A3b: 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 (TX) semantics

Bitcoin is *transaction (TX)-based*

- there are no accounts
  - addresses are not accounts
  - there are only TXs
- TXs destroy old coins, create new ones
  - a TX can have multiple inputs and multiple outputs
  - old coins are inputs of a TX
    - Sum of inputs will be divided into outputs
  - new coins are outputs of a TX
  - sum of old coins ≥ sum of new coins
    - sum of inputs ≥ sum of outputs

A transaction-based ledger (Bitcoin)

Transaction identifier (TXID or transID)

special TX without input

time

Create: #1 to Alice (25 coins)
Input: #1 Output: #2 to Bob (17), #3 to Alice (8) SIGNED(ALICE)
Input: #2 Output: #4 to Charlie (8), #5 to Bob (9) SIGNED(BOB)
Input: #3 Output: #6 to David (16), #7 to Alice (2) SIGNED(ALICE)

source: https://docs.google.com/presentation/d/13SOZqHzLRryyFXw26CCE89lKMa8X9YAhdTFgXdw0a1s/view#slide=id.p

A transaction-based ledger (Bitcoin)

time

Create: #1 to Alice (25 coins)

Input: #1

Output: #2 to Bob (17), #3 to Alice (8)

A change address

SIGNED(ALICE)

Input: #2

Output: #4 to Charlie (8), #5 to Bob (9)

SIGNED(BOB)

Input: #3

Output: #6 to David (16), #7 to Alice (2)

SIGNED(ALICE)

A transaction-based ledger (Bitcoin)

UTXO: unspent TX output

time

Create: #1 to Alice (25 coins)
Input: #1 Output: #2 to Bob (17), #3 to Alice (8) SIGNED(ALICE)
Input: #2 Output: #4 to Charlie (8), #5 to Bob (9) SIGNED(BOB)
Input: #3 Output: #6 to David (16), #7 to Alice (2) SIGNED(ALICE)

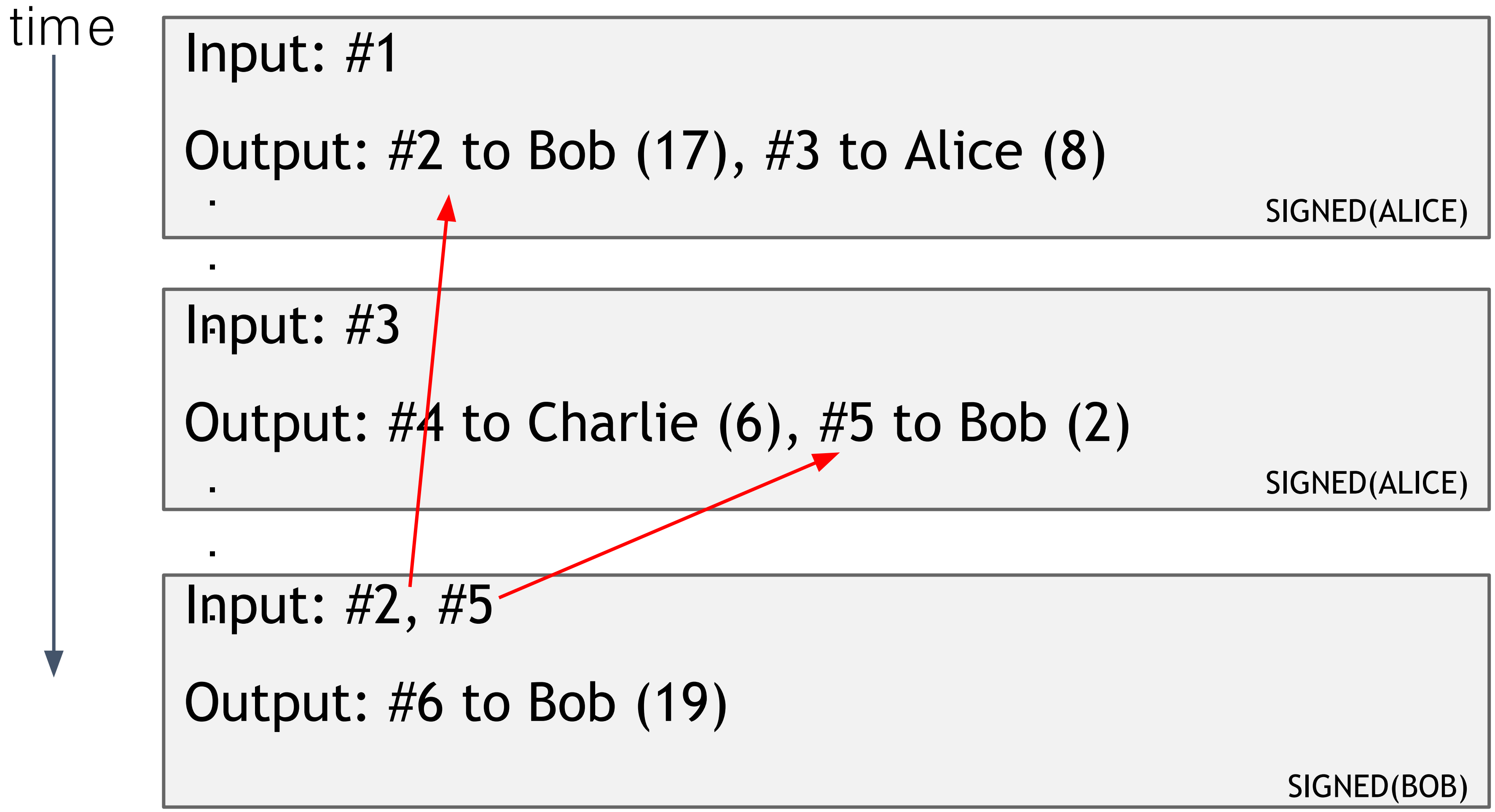
follow the hash pointers

is this valid?

OPTIMIZATION: Store all valid UTXOs

Merging value:

A TX can have multiple inputs



We need a script language to process TXs

- A transaction output doesn’t just specify an address but a script saying “this output will be redeemed by a public key that hashes to address X, along with a signature from the owner of that public key.”
  - ScriptPubKey: locking script
- Later on, a transaction input then needs to specify a script saying “Here is the public key and the signature of the recipient (whose address is X)”
  - ScriptSig: unlocking script
- Both scripts are simply concatenated (the script for the input part of the current TX and the script for the output in the corresponding prior TX), which must run successfully in order for the current TX to be valid

what to specify for input and output of a TX

- A TX can have multiple inputs and multiple outputs
- An Input specifies
  - which output of which prior TX will be used
  - owner's Signature & Public Key
    - i.e., recipient of the prior TX's output
- An Output specifies
  - How much amount
  - Who (or whose address) will receive this money, and how he will "unlock" the received money

TXs: an illustration

Input: …

TX 683

Output[0]: 2.3 coin, to address X1; X1's PubKey/signature needed to redeem money

Output[1]: 1.1 coin, to address X2; X2's PubKey/signature needed to redeem money

…

Input: TX683 output[0], X1's PubKey, X1's signature

TX 320

Output[0]: 1.3 coin, to address X3; X3's PubKey/signature needed to redeem money

Output[1]: 1.0 coin, to address X1; X1's PubKey/signature needed to redeem money

time

X1 sends 1.3 coin (out of 2.3 coin received) to X3

A realistic Bitcoin TX

this TX will be valid after lock_time

TXID

hash

metadata

```
{
  "hash":"5a42…8b6b",
  "ver":1,
  "tx_in count":1,
  "tx_out count":1,
  "lock_time":0,

  "in":[
    {
      "prev_out":{ "hash":"3be4…0260", "n":0 },
      "scriptSig":"30b6…14de 54ab…de81"
    },
  ],

  "out":[
    {
      "value":"5.12",
      "scriptPubKey":"OP_DUP OP_HASH160 69e0…d42e OP_EQUALVERIFY OP_CHECKSIG"
    }
  ]
}
```

number of inputs of this TX

number of outputs of this TX

input(s)

1st output of the prior TX 3be4…

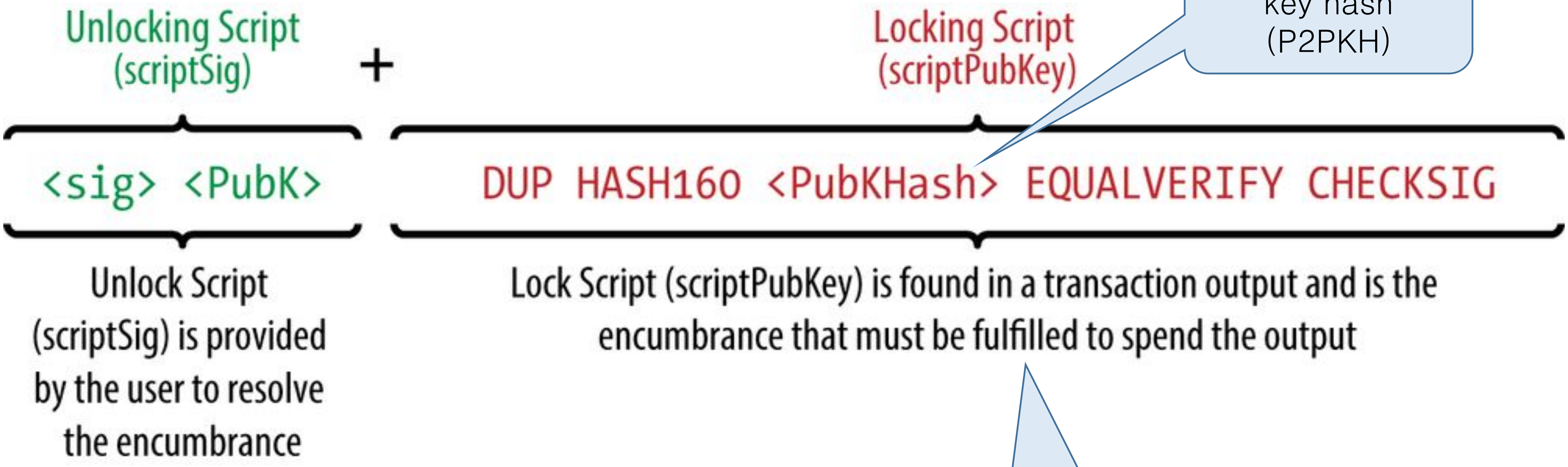
Unlocking script: signature +PubKey

output(s)

Locking script

How to process a TX

Pay to public key hash (P2PKH)

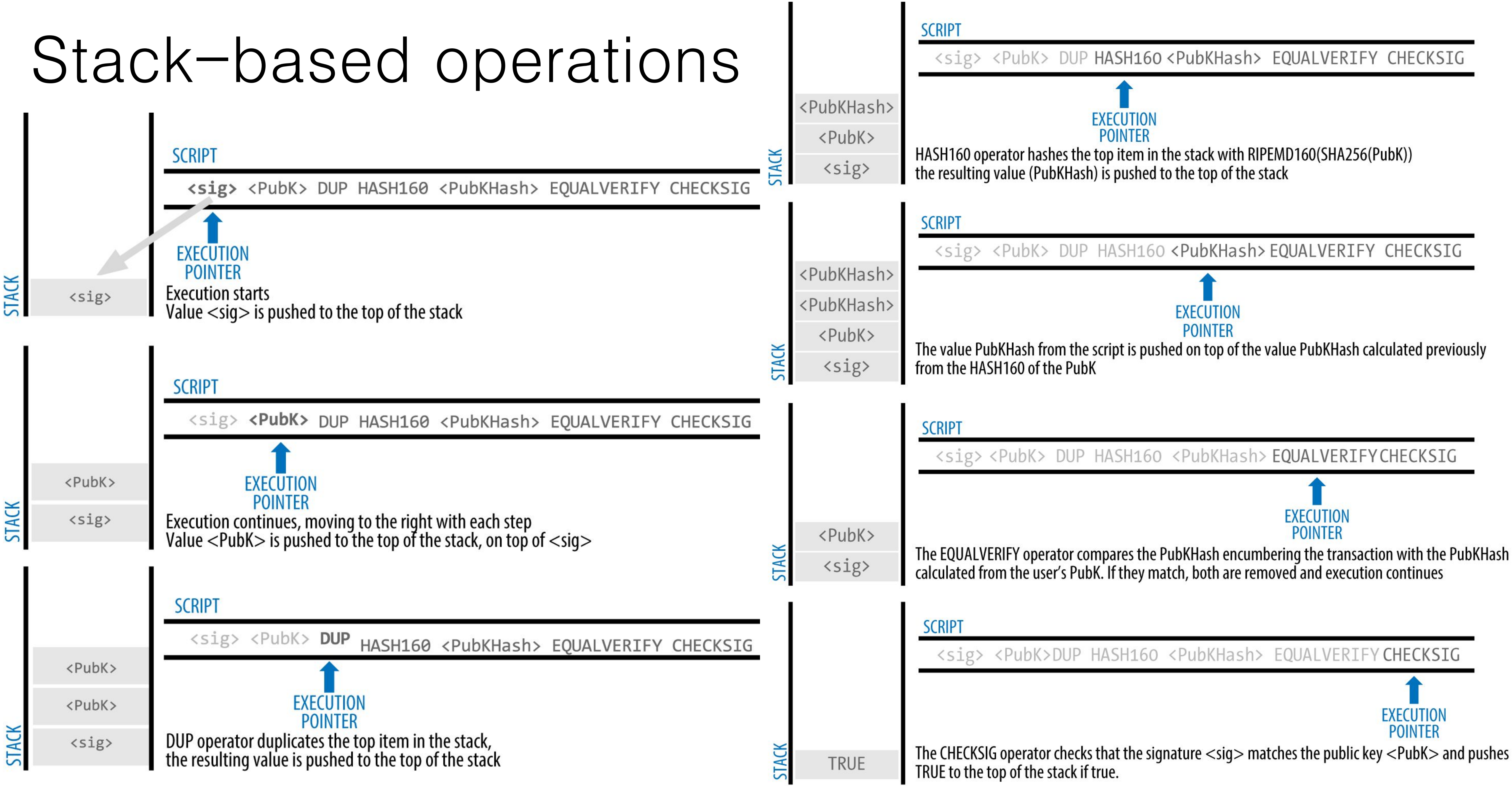


Input in the current TX

Output in the prior TX

Source: apprize.info

Stack-based operations



Source:

Transaction validation: an illustration

Source:
softblocks.co

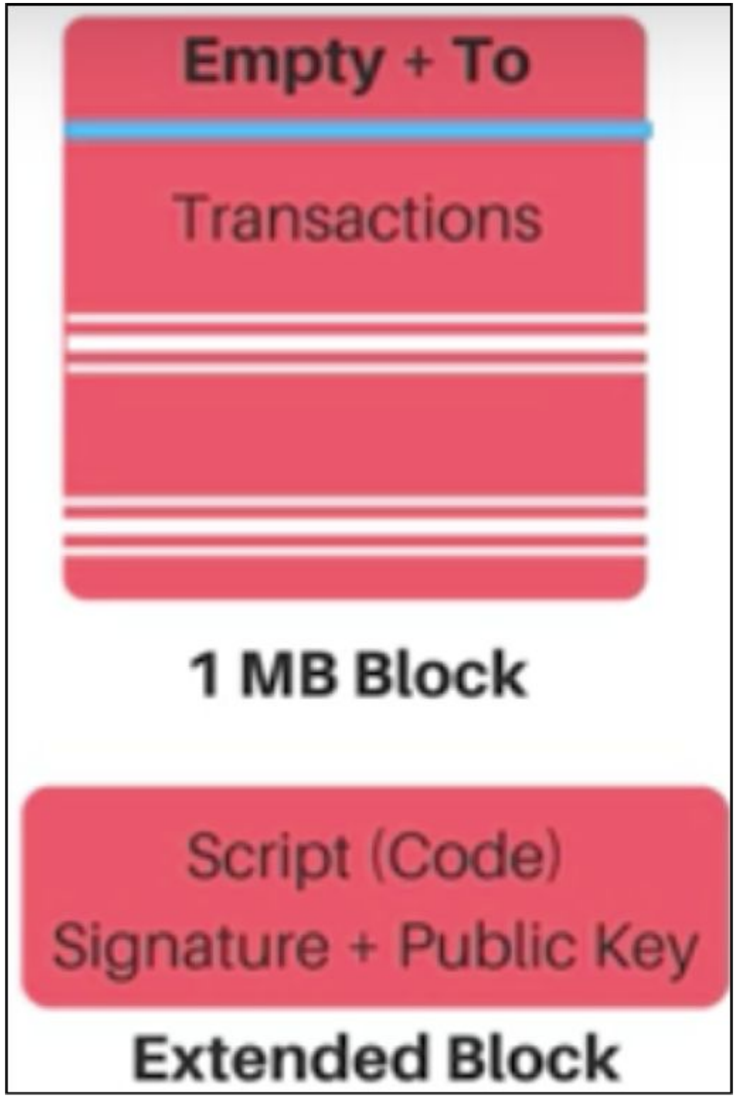
Bitcoin scripting language (“Script”)

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

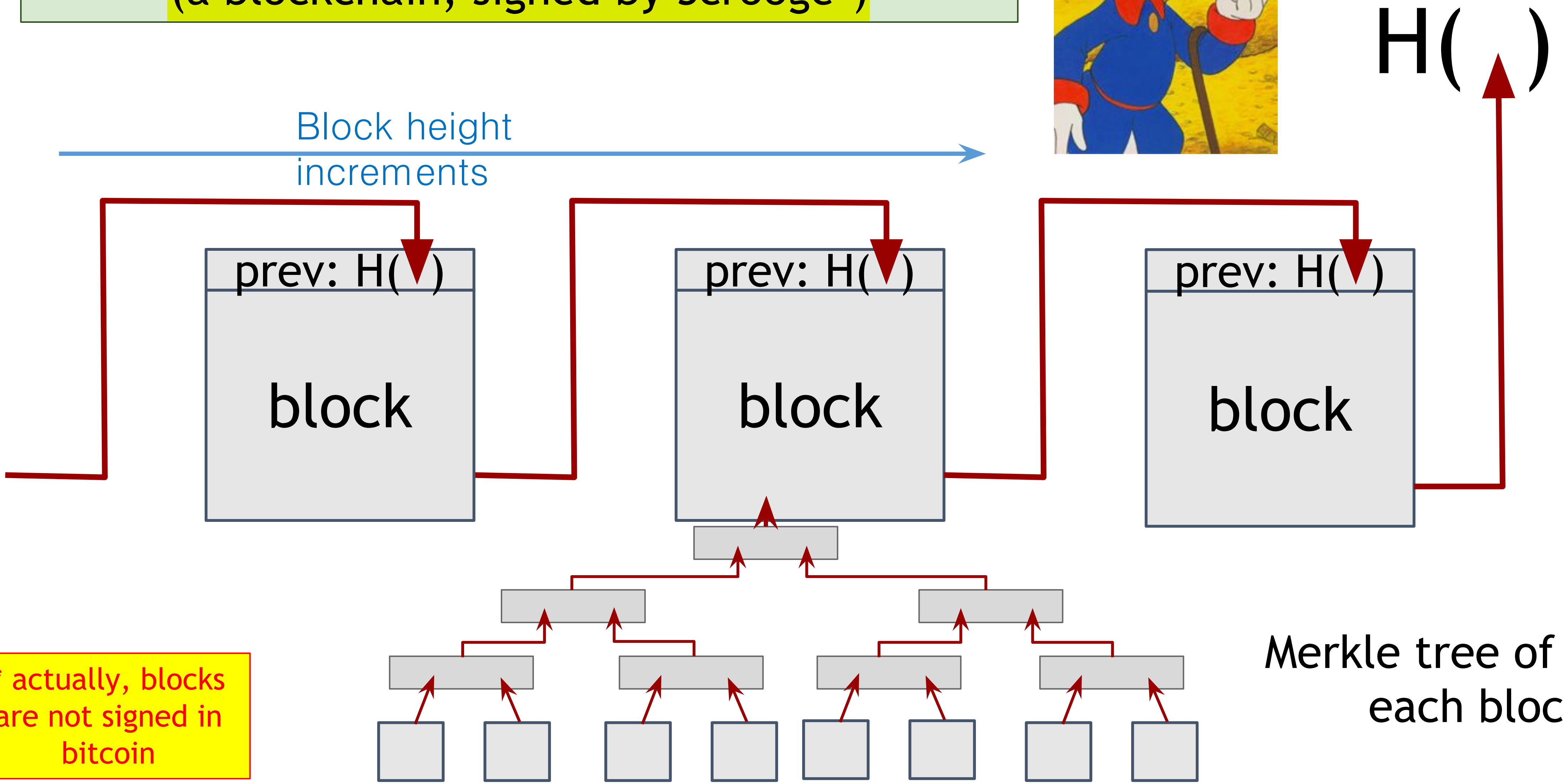


Source: blockgeeks.com

Centralized ledger (ScroogeCoin)

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

signed by $pk_{Scrooge}$



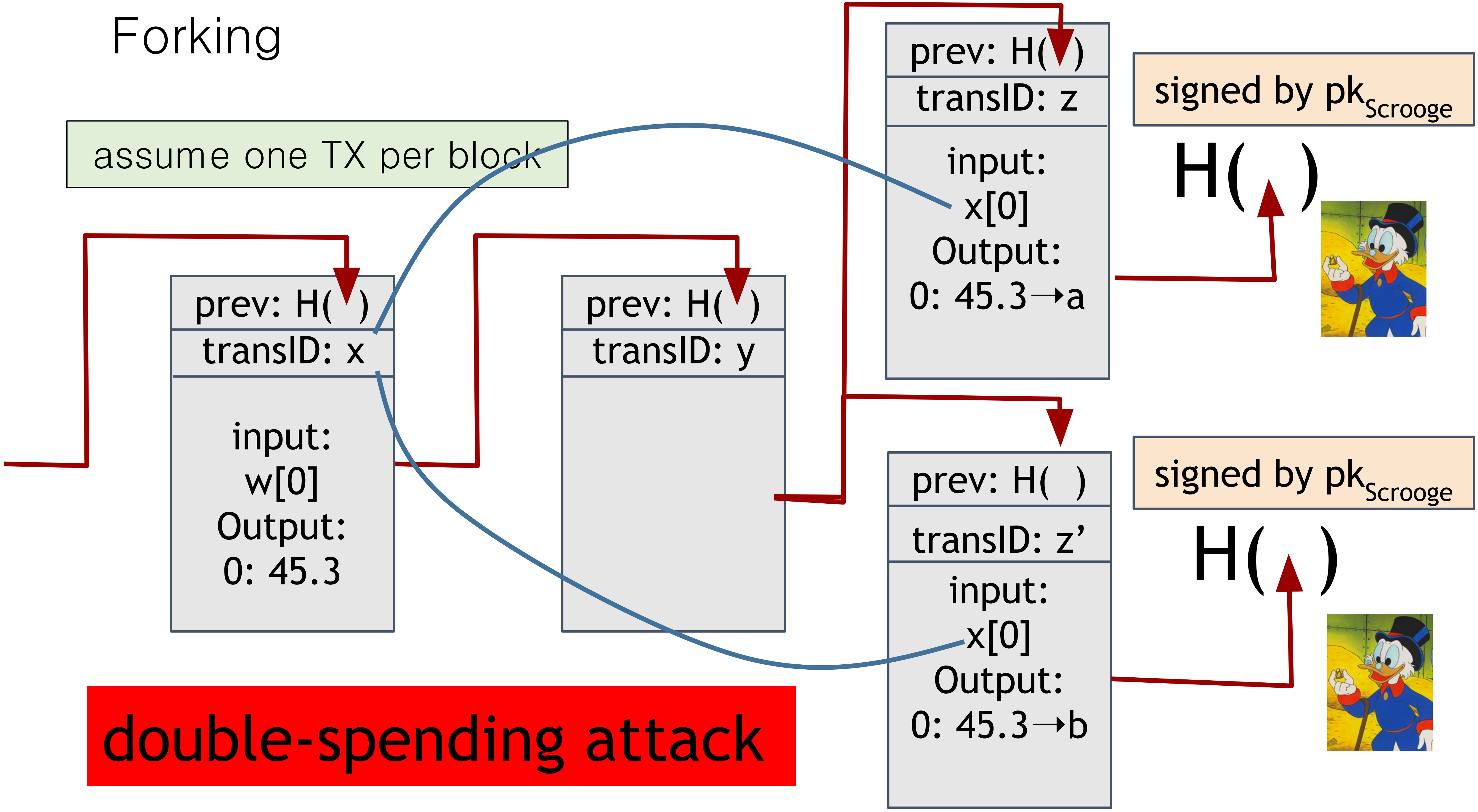
Merkle tree of TXs in each block

* actually, blocks are not signed in bitcoin

Don’t worry, I’m honest.

What if Scrooge is malicious?

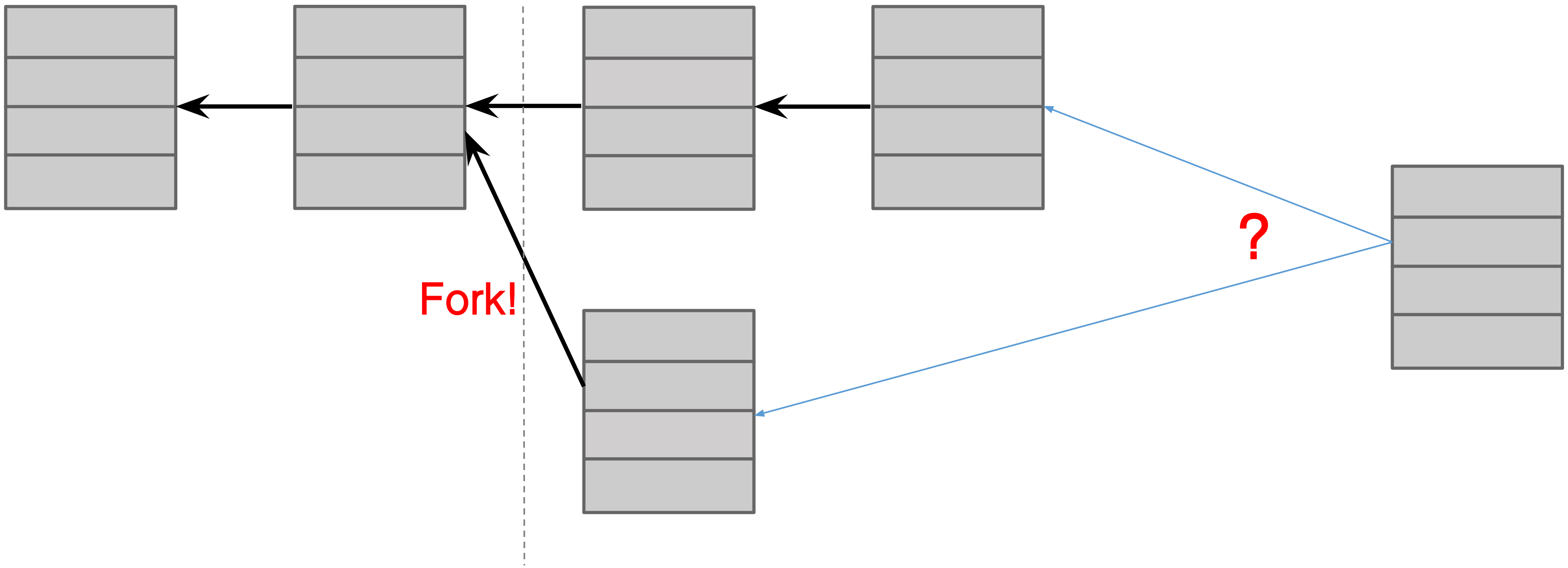
Forking



double-spending attack

how to resolve if a fork happens?

- Only one path should be taken
- In case of multiple paths (or branches)
  - Miner of the next block will try to extend the longer path



Other Scrooge problems

- Blacklist addresses
- Demand transaction fees
- Go offline
- Get hacked
  - e.g. his private key is leaked

Decentralization

We wish to avoid vulnerability to misbehavior/failure of a centralized entity

But how?

