

# Mining, Consensus, Forks

## ENFORCING RULES WITHOUT A CENTRAL AUTHORITY

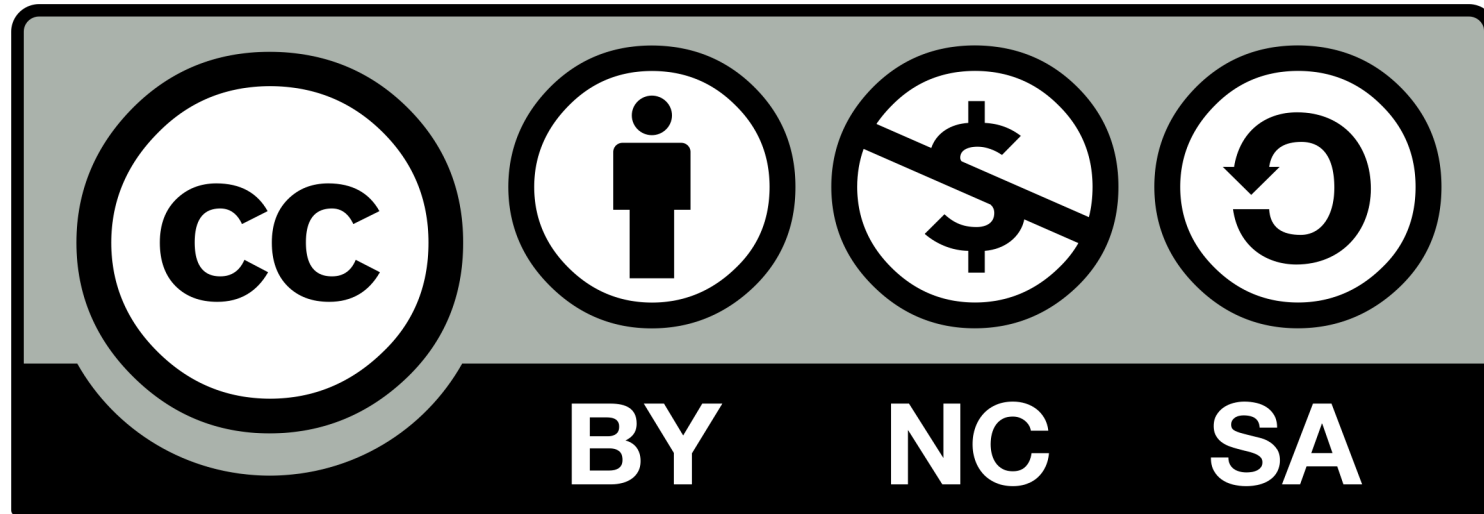
---

Stéphane Roche

06-10-2019

# CREATIVE COMMONS

Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)



# ABOUT STEPHANE



**2015**

Work at Ledger - hardware wallet company



**2017–2019**

Found Bitcoin Studio

Focus on Bitcoin education

Consultant at Chainsmiths

## Work on Ethereum


- Learn and play
- Co-found non-profit organization Asseth
- Contribute to the ERC20 Consensus smart contracts
- Dether.io



**2016–2017**

<https://www.bitcoin-studio.com>  
@janakaSteph on Twitter  
[bitcoin-studio@protonmail.com](mailto:bitcoin-studio@protonmail.com)

# OUTLINE

- 
- 1** The Bitcoin network
  - 2** Mining and Consensus
  - 3** Natural Fork
  - 4** Hard Fork
  - 5** Soft Fork

