Blockchain 2

PoW and Forks

Leander Jehl

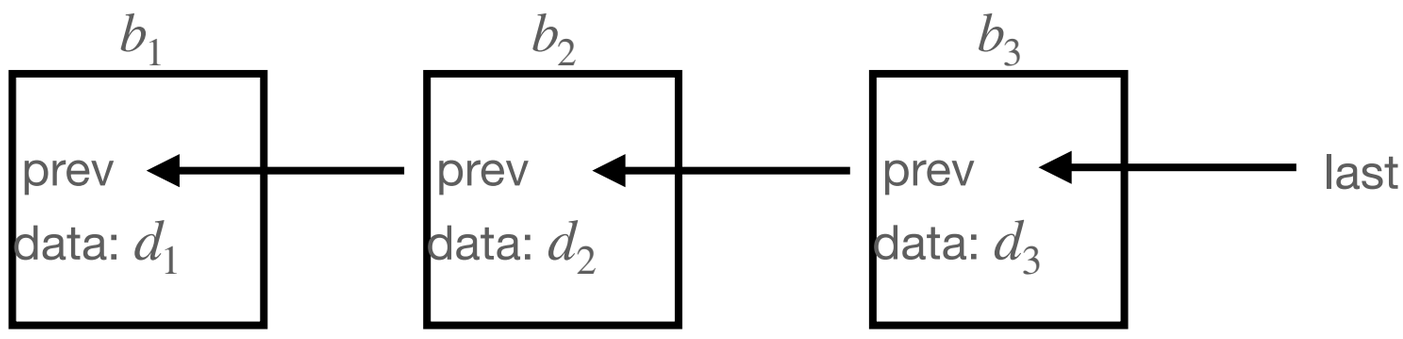
DAT655 Blockchain tech & app

Proof of Work

Proof of Work

Problem

```
type Block struct {
    prev pointer
    data bytes
    prevhash hash
}
```



- $id_b = H(b.\texttt{prevhash}||b.\texttt{data})$
- Blockchain identified by id_{b_3}
- Changing d_1 changes id_{b_3}
- Removing b_2 changes id_{b_3}
- Adding b_2' changes id_{b_3}

secured against changes

Problems:
- **Can recreate complete chain**
- **Who adds blocks?**

Proof of work

A proof of work allows to convince others that you did spend a certain amount of time/resources.

$$\pi \leftarrow f_{PoW}(\texttt{data}, d)$$

$$\texttt{bool} \leftarrow verify(\texttt{data}, \pi, d)$$

- d configurable difficulty
- f_{PoW} is long running
- $verify$ is fast

Use case:
- **Rate limit web API**

Proof of work

Verify:

- compute $hash = H(\texttt{data}||\pi)$
- check if first d bits of $hash$ are 0

$\textsc{verify}(\texttt{data}, \pi, d:)$

$hash = H(\texttt{data}||\pi)$

if first d bits of $hash$ are 0 **then**

  return `true`

return `false`

PoW:

- repeatedly try different nonces (data)

Proof of work

Some math

Lemma: For two different nonces, the probability to find a solution is independent.

Theorem: If p is the probability to find a nonce, then the expected number of trials is $\frac{1}{p}$.

Proof of work

Example:

- $d = 4$
- Probability to find a proof on one trial is $p = 2^{-4} = 1/16$
- Expected number of trials until success is $1/p = 16$
- Probability to not find a proof in 32 trials $(1 - p)^{32} = 0.127$
- $d = 5$ $\quad p = 2^{-5} = 1/32 \qquad 1/p = 32$

Proof of work

Verify:

- compute $hash = H(\texttt{data}||\pi)$
- check if first d bits of $hash$ are 0

VERIFY$(\texttt{data}, \pi, d:)$

$hash = H(\texttt{data}||\pi)$

if first d bits of $hash$ are 0 **then**

  return `true`

return `false`

- **Changing difficulty gives double/half the expected trials**
- **Amount of work needed is very variable.**

Proof of work

Better version

- **Difficulty D is hexadecimal number**

Verify:

- compute $hash = H(\texttt{data}||\pi)$
- check if $hash$ is smaller than D

$\textsc{verify}(\texttt{data}, \pi, D:)$

$hash = H(\texttt{data}||\pi)$

if $hash \leq D$ **then**

  return `true`

return `false`

PoW:

- repeatedly try different nonces

Proof of work

Difficulty adjustion example

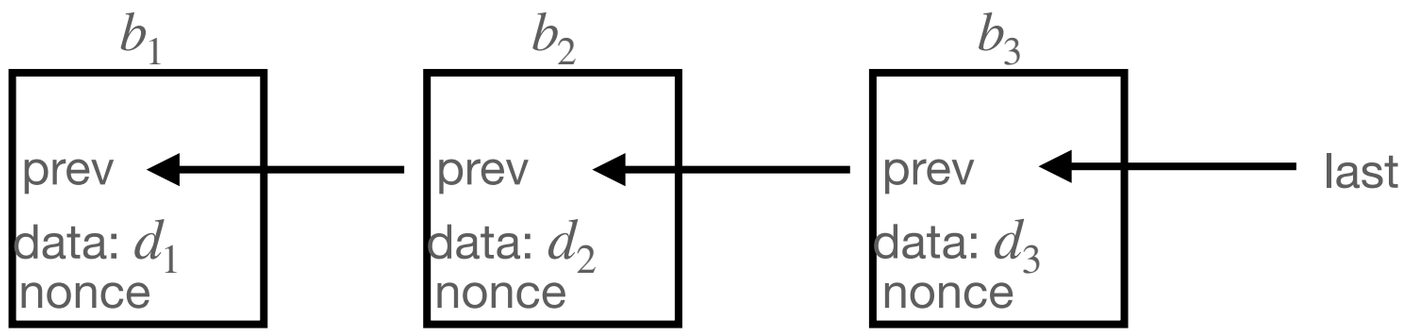
Example:

- A computer computes a PoW every 15 seconds.
- How can we adapt difficulty to get 20 seconds?

Proof of Work

In the blockchain

```
type Block struct {
    prev pointer
    data bytes
    prevhash hash
    nonce uint
}
```



- $id_b = H(b.\texttt{prevhash}||b.\texttt{data}||b.\texttt{nonce})$
- to recreate chain, need to recompute all proof of work

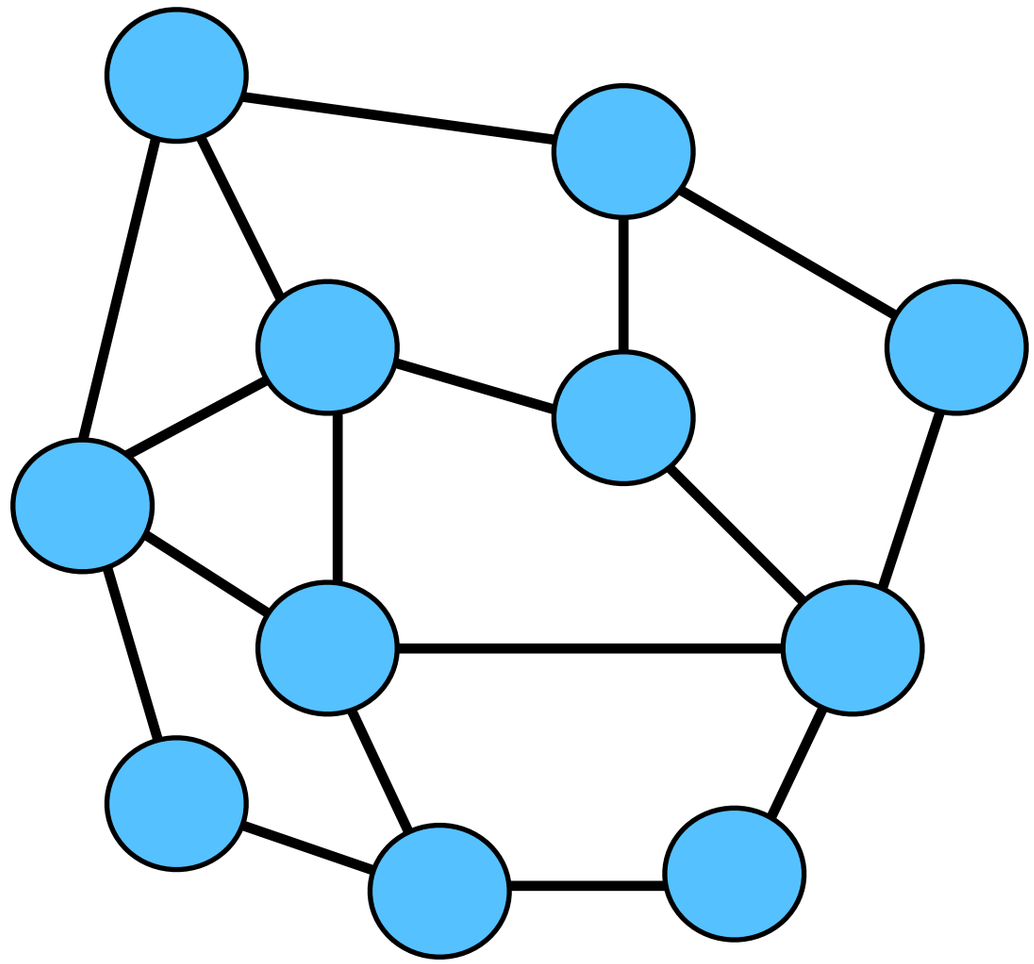
Who computes PoW?