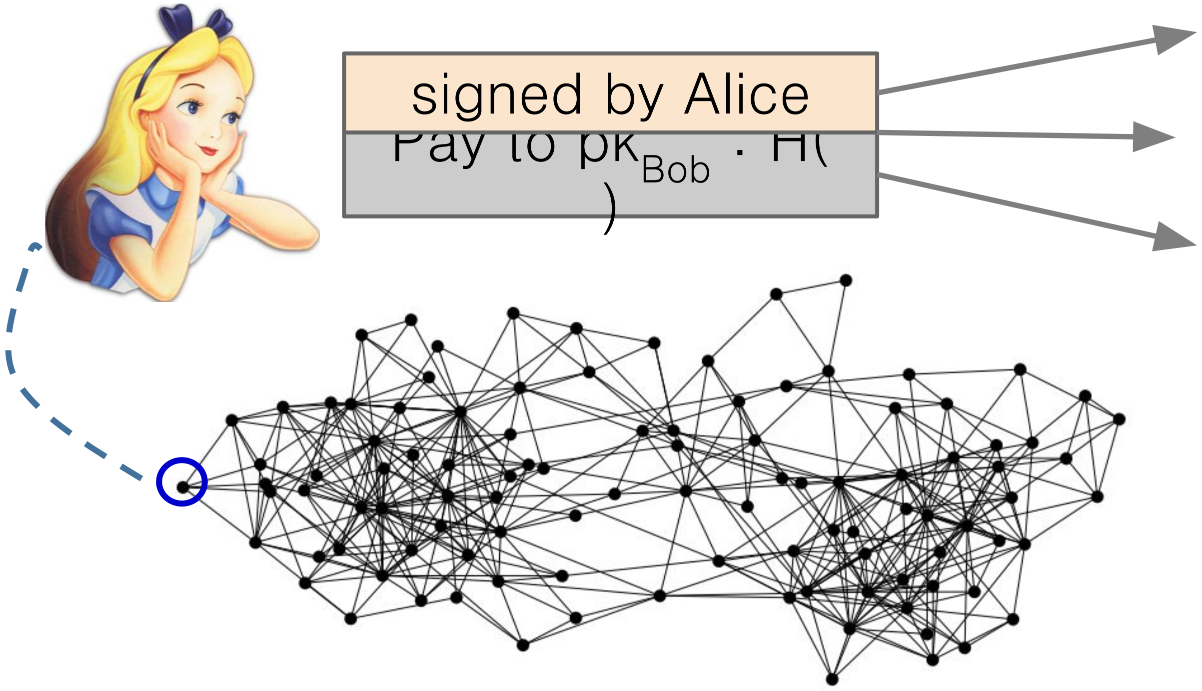
6. A decentralized ledger: Bitcoin

Two different words

- Decentralized
  - No central point
  - Power/control is split
  - Physical distances not a primary concern
  - Nodes are usually scattered
  - Examples: anarchy, Occupy Wall Street, P2P systems
- Distributed
  - Multiple nodes are scattered
  - Physical distances matter
  - Control/power is not of concern
  - Control/power can be centralized or decentralized
  - Examples: Internet, telephone networks, Airline reservation systems

Bitcoin is a peer-to-peer system

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



goal: all nodes must agree on a sequence of TXs

source: https://docs.google.com/presentation/d/13SOZqHzLRryyFXw26CCE89lKMa8X9YAhdTFgXdw0a1s/view#slide=id.p

Bitcoin consensus (simplified)

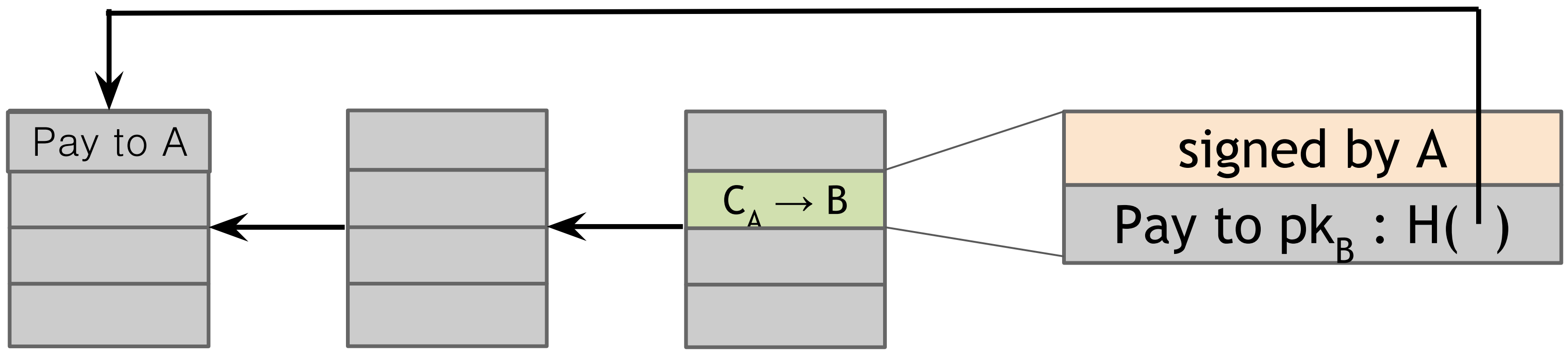
previous vs prior

1. Transactions are broadcast to all nodes
   - invalid TXs are ignored
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
   - Invalid blocks are ignored
4. If more than one nodes broadcast (different) blocks with the same height, the next node will choose one of them
5. Longest chain is considered canonical

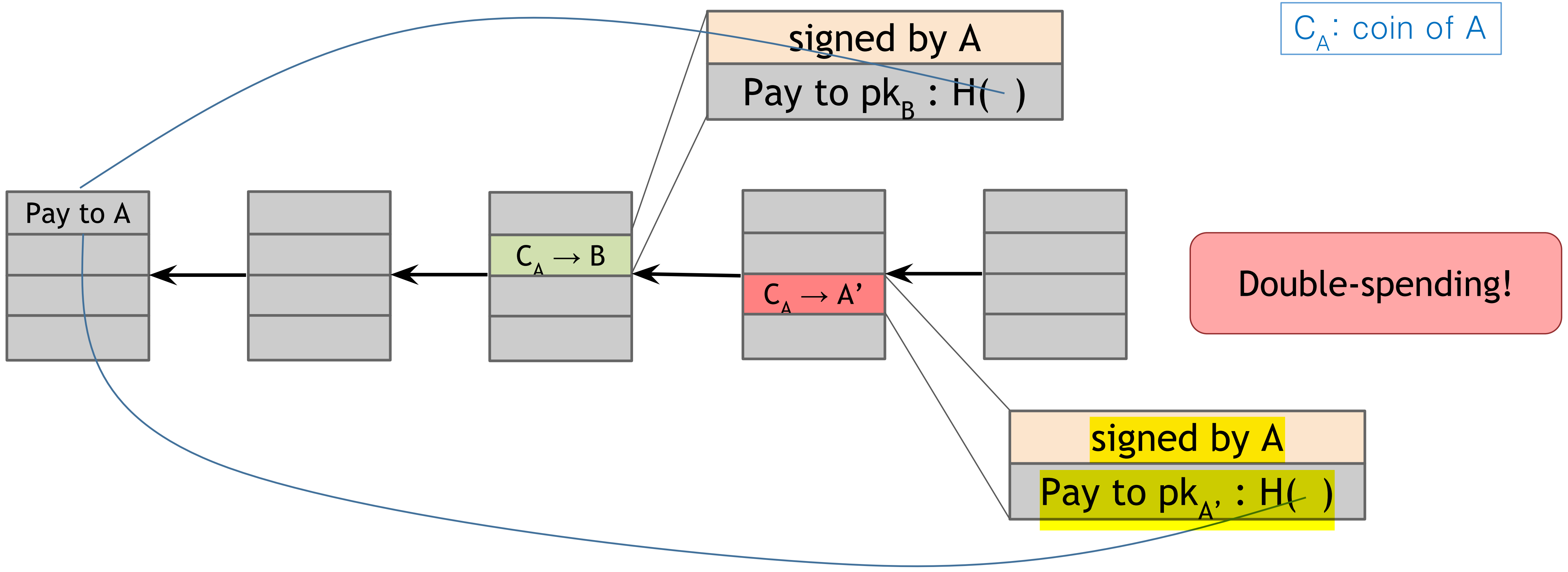
Leads * how to select or elect a random node? ith “h ** actually, blocks are not signed

What can a malicious node do?

C_A: coin of A



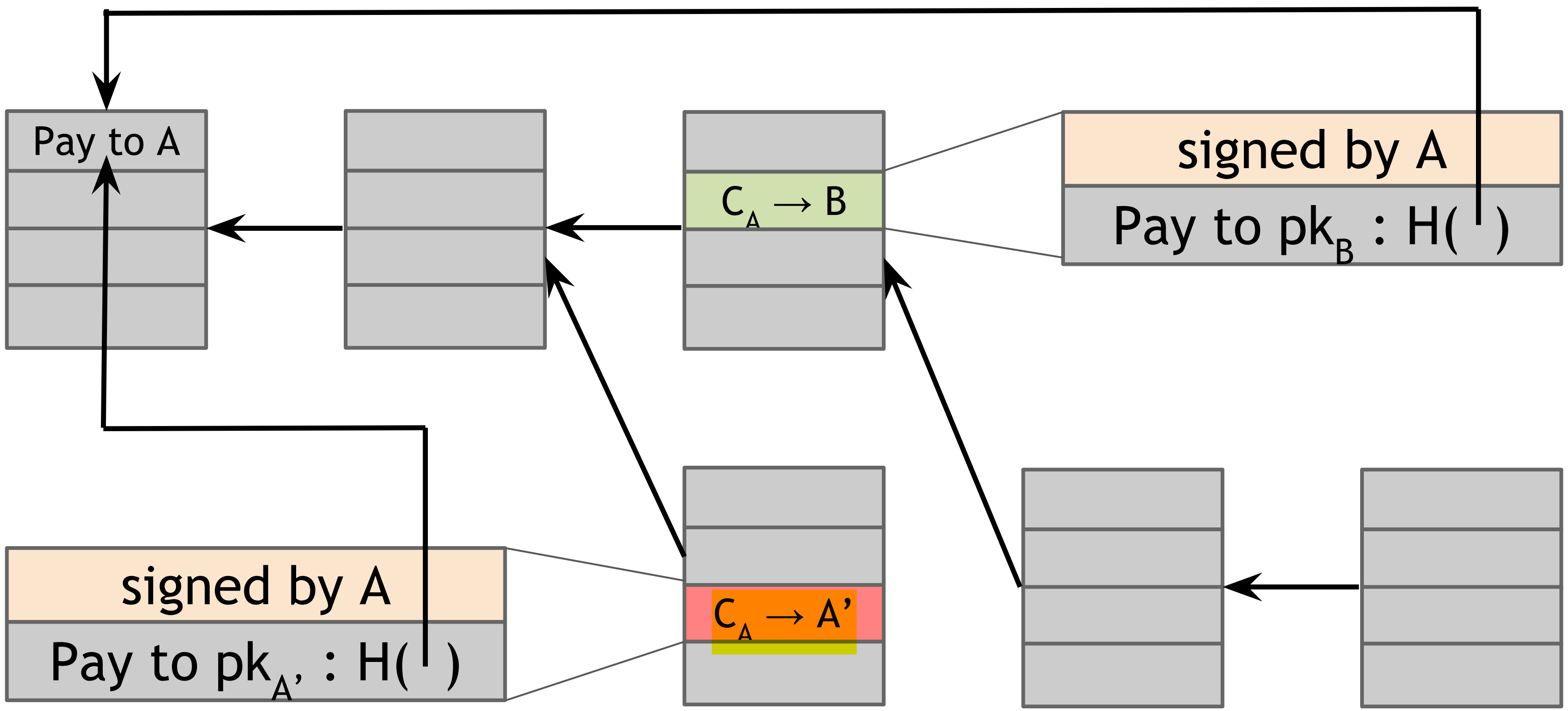
What can a malicious node do?



is this possible?

What can a malicious node do?