1

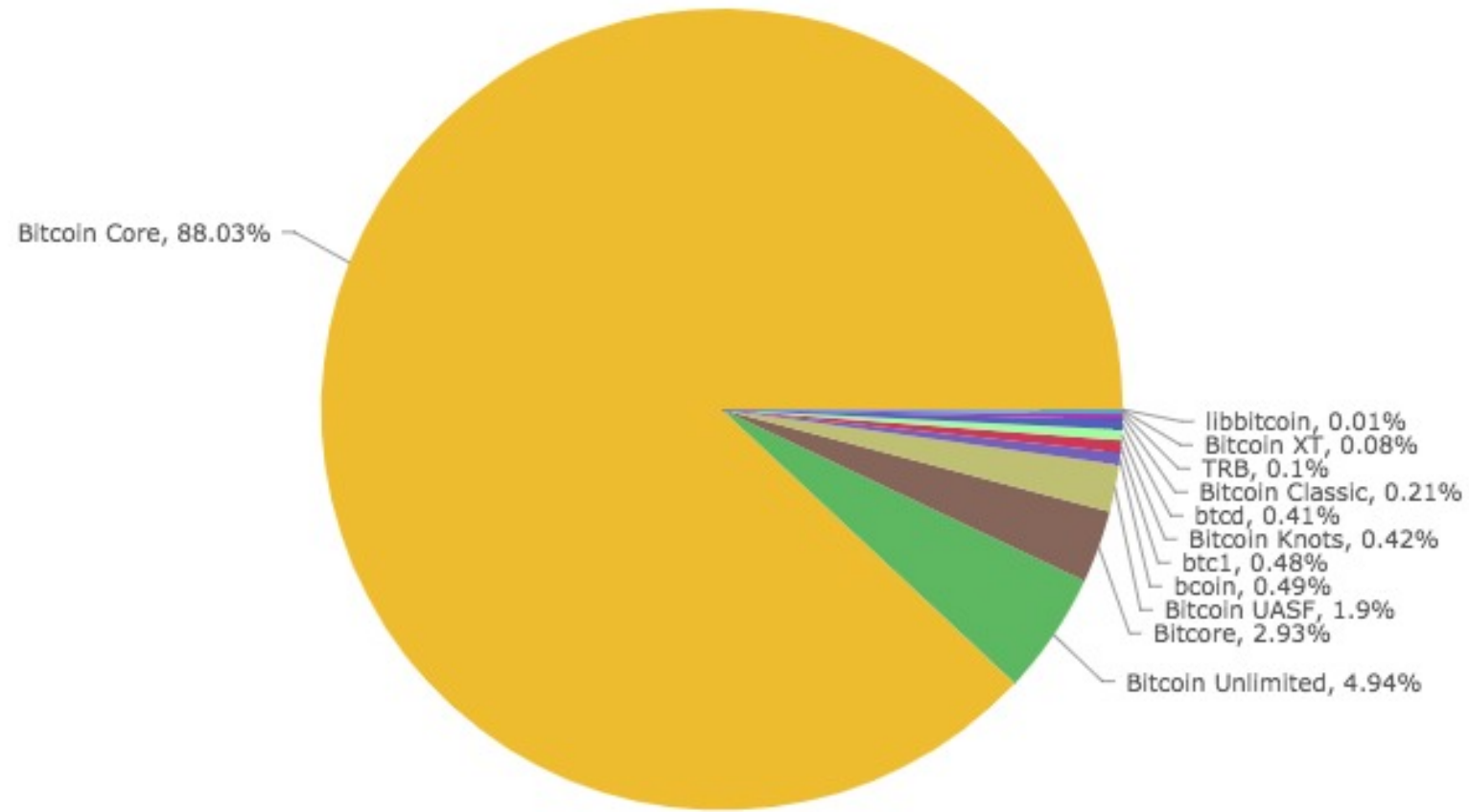
# **THE BITCOIN NETWORK**

# PROTOCOL & IMPLEMENTATIONS

- BIPs are a set of specifications that affects Bitcoin in general, not a specific implementation
- But ultimately, the reference client is the specification
  - The Bitcoin protocol is specified by the behavior of the reference client
  - Consensus is determined by the software that the majority of the network runs
  - Consensus is not determined by a natural language specification
- And a few other implementations
  - Btcd (Go)
  - BitcoinJ (Java)
  - Bcoin (Javascript)
  - Parity-bitcoin (Rust)
  - ...

