1b: Bitcoin & blockchain

* Many slides from Joseph Bonneau@NYU

outline

1. Cryptographic hash function
2. Hash pointer, Blockchain, Merkle tree
3. Digital signatures
4. Simple cryptocurrencies
5. Transaction semantics
6. A decentralized ledger: bitcoin
7. Bitcoin P2P networking
8. Finding a valid block
9. The rules of Bitcoin

5. Transaction semantics

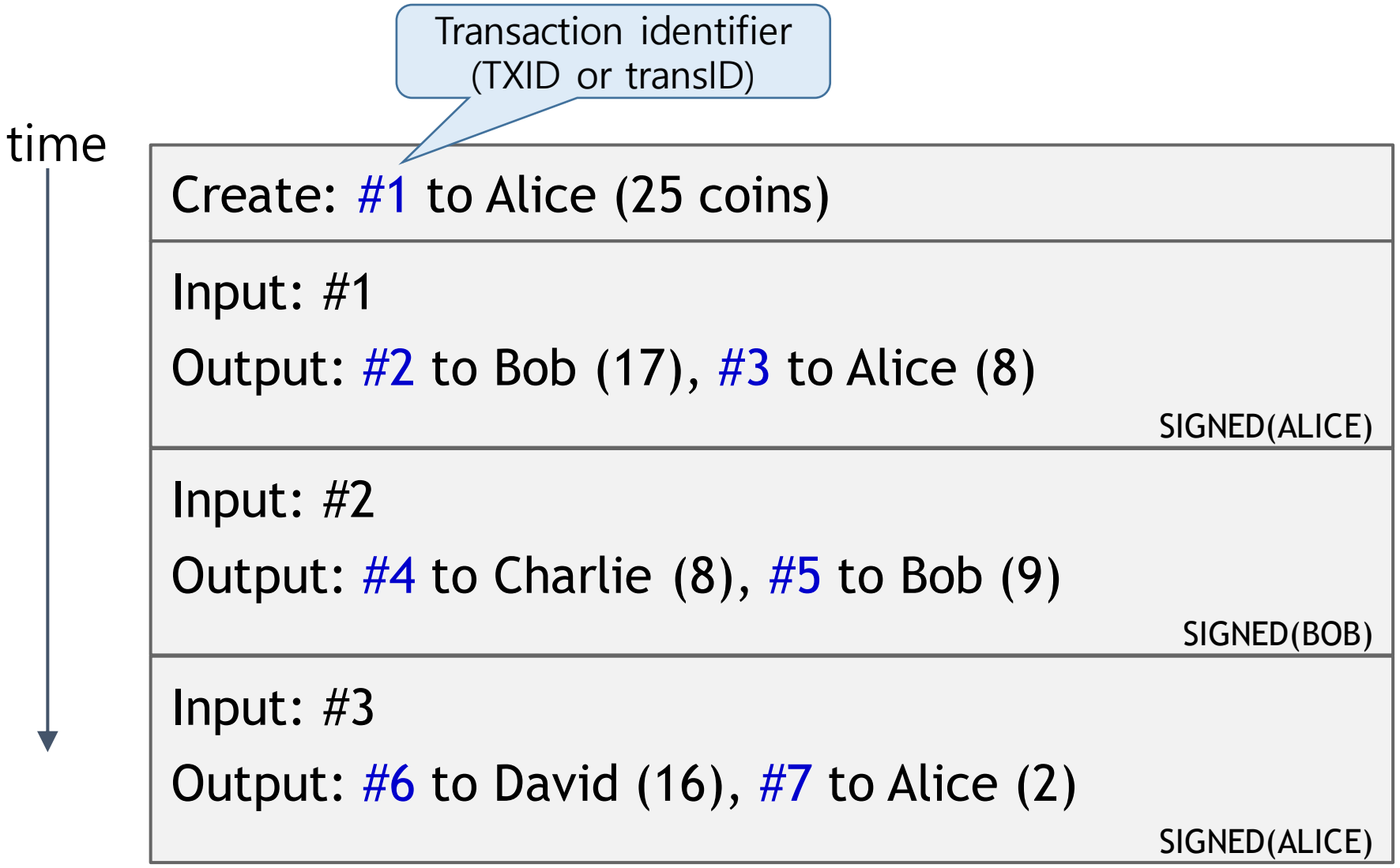
Bitcoin is *transaction-based*

There are **no "accounts"**

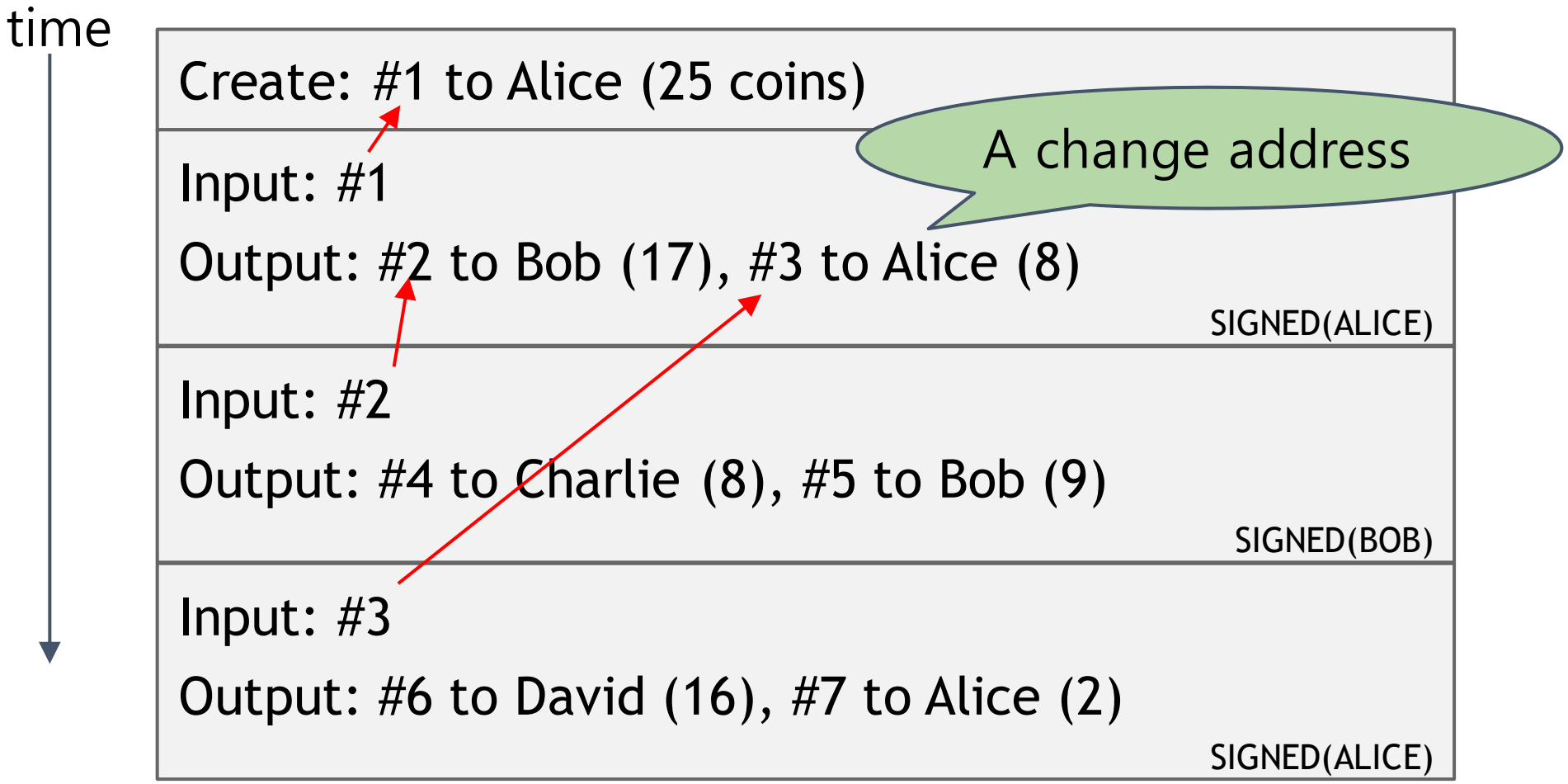
Transactions (TXs) destroy old "coins", create new ones

- Sum of new coins ≤ sum of old coins

A transaction-based ledger (Bitcoin)

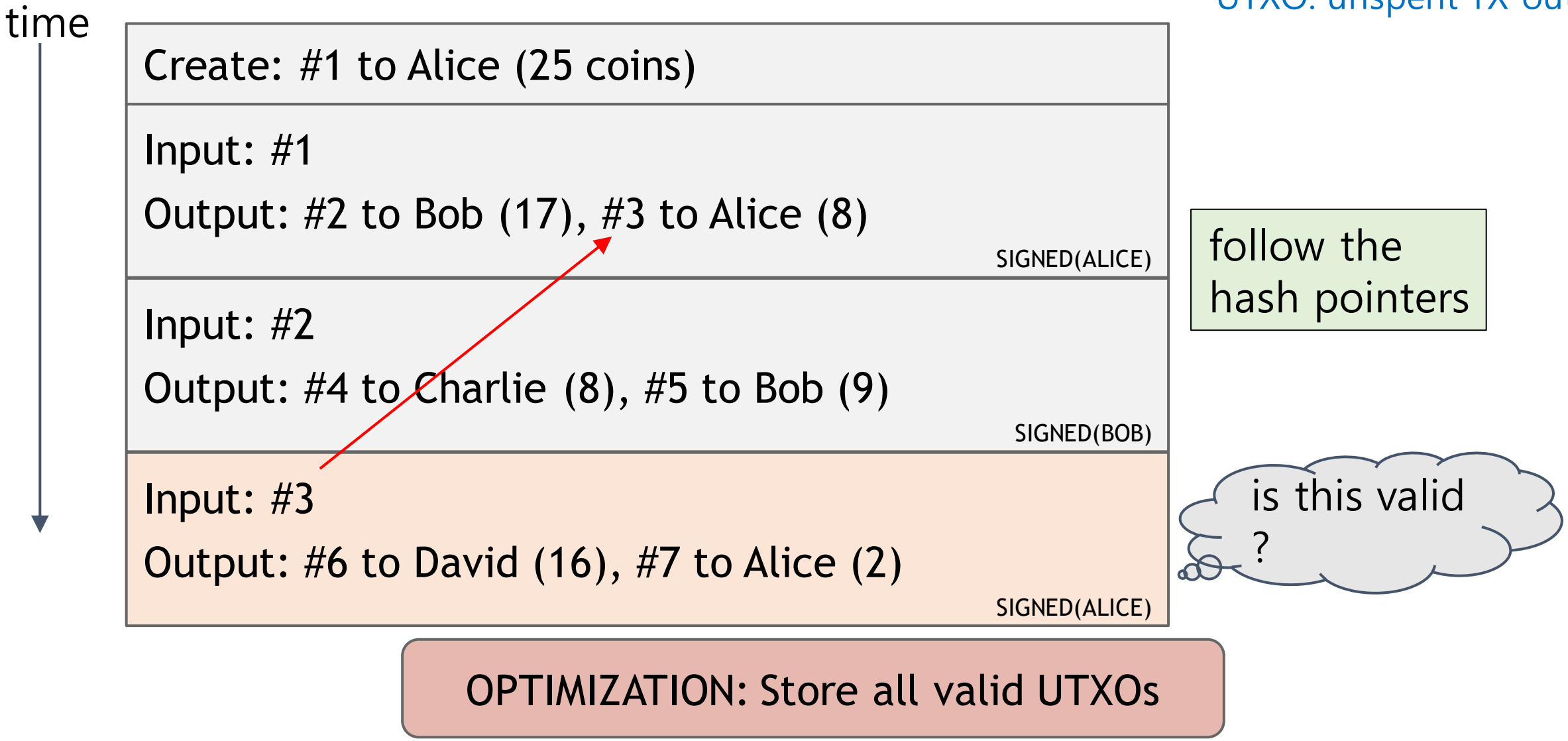


A transaction-based ledger (Bitcoin)

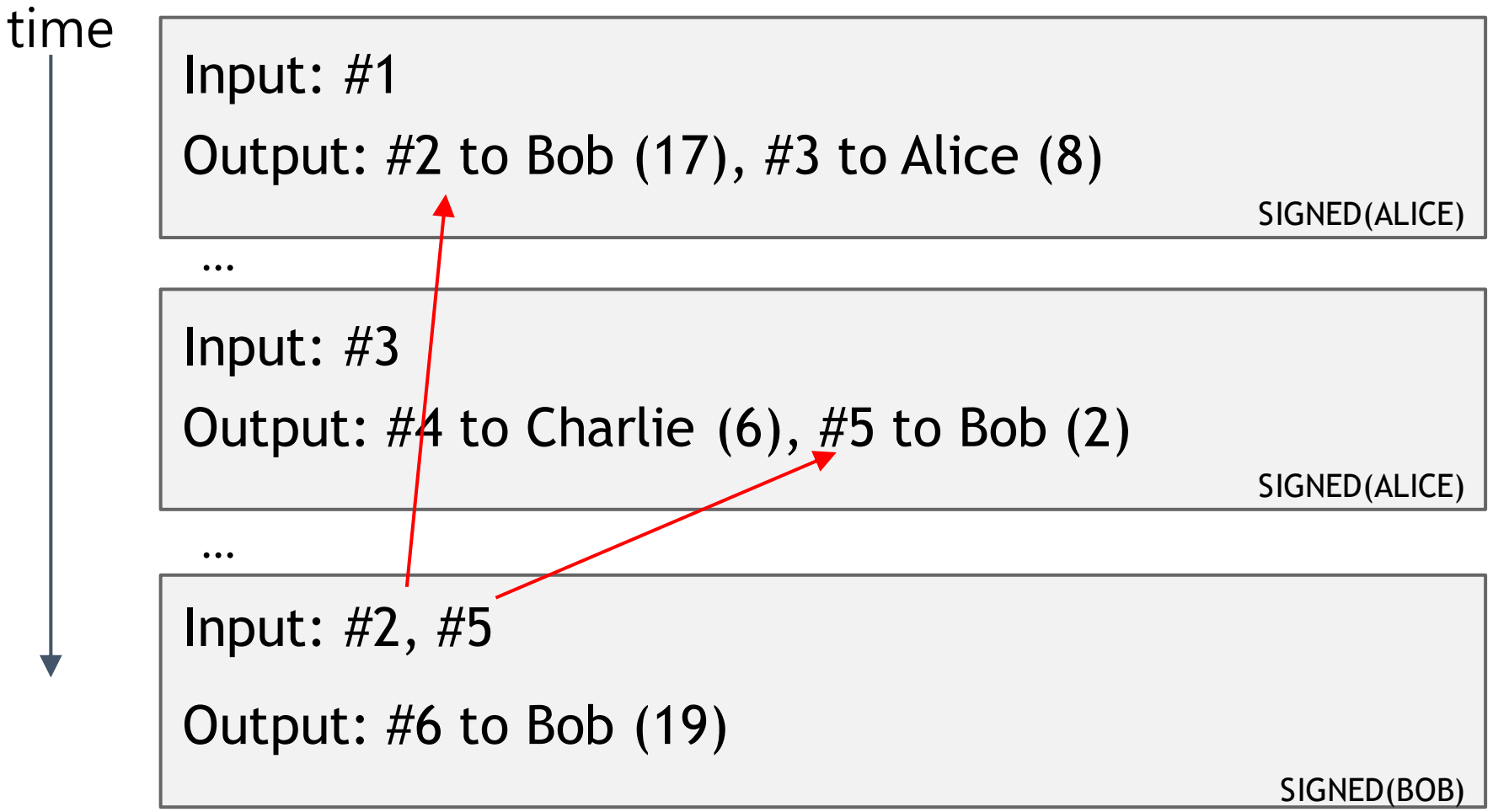


A transaction-based ledger (Bitcoin)