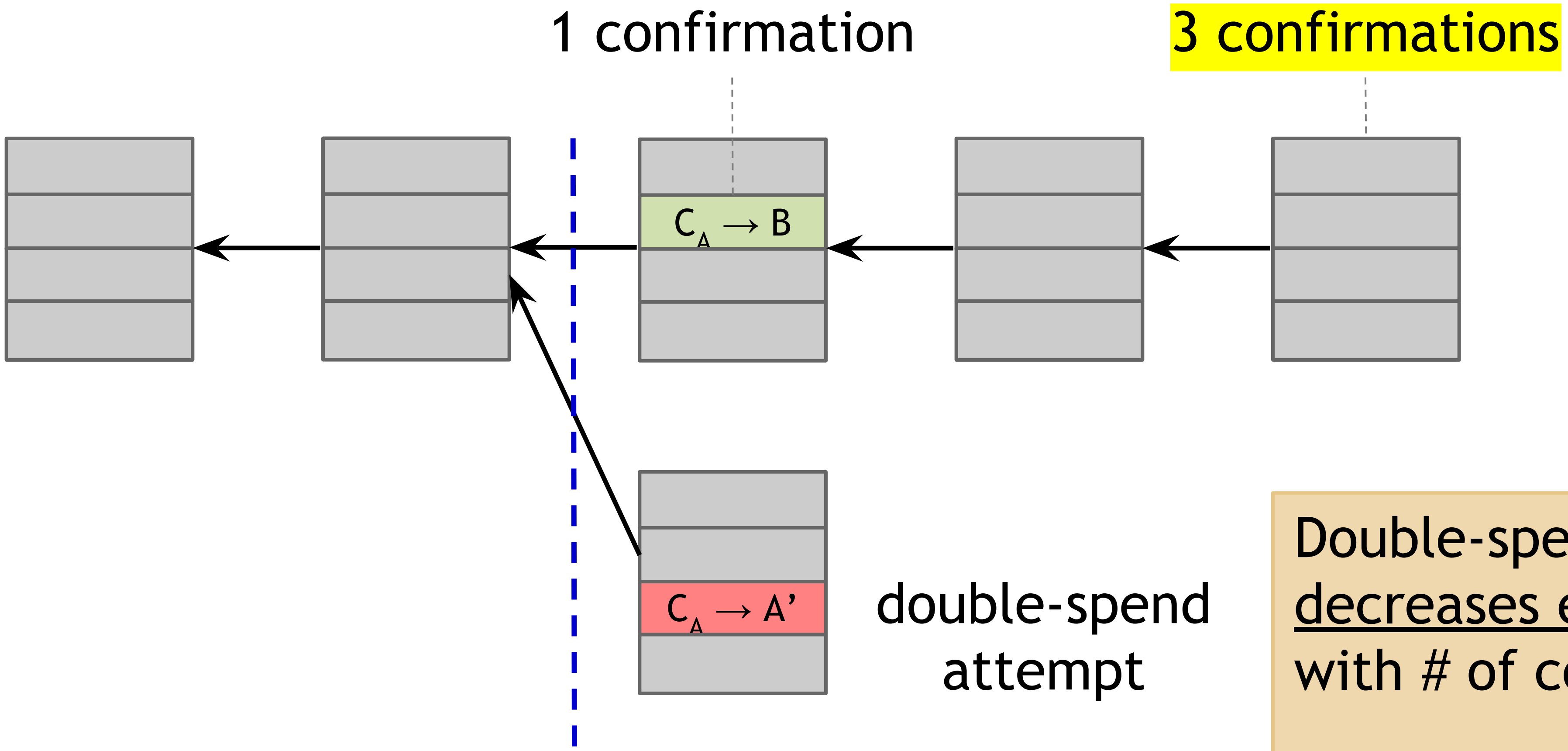
C_A: coin of A



Double-spend

Honest nodes will extend the <u>longest valid branch</u>

From a merchant B's point of view



Hear about $C_A \rightarrow B$ transaction:
0 confirmations

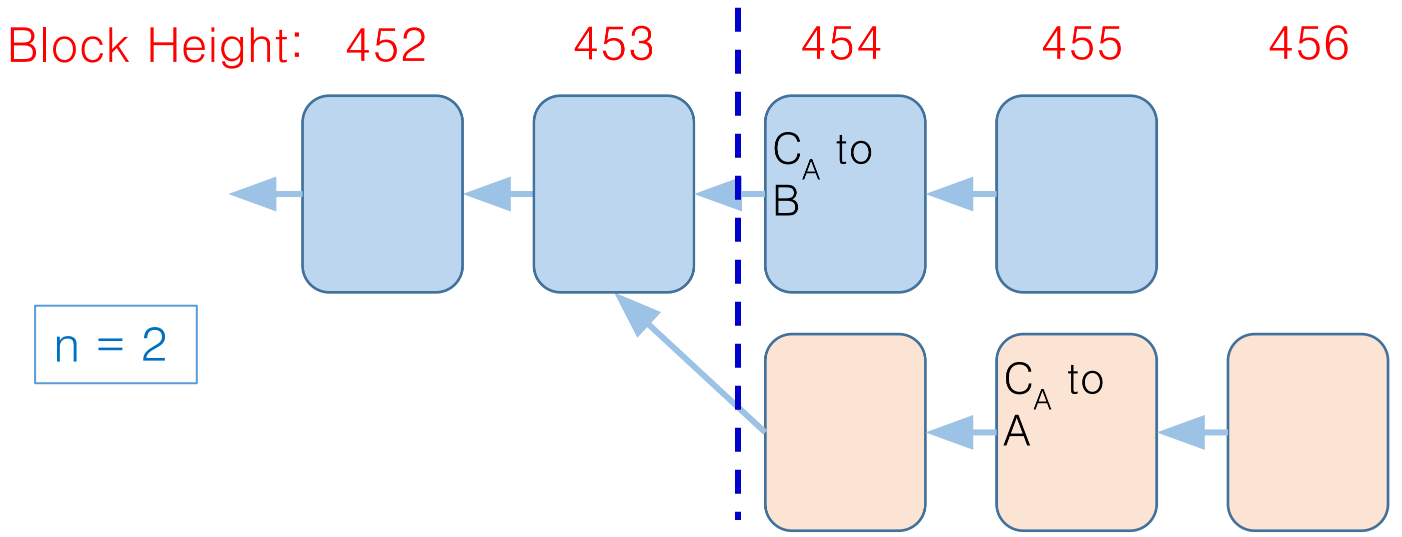
Double-spend probability <u>decreases exponentially</u> with # of confirmations

Most common heuristic:
6 confirmations

source: https://docs.google.com/presentation/d/13SOZqHzLRryyFXw26CCE89lKMa8X9YAhdTFgXdw0a1s/view#slide=id.p

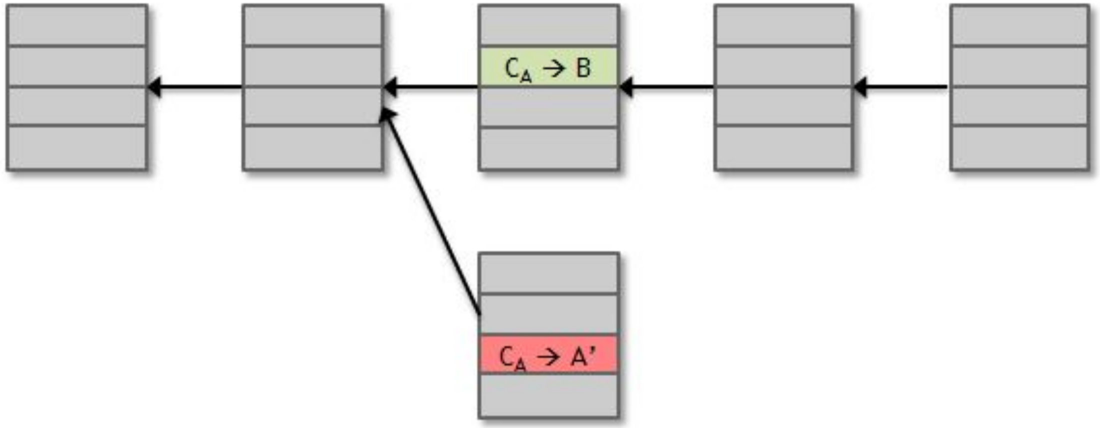
51% (or majority) attack

- The attacker submits to the merchant/network a transaction which pays the merchant
- He has also been 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



Then other miners will join the lower branch

Basic properties



- Protection against invalid transactions is cryptographic
  - Signature will not be valid (unforgeability)
- Protection against double-spending relies on consensus
  - We assume at least 51% will be honest
- You're never 100% sure a transaction is in the blockchain
  - i.e. the longest path includes the TX

how to reach a consensus?

- ordering of TXs is important
  - TX propagation is different for individual participants
- getting the votes of a majority of participants
- issues?

when voting is needed?

who initiates voting?

who counts votes?

who are participants?

a fork happens

anyone who sees a fork

everybody

some participants offline

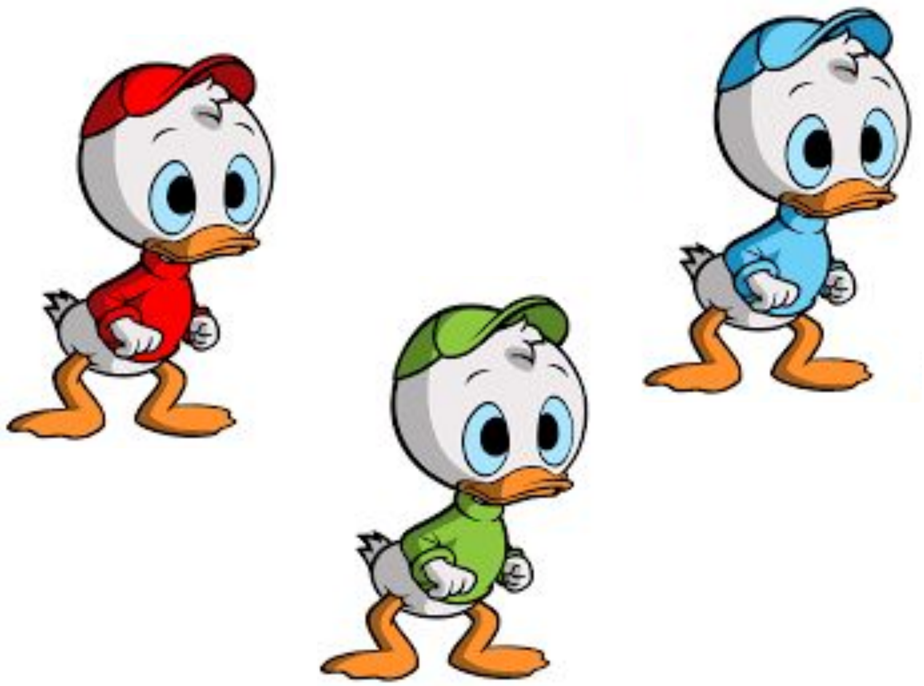
number of participants is continuously changing

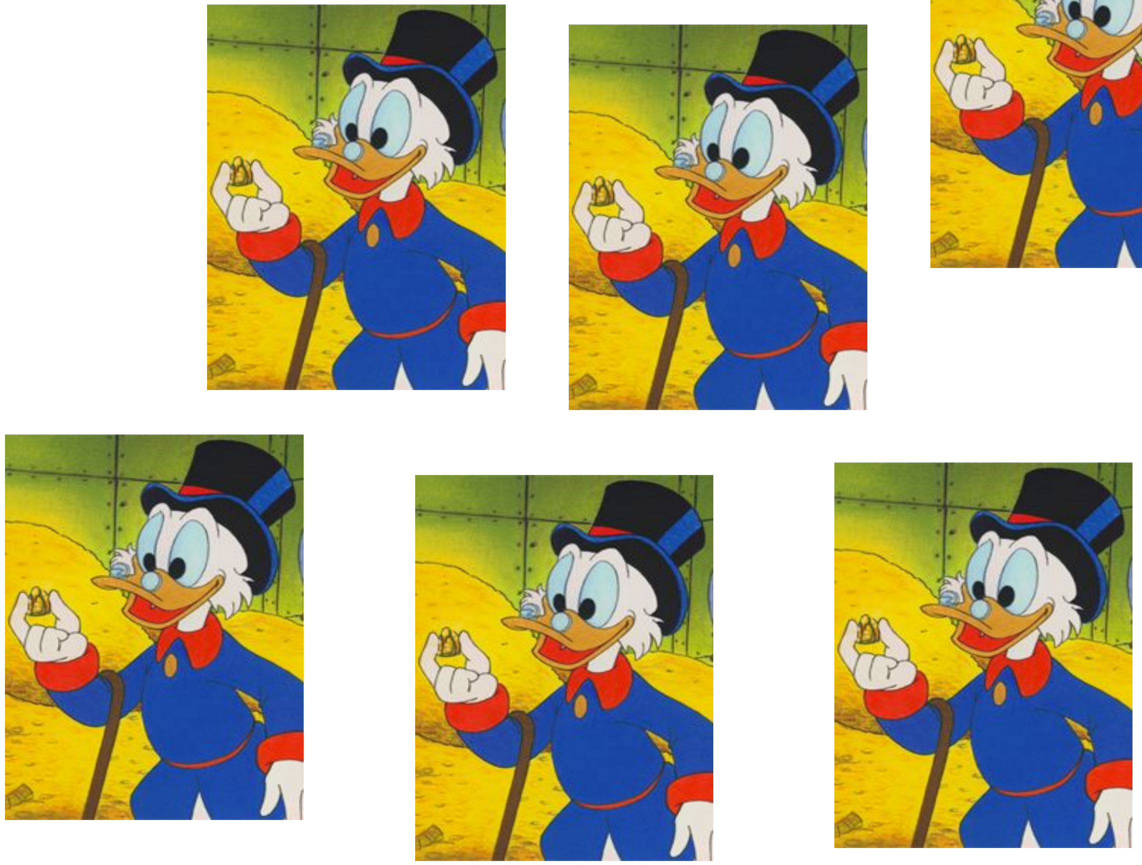
eligibility of voting

what else?

anyone with a PK?

Honest majority of whom?





Recall: addresses can be freely created

Solution: "vote" by CPU power

a random* node constructs a block

<u>Bitcoin mining puzzle:</u>

Given previous block prev, new block curr:

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

d is a difficulty parameter

First solution wins

Puzzle friendliness
H(id || x)

Miners are rewarded for solutions

Creator of a block gets to

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

“Block reward” currently 6.25 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 (PoW)

Miners rewarded by creating new bitcoins

Bitcoin miners

- Bitcoin depends on miners to:
    - Receive and store TXs
        - Validate incoming TXs, which form a Merkle tree
    - Construct blocks and solve puzzles
    - Vote (by hash power) on consensus
        - Solving puzzles is kind of voting
    - broadcast “blocks” in the blockchain
- Miners rewarded with new coins and TX fees

what each miner does iteratively

0. Joins the network
1. listens for transactions
   - Validates all proposed transactions
2. Listens for new blocks from other miners
2. meanwhile, constructs his own block out of TXs
   - Finds a nonce to make his block valid
3. If a new block (from others) is broadcast, validates and accepts it
   - Gives up solving current puzzle, go to 2
3. If nonce is found, broadcasts his own block
4. Hopes everybody accepts his block
5. Profits!