# Bitcoin Nodes (2018-01-31)

coin.dance



Bitcoin Core  
Bitcoin Knots  
libbitcoin

Bitcoin Unlimited

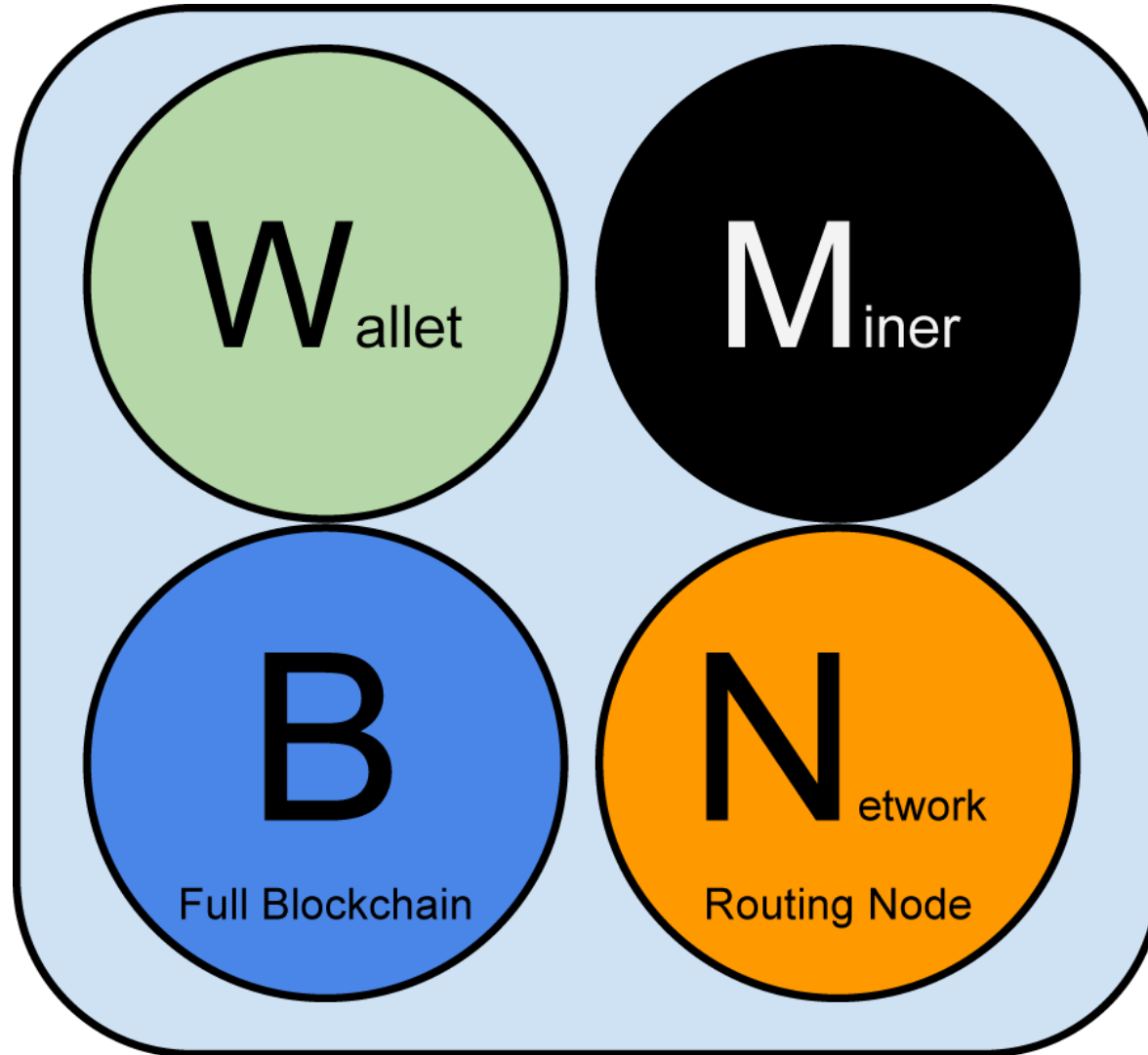
Bitcore  
btcd

Bitcoin UASF  
Bitcoin Classic

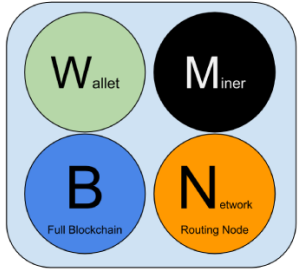
bcoin  
TRB

btc1  
Bitcoin XT

# The four functions of a node

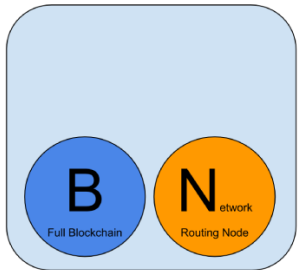






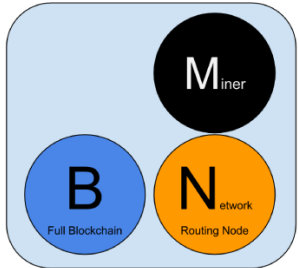
## Reference Client (Bitcoin Core)

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



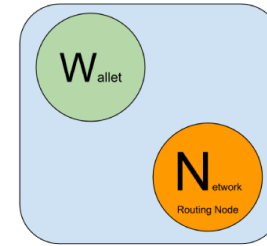
## Full Block Chain Node

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



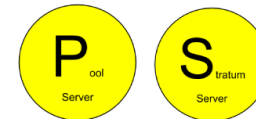
## Solo Miner

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.



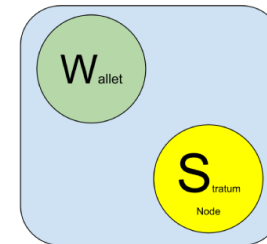
## Pool Protocol Servers

Gateway routers connecting the bitcoin P2P network to nodes running other protocols such as pool mining nodes or Stratum nodes.



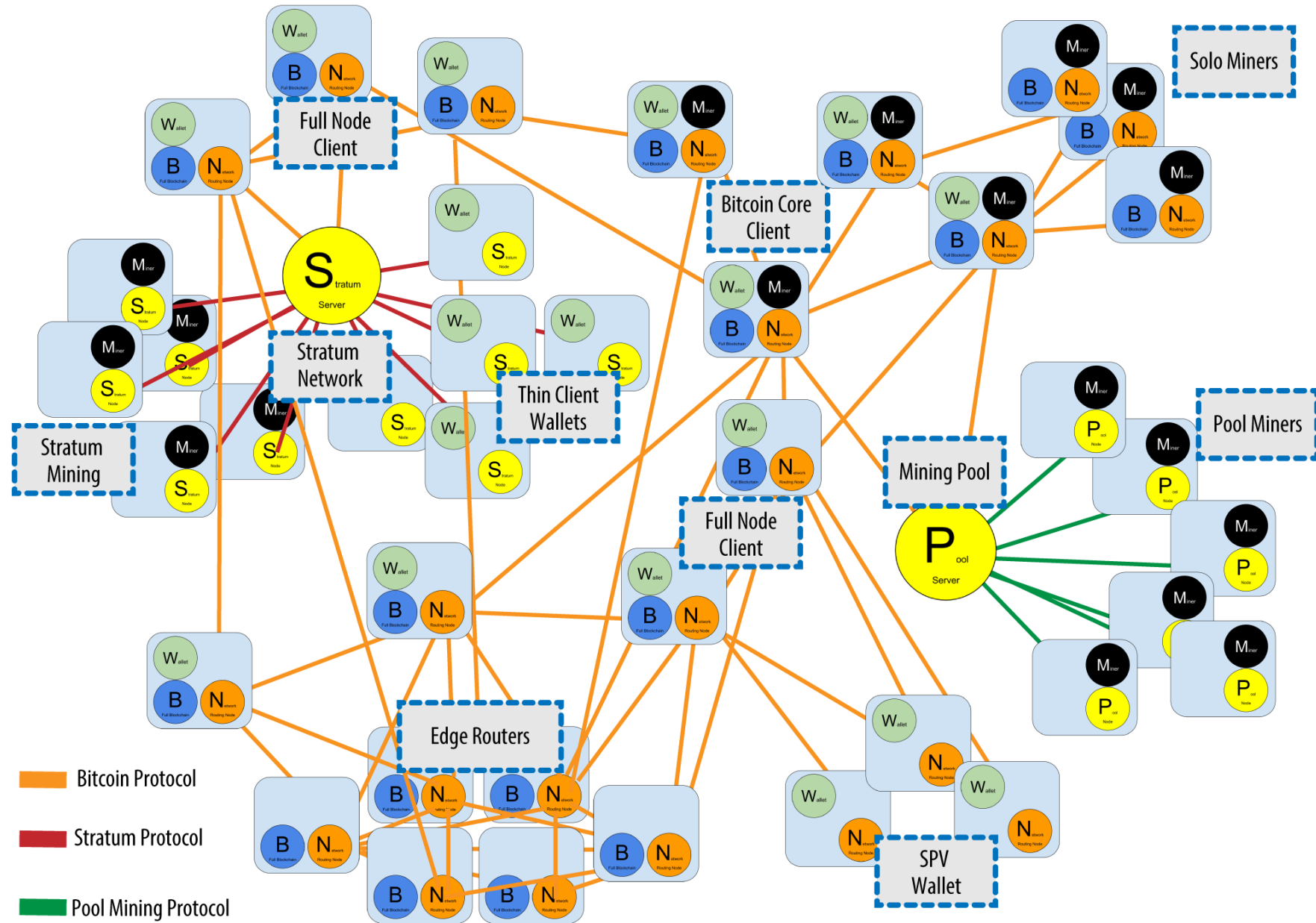
## Mining Nodes

Contain a mining function, without a blockchain, with the Stratum protocol node (S) or other pool (P) mining protocol node.