UTXO: unspent TX output



Merging value



A real Bitcoin transaction

metadata

input(s)

output(s)


```
{
  "hash":"5a42590fbe0a90ee8e8747244d6c84f0db1a3a24e8f1b95b10c9e050990b8b6b",
  "ver":1,
  "vin_sz":2,
  "vout_sz":1,
  "lock_time":0,
  "size":404,
  "in":[
    {
      "prev_out":{
        "hash":"3be4ac9728a0823cf5e2deb2e86fc0bd2aa503a91d307b42ba76117d79280260",
        "n":0
      },
      "scriptSig":"30440....3f3a4ce81"
    },
    {
      "prev_out":{
        "hash":"7508e6ab259b4df0fd5147bab0c949d81473db4518f81afc5c3f52f91ff6b34e",
        "n":0
      },
      "scriptSig":"304602210....3f3a4ce81"
    }
  ],
  "out":[
    {
      "value":"10.12287097",
      "scriptPubKey":"OP_DUP OP_HASH160 69e02e18b5705a05dd6b28ed517716c894b3d42e   OP_EQUALVERIFY OP_CHECKSIG"
    }
  ]
}
```


TXID

Transactions: scripting language

- A transaction output doesn't just specify a public key but a script saying "this output can be redeemed by a public key that hashes to X, along with a signature from the owner of that public key."
  - ScriptPubKey: locking script
- The according transaction input then needs to specify a redeem script saying "Here is the public key and the signature you are looking for"
  - ScriptSig: unlocking script
- Both scripts are simply concatenated (the input and the earlier output script) and the resulting script must run successfully in order for the transaction to be valid

Transaction inputs

previous transaction

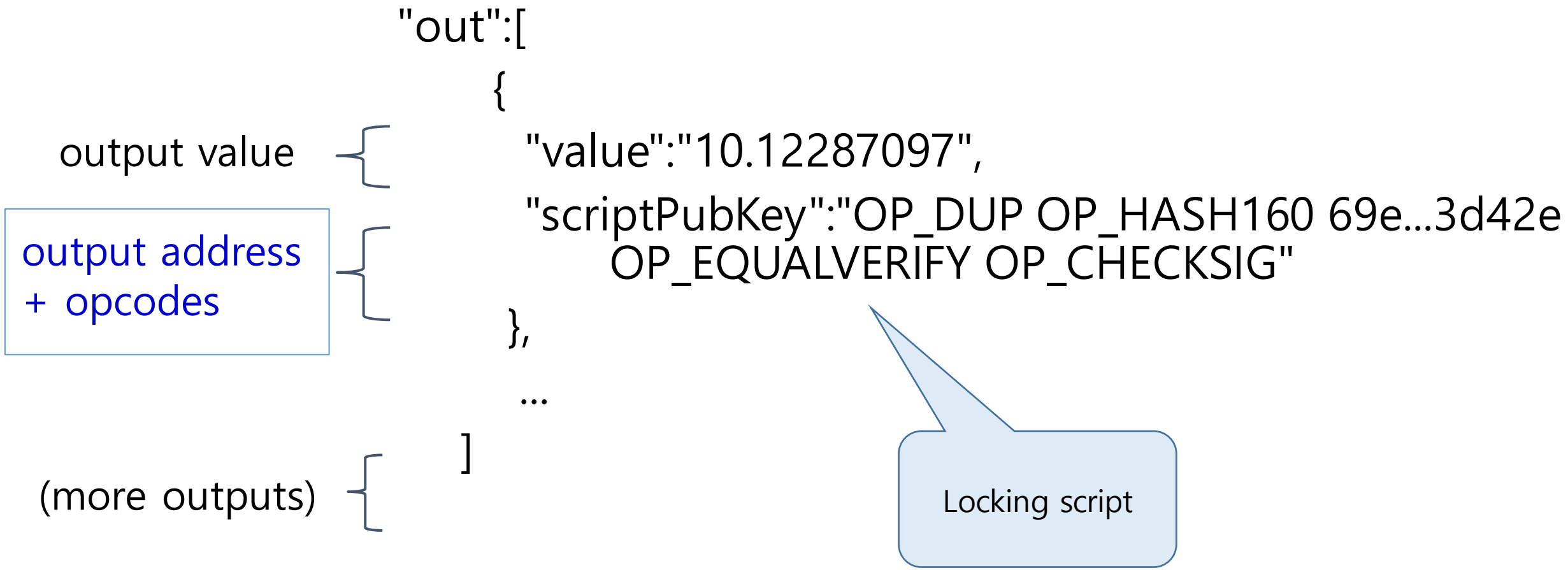
Signature + public_key

(more inputs)

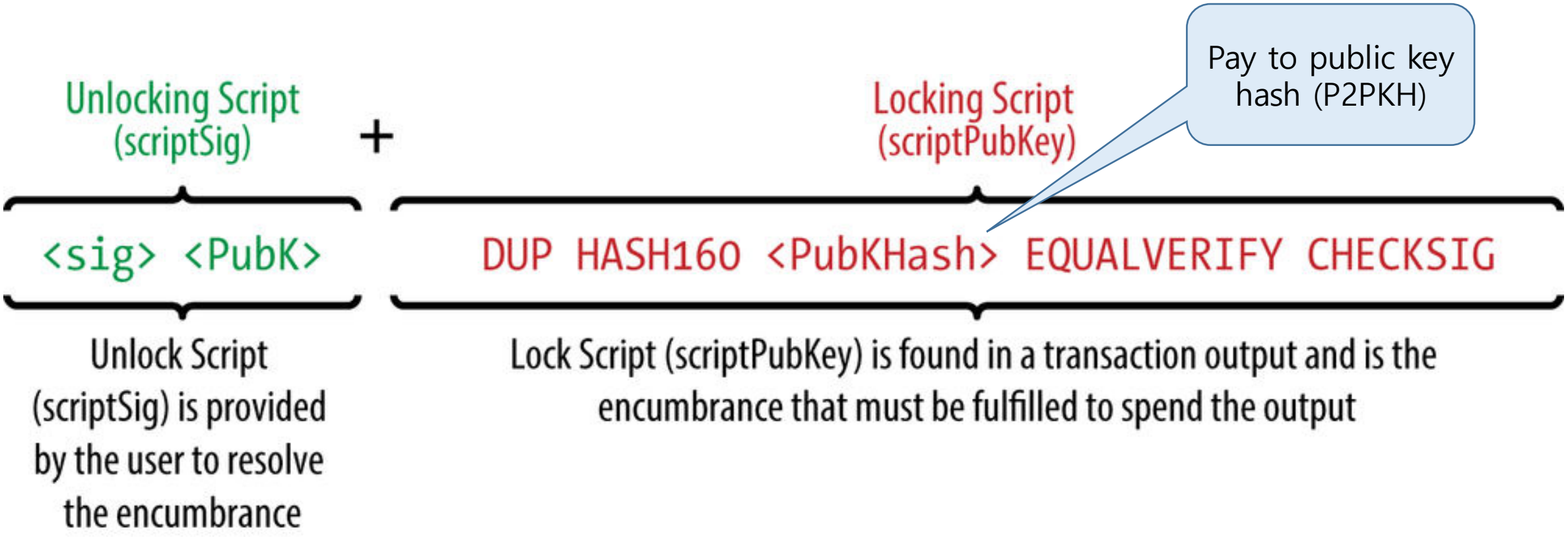
```
"in":[
    {
      "prev_out":{
        "hash":"3be4...80260",
        "n":0
      },
     "scriptSig":"30440....3f3a4ce81"
    },
  ...
  ],
```