very hard

Energy consuming

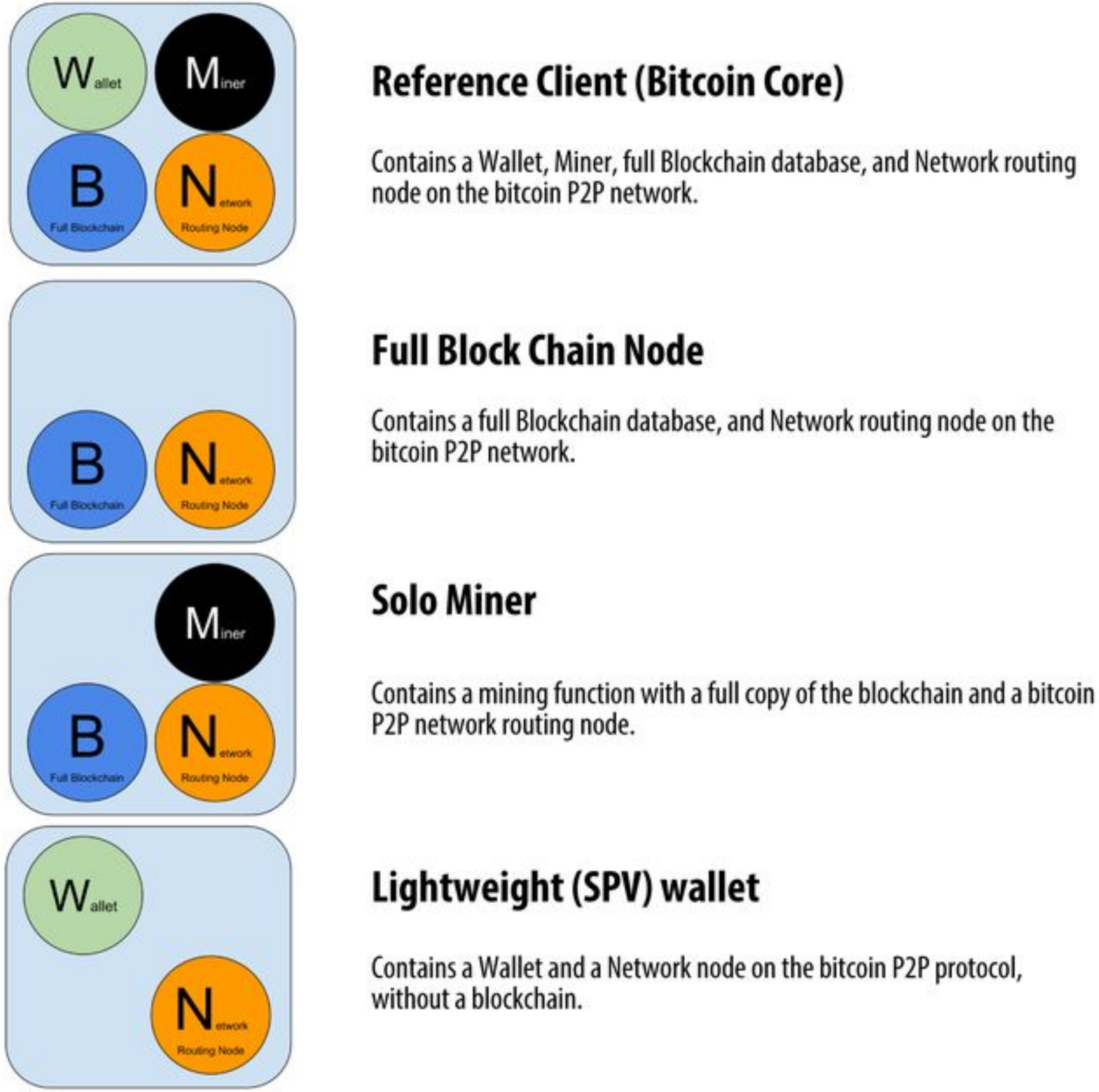
7. Bitcoin P2P networking

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.

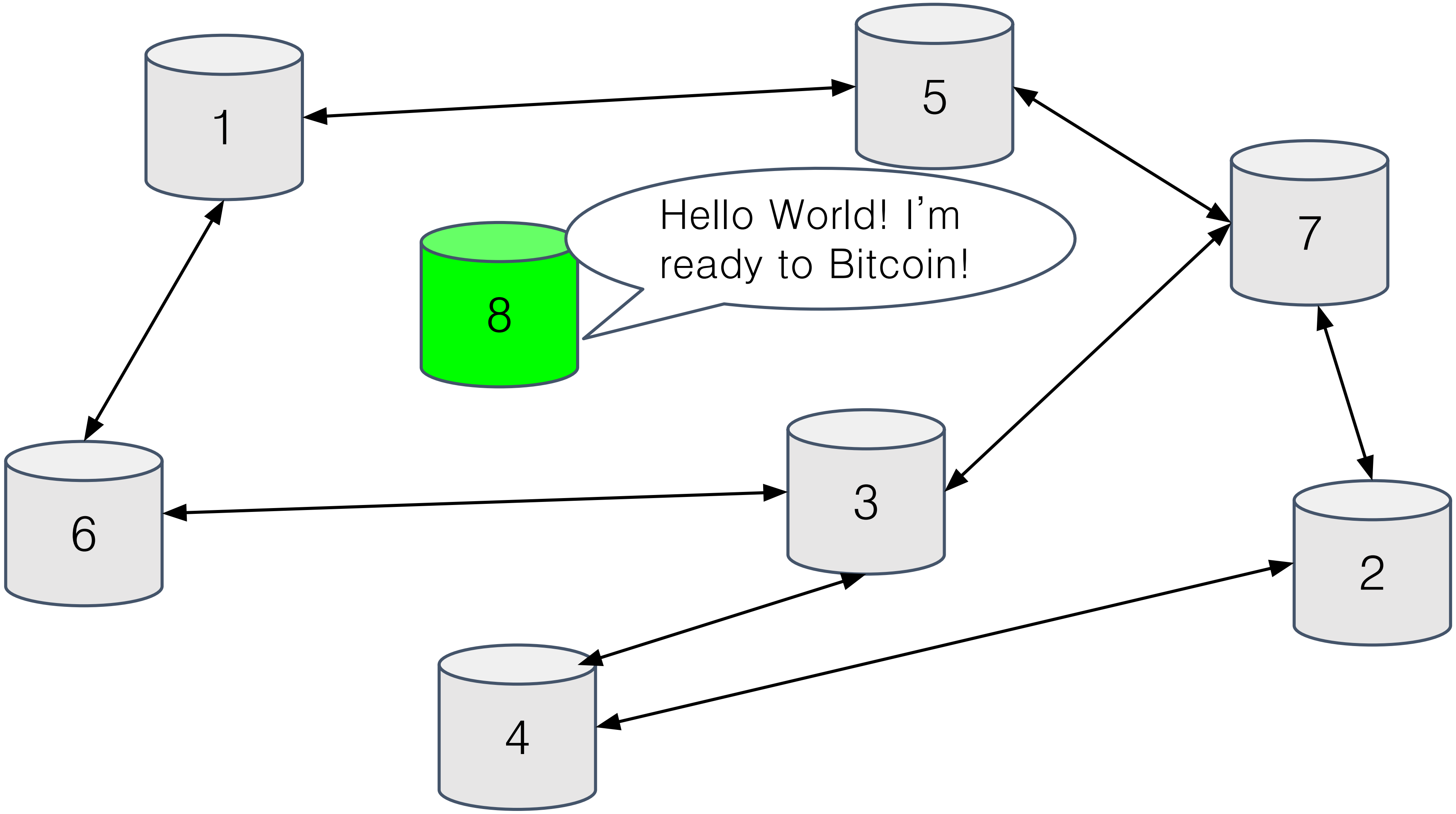
P2P networks are inherently resilient, decentralized, and open.



