Blockchain and Cryptocurrencies

Week 5 — Chapter 3 (part 2): Mechanics of Bitcoin

Prof. Dr. Peter Thiemann

Albert-Ludwigs-Universität Freiburg, Germany

SS 2020

Mechanics of Bitcoin

draws on material from

- Bitcoin and Cryptocurrency Technologies
- Bitcoin, Blockchain, and Cryptoassets
- Antonopoulos: Mastering Bitcoin

Contents

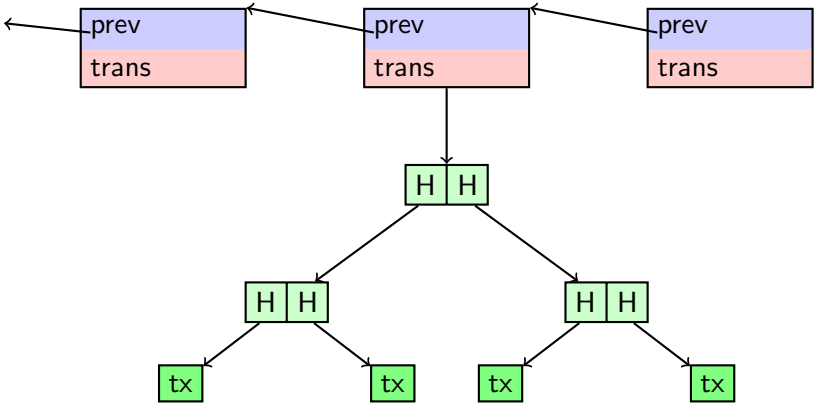
Blocks

Why blocks?

- transactions are grouped in blocks of 1 MB
- mainly for efficiency:
  - it keeps the chain shorter
  - it helps to maintain a reasonable transaction rate

(10 min für Consensus!)

Structure of a Block



hash chain of block headers

Merkle tree of transactions

Block Header (binary format)

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	Difficulty Target	The Proof-of-Work algorithm difficulty target for this block
4 bytes	Nonce	A counter used for the Proof-of-Work algorithm

- only the block header is hashed!

Coinbase Transactions

1. minting new bitcoins
2. single input, single output
3. input does not redeem previous output $\Rightarrow$ hash all zeroes
4. as of June 2020, the output is 6.25 BTC plus transaction fees
5. there is a special coinbase parameter which is arbitrary When the nonce is not possible

Contents

The Bitcoin Network

- Bitcoin relies on a **peer-to-peer network**

The Bitcoin Network

- Bitcoin relies on a **peer-to-peer network**
- It consists of peers (nodes) having equal rights
  $\Rightarrow$ no notion of servers or clients

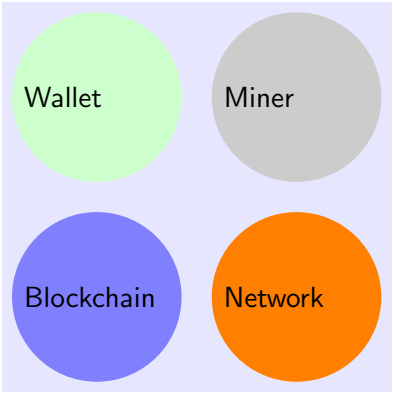
The Bitcoin Network

- Bitcoin relies on a **peer-to-peer network**
- It consists of peers (nodes) having equal rights
  $\Rightarrow$ no notion of servers or clients
- Nodes are connected, not necessarily by a direct link

The Bitcoin Network

- Bitcoin relies on a **peer-to-peer network**
- It consists of peers (nodes) having equal rights
  $\Rightarrow$ no notion of servers or clients
- Nodes are connected, not necessarily by a direct link
- An overlay network independent of the underlying network (i.e., the Internet)

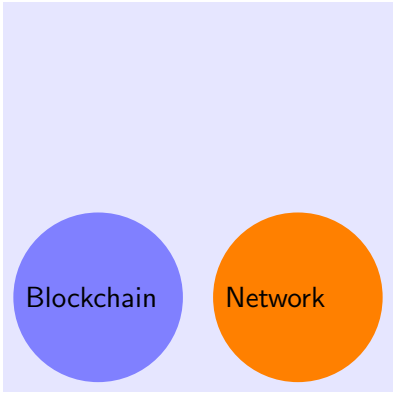
Nodes Types and Roles



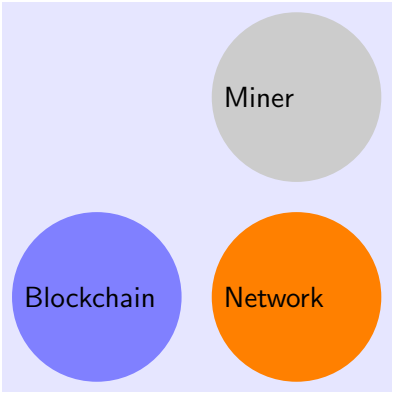
The **reference client** (Bitcoin core) includes all roles