PoW in peer to peer network

Proof of Work Network

Problem

- Store blockchain on many nodes
- Unknown nodes, anyone can join
- Decide what is the next block

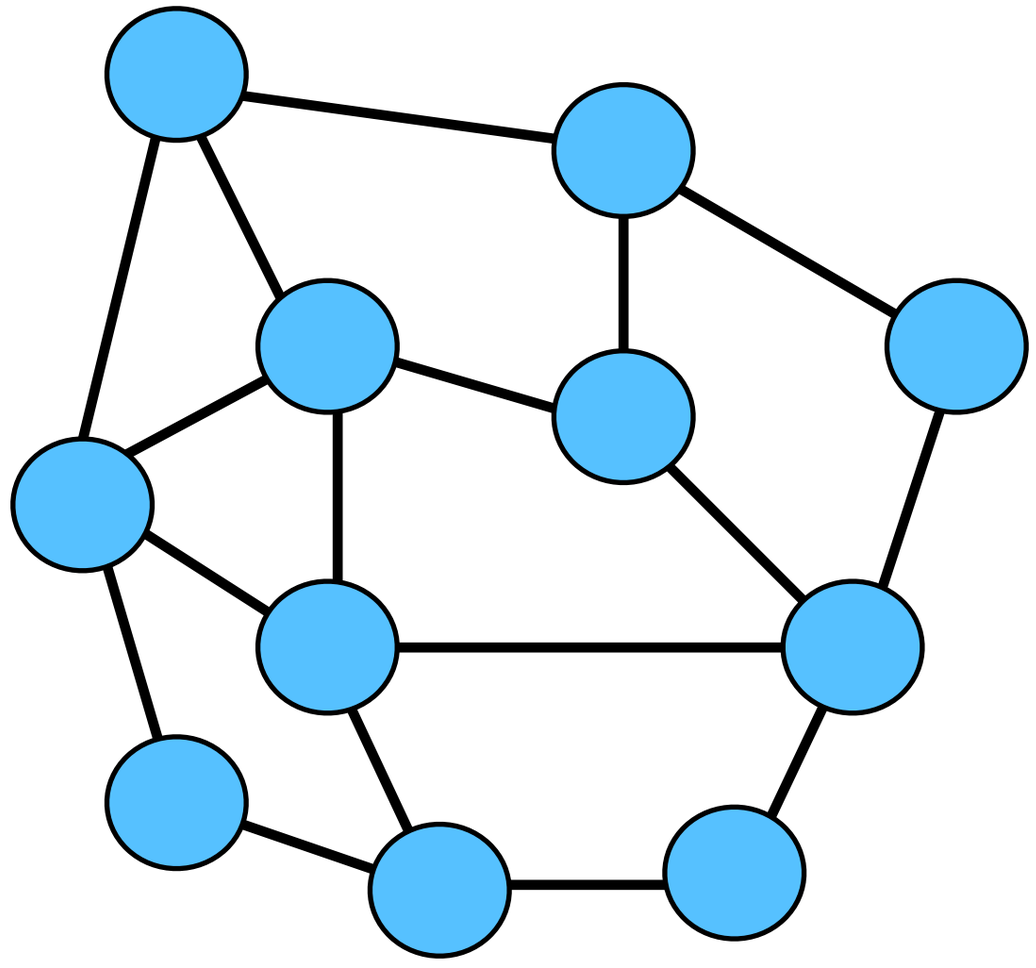


Problems:
- **No list of all ids exists**
- **New ids/nodes can be added easily**

Proof of Work Network

A Peer to Peer network

- several independent nodes
- nodes are well connected
- every node stores the blockchain
- transactions is broadcast to all nodes

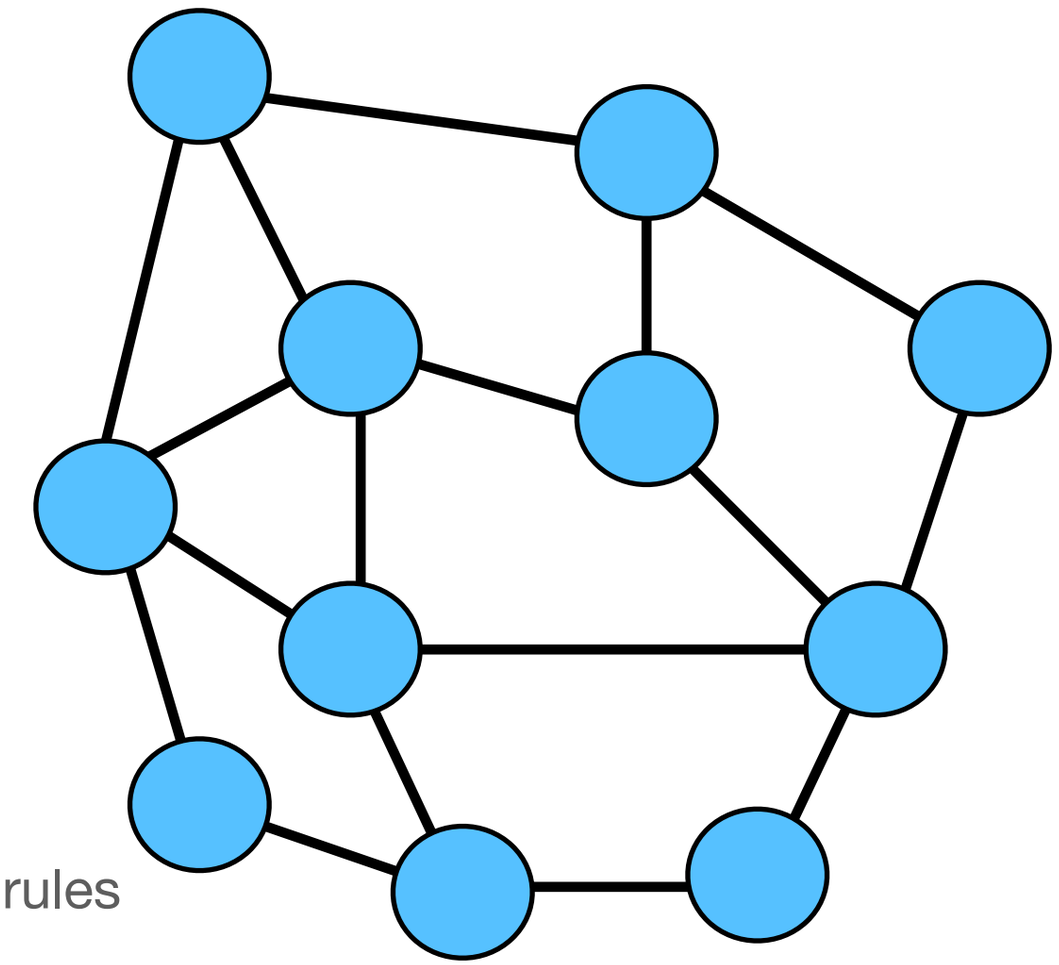


Proof of Work Network

Proof of work workflow

Every node does:

- collect transaction to form block data
- try to solve PoW (*find nonce*)
- the first to solve PoW publishes block to everybody
- all check PoW, *validate Block,* apply transactions, continue



can enforce rules

Proof of Work Network

Properties

Censorhip resistance

- One node cannot prevent a transaction to be put in a block.

Fault tolerance

- Individual nodes may fail.