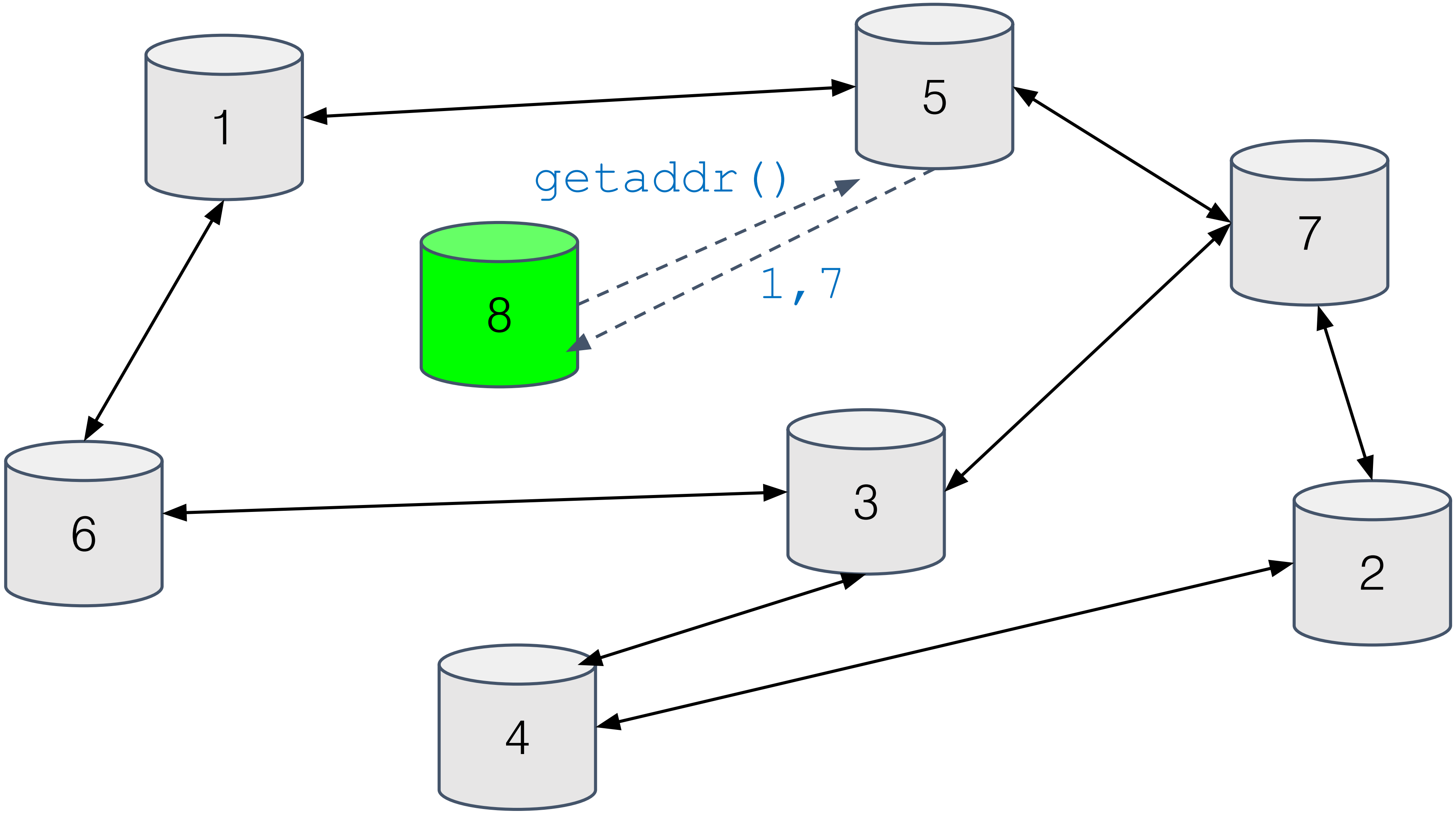
*SPV: simplified payment verification

Source: O'Reilly Media

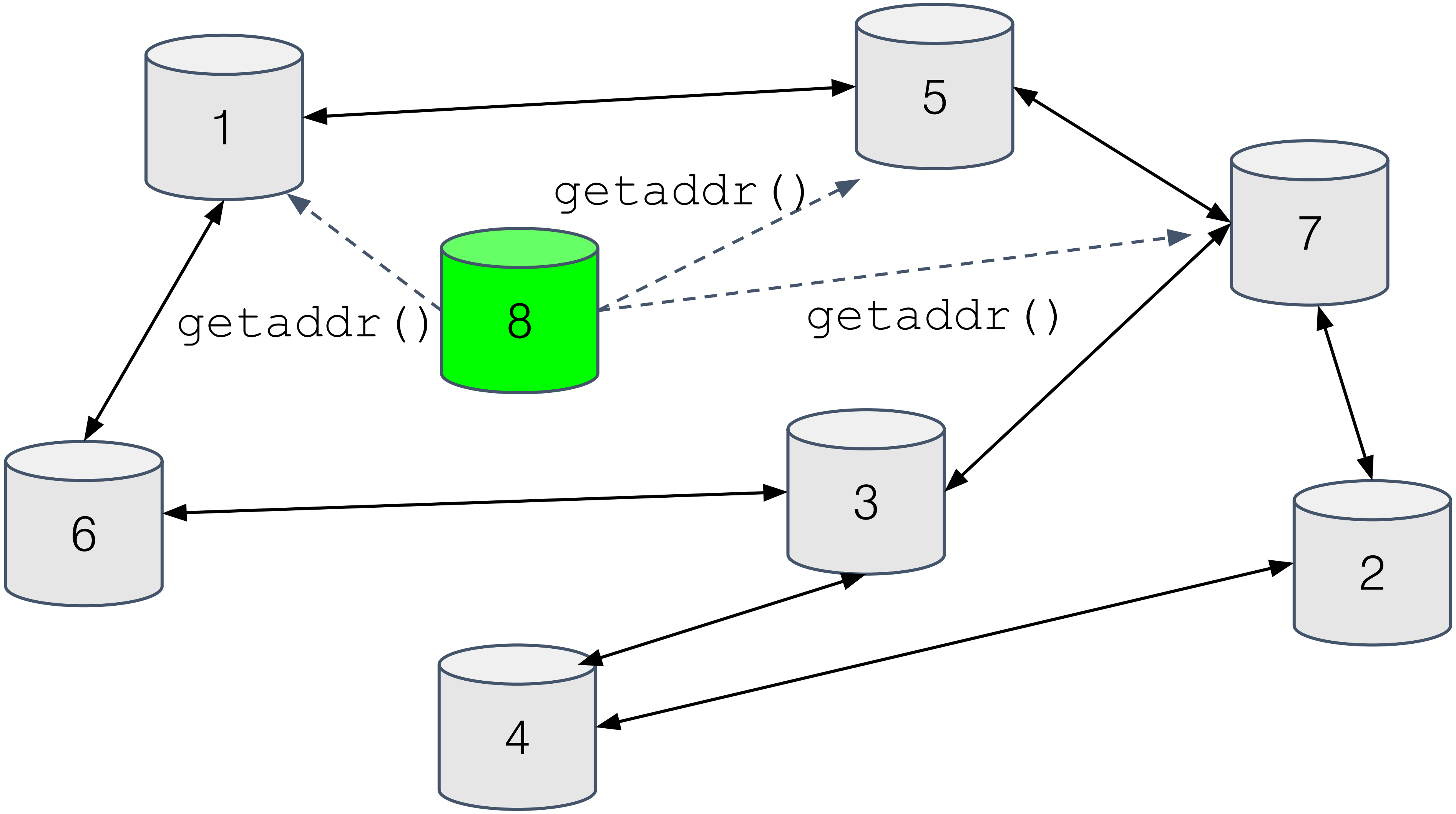
Joining the Bitcoin P2P network



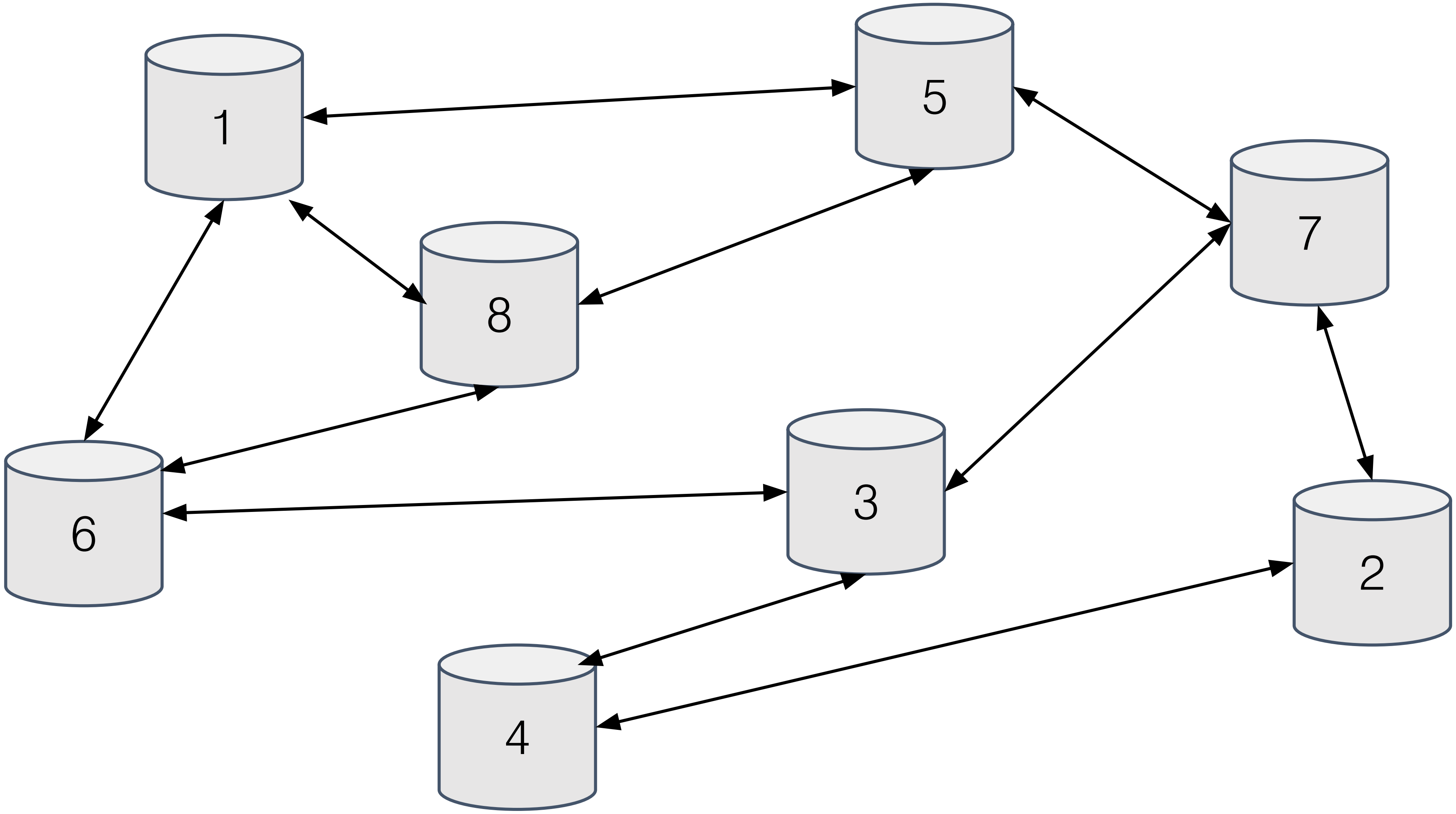
Joining the Bitcoin P2P network



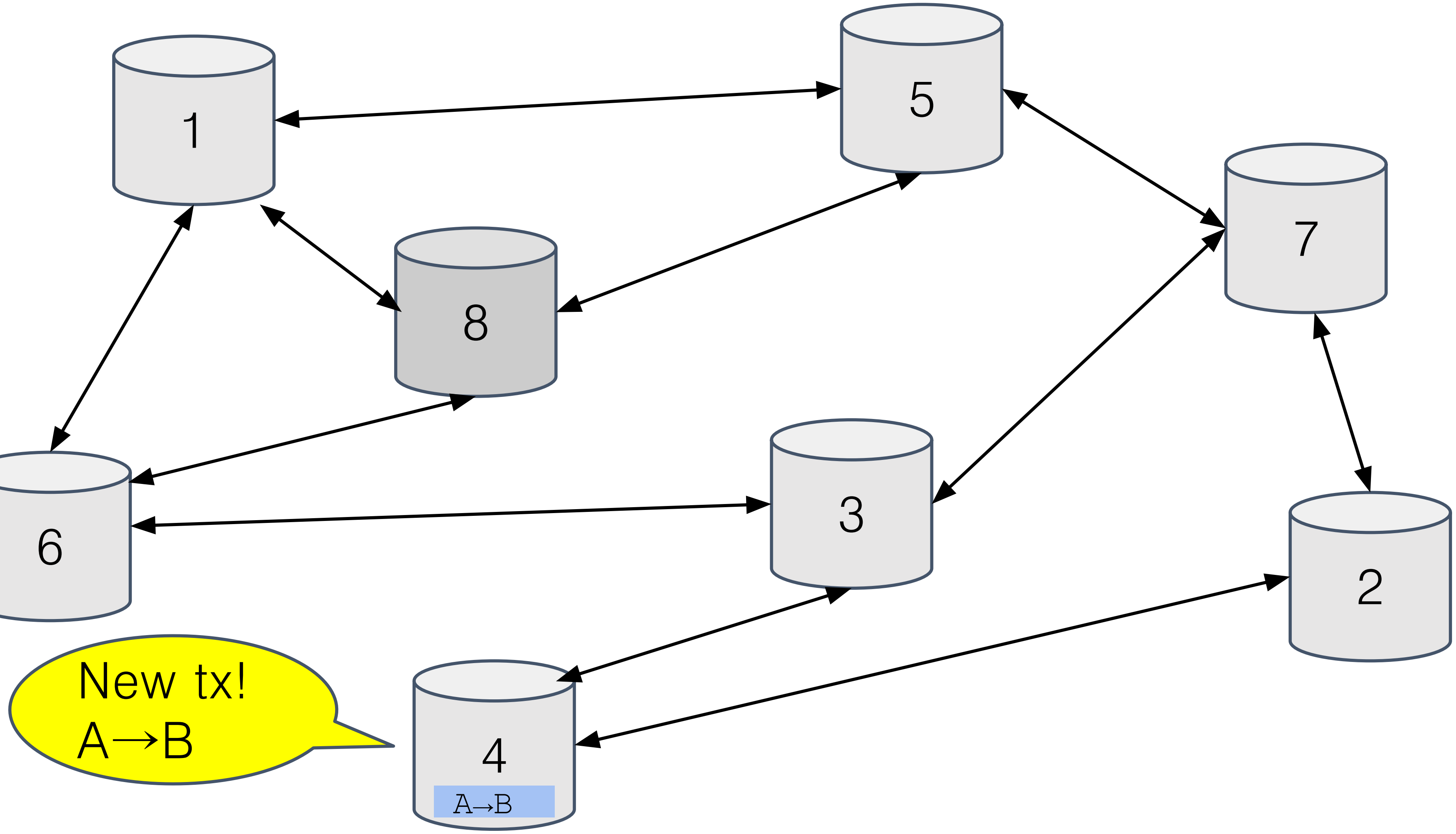
Joining the Bitcoin P2P network



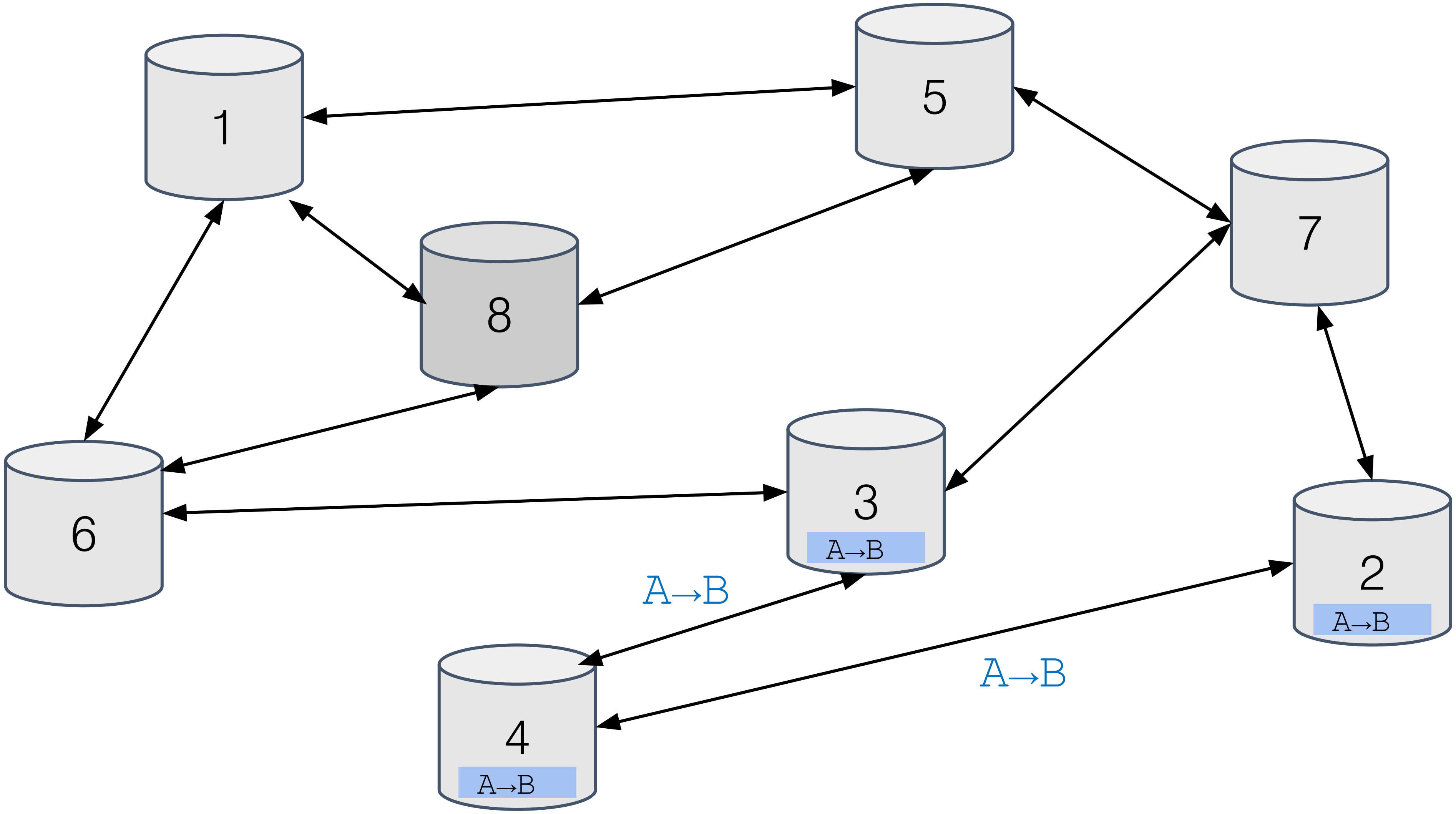
Joining the Bitcoin P2P network



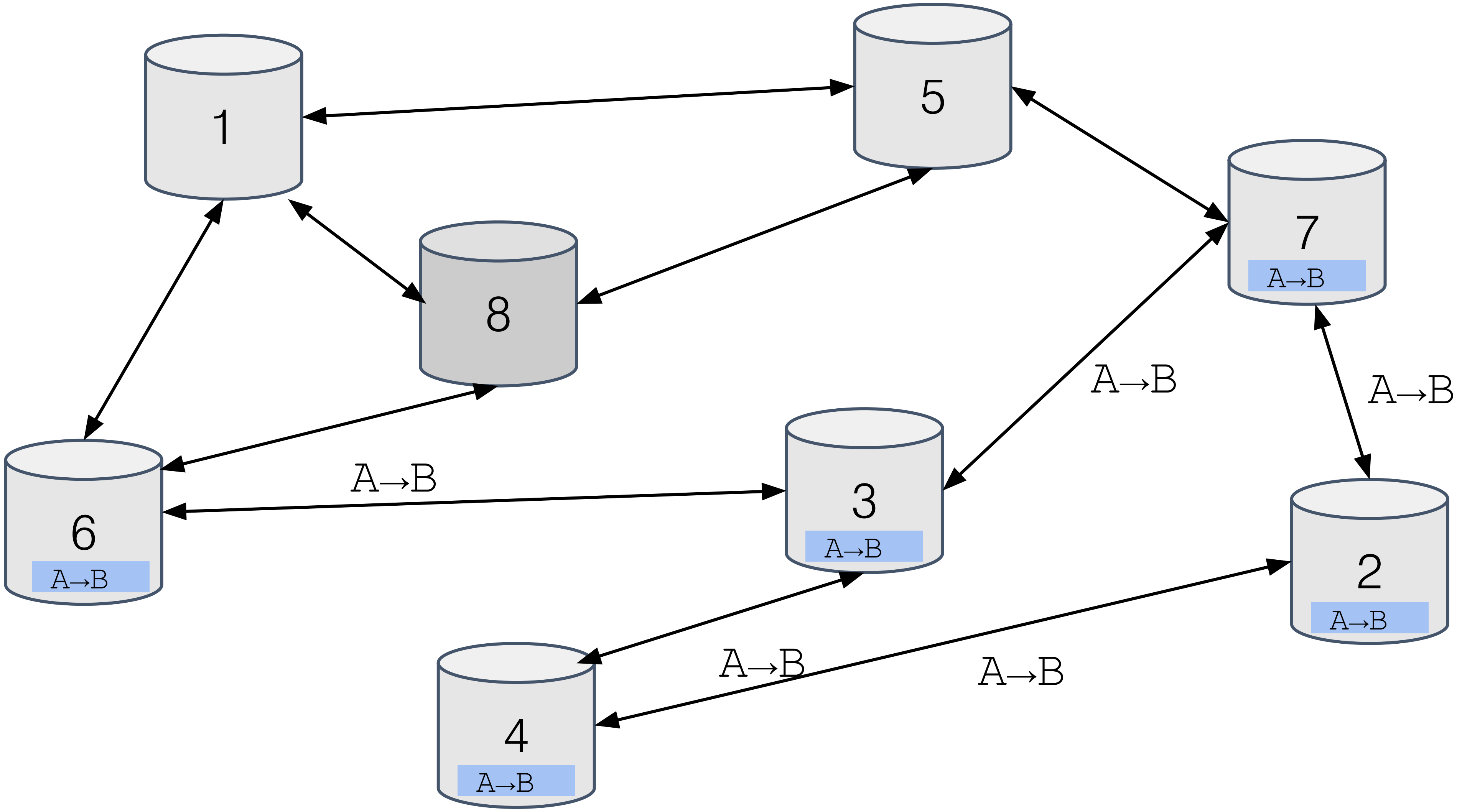
Transaction propagation (flooding)