## Lightweight (SPV) Stratum wallet

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



# RECAP

- Multiple software implementations
- Multiple node functionalities / roles
- Each node has its own view of the blockchain and the network

2

## **MINING AND CONSENSUS**

# MINING

- Proof-of-Work consensus algorithm
- Secures the Bitcoin system
  - Bitcoin hash rate currently around 20 exahash/s
- Currency creation
- Enables the emergence of network-wide *consensus without a central authority*

Last login: Sat Feb 3 15:16:02 on ttys000

# steph @ MBP-de-stepthane in ~ [15:50:27]

\$ echo -n "I am Satoshi Nakamoto" | shasum -a 256  
5d7c7ba21cbbcd75d14800b100252d5b428e5b1213d27c385bc141ca6b47989e -

# steph @ MBP-de-stepthane in ~ [15:50:52]

\$ echo -n "I am Satoshi Nakamoto1" | shasum -a 256  
f7bc9a6304a4647bb41241a677b5345fe3cd30db882c8281cf24fbb7645b6240 -

# steph @ MBP-de-stepthane in ~ [15:51:00]

\$ echo -n "I am Satoshi Nakamoto2" | shasum -a 256  
ea758a8134b115298a1583ffb80ae62939a2d086273ef5a7b14fbfe7fb8a799e -

# steph @ MBP-de-stepthane in ~ [15:51:07]

\$ echo -n "I am Satoshi Nakamoto3" | shasum -a 256  
bfa9779618ff072c903d773de30c99bd6e2fd70bb8f2cbb929400e0976a5c6f4 -

# steph @ MBP-de-stepthane in ~ [15:51:13]

\$ echo -n "I am Satoshi Nakamoto4" | shasum -a 256  
bce8564de9a83c18c31944a66bde992ff1a77513f888e91c185bd08ab9c831d5 -

# BLOCK HEADER STRUCTURE

Size	Field	Description
4 bytes	Version	A version number to track software/protocol upgrades
32 bytes	Previous Block Hash	A reference to the hash of the previous (parent) block in the chain
32 bytes	Merkle Root	A hash of the root of the merkle tree of this block's transactions
4 bytes	Timestamp	The approximate creation time of this block (seconds from Unix Epoch)
4 bytes	Target	The Proof-of-Work algorithm target for this block
4 bytes	Nonce	A counter used for the Proof-of-Work algorithm

# DECENTRALIZED CONSENSUS

- Trust model: centralized vs decentralized
- Emergent consensus (arising casually or unexpectedly)
- Independent nodes following the same set of rules
- Independent nodes assembling their own copy of the authoritative, trusted, public, global ledger



# INTERPLAY OF FOUR PROCESSES

- Independent transaction verification
- Independent transaction aggregation into *candidate blocks* + PoW
- Independent block verification and assembly into the chain
- Independent selection of the greatest-cumulative-work valid chain

# RECAP

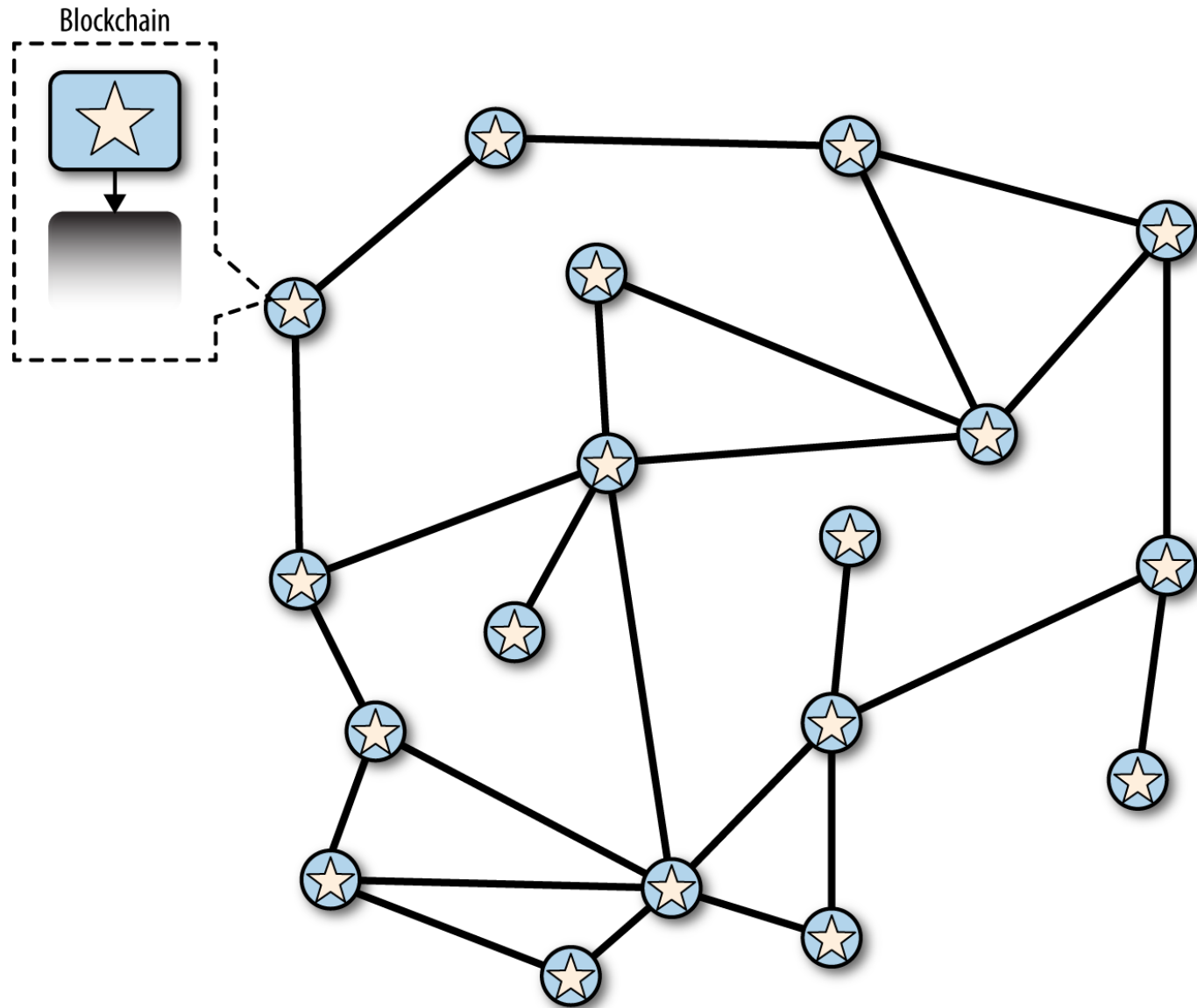
- PoW mining, independent validation and selection of the longest chain work together to produce *decentralized consensus*
- The consensus mechanism depends on having a majority of the miners acting honestly out of self-interest