Unlocking script

Transaction outputs

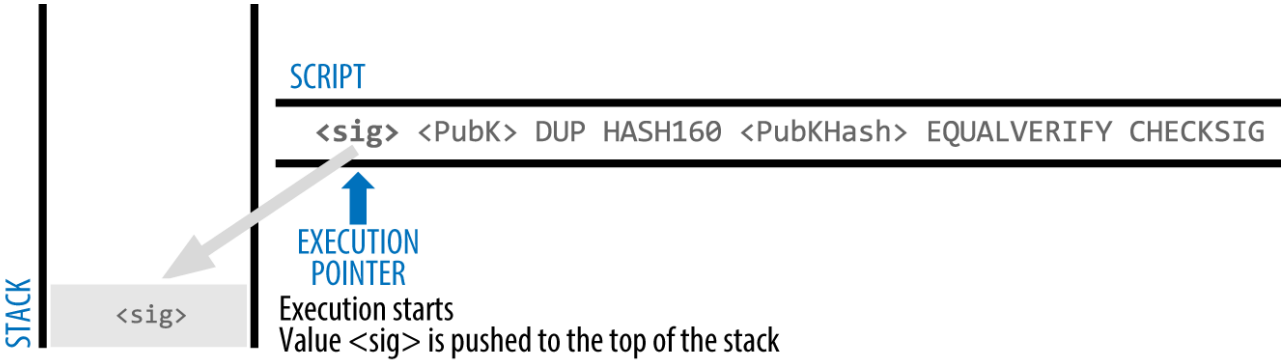


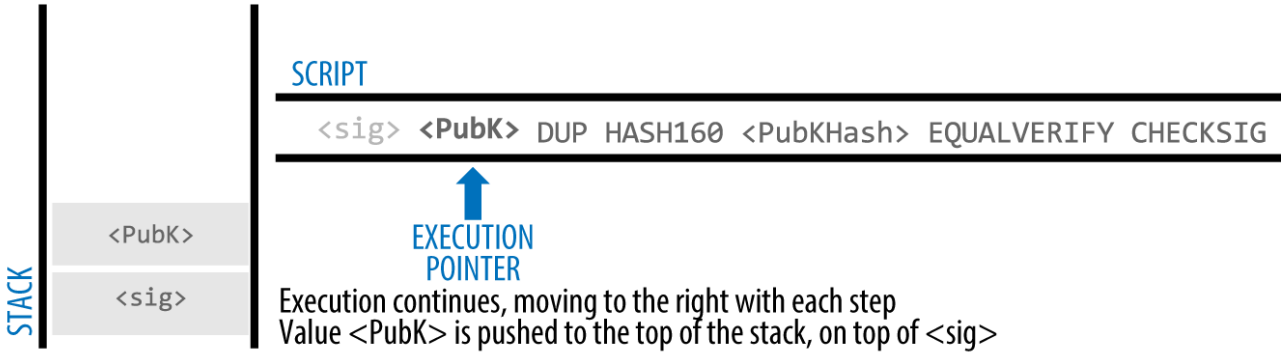
How to process transaction

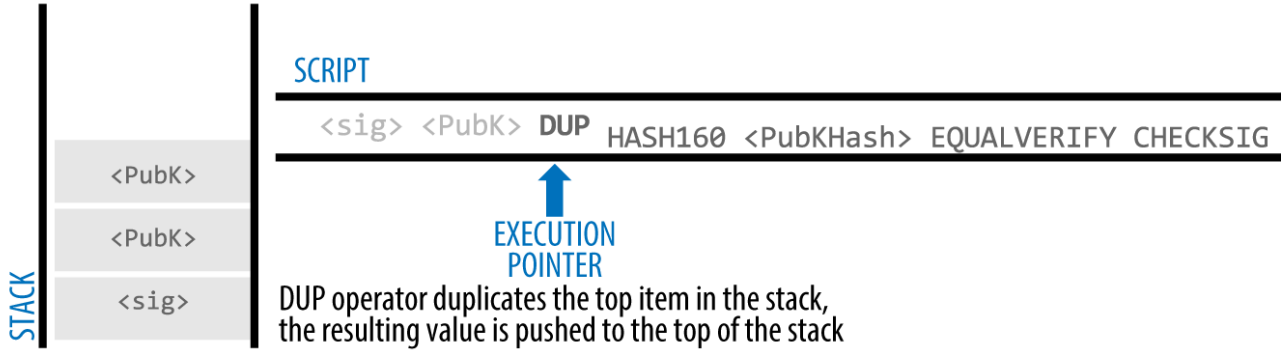


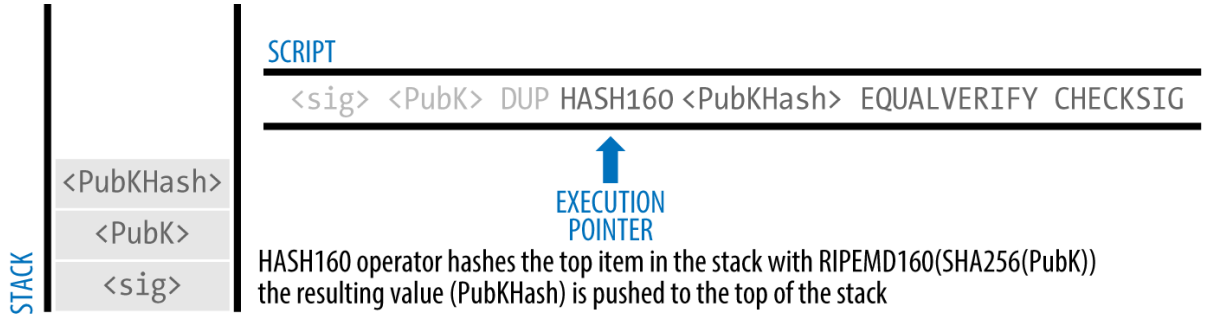
Source: apprize.info

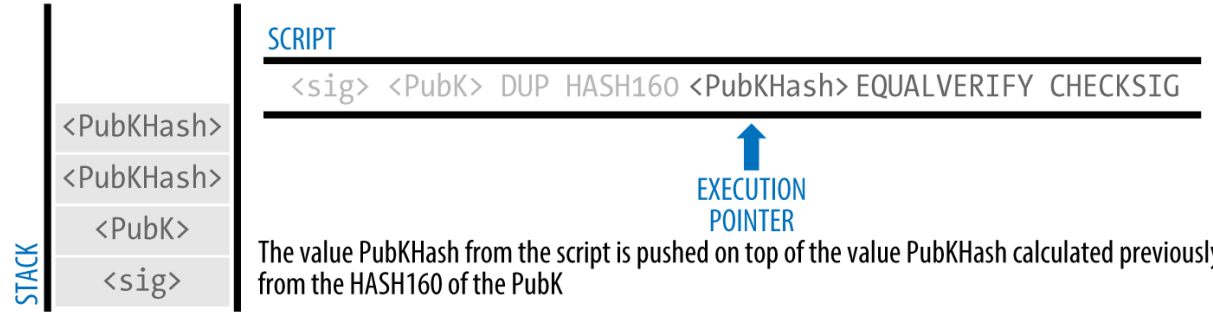
Stack-based operation

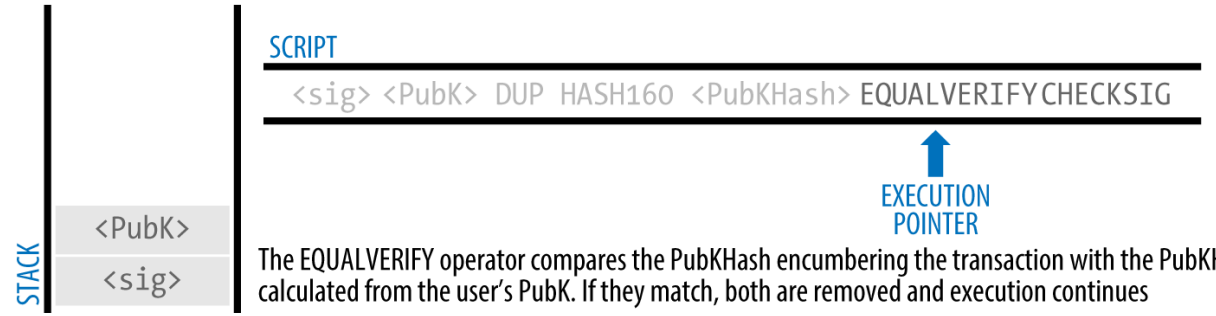


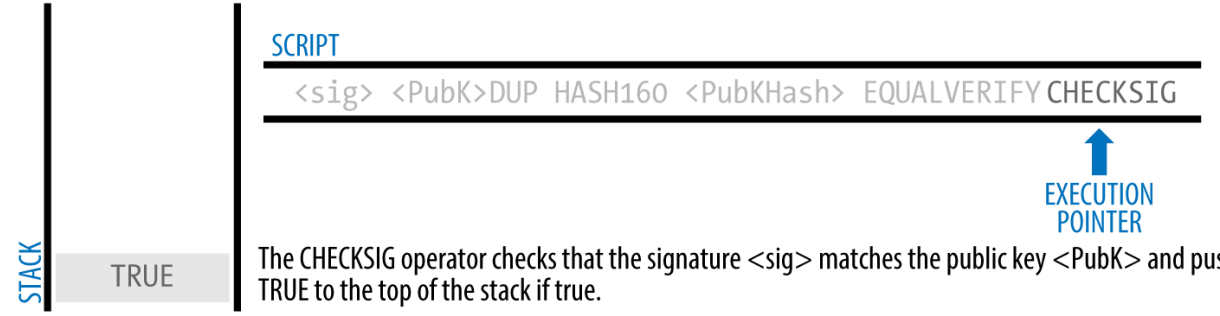












Source: apprize.info

Transaction validation: an illustration

Source: softblocks.co

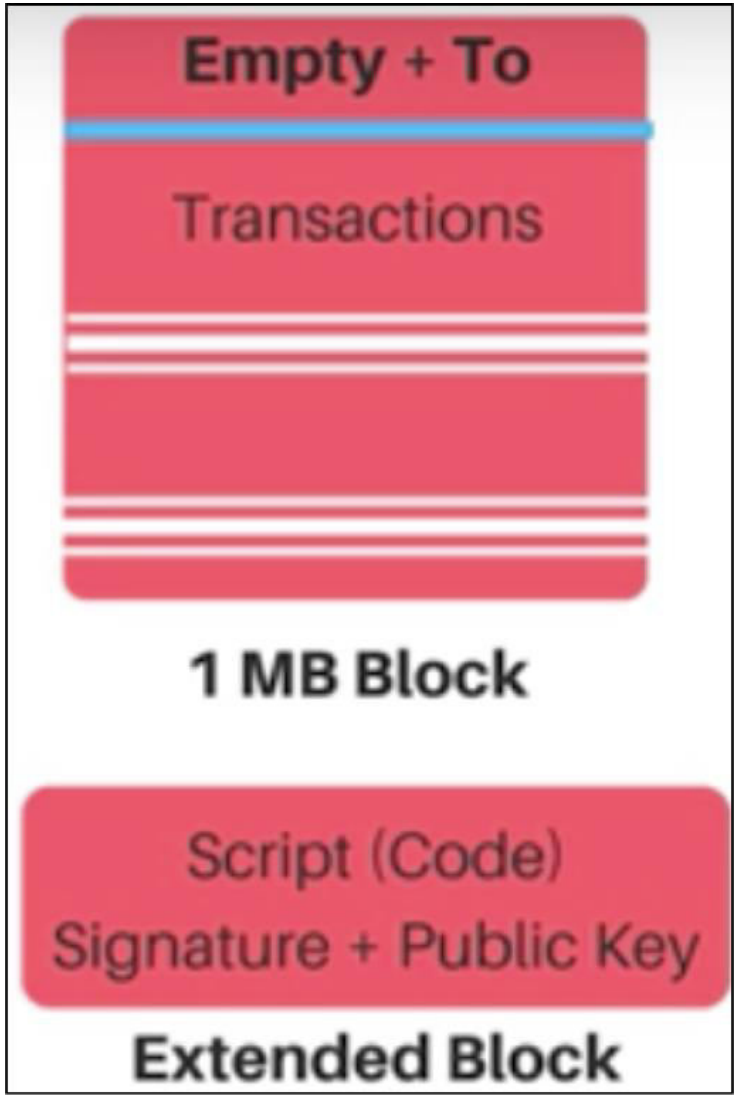
Bitcoin scripting language (“Script”)

Design goals

- Built for Bitcoin (inspired by Forth)
- Stack-based
- Simple, finite
- No looping
- Support for cryptography
    - MULTISIG addresses

After SegWit

- Segregated witness (SegWit) effectively increases the block capacity



Source: blockgeeks.com

Centralized ledger (ScroogeCoin)

Scrooge publishes ledger of all transactions
(a blockchain, signed by Scrooge)

signed by $pk_{Scrooge}$

H( )

Block height increments

prev: H( )
trans

prev: H( )
trans

prev: H( )
trans

Merkle tree of TXs in each block

Don’t worry, I’m honest.

What if Scrooge is malicious?

Forking

prev: H( )
transID: x
input:
w[0]
Output:
0: 45.3

prev: H( )
transID: y

prev: H( )
transID: z
input:
x[0]
Output:
0: 45.3→a

signed by $pk_{Scrooge}$

H( )

prev: H( )
transID: z'
input:
x[0]
Output:
0: 45.3→b

signed by $pk_{Scrooge}$

H( )

double-spending attack

Other Scrooge problems

- Blacklist addresses
- Demand transaction fees
- Go offline
- Get hacked

Decentralization

Can we avoid vulnerability to misbehavior by one entity?

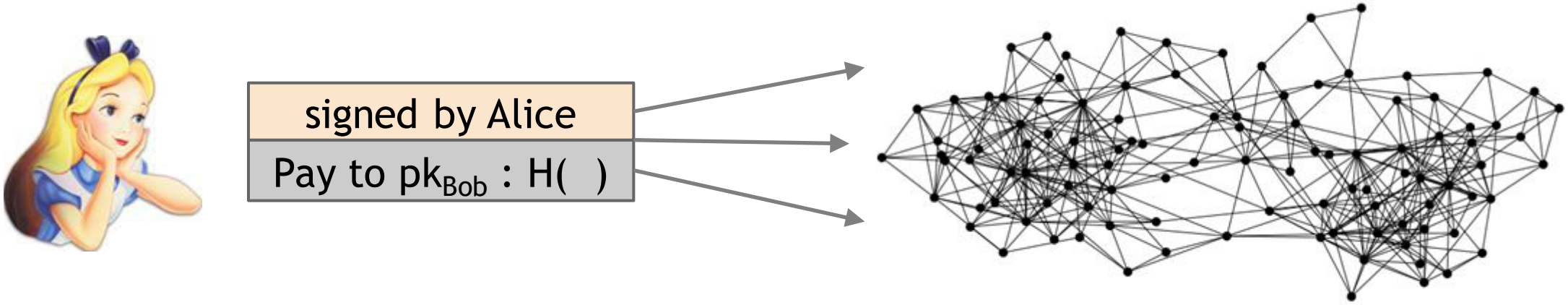
6. A decentralized ledger: Bitcoin

Two different words

- Decentralized vs. distributed

Bitcoin is a peer-to-peer system

When Alice wants to pay Bob:
she broadcasts the transaction to all Bitcoin nodes



All nodes must agree on a sequence of transactions

consensus

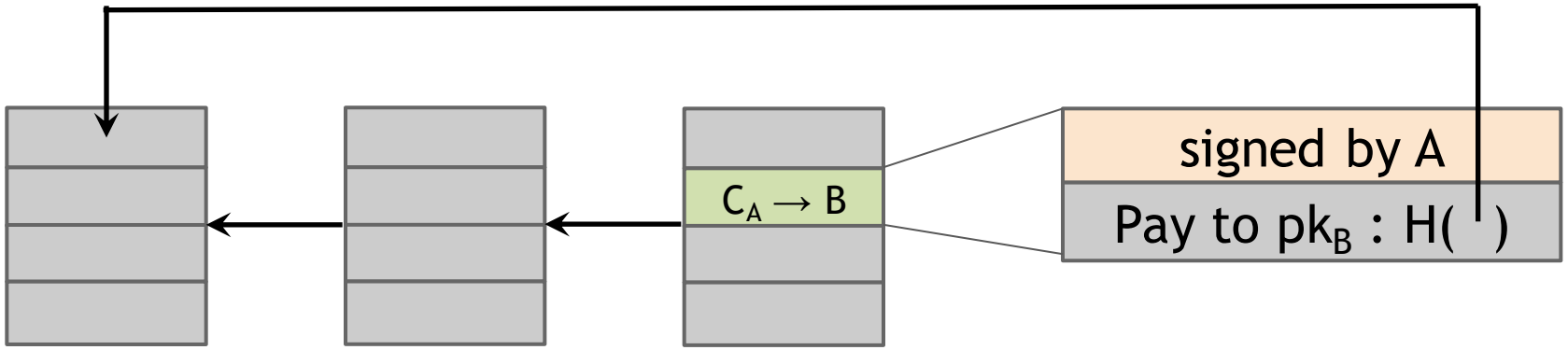
- Consensus is the process by which a network of nodes provides a guaranteed ordering of transactions and validates transactions
- Lottery-based algorithms
  - scale to a large number of nodes since the winner of the lottery proposes a block and transmits it to the rest of the network for validation
  - may lead to forking when two “winners” propose a block. Each fork must be resolved, which results in a longer time to finality
  - Proof-of-Work (PoW), Proof-of-Stake (PoS)
- voting-based algorithms
  - When a majority of nodes validates a transaction or block, consensus exists and finality occurs (thus low latency finality)
  - typically require nodes to transfer messages to each of the other nodes on the network; the more nodes that exist on the network, the more time it takes to reach consensus (poor scalability)
  - Kafka, PBFT

Bitcoin consensus (simplified)

1. Transactions are broadcast to all nodes
2. In each round a random* node signs a block of new transactions, including the hash of the previous block
3. Other nodes *accept* the block if all transactions are valid
   - The block is propagated over the P2P network
4. Invalid blocks are ignored
5. If more than one nodes broadcast (different) blocks with the same height, the next node will choose one of them
6. Longest chain is considered canonical

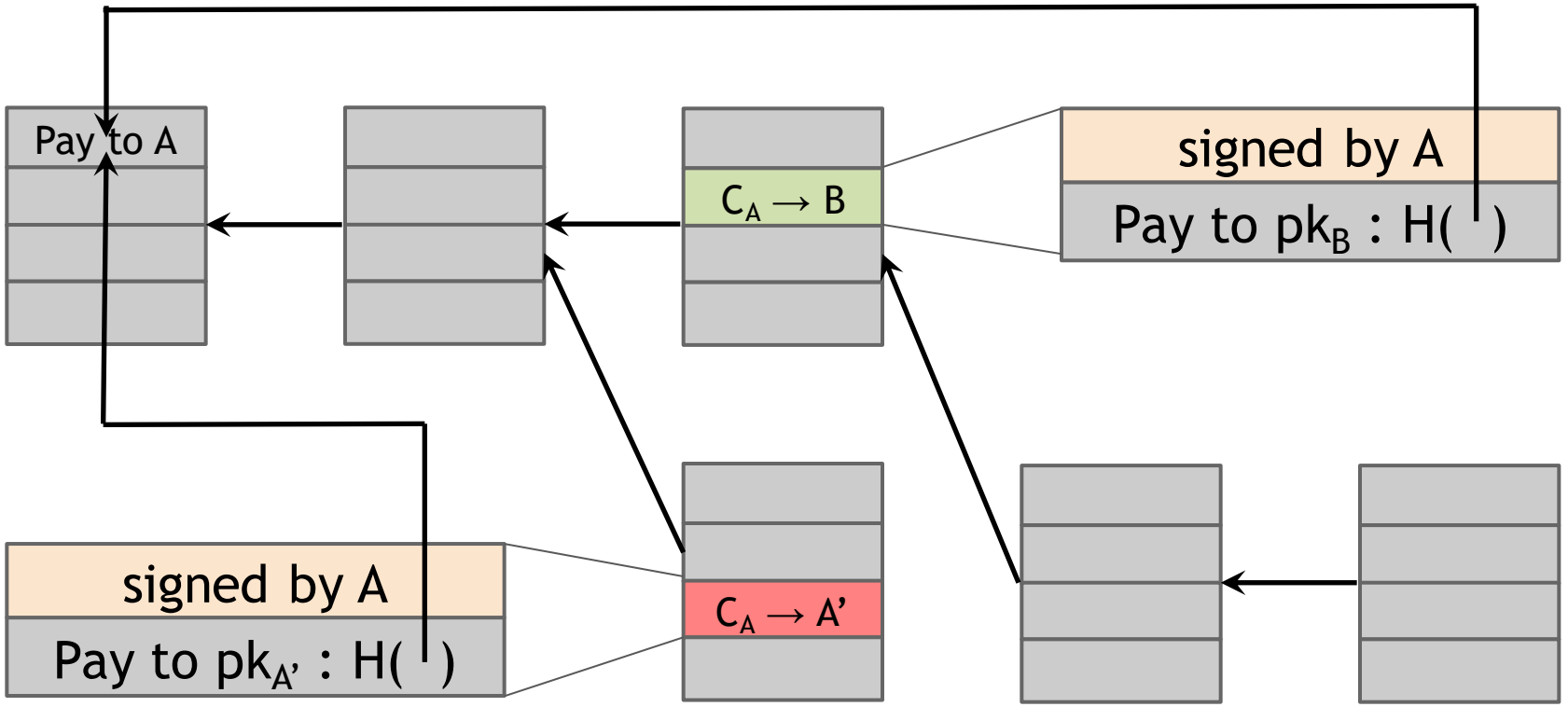
Leads to a valid canonical chain with “honest majority”

What can a malicious node do?