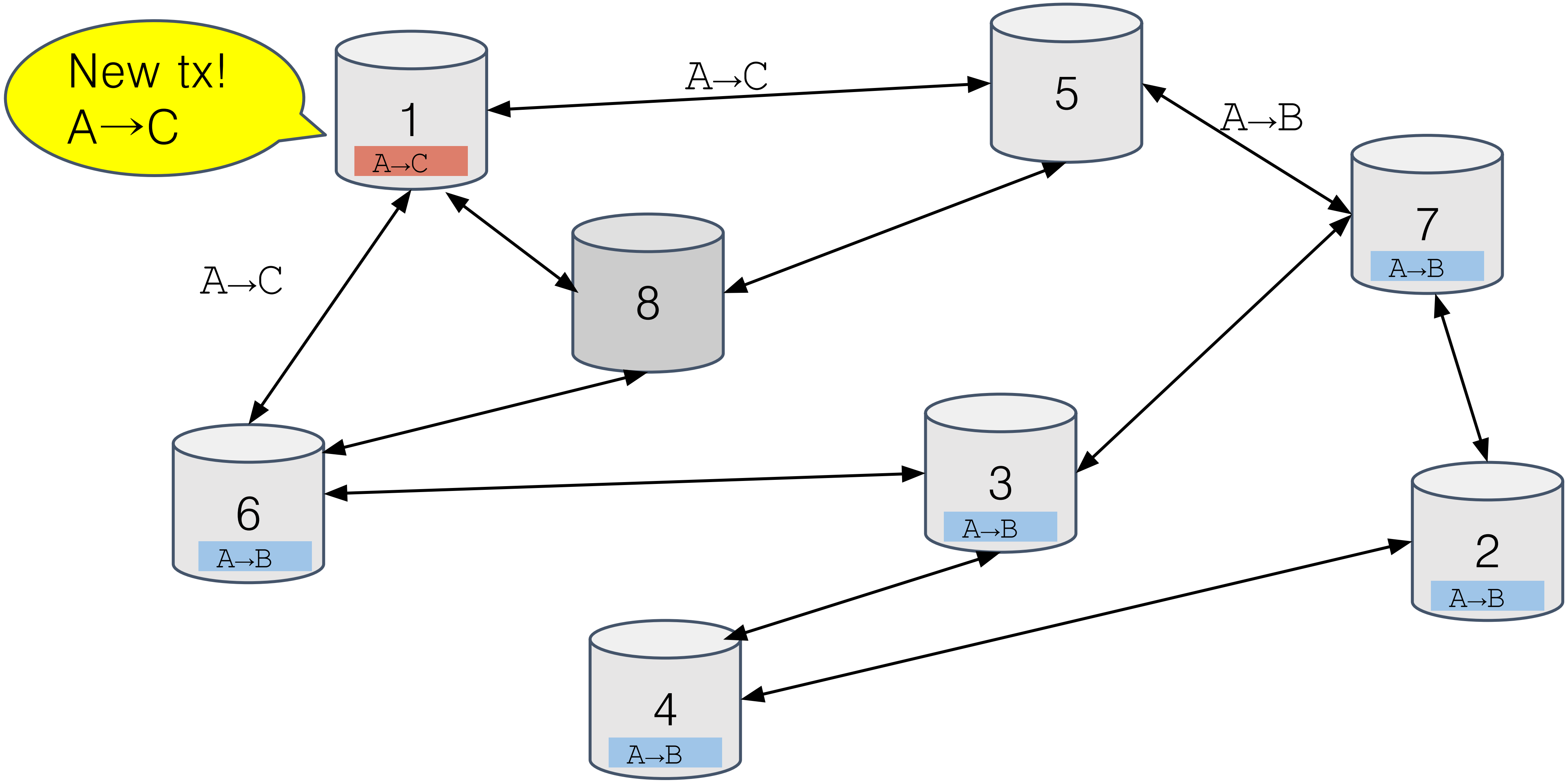
Transaction propagation (flooding)



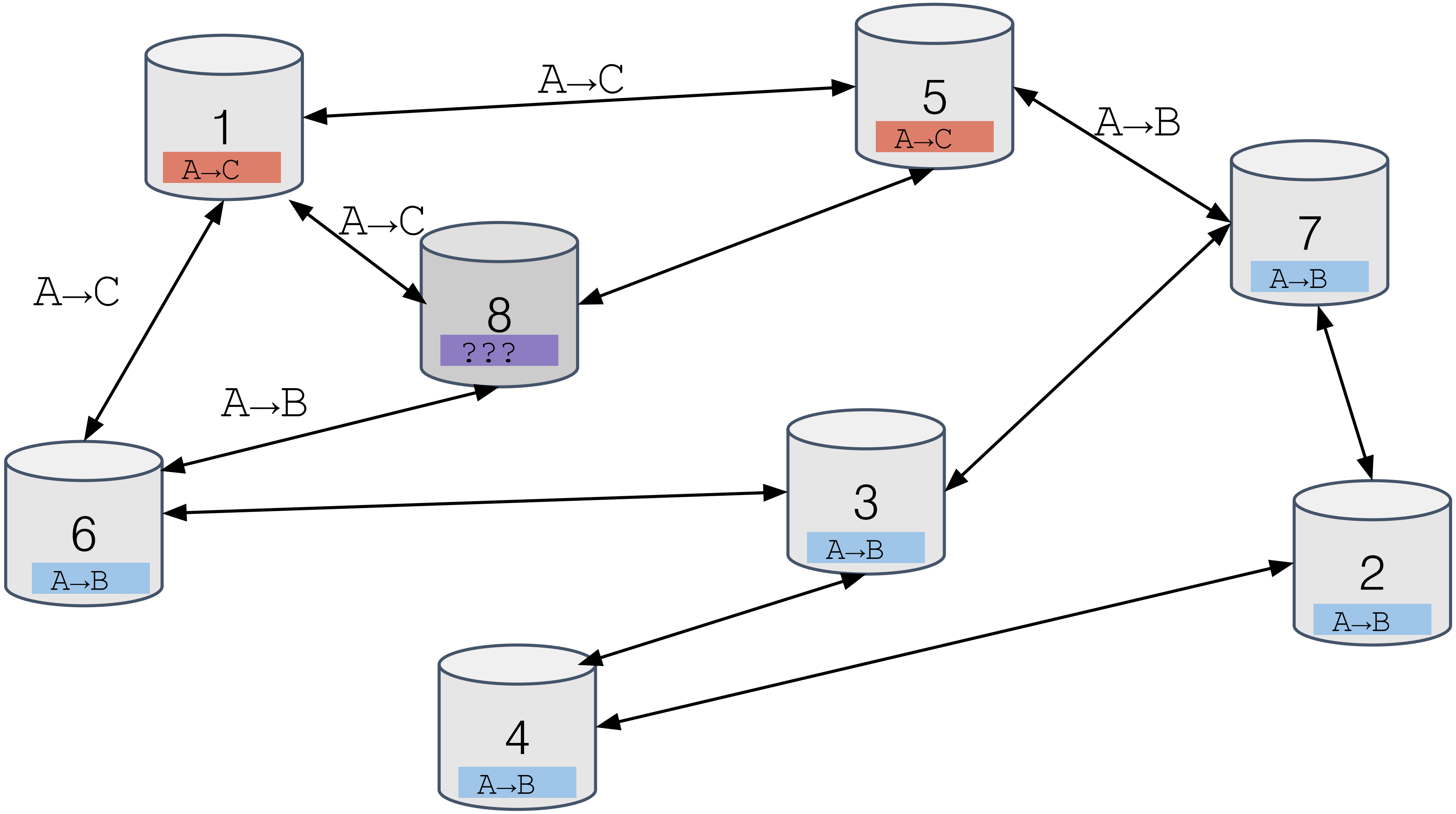
Transaction propagation (flooding)



