain can be shorter than the previous chain!
 - A higher level of social coordination is required

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 $\geq 50\%$ of the staked resources.
- In **PoW**, the hard fork can only invalidate transaction **WHILE** the attacker is still in control of $\leq 50\%$ of the computational power.

Chain validation

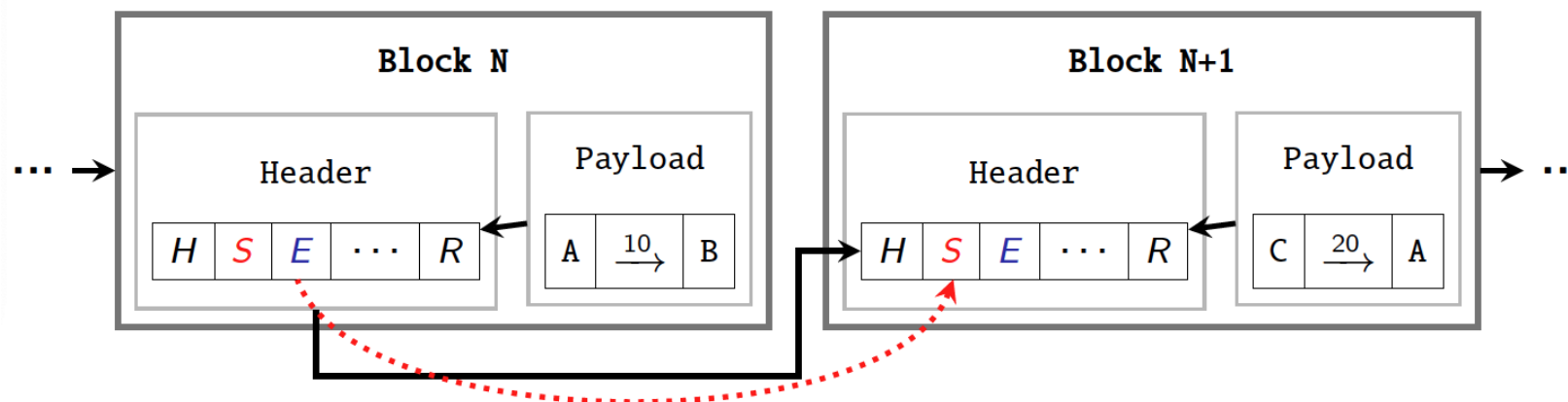
- If Alice shows Bob, the Pizzeria owner, the following blockchain, why would Bob accept it? Why would Bob believe that:
 - It is **hard** for Alice to produce such a chain of blocks
 - There does not exist a **better** chain of blocks as of now



- With PoS, forging a blockchain would be **easy**! (Bcs. Validators chosen based on their stack)

Chain validation

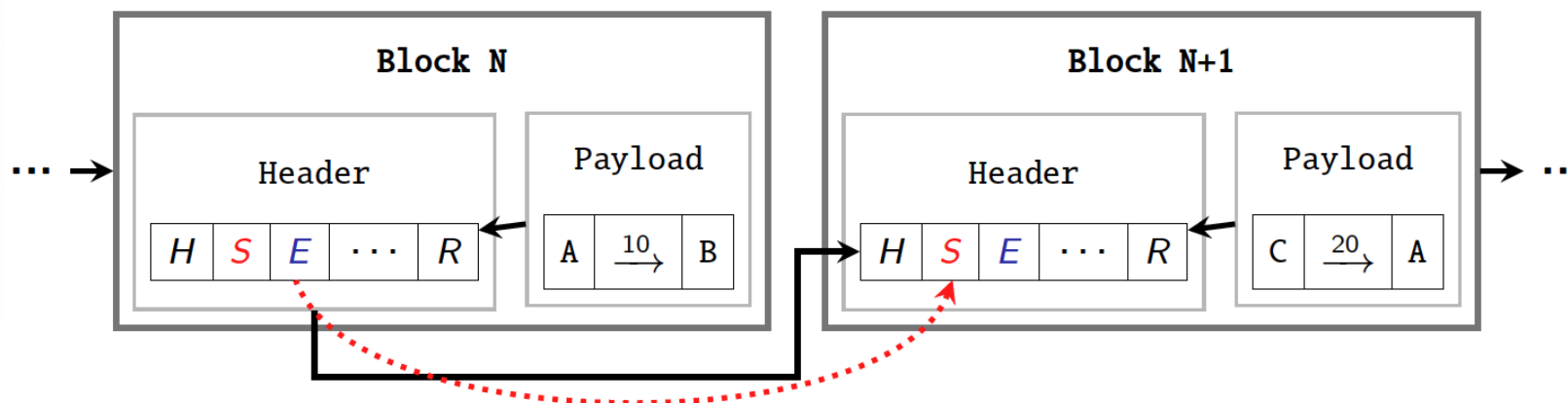
- This turns out to be an extremely complicated problem!