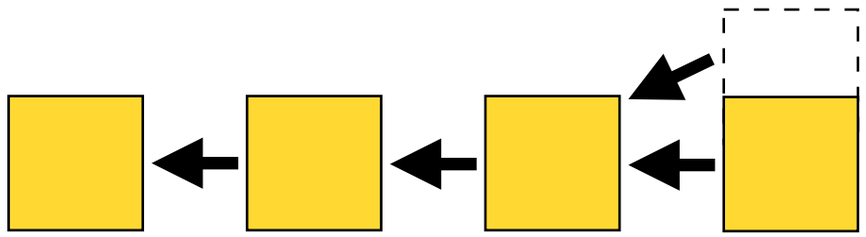
Rate control

- Difficulty of PoW determines how fast blocks are created.
- Maximum on data size gives rate limit on transactions.

Proof of Work Network

Proof of work workflow

Moving to a new block,
a node has the same chances to find a PoW.



Difficulty adjustment

Number of nodes in the system may change, need to adjust difficulty.

Idea: Include a timestamp in every block.

- **Need to validate timestamp on new block!**

At regular intervals, check average block delay, adjust difficulty.

Proof of Work Network

Rewards

Each transaction pays fees

For every new block a block reward is payd out/created

- A block includes pk of the node that receives the reward. **Coinbase transaction**
- Each nodes has a differnt block and needs a different nonce.

Interesting:

- Block rewards make the system run, even without transactions.
- Fees ensure nodes do actually include transactions.

Proof of Work Network

Fees

Each transaction pays fees

Each block may contain at most 1MB of transactions.

Include transactions which pay most fee per byte.

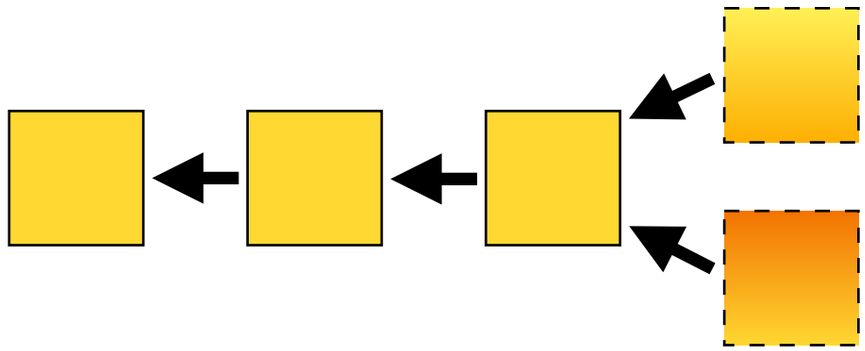
Interesting:

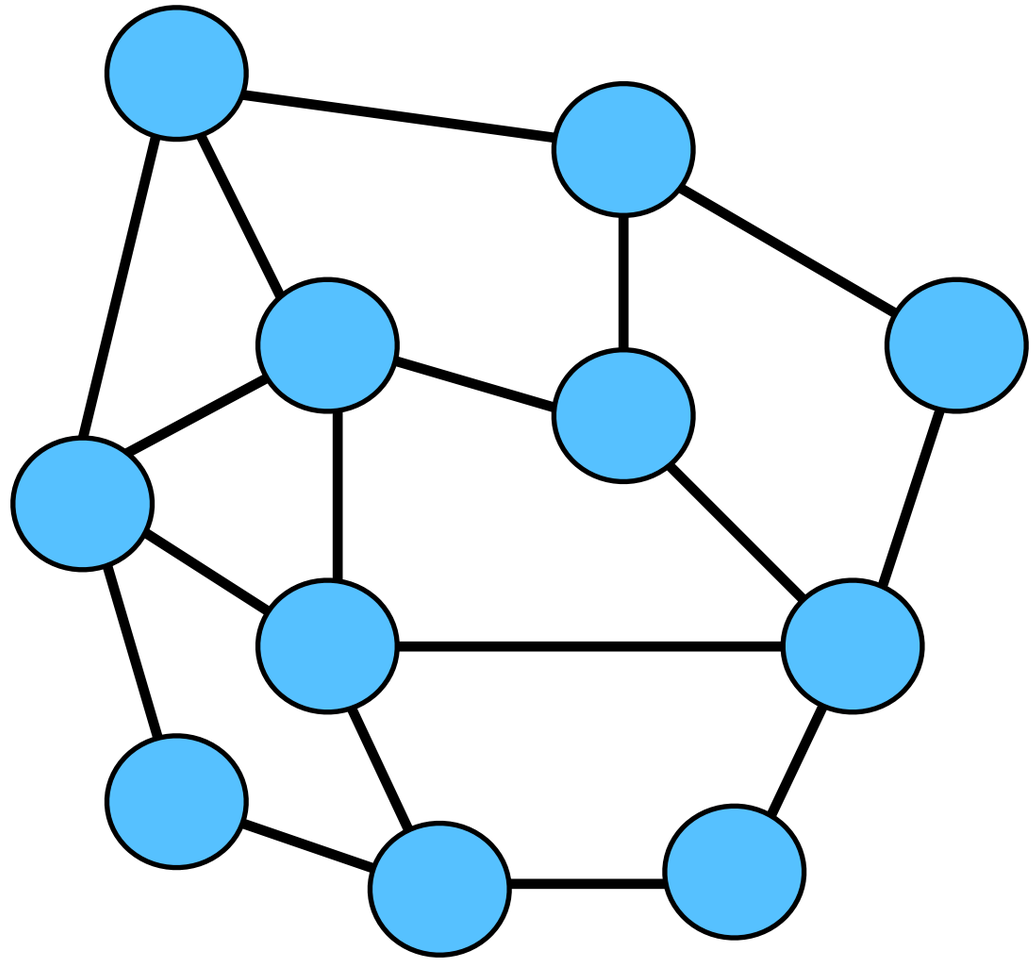
- Transaction with large fee included before transaction with small fee.
- Fee independent from transaction value.

Forks and longest chain rule

Forks

- A fork is if multiple blocks have the same predecessor





- Why: Two blocks found “concurrently”

Forks

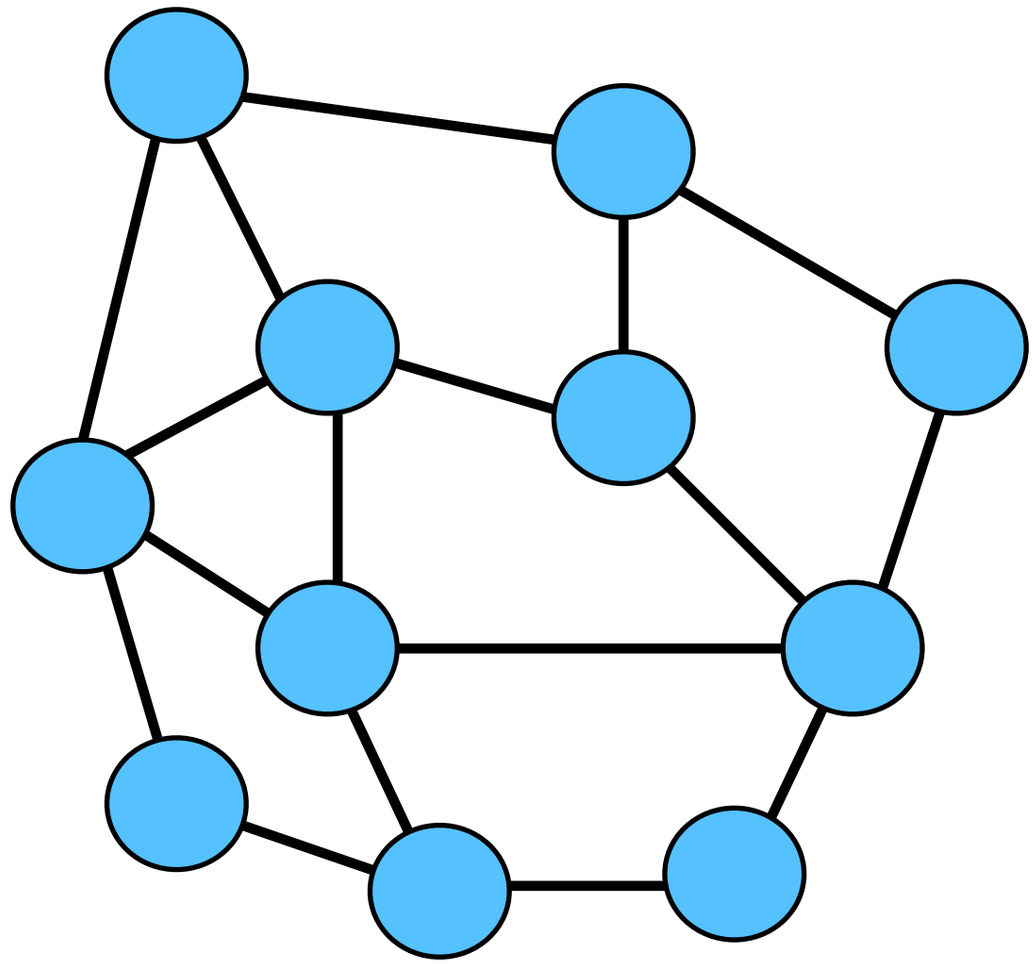
Proof of work workflow

Every node does:

- collect transaction to form block data
- try to solve PoW (*find nonce*)
- the first to solve PoW publishes block to everybody

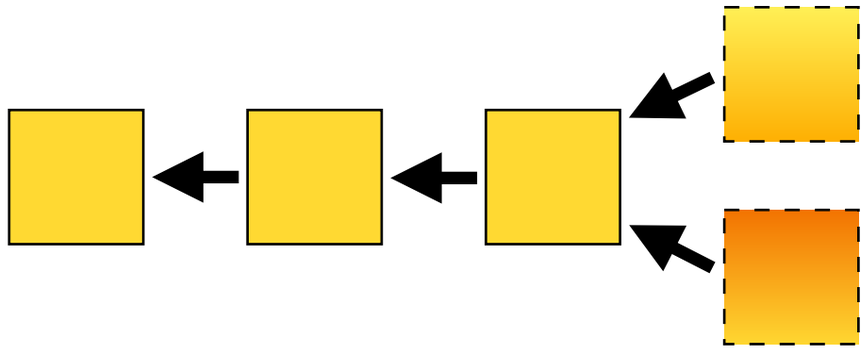
another block found before end of propagation

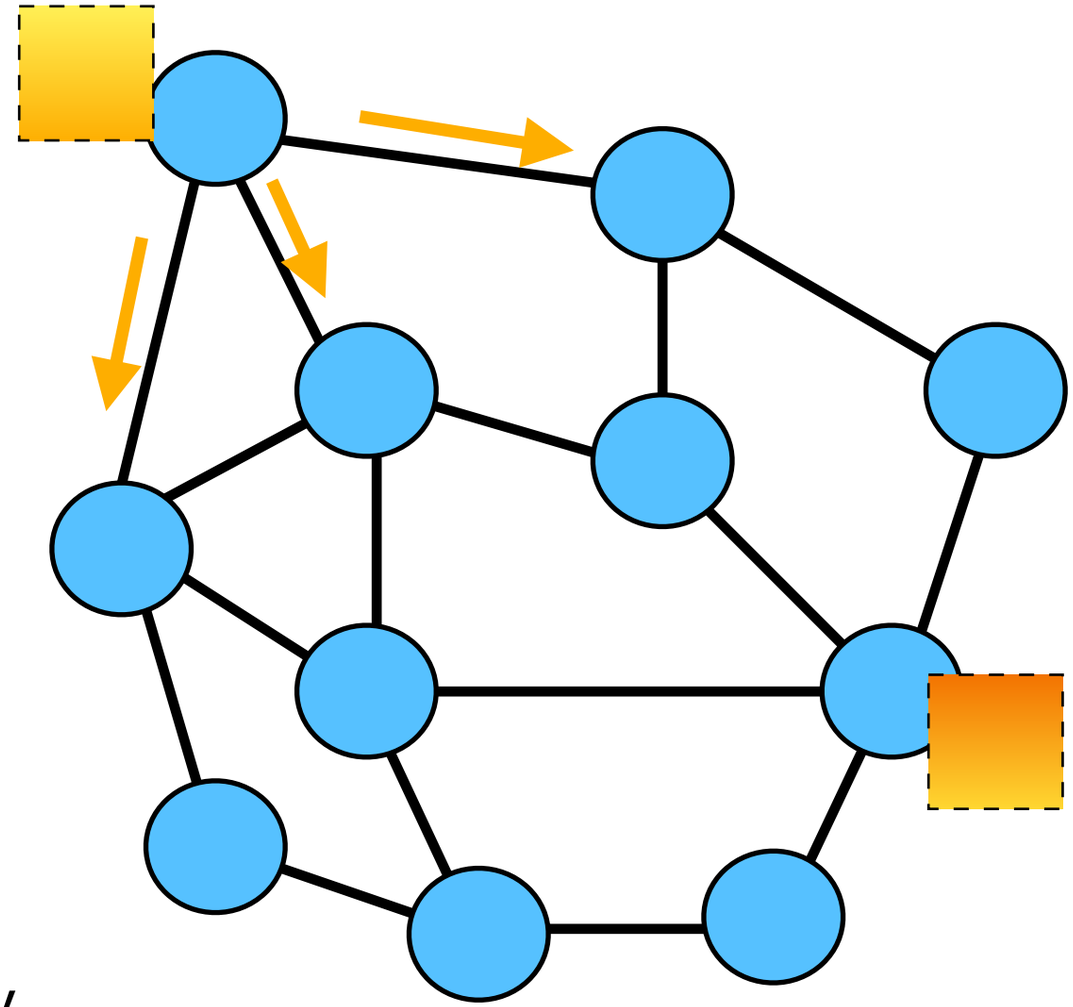
- all check PoW, *validate Block,* apply transactions, continue



Forks

- A fork is if multiple blocks have the same predecessor



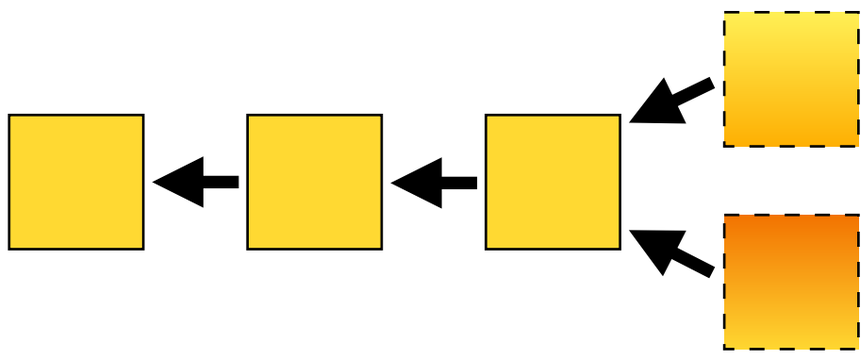


- Why: Two blocks found “concurrently”
- Bitcoin 2013: avg. 12.6sec block delivery [Decker, Wattenhofer]

Forks

Longest chain rule

- If a fork exists, all nodes should adopt the longest chain.

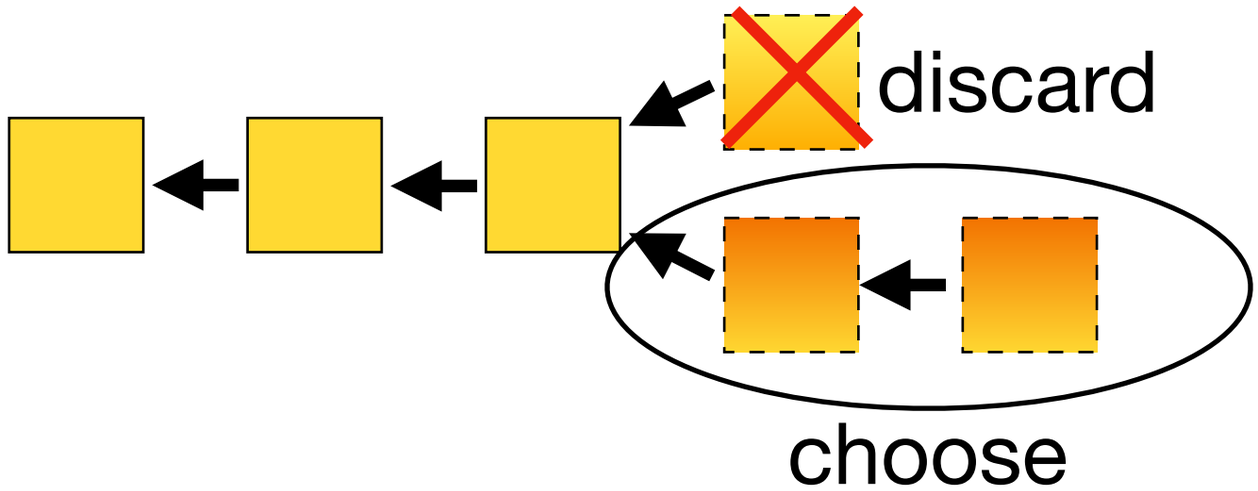


undecided

Forks

Longest chain rule

- If a fork exists, all nodes should adopt the longest chain.