Nodes may differ on transaction pool



Nodes may differ on transaction pool

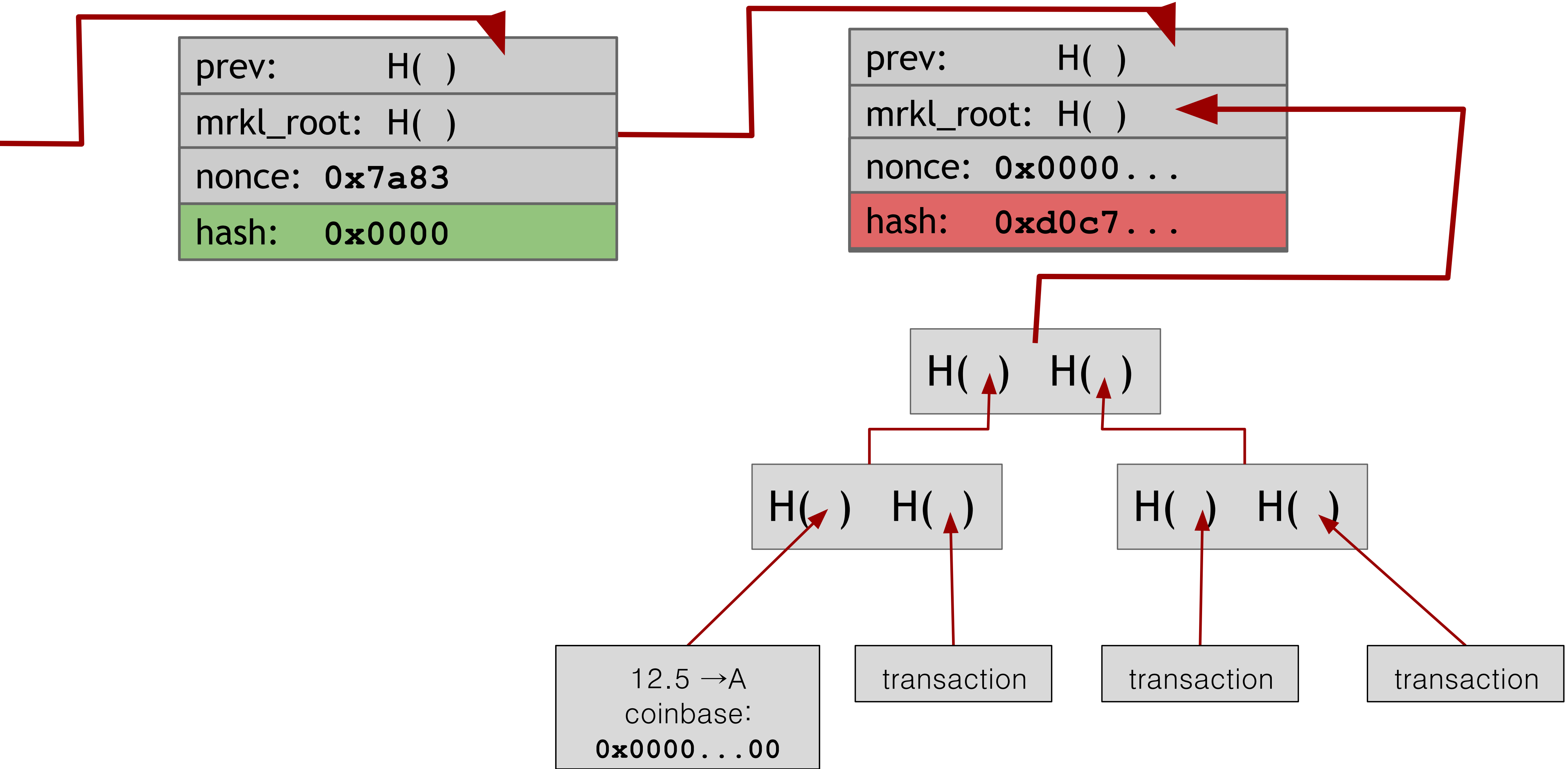


8. Finding a valid block

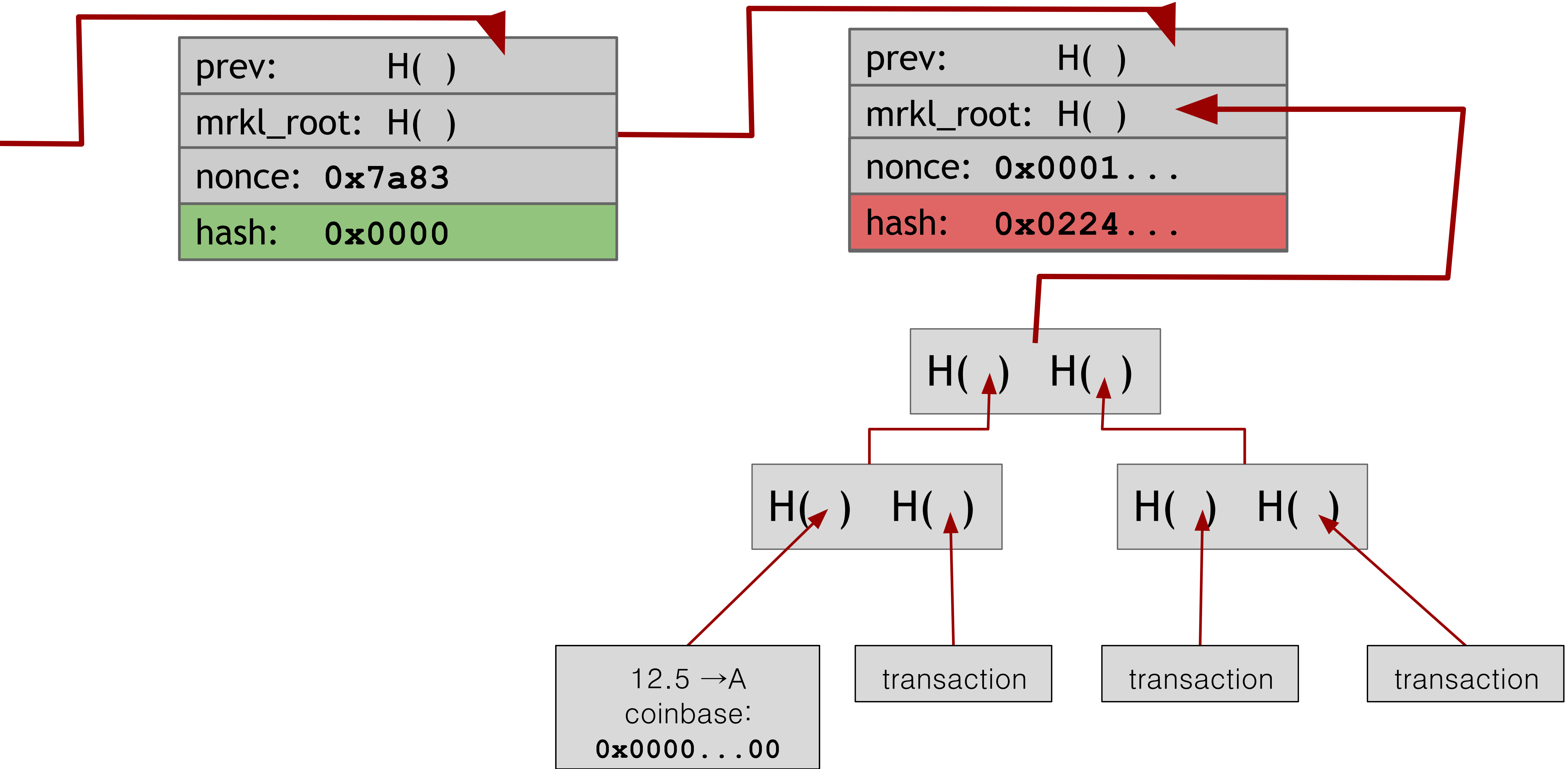
Who solves the puzzle confirms a TX block

- 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

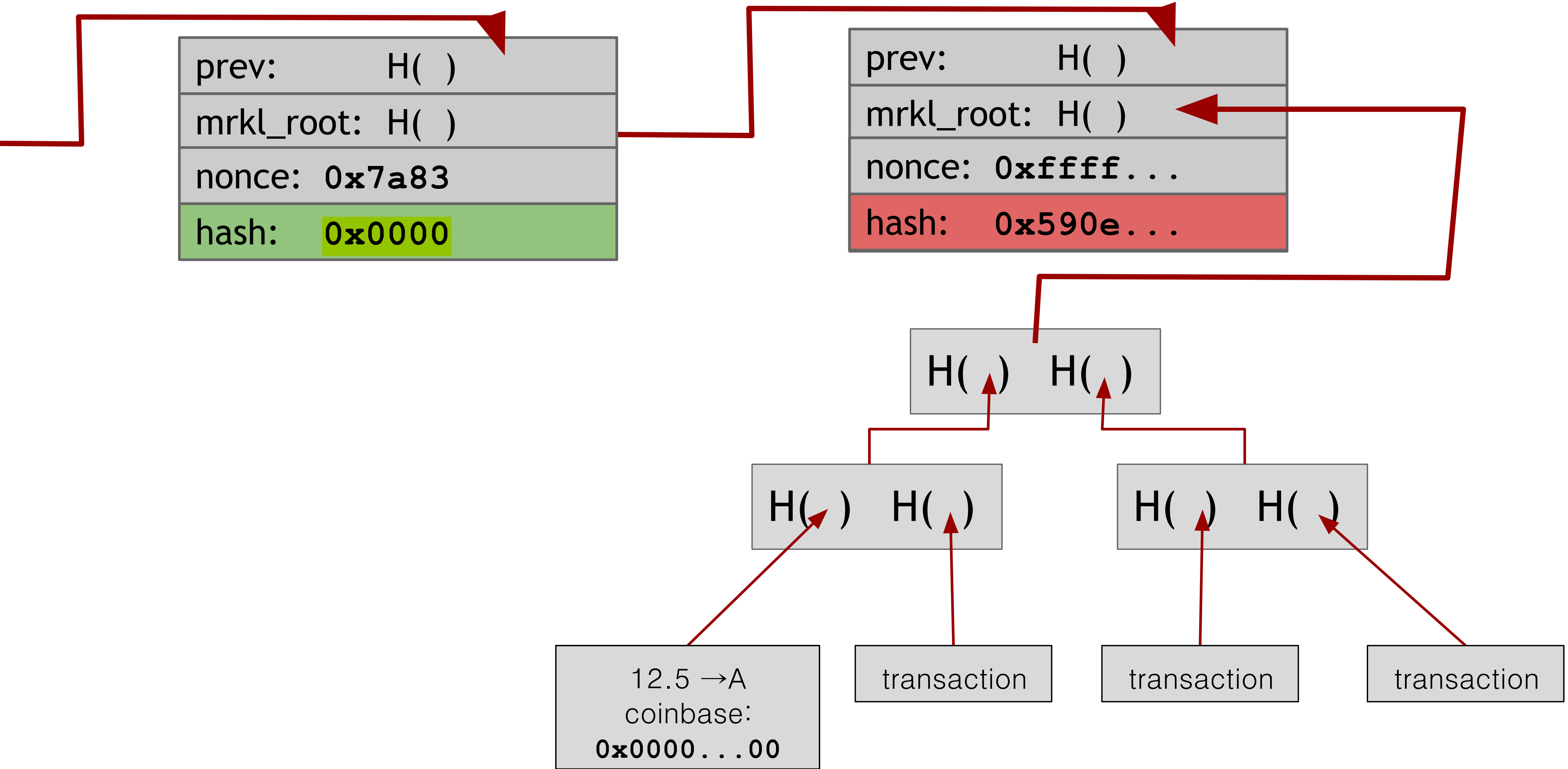
Finding a valid nonce is the hard part!



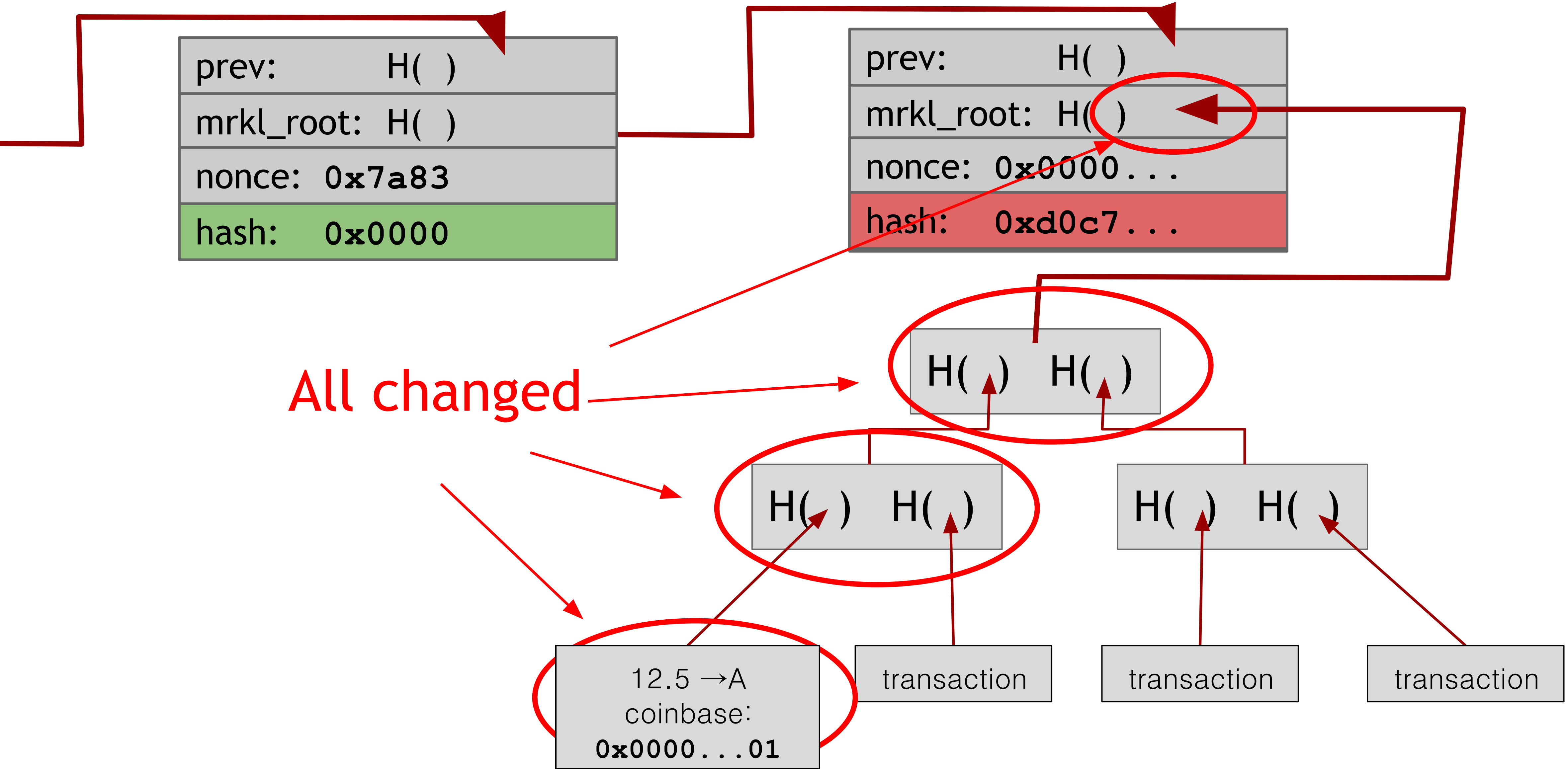
Finding a valid nonce is the hard part!



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

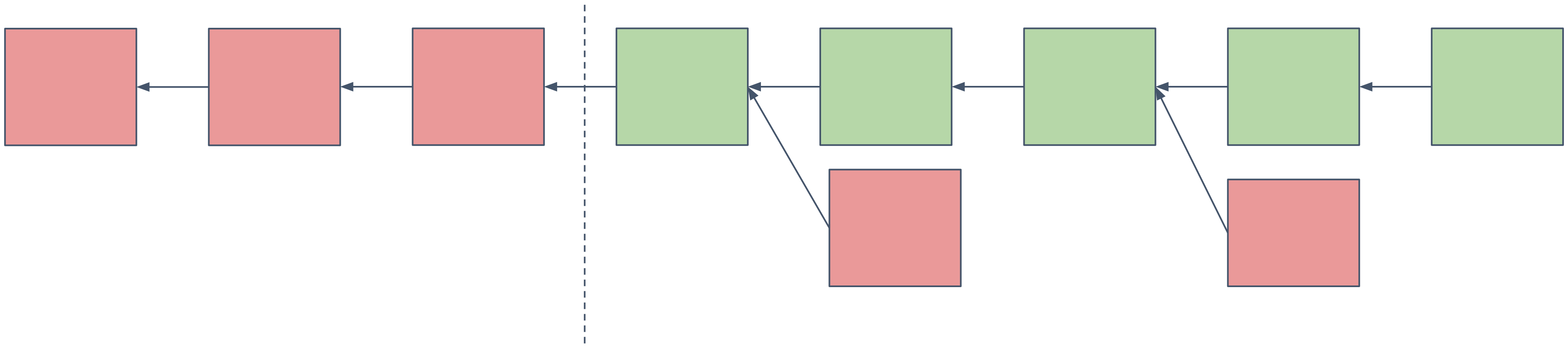
Originally designed by Satoshi

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

Soft fork



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

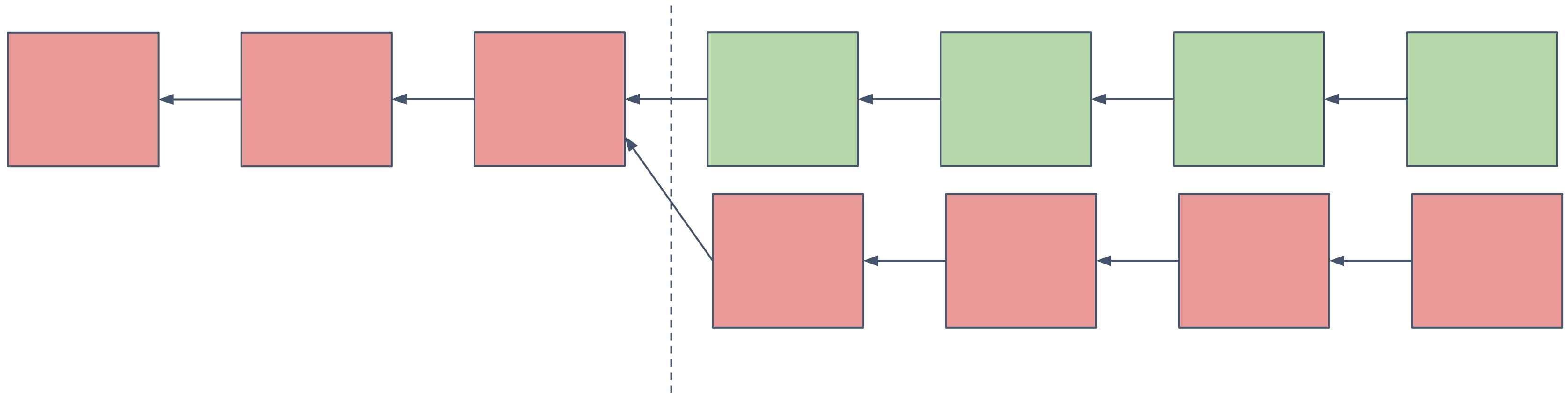
- They generate new blocks, which are not valid to old nodes

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

- they have their own chain

the ledger is replicated into two separate ones at the moment of hard fork

Hard fork



If a new TX is compatible, replay attack!
(the receiver broadcasts the TX on the other chain)

the owner can use her coins on both chains.
so her money becomes twice its value?

Altcoin

Alternative coin

Re-writes the rules from scratch

(usually) starts a new genesis block