- **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!

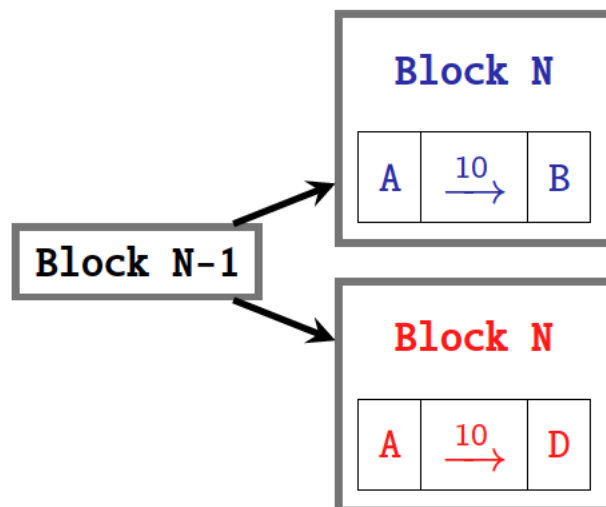


- **S** - Signature of the proposer of this block
- **E** - Election packet that records how this proposer is elected

Q: What are the issues with this scheme?

The Nothing-at-Stake problem

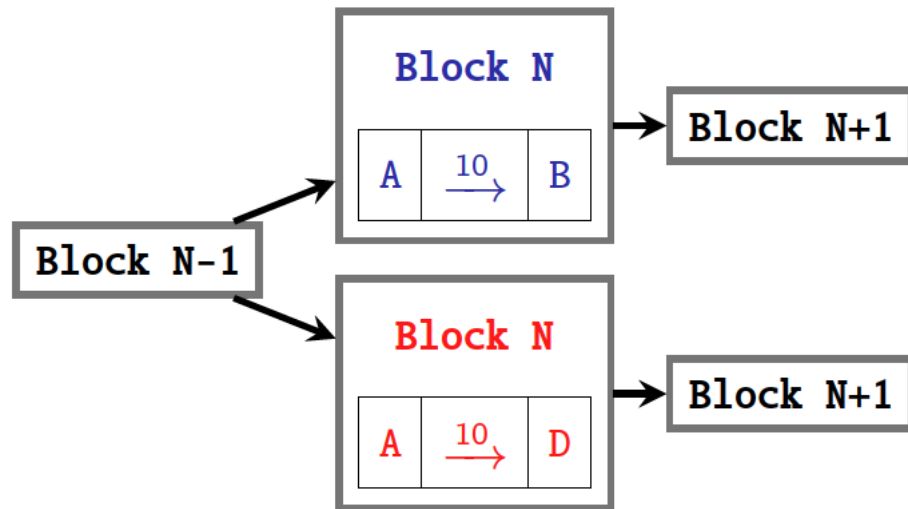
- Alice has some small stake (e.g., 1%) and can be elected as a block proposer:



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

The Nothing-at-Stake problem

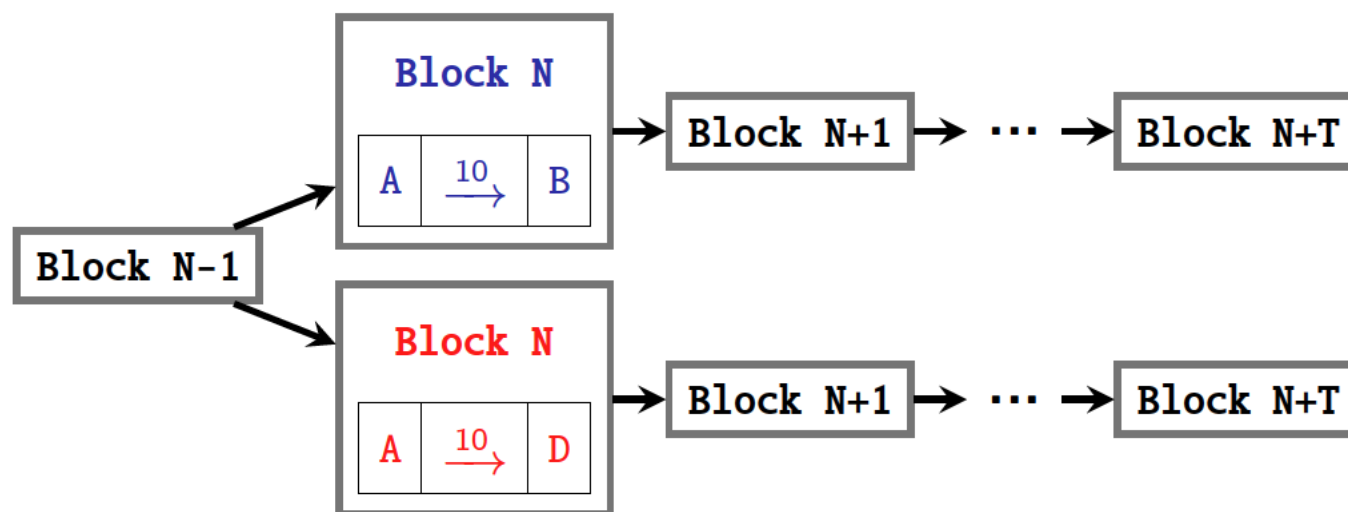
- Alice has some small stake (e.g., 1%) and can be elected as a block proposer:



- 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

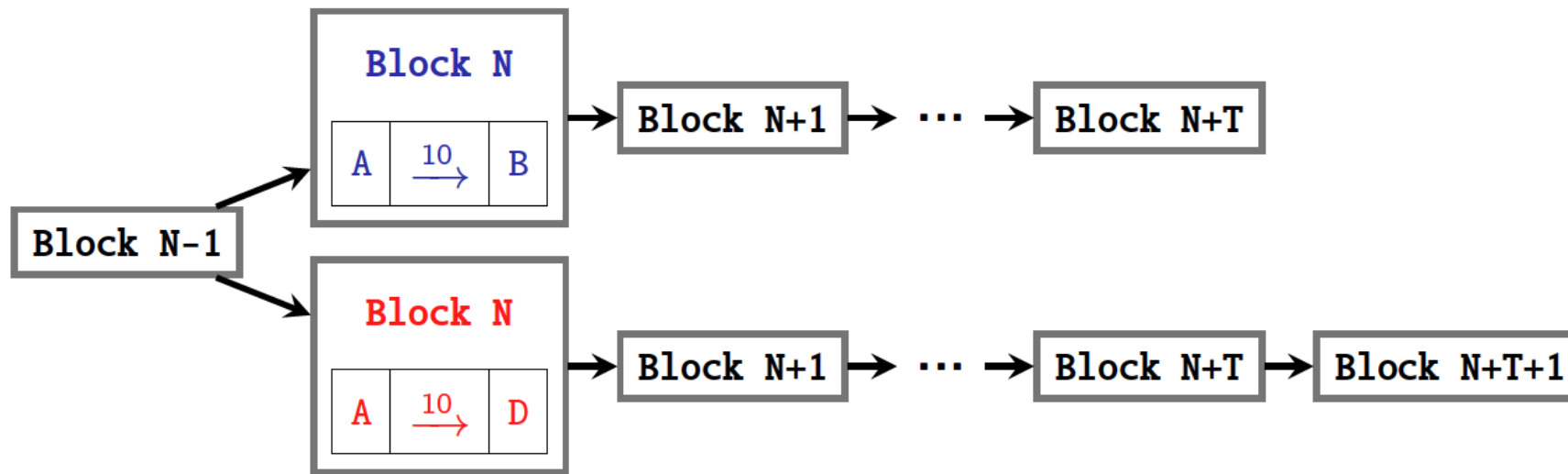
- Alice has some small stake (e.g., 1%) and can be elected as a block proposer:



- 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:



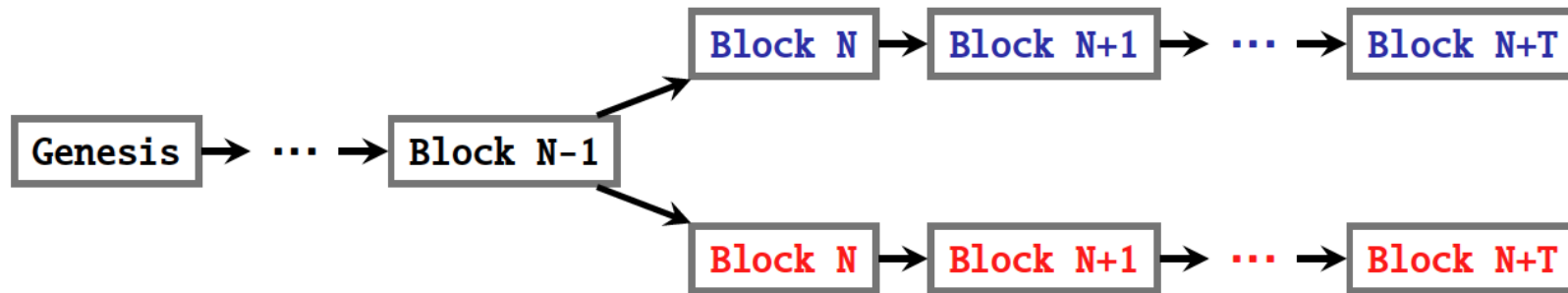
- When it's Alice's turn again, she only appends a block to the chain that is more favorable to her. The other chain **dies as a result**.
- This is sometimes called the **1% attack**.

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.

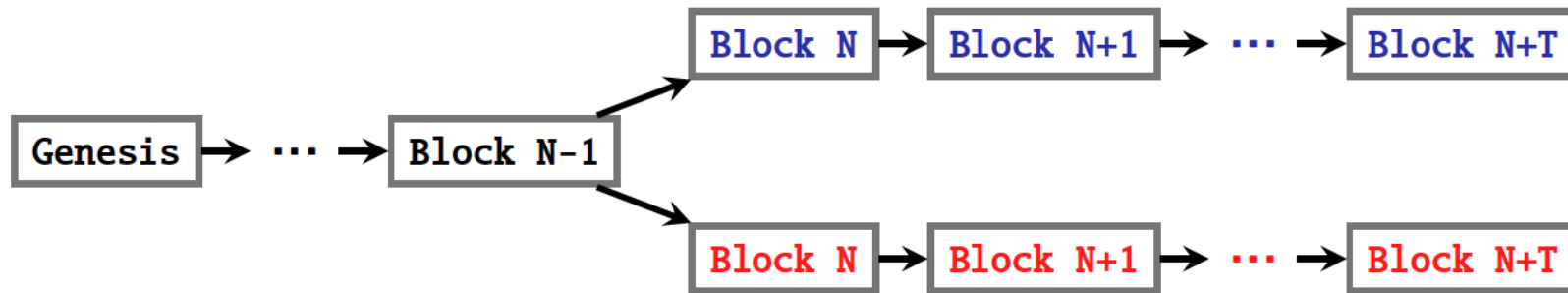
Long-range attacks (The bootstrapping problem)

- 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 bootstrapping problem)

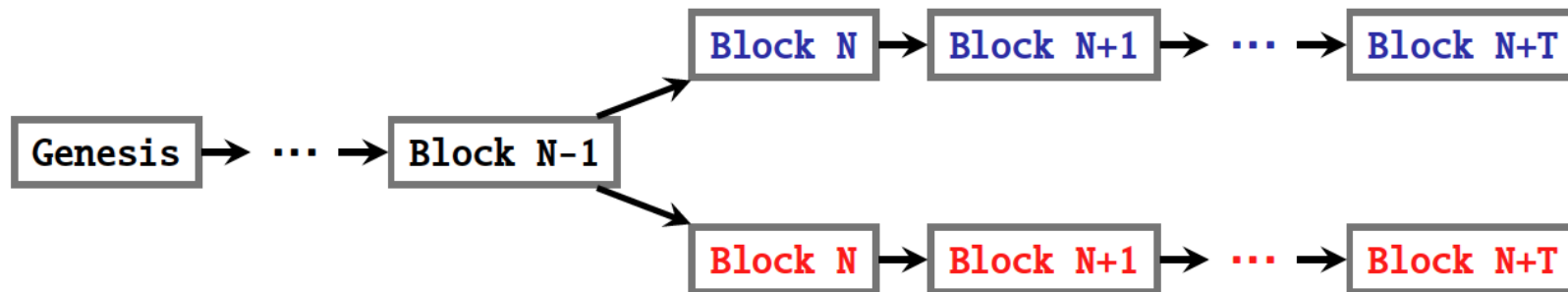
- A validator node could forge an entire chain by itself
- If Bob, a new user, joins the network, which chain should he accept?



Q: Why is this not a problem in PoW?

Long-range attacks (The bootstrapping 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 bootstrapping 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**.