Forks

Longest chain rule

- If a fork exists, all nodes should adopt the longest chain.

Problems:

- Blocks & Transactions in smaller chain are discarded
  - Miners loose reward
  - Some transactions may be only in one fork
  - Two conflicting transactions may be included in different forks (double spend)

Forks

Math: How likely is a fork

p_{sec} probability a block is found in one second

δ average time to get a block from the network

Theorem:

$$P[\text{fork}] = 1 - (1 - p_{sec})^{\delta}$$

Forks

Reparametrization

Fork probability depends on

- Network delay
  time to propagate a block
- PoW difficulty

lower difficulty → more forks

lower difficulty → faster blocks → more throughput

Forks

Reparametrization

Fork probability depends on

- Network delay
  time to propagate a block
- PoW difficulty

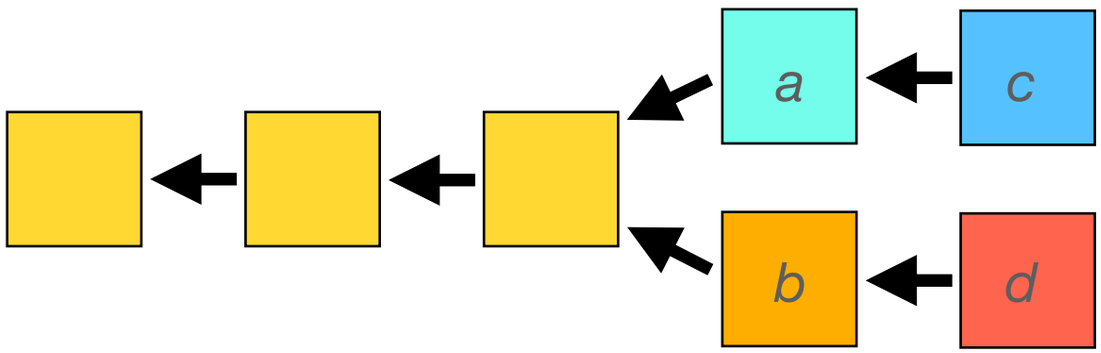
larger blocks → more throughput

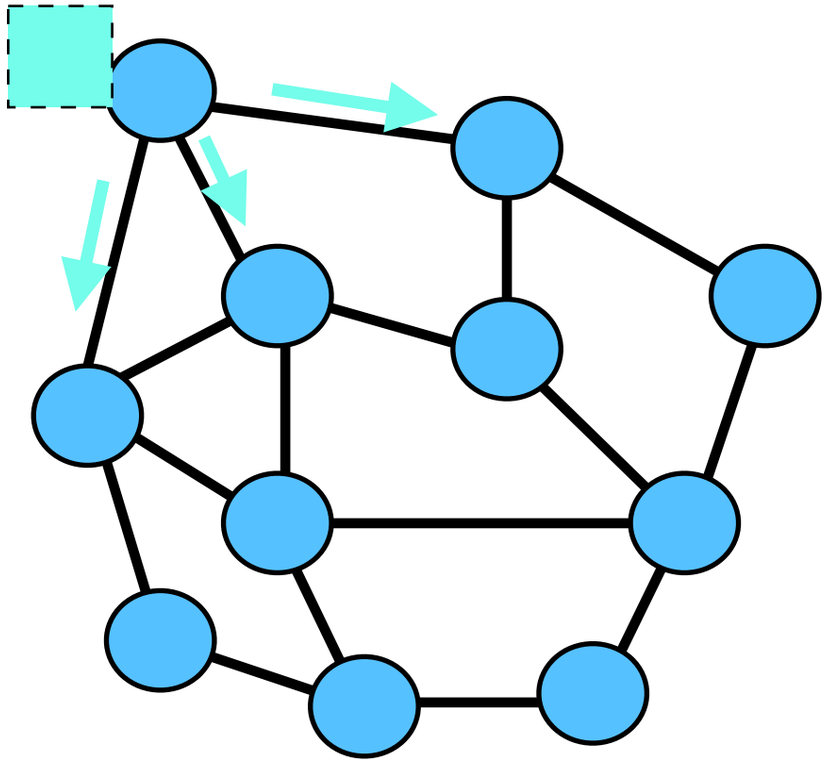
larger blocks → slow propagation → more forks

Forks

Multiple forks

- Multiple forks may arrise after each other.
- E.g. *b* found while *a* was propagated,
  - *d* found while *c* was propagated.

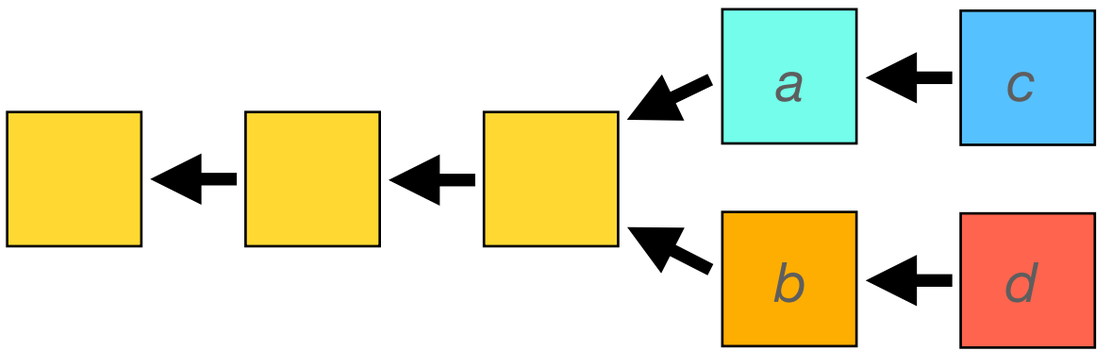


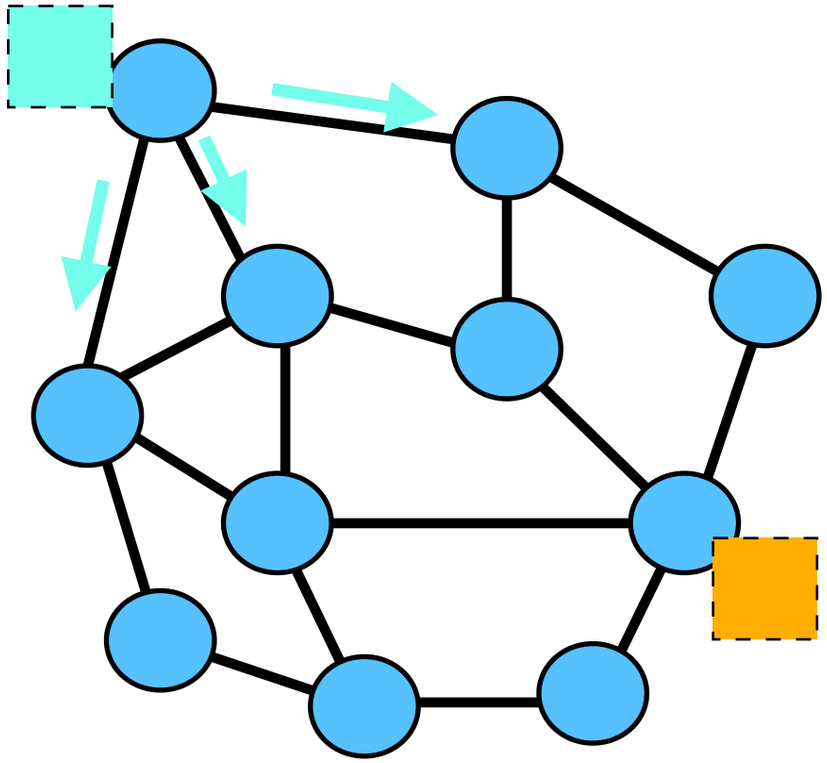


Forks

Multiple forks

- Multiple forks may arrise after each other.
- E.g. *b* found while *a* was propagated,
  - *d* found while *c* was propagated.