C_A: coin of A

What can a malicious node do?

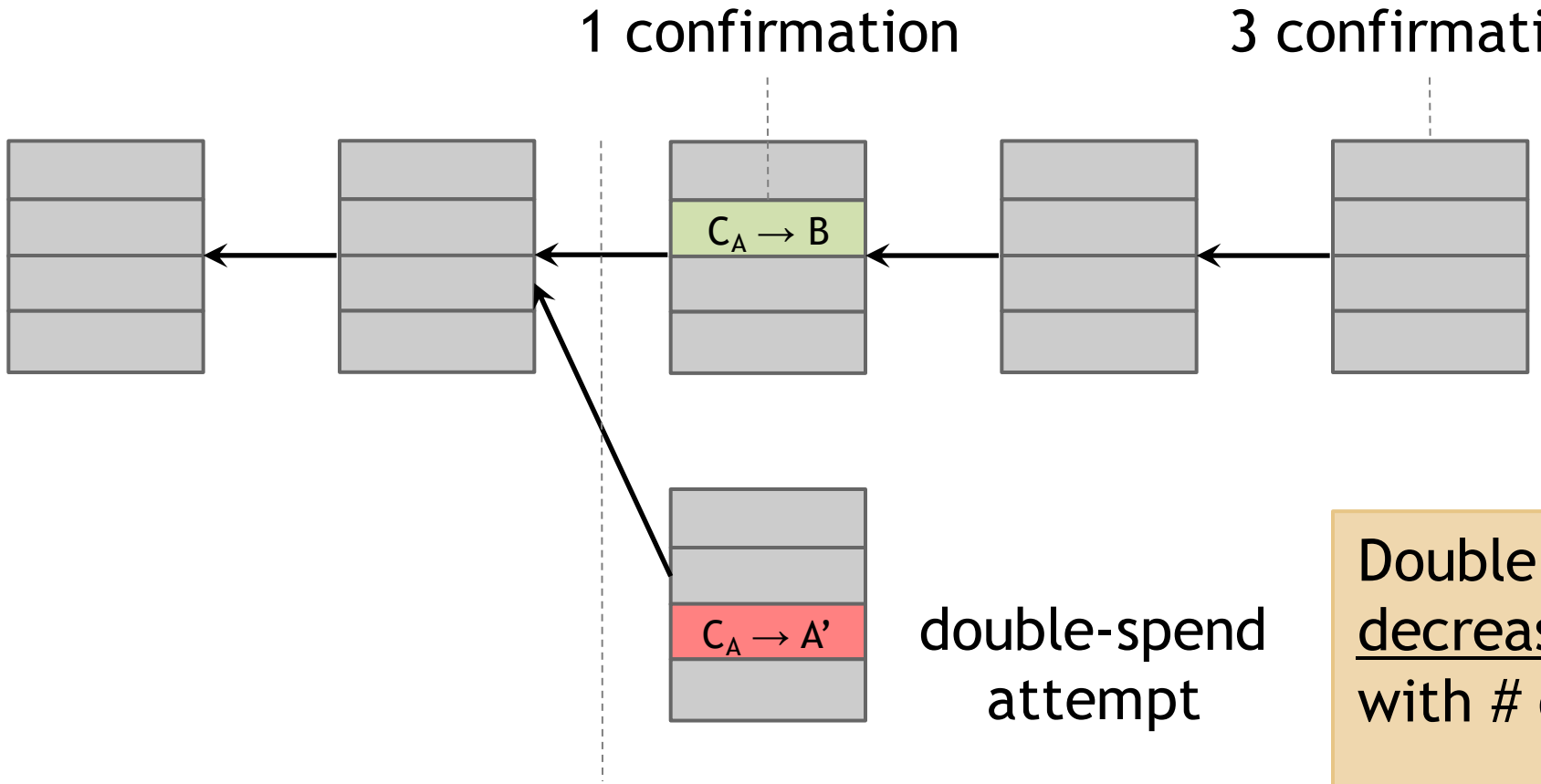


Double-spend

Honest nodes will extend the longest valid branch

C_A: coin of A

From a merchant's point of view

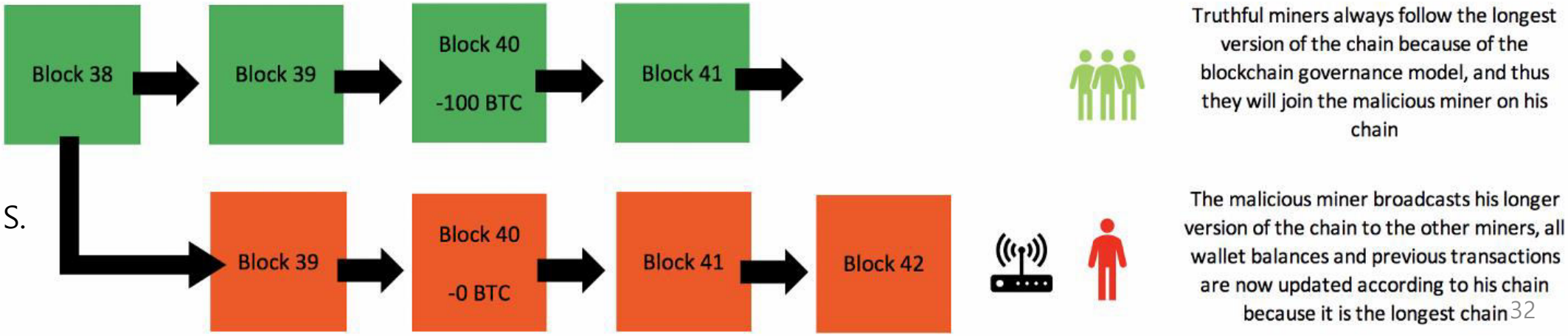


Double-spend probability <u>decreases exponentially</u> with # of confirmations

Most common heuristic:
6 confirmations

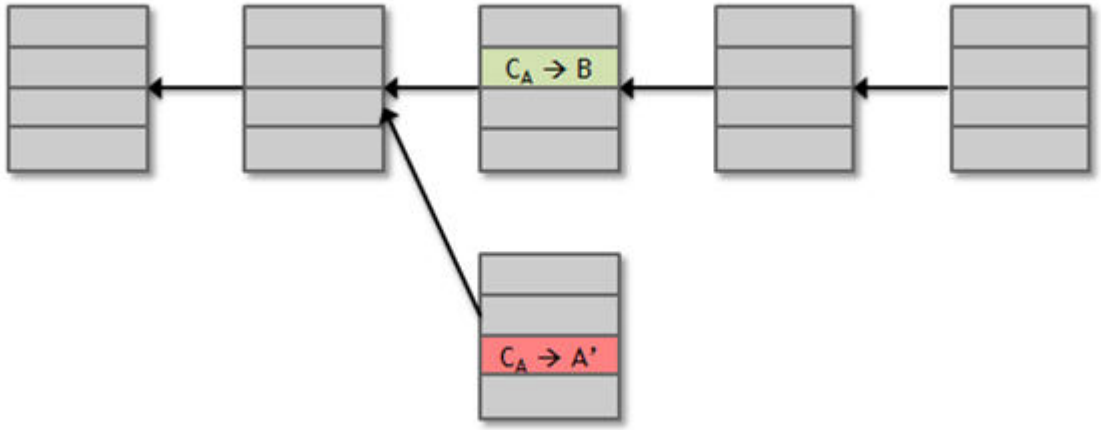
51% (or majority) attack

- The attacker submits to the merchant/network a transaction which pays the merchant
- He also privately mining a blockchain fork in which a double-spending transaction is included instead
- After waiting for n confirmations, the merchant sends the product
- If the attacker happened to find more than n blocks at this point, he releases his fork and regains his coins
- Or he can try to continue extending his fork with the hope of being able to catch up with the network.



Source: Jimi S. @Medium

Basic properties

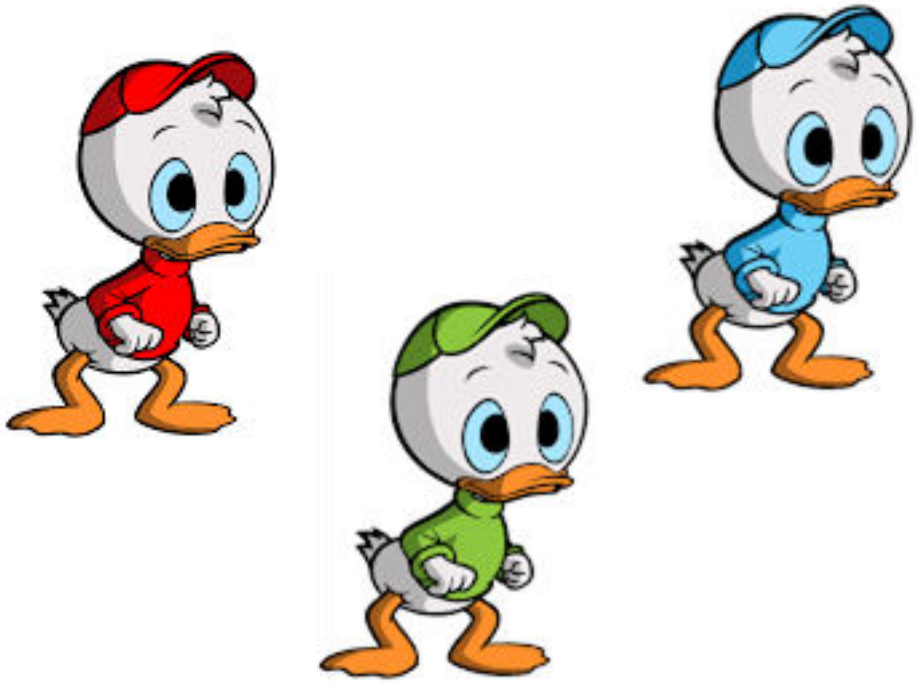


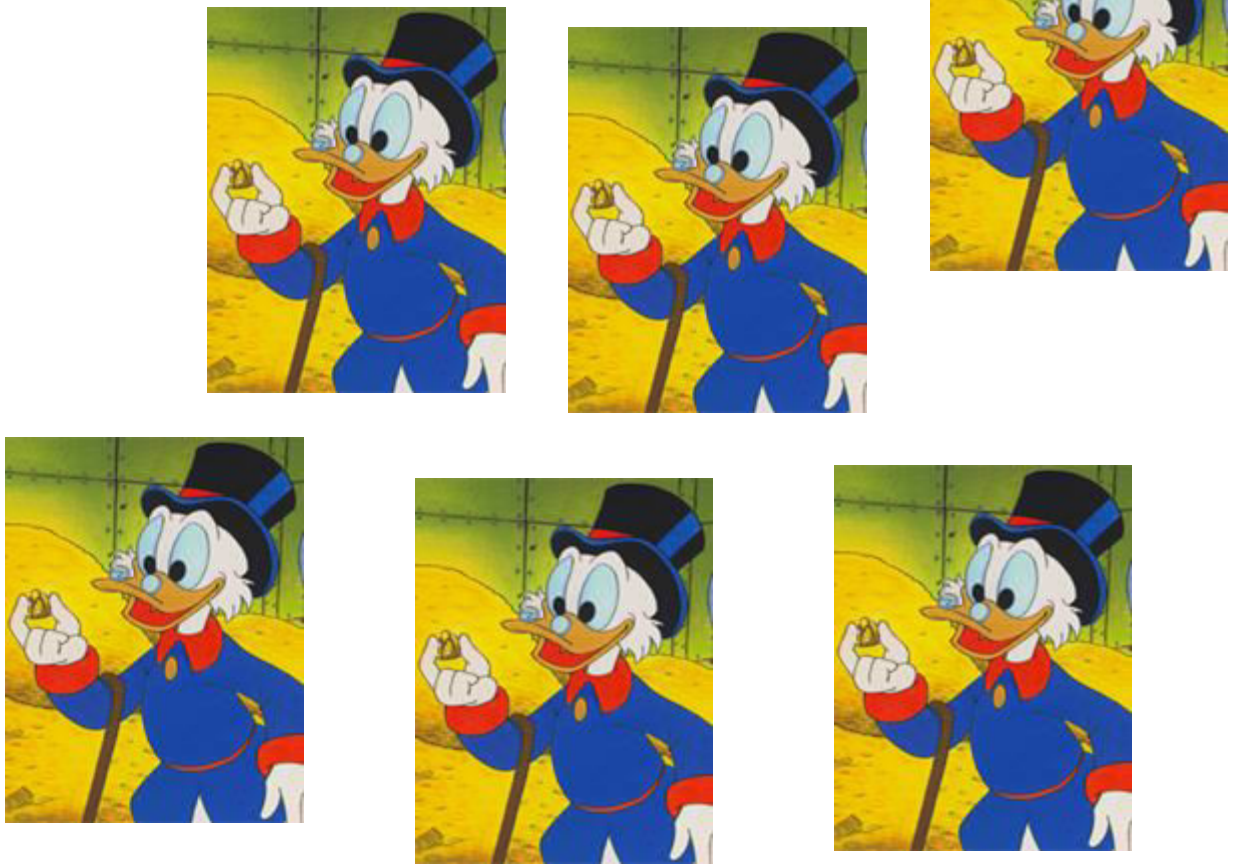
Protection against invalid transactions is cryptographic

Protection against double-spending relies on consensus

You’re never 100% sure a transaction is in the blockchain

Honest majority of whom?





Recall: addresses can be freely created

Solution: “vote” by CPU power

Bitcoin mining puzzle:

Given hashes of the previous and current blocks prev, curr:

Find a nonce such that $H(\text{prev}|\text{curr}|\text{nonce}) < 2^{256-d}$

d is a difficulty parameter

First solution wins

Miners are rewarded for solutions

Creator of block gets to

- include special coin-creation transaction in the block
- choose recipient address of this transaction

“Block reward” currently 12.5 BTC, halves every 4 years

Transaction fees also kept

Rewarded only if block is on eventual consensus branch!