3

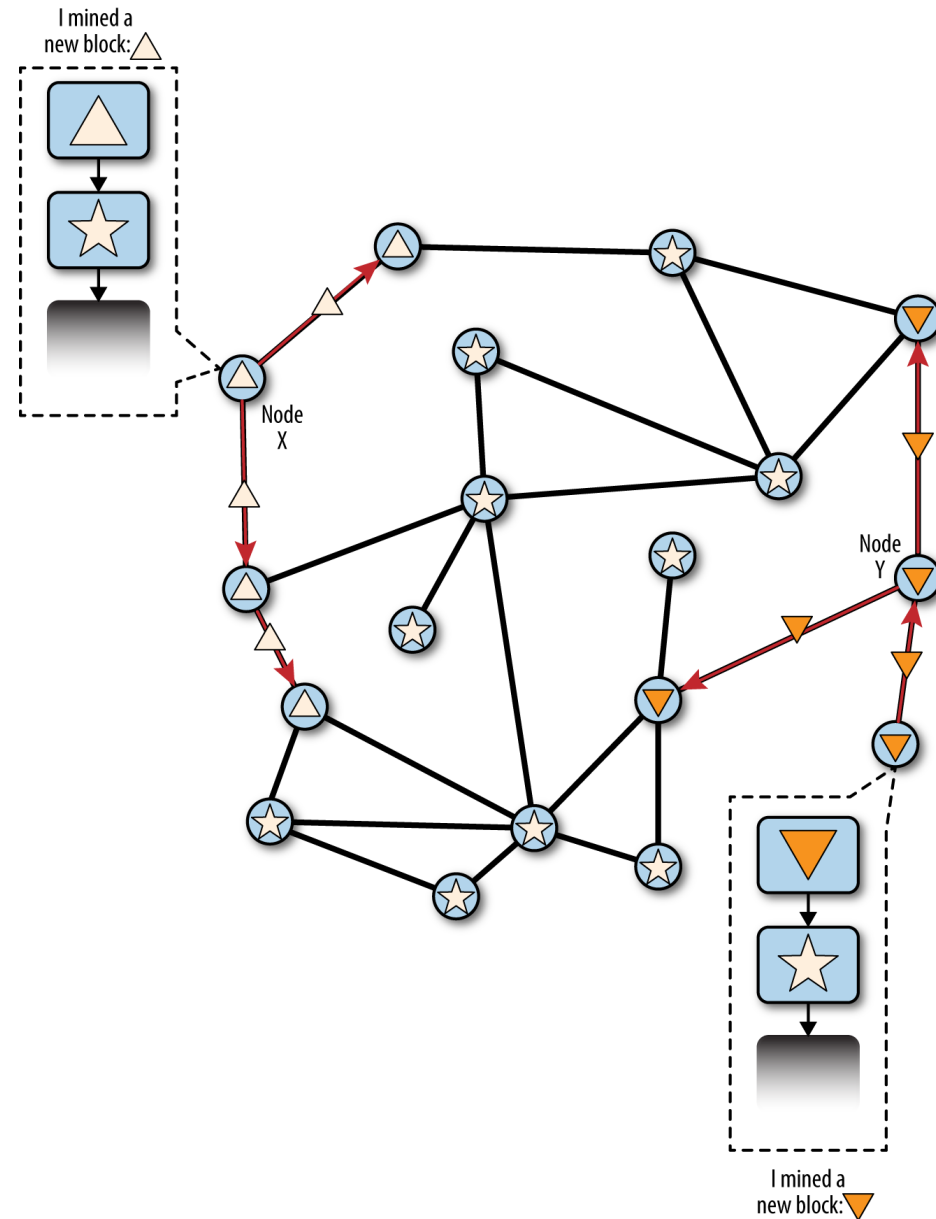
**NATURAL FORKS**

- Two *candidate blocks* extends the same *parent*, competing to form the longest blockchain
- Occurs naturally as a result of transmission delays
- Resolved by the independent selection of the *greatest-cumulative-work* chain