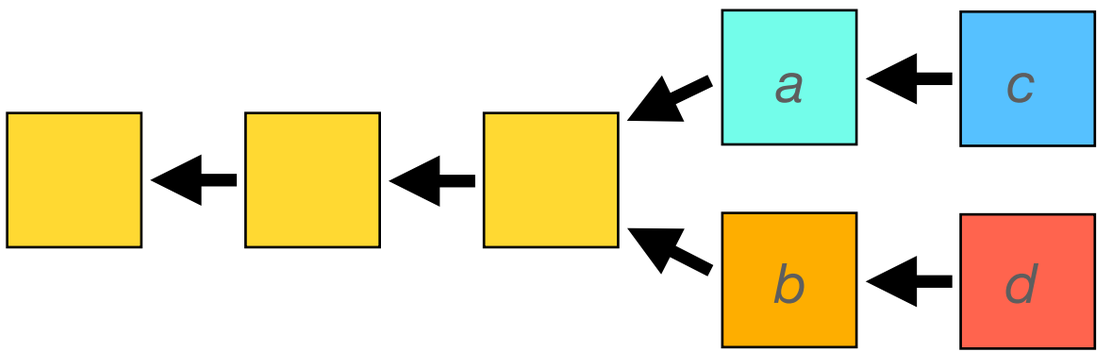


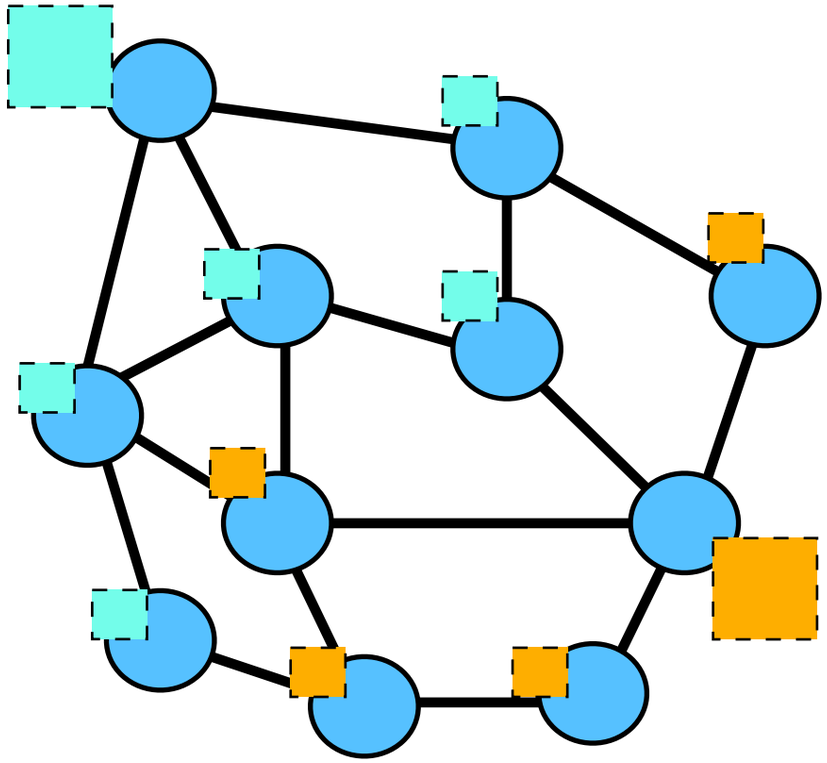


Forks

Multiple forks

- Multiple forks may arrise after each other.
- E.g. *b* found while *a* was propagated,
  - *d* found while *c* was propagated.

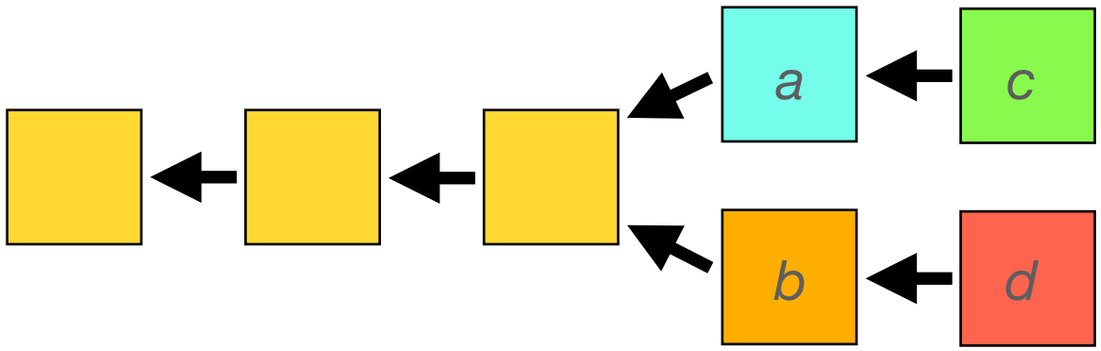


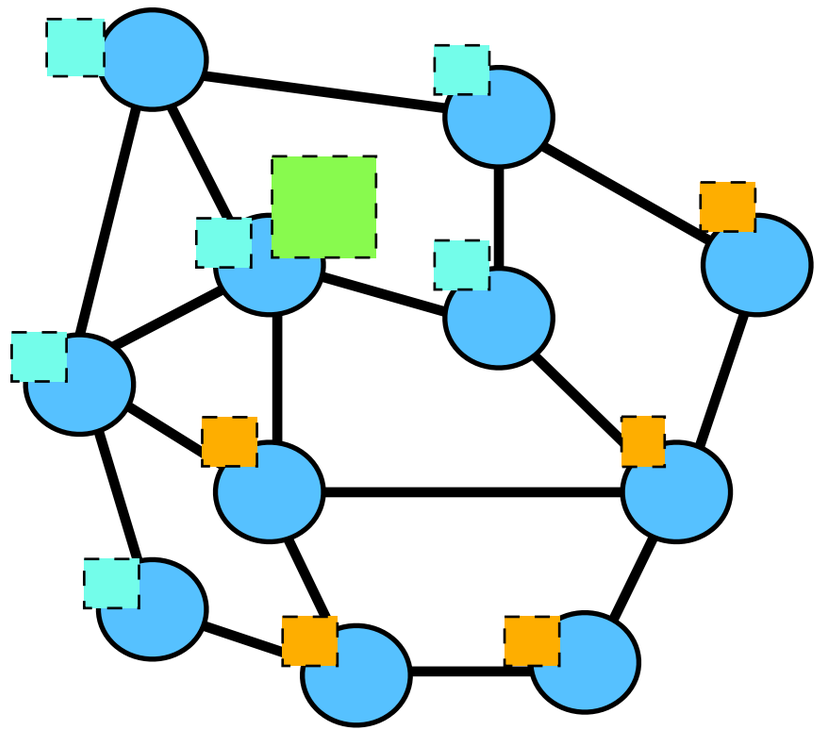


Forks

Multiple forks

- Multiple forks may arrise after each other.
- E.g. *b* found while *a* was propagated,
    - *d* found while *c* was propagated.