Recap

Bitcoins created by special mining transactions

Bitcoins owned by public keys (addresses)

Bitcoin transfers authorized by digital signatures

Blockchain records all transfers, prevents double spends

Miners extend blockchain by solving proof of work

Miners rewarded by creating new bitcoins

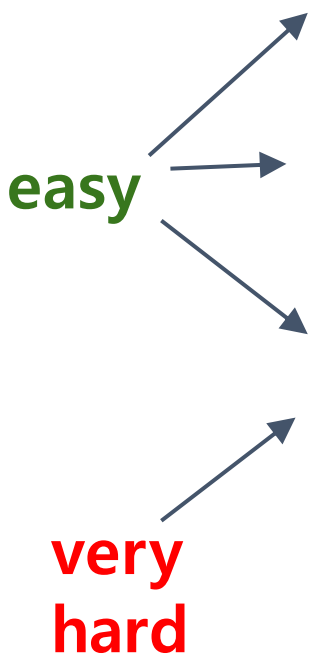
Bitcoin miners

Bitcoin depends on miners to:

- Store and broadcast the block chain
- Validate new transactions, which form a Merkle tree
- Vote (by hash power) on consensus

Miners rewarded with new coins

Mining Bitcoins in 6 easy steps



1. Join the network, listen for transactions
    a. Validate all proposed transactions
2. Listen for new blocks, maintain block chain
    a. When a new block is proposed, validate it
3. Assemble a new valid block
4. Find a nonce to make your block valid
5. Hope everybody accepts your new block
6. Profit!

7. Bitcoin P2P networking

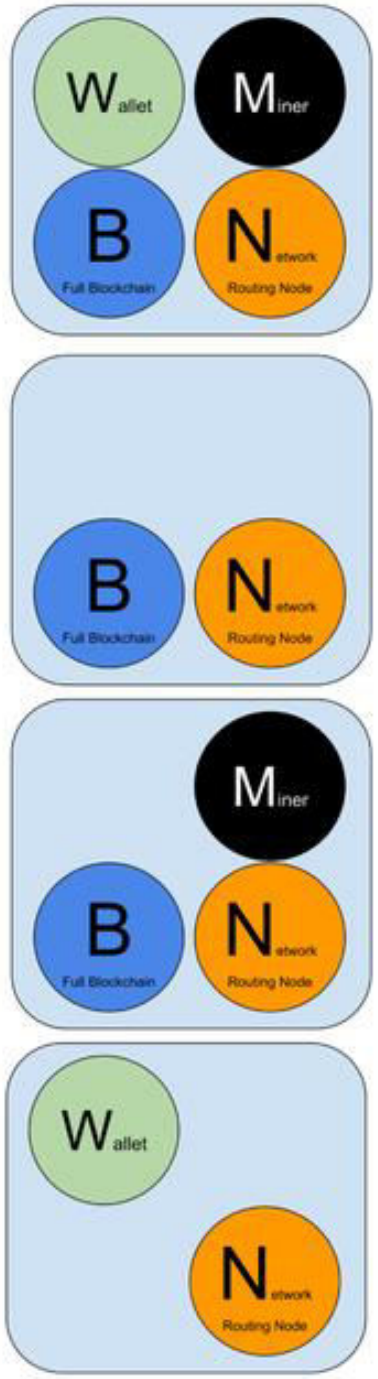
Bitcoin P2P network

Ad-hoc protocol (runs on TCP port 8333)
Ad-hoc network with random topology
All nodes are equal
Flat topology
New nodes can join at any time
Forget non-responding nodes after 3 hr

There is no server, no centralized service, and no hierarchy within the network.

P2P networks are inherently resilient, decentralized, and open.

*SPV: simplified payment verification



Reference Client (Bitcoin Core)

Contains a Wallet, Miner, full Blockchain database, and Network routing node on the bitcoin P2P network.

Full Block Chain Node

Contains a full Blockchain database, and Network routing node on the bitcoin P2P network.

Solo Miner

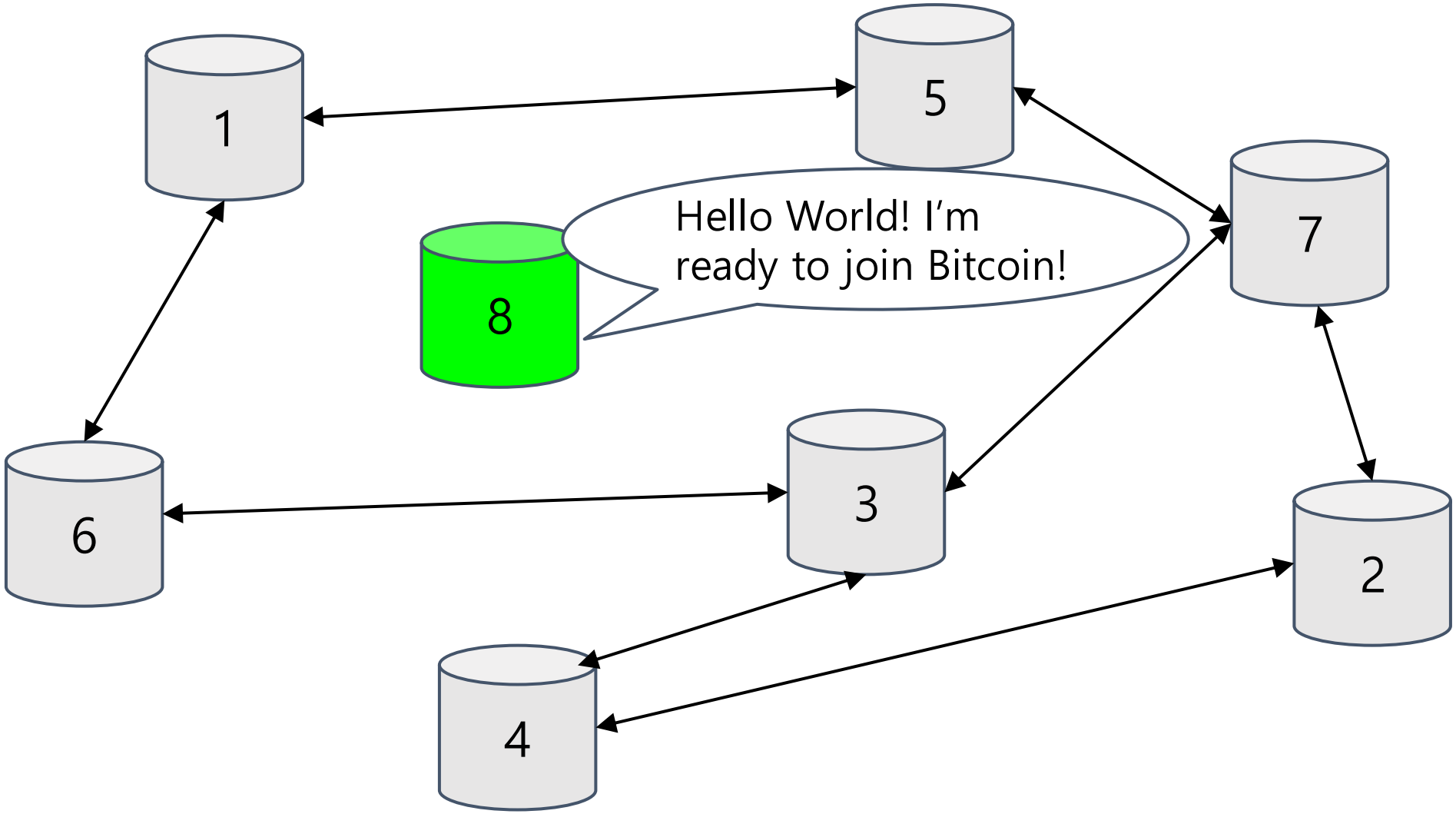
Contains a mining function with a full copy of the blockchain and a bitcoin P2P network routing node.

Lightweight (SPV) wallet

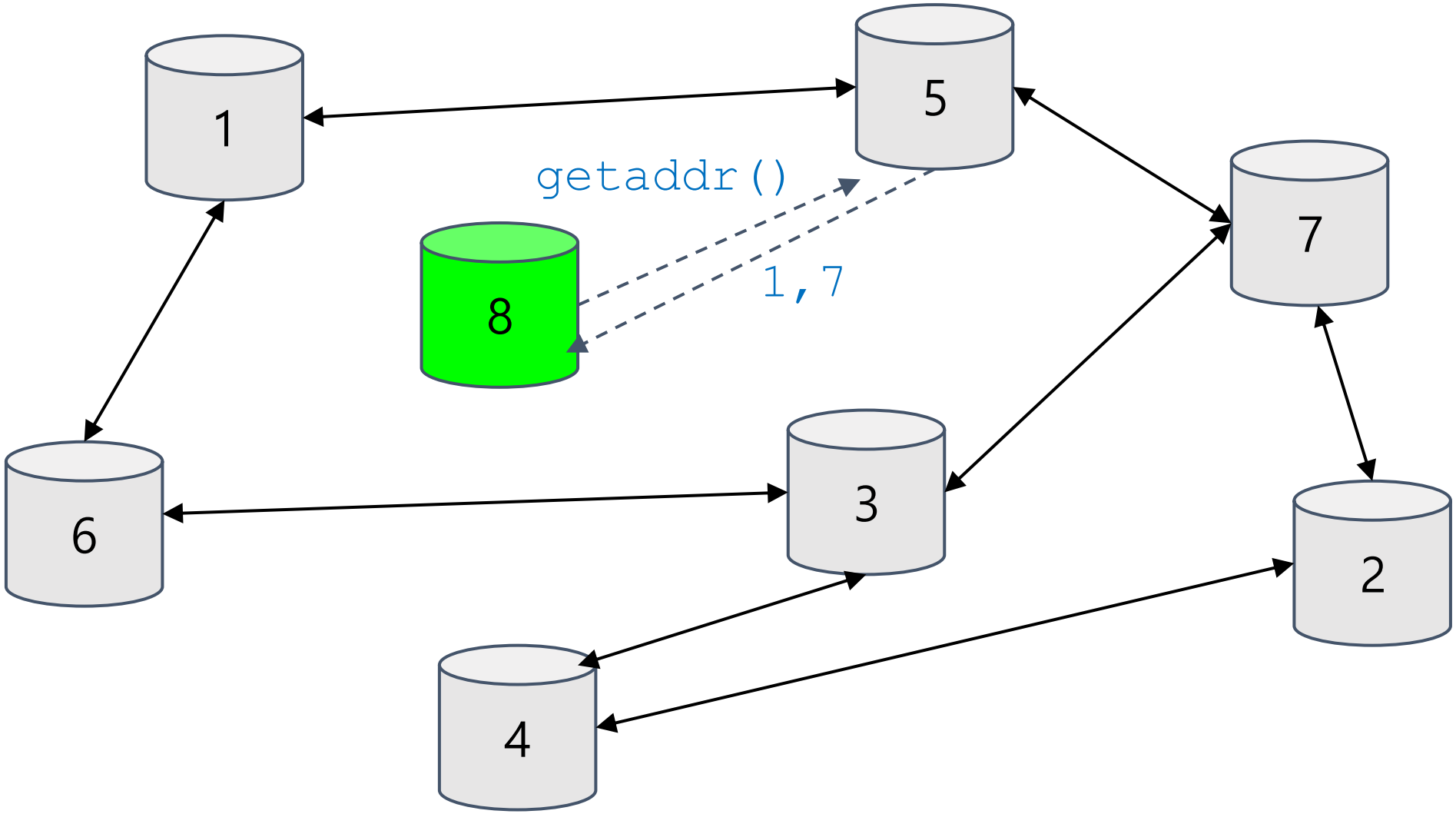
Contains a Wallet and a Network node on the bitcoin P2P protocol, without a blockchain.