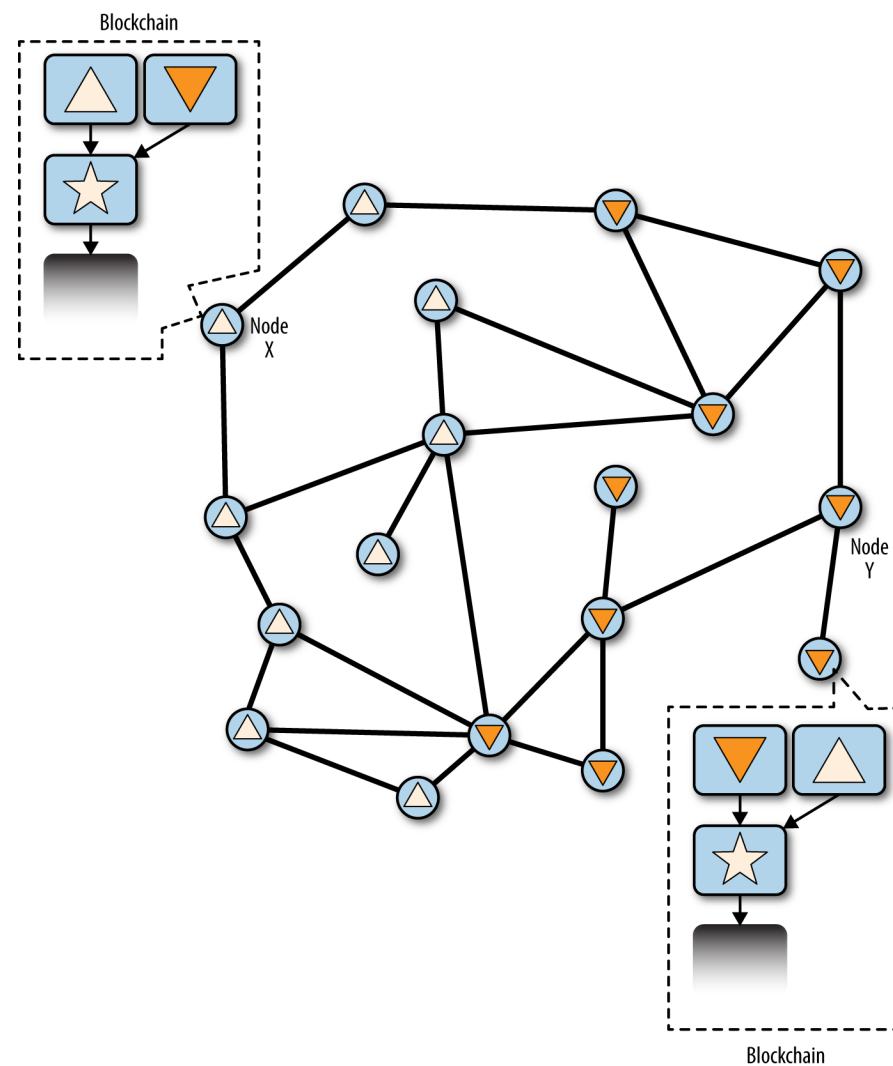
Before the fork—all nodes have the same perspective