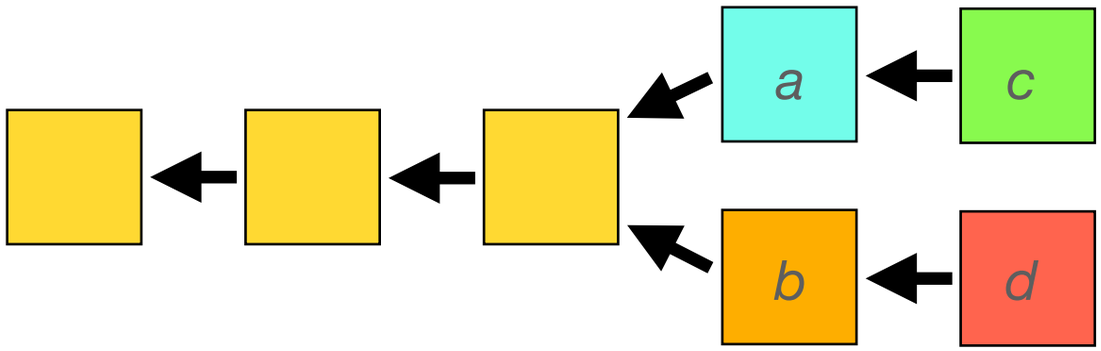


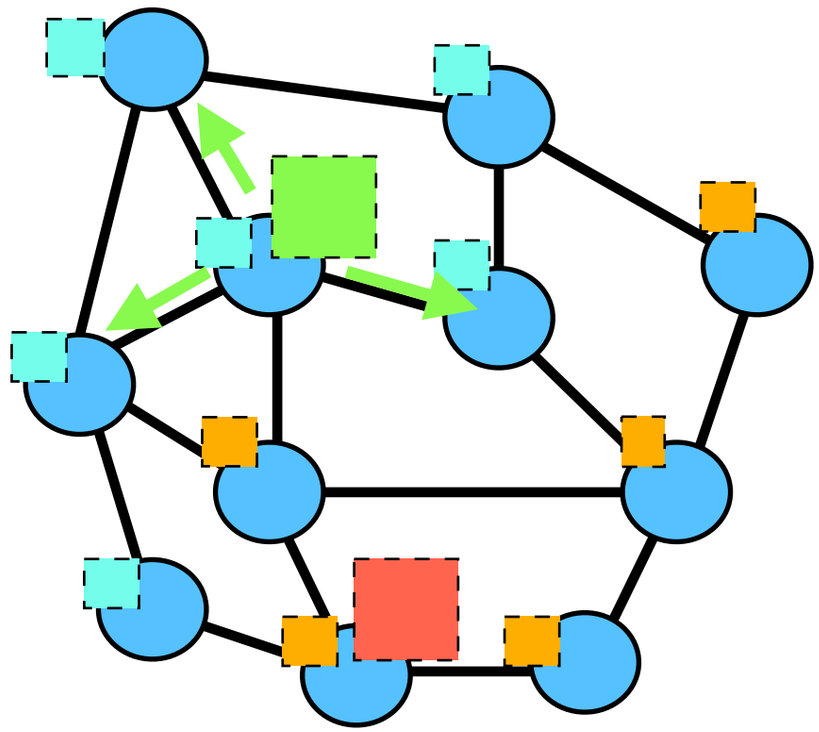


Forks

Multiple forks

- Multiple forks may arrise after each other.
- E.g. *b* found while *a* was propagated,
    - *d* found while *c* was propagated.



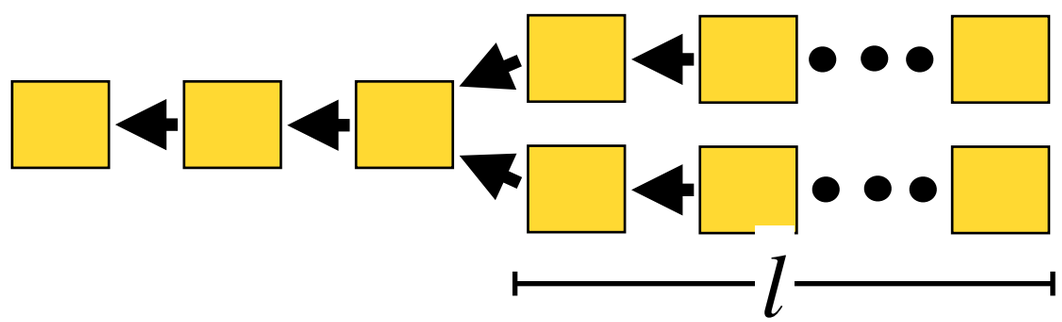


- Probability for second for smaller than the first.

Forks

Multiple forks

- Multiple forks may arrise after each other.
- Probability for second for smaller than the first.
- Probability for l forks decreases exponentially
- $P[l \times \text{fork}] \leq P[\text{fork}]^l$



Wait for l blocks to consider a transaction confirmed.

Attacks

Possible attacks

Proof of work workflow

Every node does:

- collect transaction to form block data
- try to solve PoW (*find nonce*)
- the first to solve PoW publishes block to everybody

Attack:
don’t publish block

- all check PoW, *validate Block,* apply transactions, continue

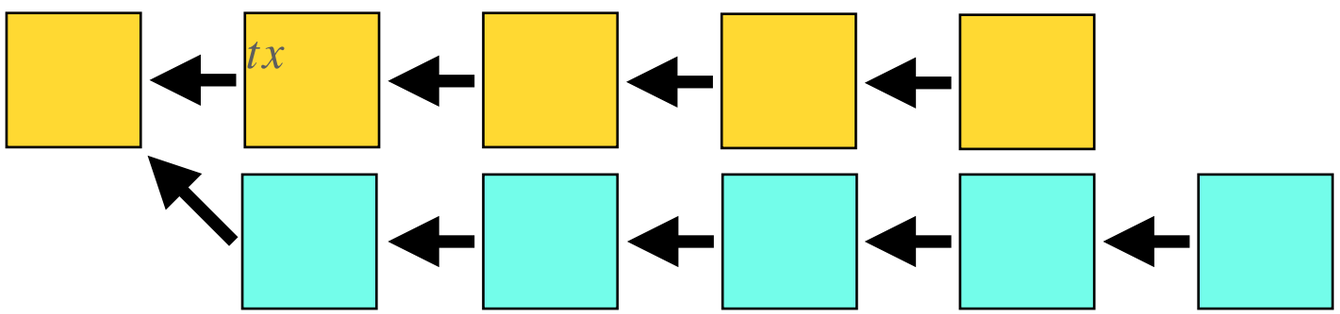
Attack:
don’t extend at right place

Attacks

51% attack

- Assume the attacker has $\alpha > 50\,\%$ of the hashing power.
    - Attacker can grow a private chain faster than the public chain.

A private chain is a fork with blocks not propagated through the network.



Attacker can:
- **Double spend**
- **Get all the reward**

Attacks

Stubborn mining:

- Attacker does not follow longest chain rule.

Selfish mining:

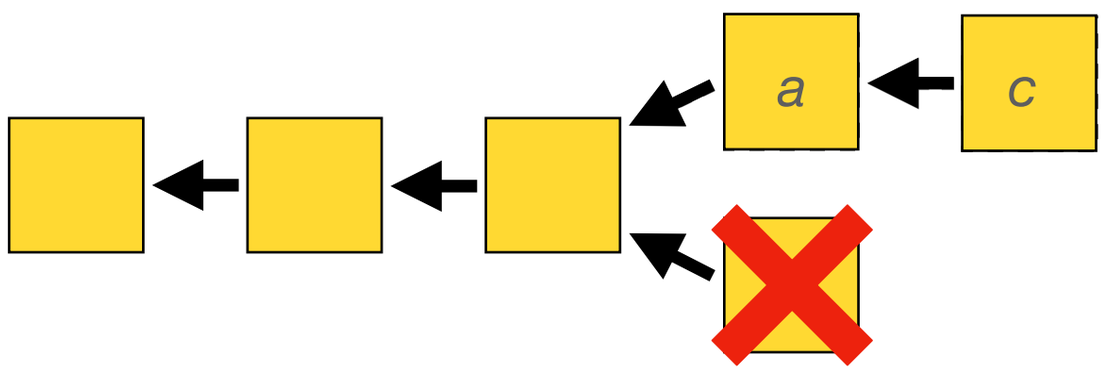
- Attacker keeps blocks secret.

Attacks

Selfish mining

Case 1, successfull attack:

1. attacker finds block *a,* keeps it secret
2. attacker finds block *c,* keeps it secret
3. other nodes find block *b* and propagate it
4. attacker propagates blocks *a* and *c*

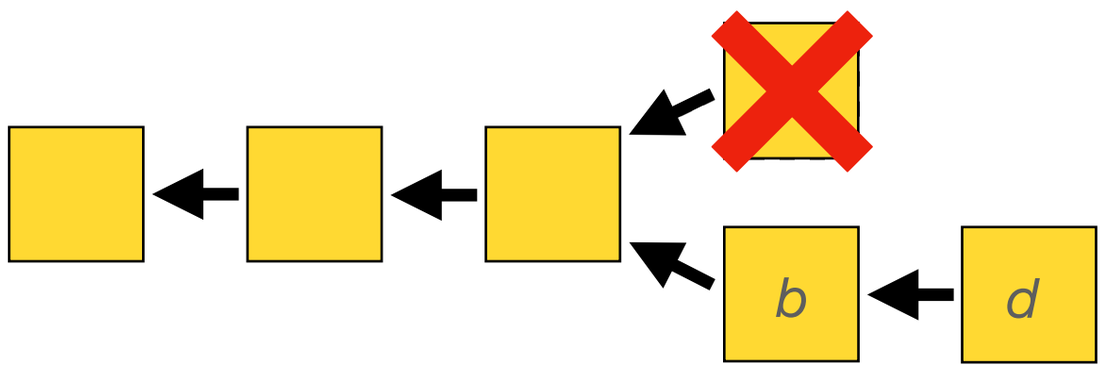


Attacks

Selfish mining

Case 2, unsuccessfull attack:

1. attacker finds block *a,* keeps it secret
2. other nodes find block *b* and propagate it
3. attacker propagates block *a*
4. other nodes find block *d* extending *b*