Source: O'Reilly Media

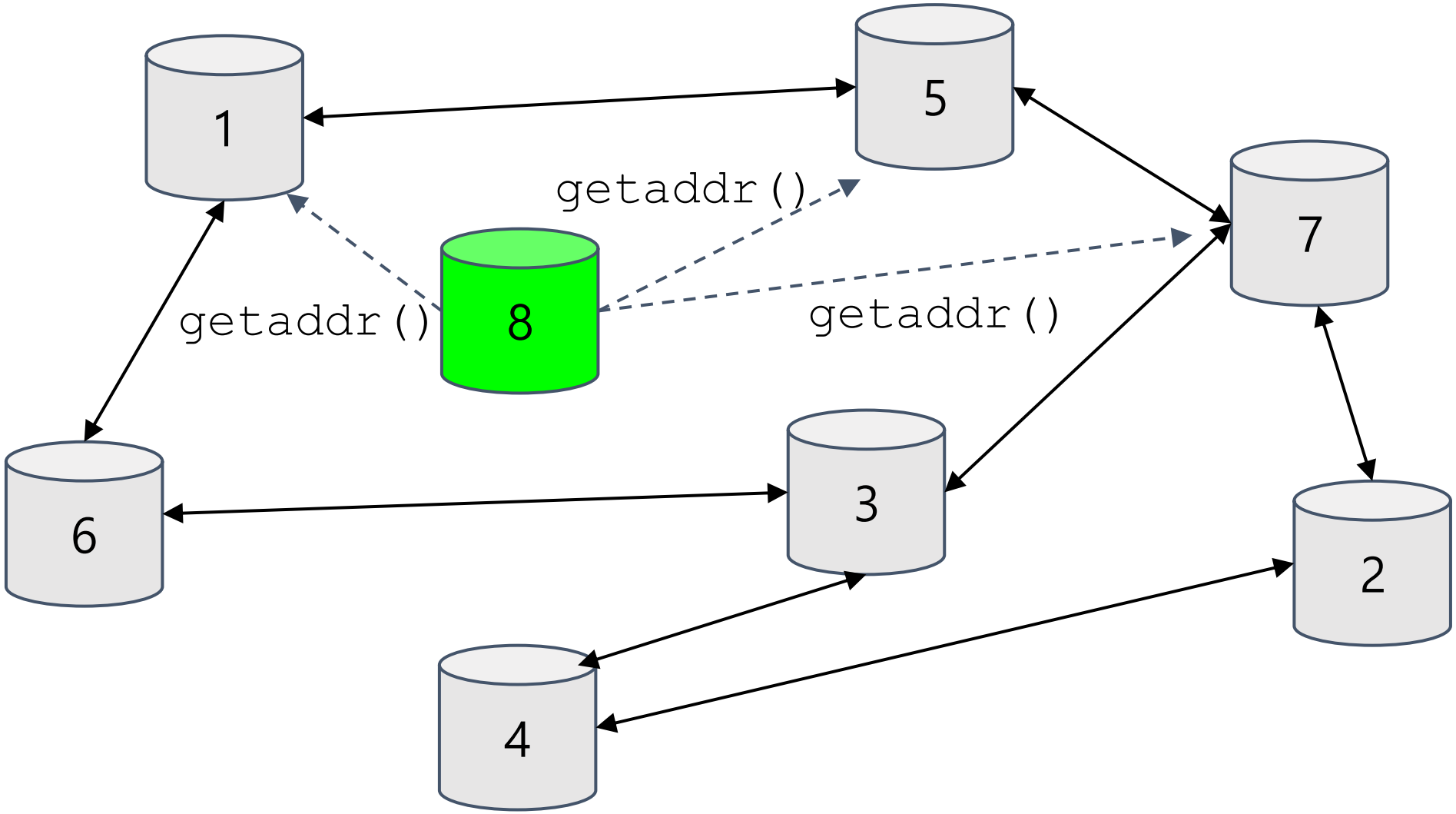
Joining the Bitcoin P2P network



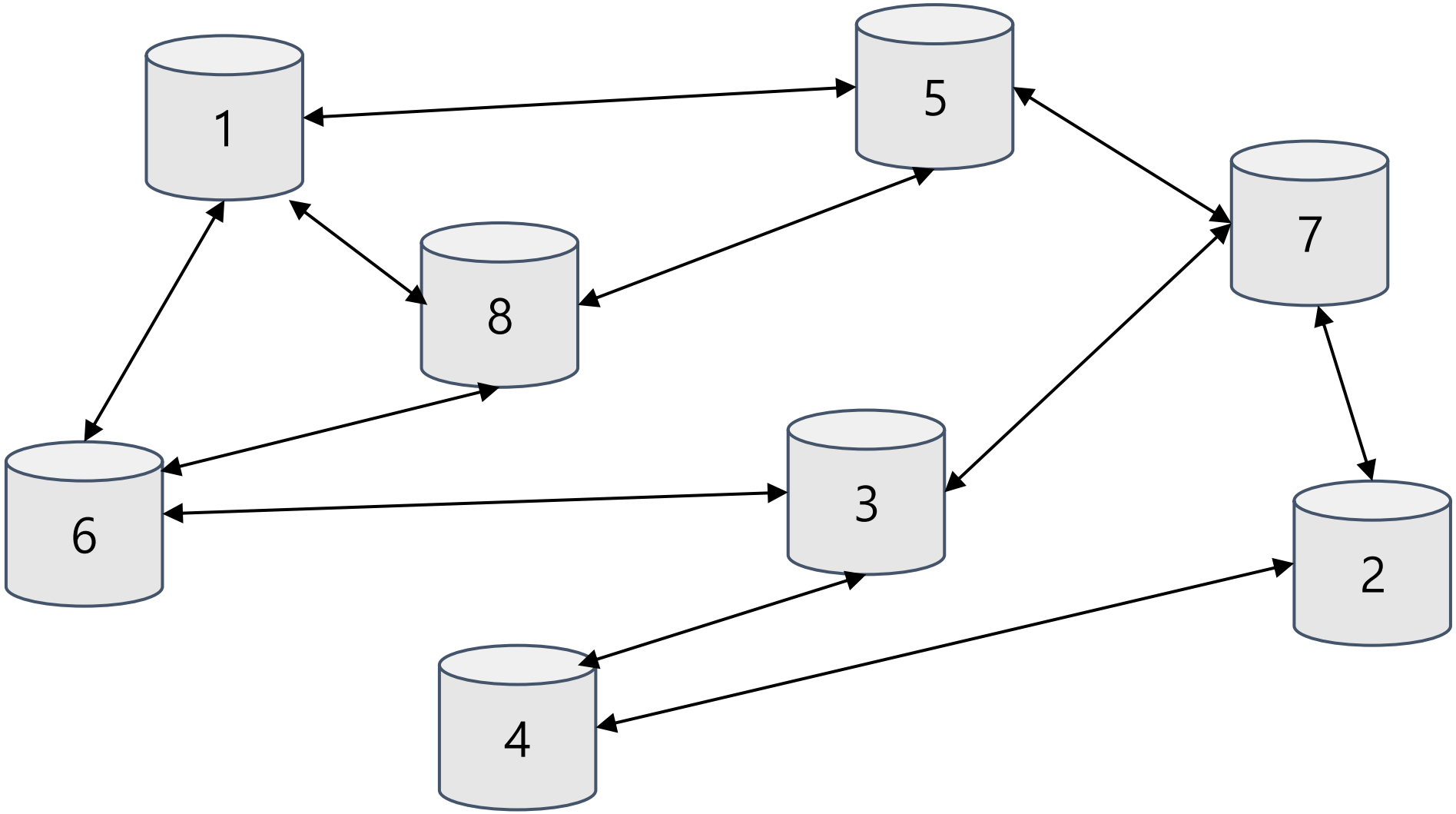
Joining the Bitcoin P2P network



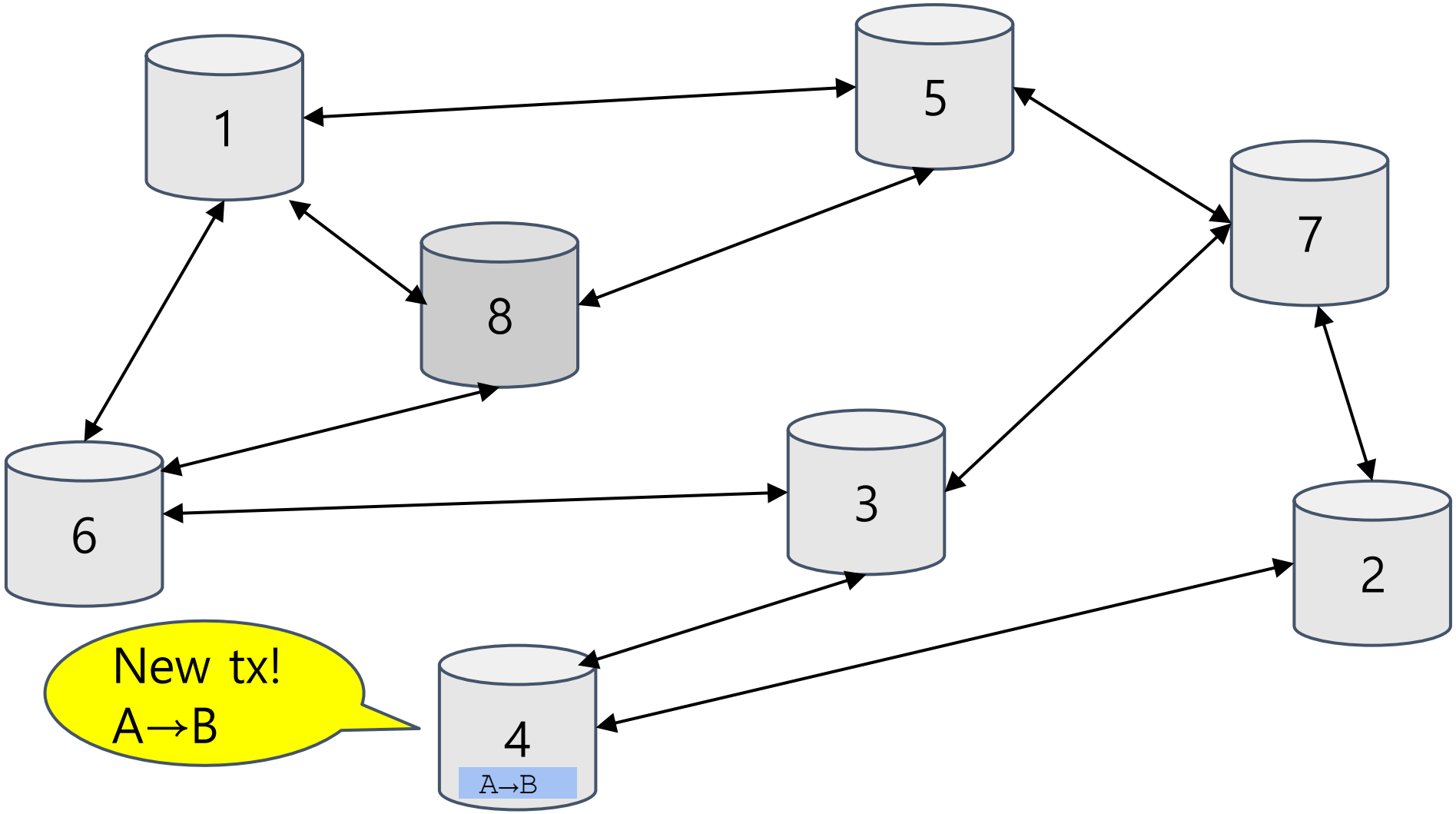
Joining the Bitcoin P2P network



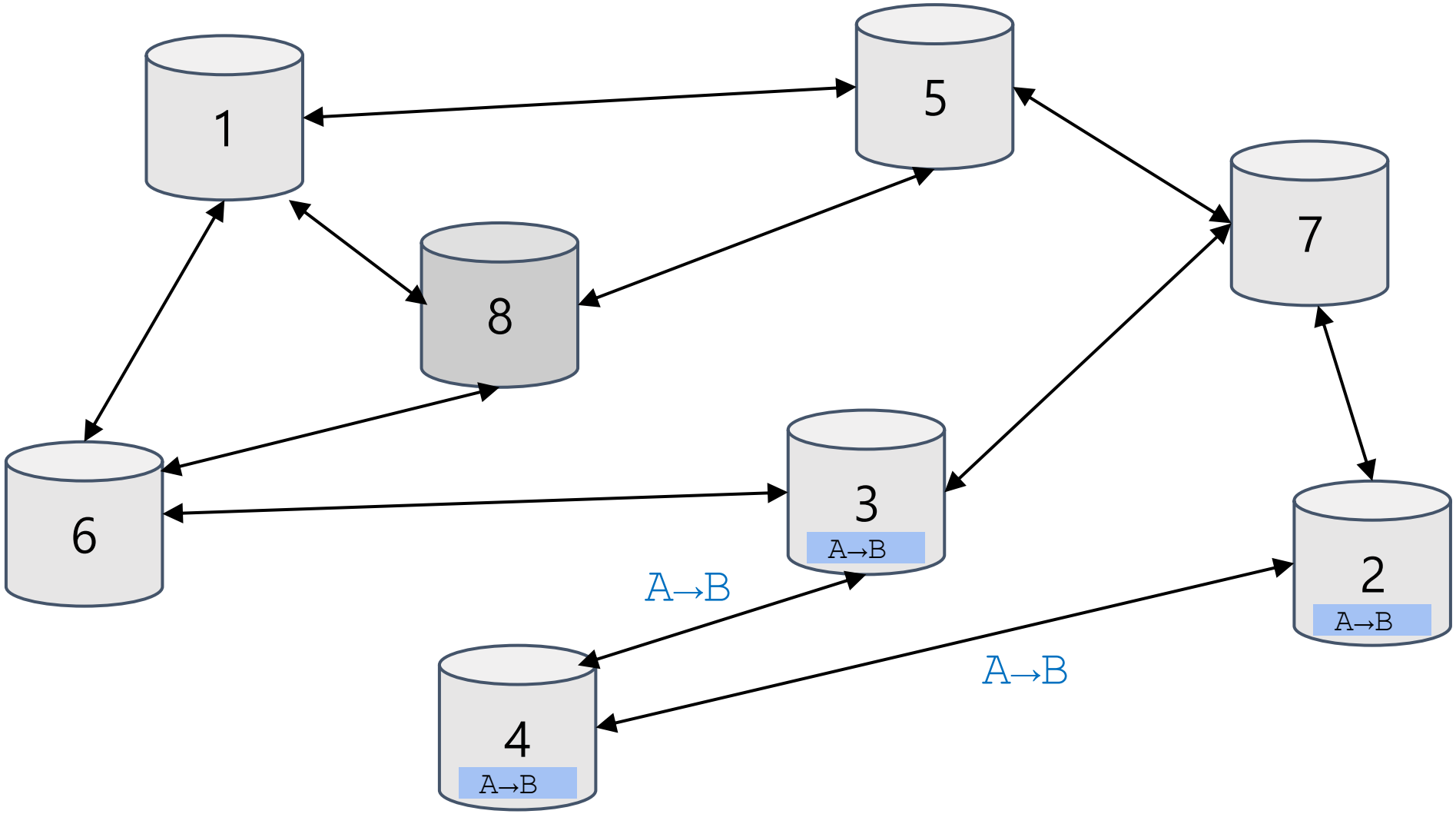
Joining the Bitcoin P2P network



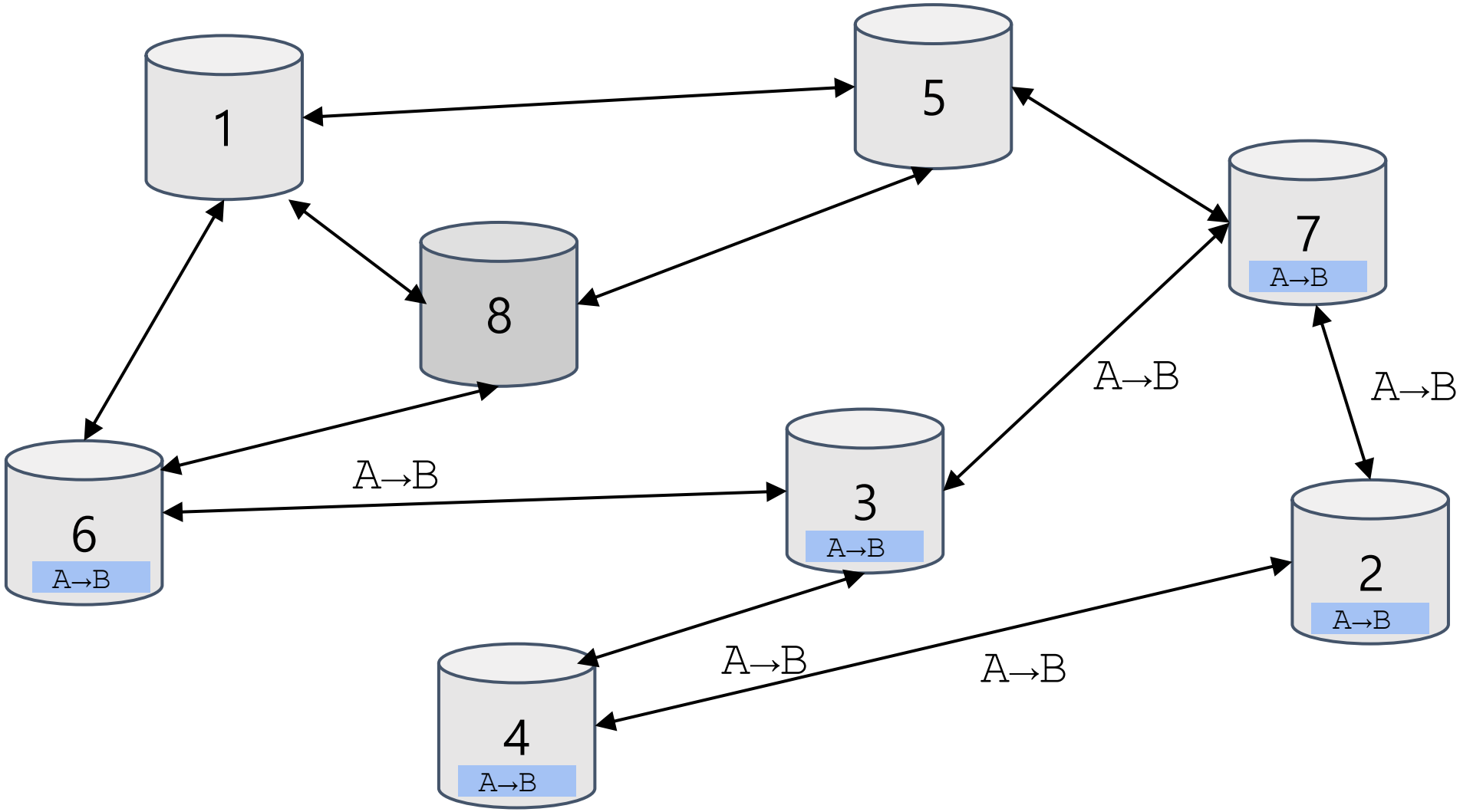
Transaction propagation (flooding)