# Blockchain fork event: two blocks found simultaneously

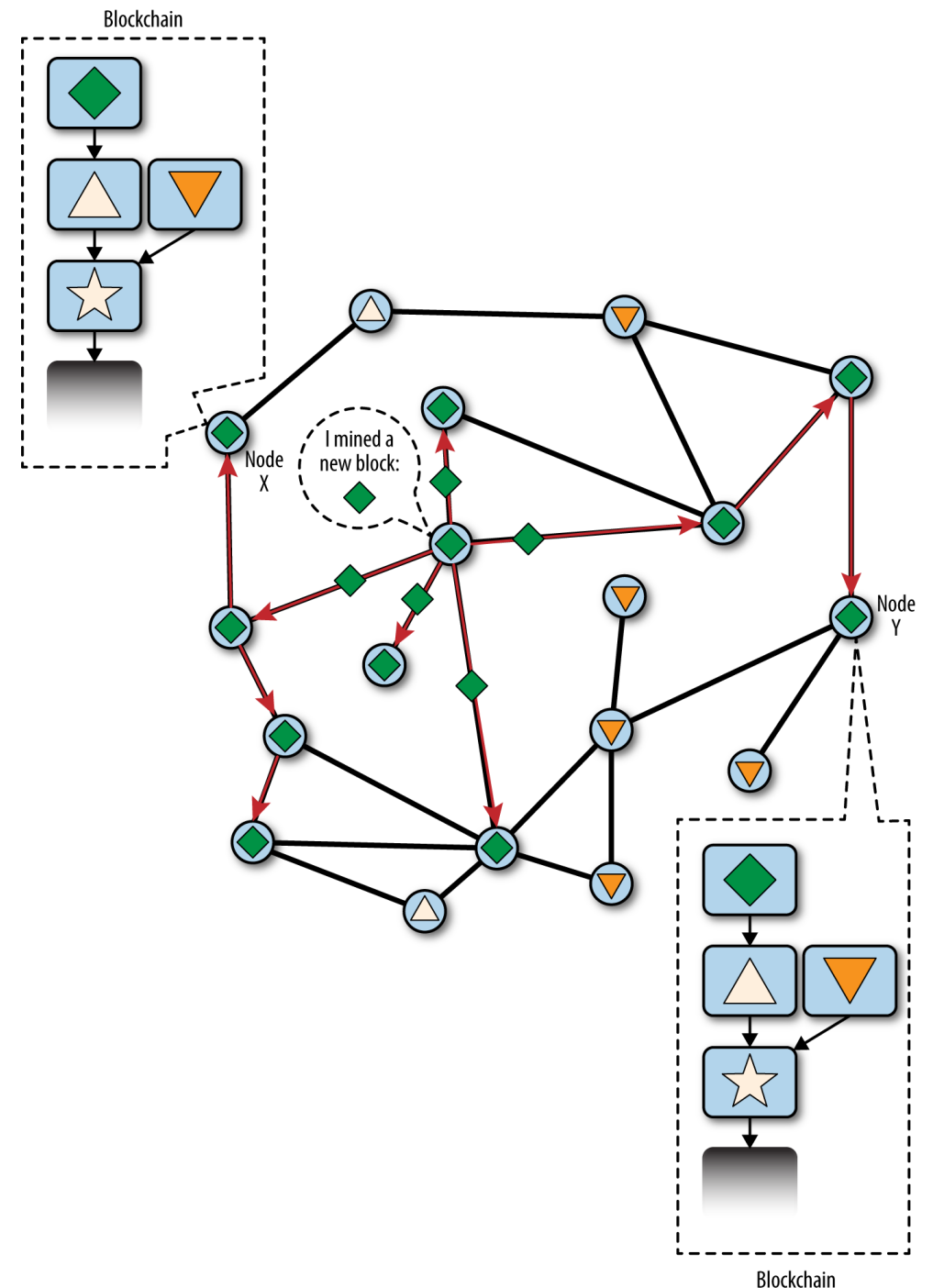


# Blockchain fork event: two blocks propagate, splitting the network



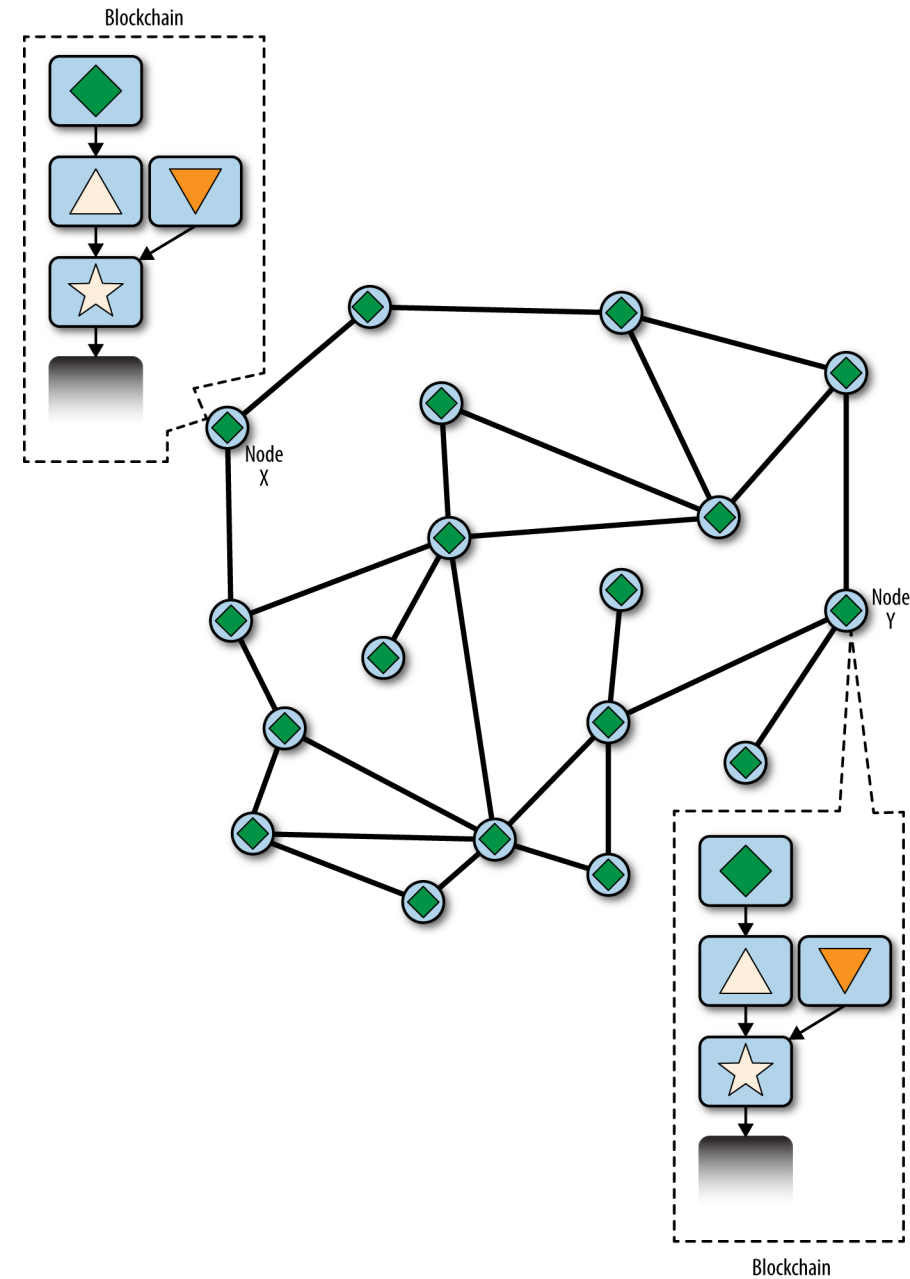
A new block extends one fork, reconverging the network

- Some nodes are forced to revise their view of the blockchain (chain reconvergence/reorganization)





The network reconverges on a new longest chain



# RECAP

- Forks happens regularly and naturally
- Usually resolved after 1 block
- Bitcoin's block interval of 10 minutes is a design compromise between fast confirmation times (settlement of transactions) and the probability of a fork

4

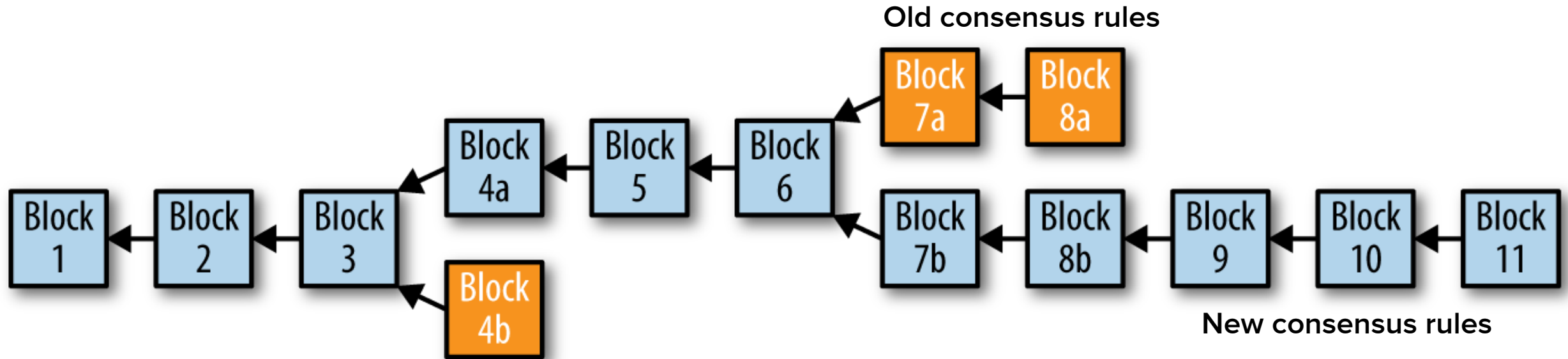
**Hard Fork**

# REASONS CONSENSUS RULES MAY DIFFER

- Validation of transactions or blocks
- Implementation of Bitcoin scripts or cryptographic primitives (ECDSA, ...)
- *Implicit consensus constraints* imposed by system limitations or implementation details (database, ...)