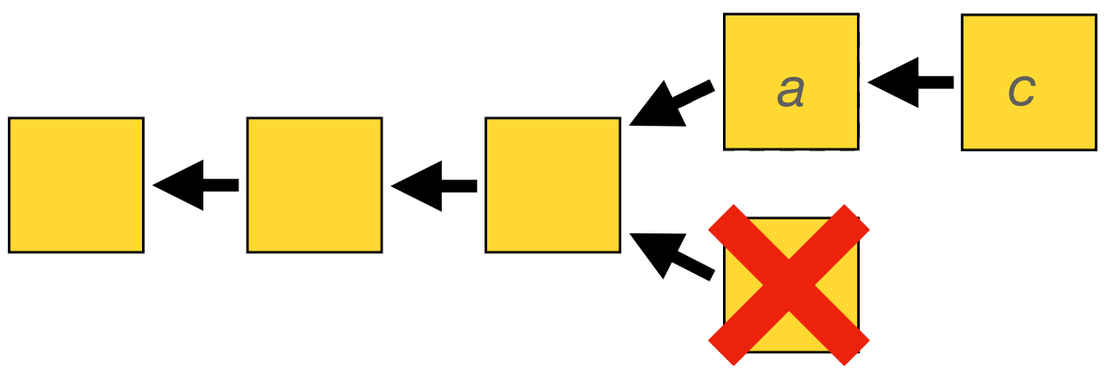


Attacks

Selfish mining

Case 3, kind of successfull attack:

1. attacker finds block *a,* keeps it secret
2. other nodes find block *b* and propagate it
3. attacker propagates block *a*
4. some node finds block *c* extending *a*

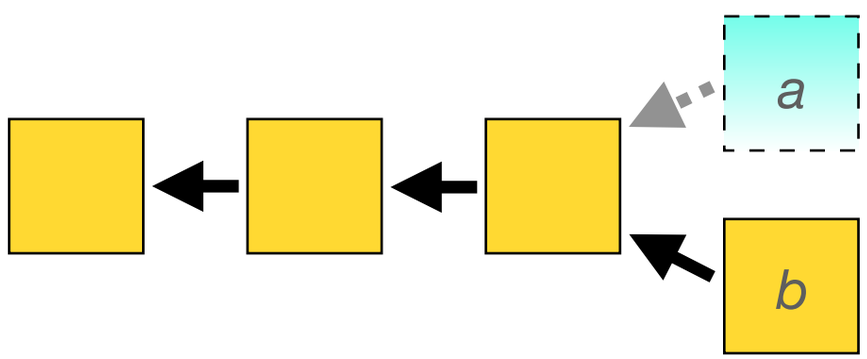


Attacks

Selfish mining

To get Case 3 instead of Case 2 attacker needs to

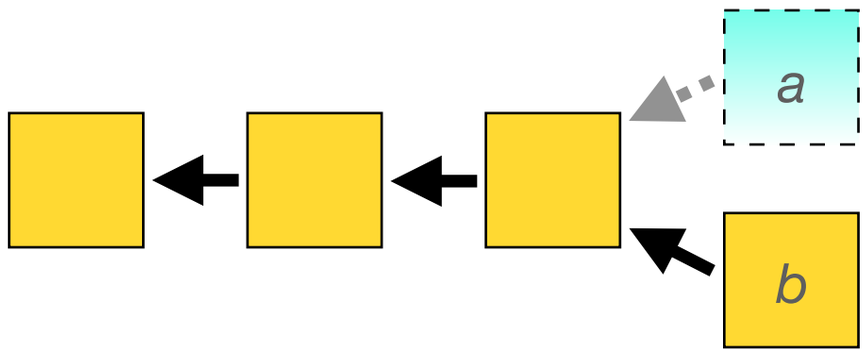
- detect new blocks fast
- propagate its block faster



Attacks

Selfish mining - take away

- Attacker does not get more blocks, but others get less.
- Good control of network makes attack work better.



Attacks

Selfish mining

Algorithm 6 Selfish mining

Idea: Mine secretly, without immediately publishing newly found blocks
Let l_p be length of the public chain
Let l_s be length of the secret chain
if a new block b_p is published, i.e. l_p has increased by 1 **then**
if $l_p > l_s$ **then**
    Start mining on b_p
else if $l_p = l_s$ **then**
    Publish secretly mined block b_s
    Mine on b_s and immediately publish new block
else if $l_p = l_s - 1$ **then**
    Push all secretly mined blocks

Attacks

Selfish mining

α the attackers hashing power, and
γ be the attackers network power.

Selfish mining is profitable, if

$\alpha > 0.33$

$\alpha > 0.25$ and $\gamma > 0.5$

$\alpha > 0$ and $\gamma = 1$

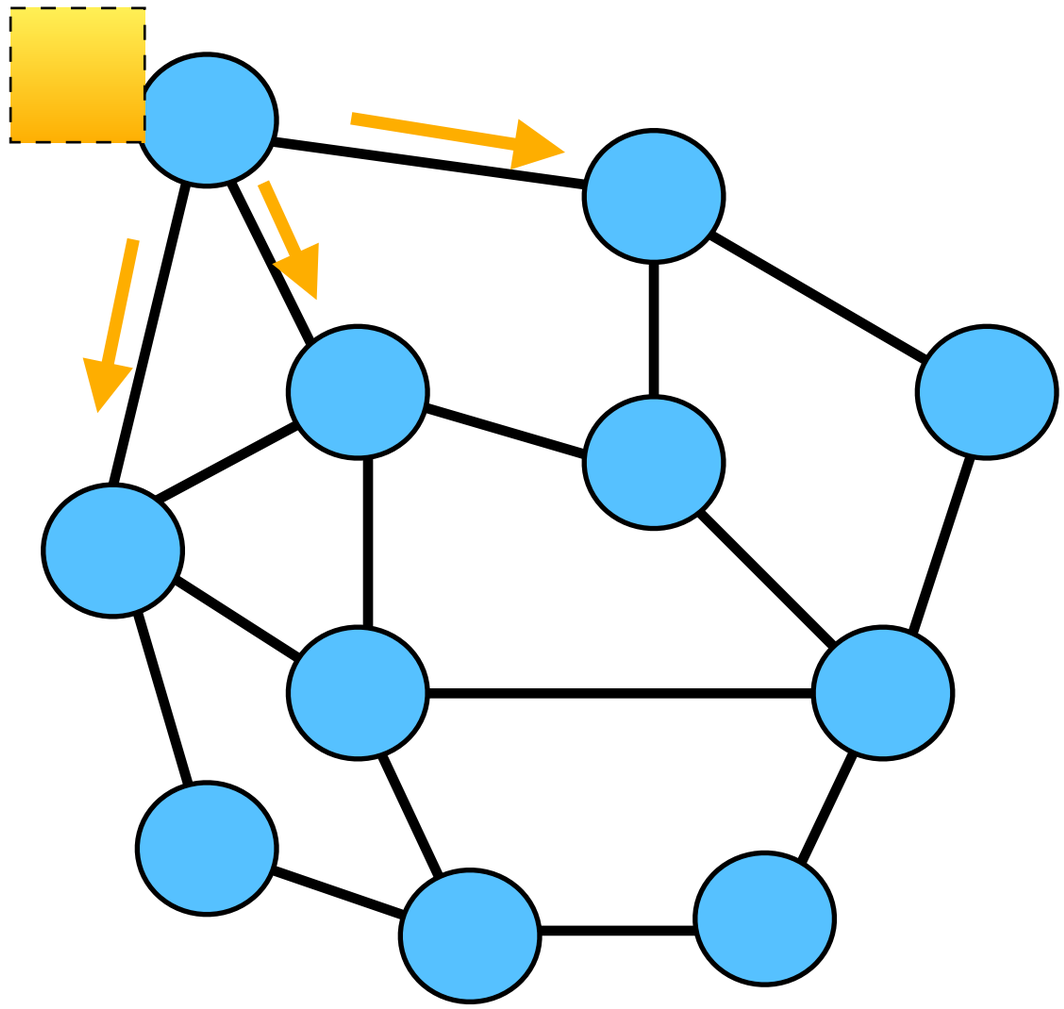
Attacks

Delivery denial

Broadcast block:

- Broadcast inventory message including block hash
- Receiving new inventory, request block
- Send block

Block is only send from one neighbor



Attacks

Delivery denial

Broadcast block:

- Broadcast inventory
- Request block
- Send block

Attack

- Broadcast inventory
- Do not send out blocks

  Victims wait for timeout.

Bitcoin

Downsides

Throughput at most 7tx per second

Confirmation latency approx 1h

Enormous energy consumption