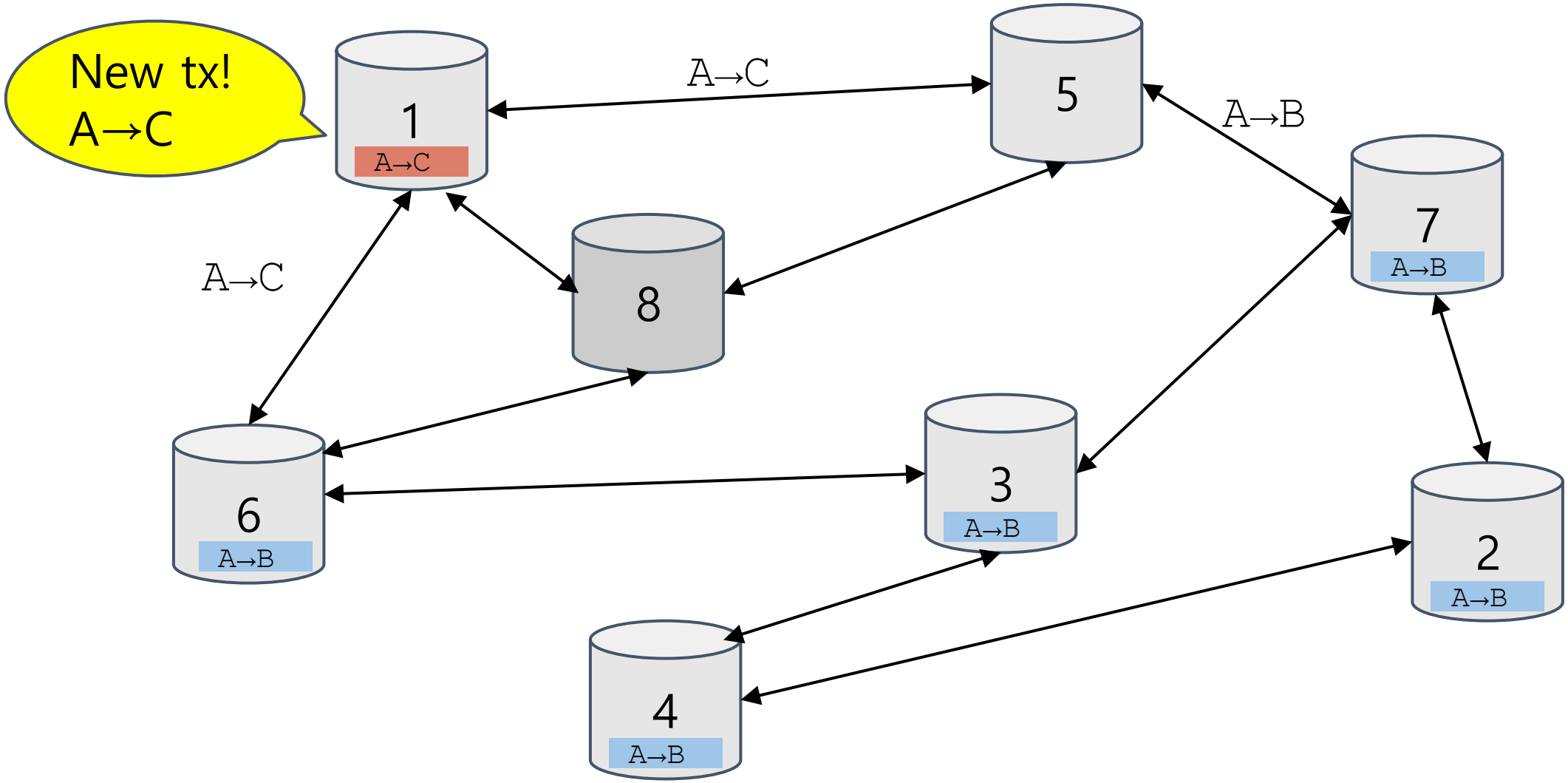
Transaction propagation (flooding)



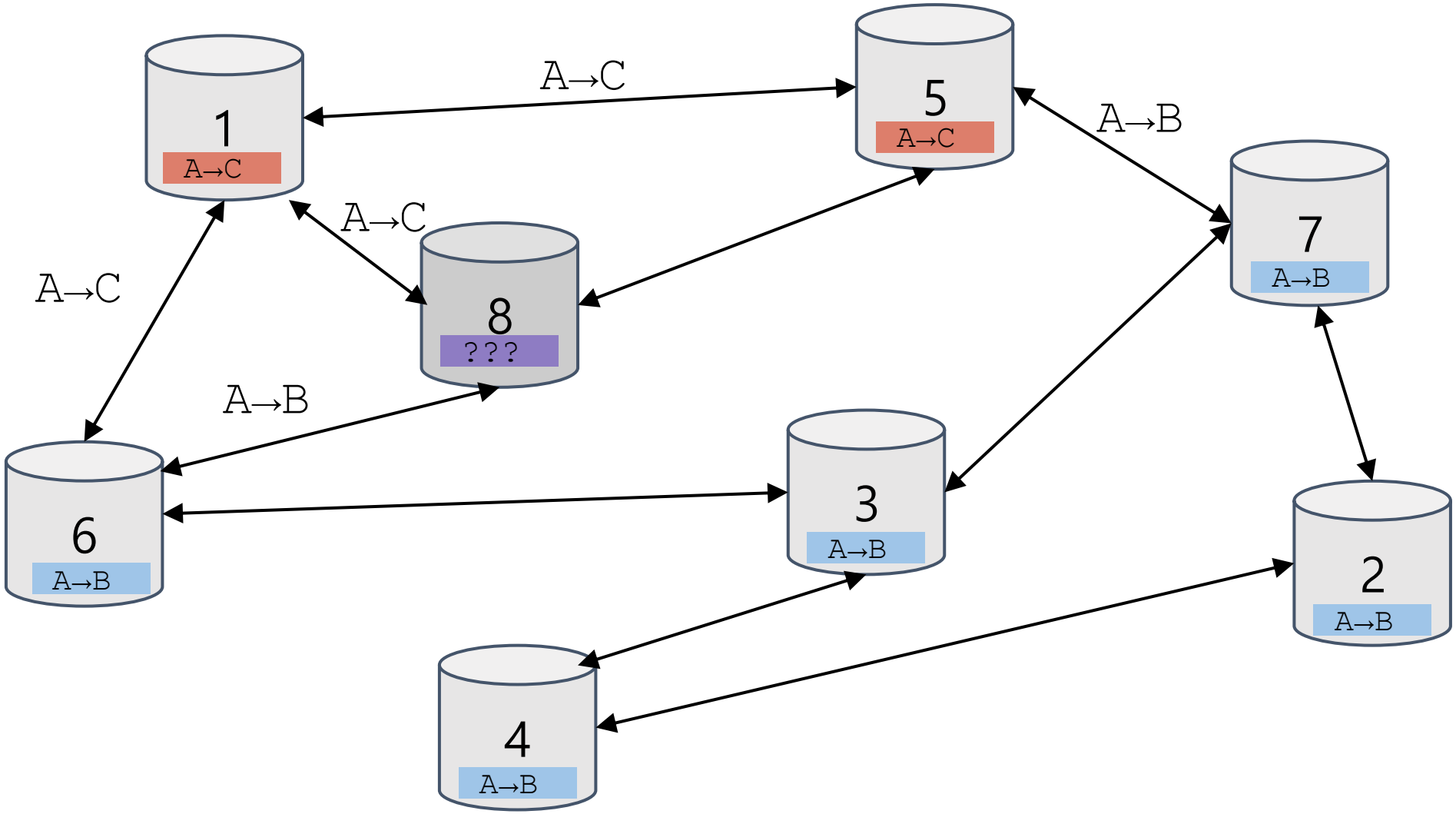
Transaction propagation (flooding)



Nodes may differ on transaction pool



Nodes may differ on transaction pool

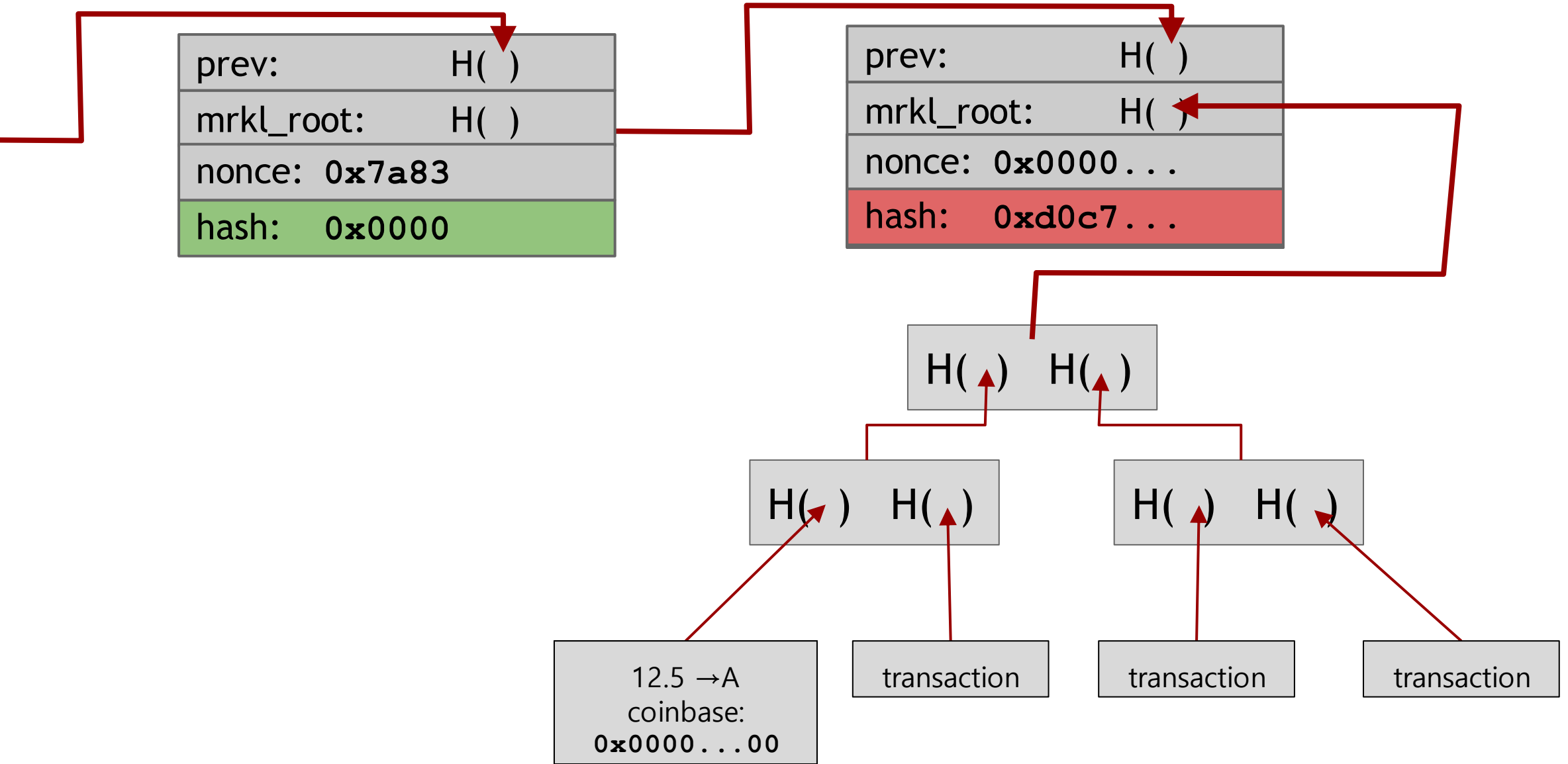


8. Finding a valid block

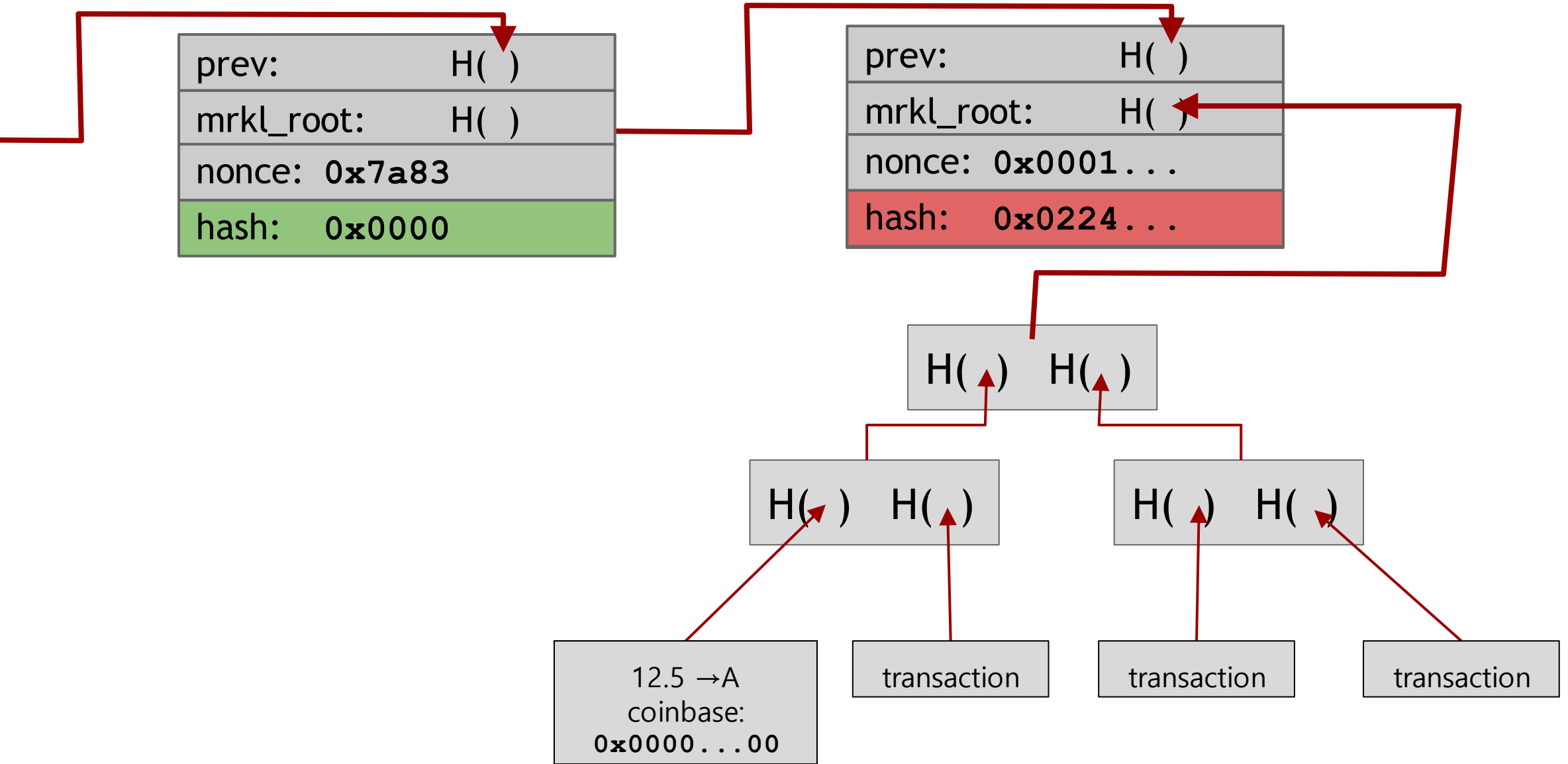
Who solves the puzzle confirms a TX block