- **Forward-incompatible: non-upgraded nodes will find themselves in a partitioned network**
- **Seen as risky because they force a minority to either upgrade or remain on a minority chain**
- **Highly controversial in the Bitcoin development community**

# HARD FORK



# FOUR STAGES

Software fork



Network fork



Chain fork



Mining fork

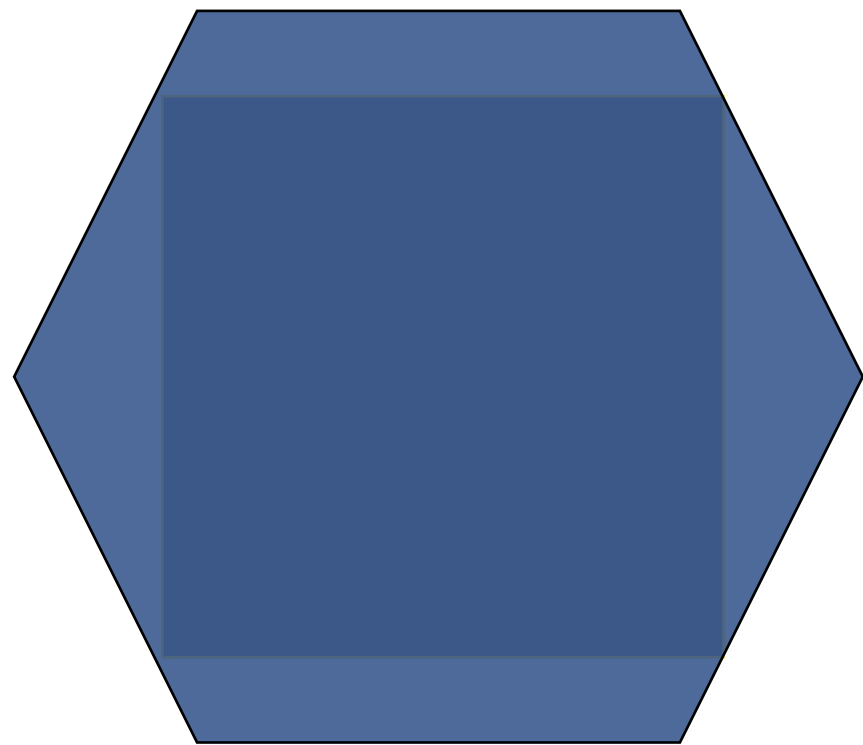
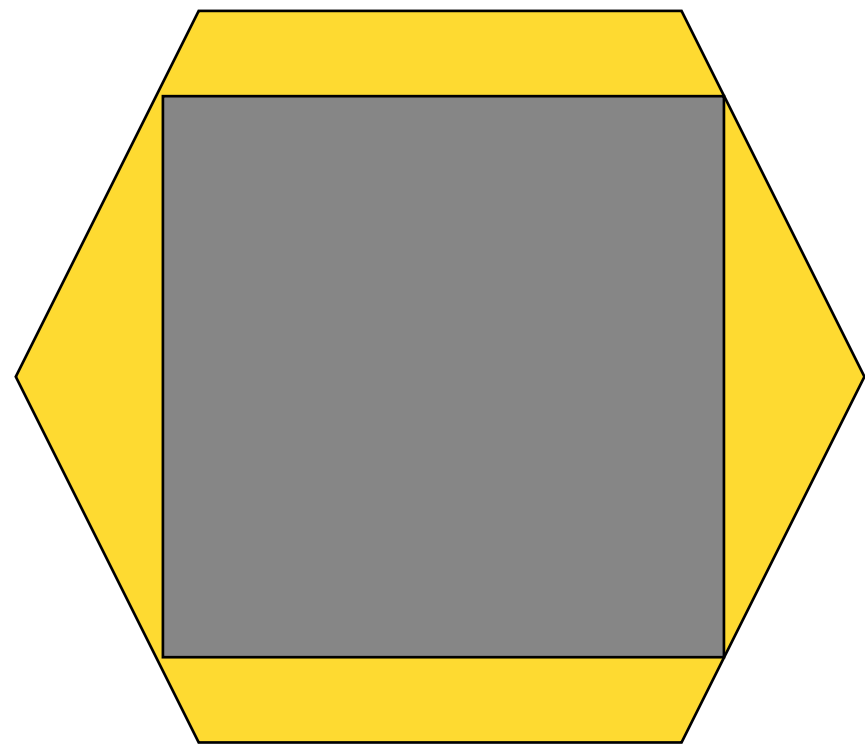
5

**Soft Fork**



- **Not backward-compatible: upgraded nodes must consider tx and blocks created under the old rules as invalid**
- **Forward-compatible: non-upgraded nodes must consider tx and blocks created under the new rules as valid**
- **The new rules can only limit what is valid, constrains the consensus rules (mostly re-interpretation of NOP opcodes)**
- **Requires a majority of the miners upgrading to enforce the new rules (not 100%)**

If consensus rules were shapes



# SOFT FORK SIGNALING AND ACTIVATION

- Miners signal readiness
- New rules activated after a threshold
- **IsSuperMajority / BIP-34**
  - Block version as an integer
  - Used for the version 2, 3 (BIP-66 / Strict DER) and 4 (BIP-65 / CLTV) upgrades
- **BIP-9**
  - Block version as a bit field
  - Much more flexible
  - Used for CSV and associated BIPs 68, 112, 113

# CRITICISMS OF SOFT FORKS

- Technical debt
- Validation relaxation
- Irreversible upgrades (without Hard Fork)

# CONCLUSION

- Decentralized consensus made by
  - Independent nodes following a set of rules
  - Mining process (which is part of the rules)
- A blockchain/coin is defined by the sum of its consensus rules
- Consensus software development is difficult and tricky
- Soft or Hard Fork = tradeoffs