- Find an input for a SHA-256 hash function for an output whose value is less than a target
- Difficulty is adjusted by changing the target value
  - target is decreased if a puzzle is solved less than 10 min on average for the last 2016 blocks and is increased otherwise
  - time_2016: time taken to confirm the last 2016 blocks (about 2 weeks)
- New_difficulty = current_difficulty * 20160 min / time_2016

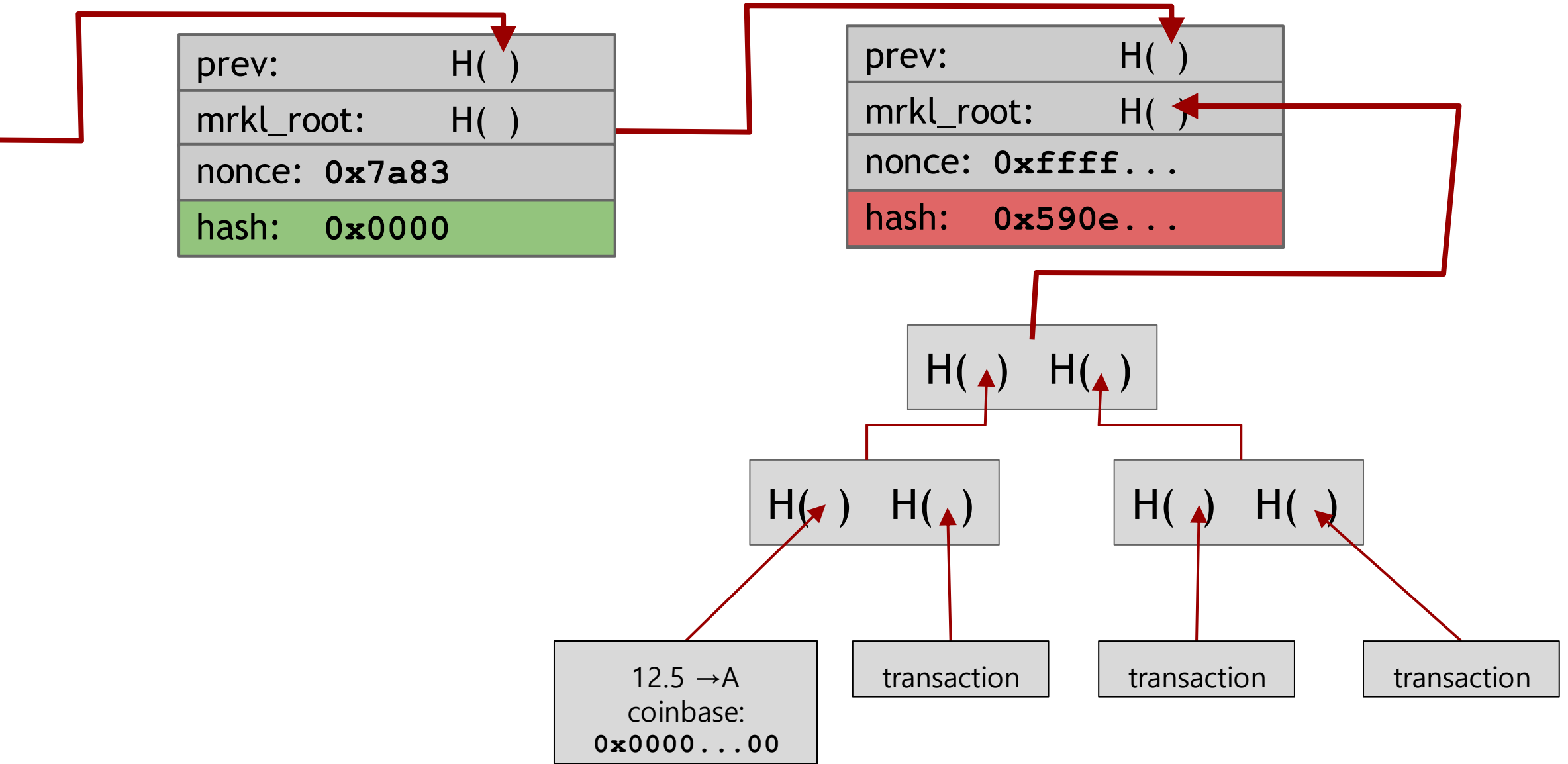
Finding a valid nonce is the hard part!



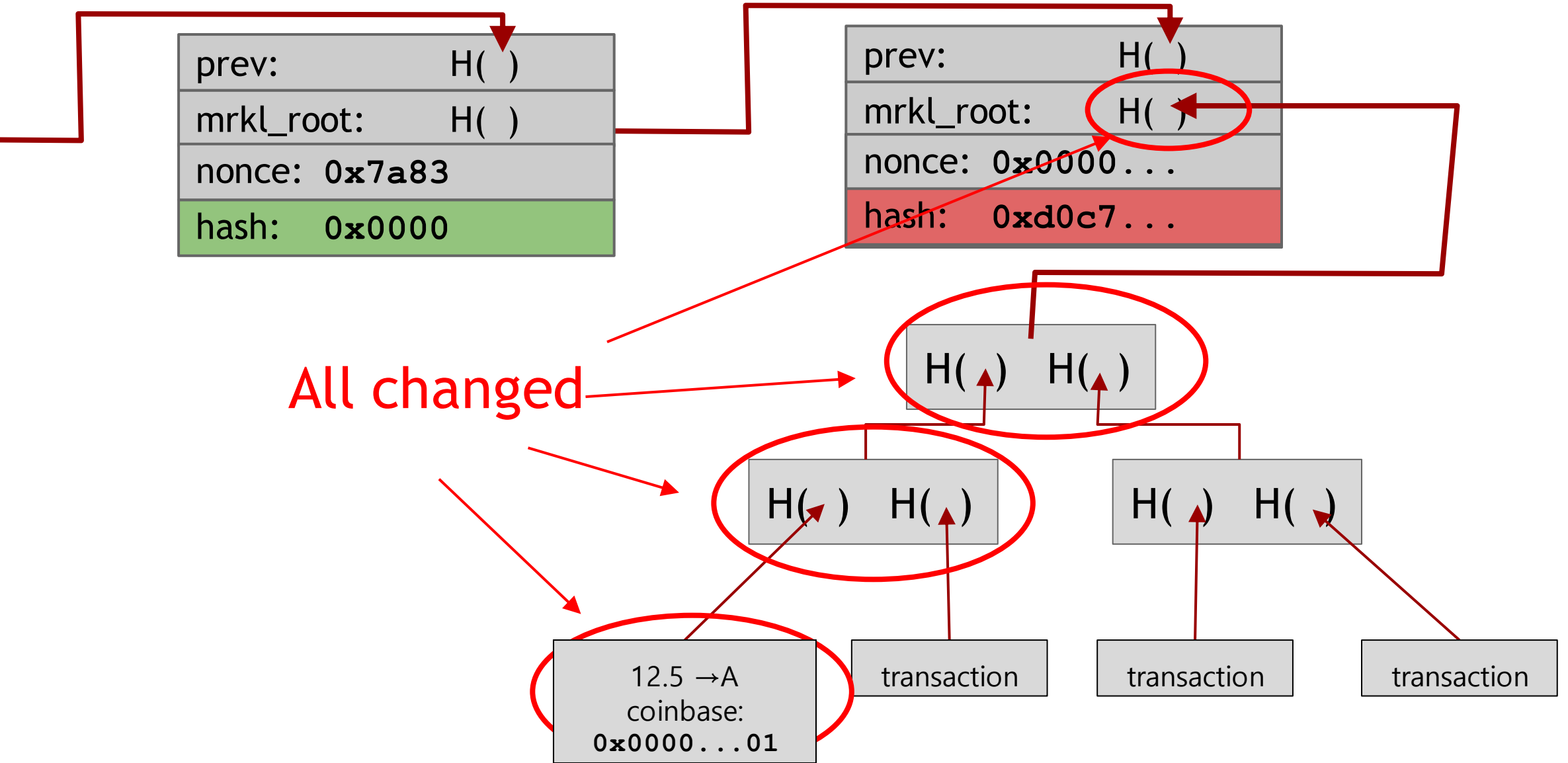
Finding a valid nonce is the hard part!



Finding a valid nonce is the hard part!



Finding a valid nonce is the hard part!



SHA-256 is “puzzle-friendly”

Optimization-free

No better strategy than trying random nonces

Progress-free

You don’t get any closer the more work you do

Parameterizable

Easy to adjust difficulty

9. The rules of Bitcoin

Bitcoin requires 3 layers of “consensus”

Agree on the protocol

Agree on a blockchain to use

Agree that coins are valuable

Agreement on the blockchain


THE TIMES

Eat Out from £5

Israel prepares to send tanks and troops into Gaza

Chancellor on brink of second bailout for banks

99p

Genesis block chosen by Satoshi

Hard-coded into all clients

New genesis block=new coin!

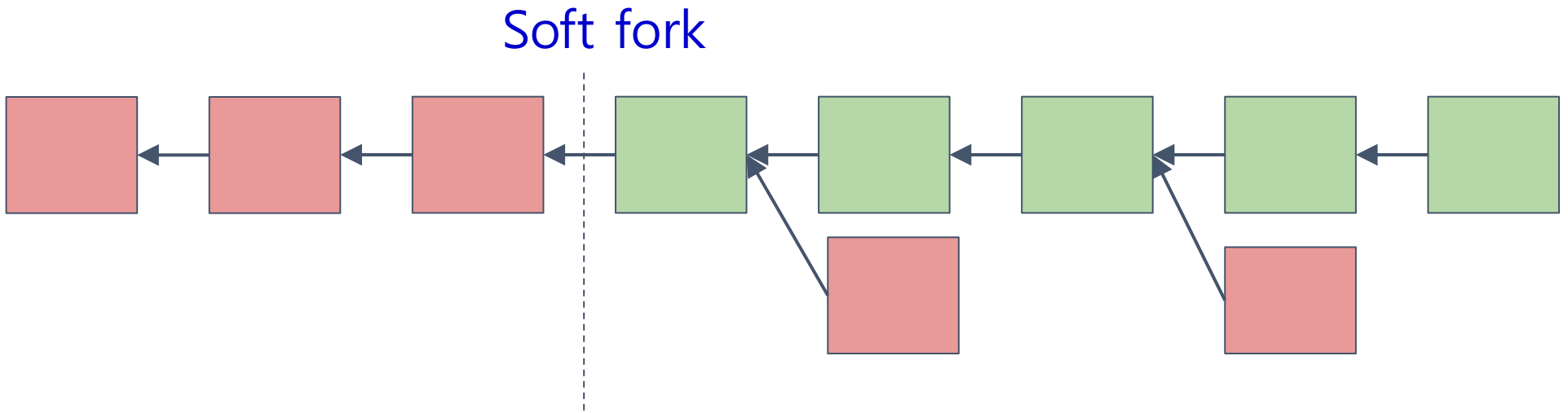
Agreement on the protocol

Originally designed by Satoshi

BUT... can change!

Soft forks

- Backwards-compatible (e.g. block size changes from 1MB to 0.5MB)
- Majority (New rules) agrees to change the protocol
  - validation rules become *stricter* only
- Minority (Old rules) can read new blocks, but their mined blocks are not supported by new-rule nodes



Hard forks

Backwards-incompatible (e.g. block size from 1MB to 2MB)

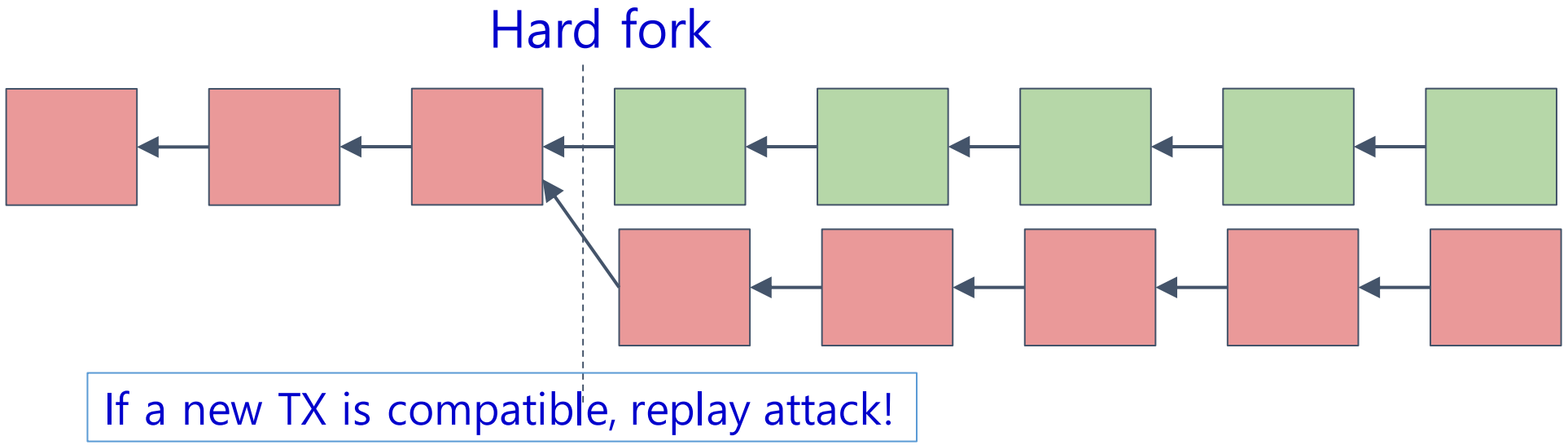
Existing TXs are on both chains

Majority (New Rules) agrees to change the protocol

- They generate new blocks, which are not valid to old nodes

Minority (Old Rules) can't read/write new blocks

- they have their own chain



Altcoin

Alternative coin
Re-writes the rules from scratch
(usually) starts a new genesis block