- Wallet — safe keeping of private keys for end users
- Miner — minting new bitcoins by creating new blocks
- Blockchain — verification of block traffic, keeping a copy of the blockchain
- Network — routing messages, maintaining connectivity needed for every node

Full node

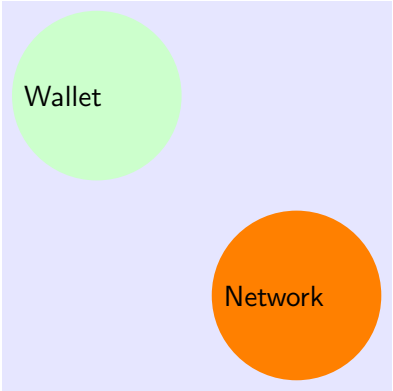


Solo miner



- Alternative: miner farms
- clients do just mining and networking
- coordinated by a (full node) server
- miner farms use a different protocol

Light Weight (SPV) Wallet



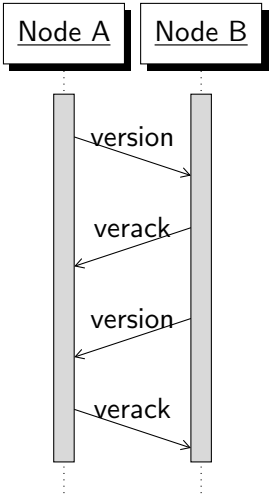
- SPV = Simplified Payment Verification
- only block headers are kept locally, not the full chain

$\Rightarrow$ 1000x decrease in storage

- do not have full picture of all UTXOs
- need to obtain transaction data from surrounding full nodes to verify, but is prone to double spending attacks
- asking for specific transactions can compromise anonymity $\Rightarrow$ Bloom filters

Contents

Network Discovery

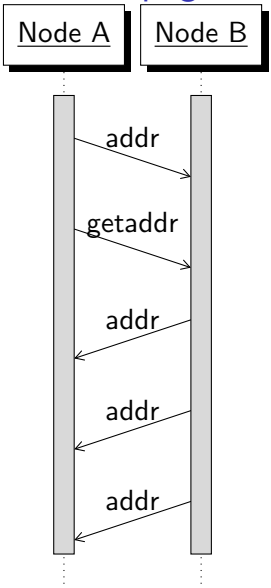


the version message for initial contact with at least one bitcoin node

```
{"PROTOCOL_VERSION": 70015,
 "nLocalServices": 1033,
 "nTime": 1591903131,
 "addrYou": "122.51.104.28",
 "addrMe": "82.38.163.188",
 "subver": "/Satoshi:0.20.0/",
 "BestHeight": 634259
}
```

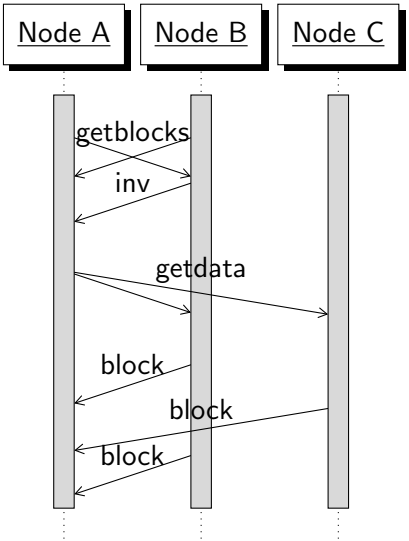
second part is optional

Address Propagation and Discovery



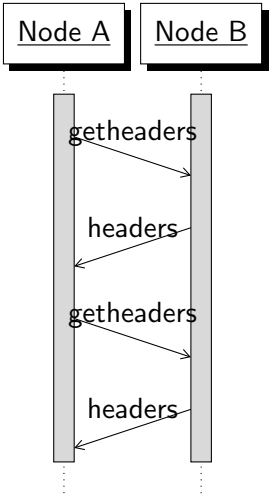
- When B receives A's getaddr message, it randomly selects up to 2500 recent addresses from its pool and sends them to A
- A connection times out after 90 minutes

Node Synchronization



- (new node start with just the genesis block)
- after connection establishment, peers exchange hash of top block
- peer with the longer chain sends inv (up to 300 hashes)
- getdata requests blocks by hash, spreading the load

Header Synchronization (SPV Nodes)



- SPV nodes use getheaders to obtain only the headers
- The headers message contains up to 2000 headers

Contents

SPV can compromise anonymity

- SPV nodes need to retrieve blocks / transactions in connection with the payment they are processing
- This selective download can reveal the addresses in the node's wallet
- To avoid revelation, SPV nodes request transaction data using **Bloom filters**
- A Bloom filter narrows down the set of transactions without revealing the selection criterion

Bloom Filters

Definition

A Bloom filter bf is a probabilistic datastructure that provides a fuzzy encoding of a set. Its operations are

- bf.add(x) includes x in the Bloom filter.
- bf.elem(x) checks membership of x. If it returns False, then x has never been added to the filter. If x has been added, then it returns True.

Bloom Filters

Definition

A Bloom filter bf is a probabilistic datastructure that provides a fuzzy encoding of a set. Its operations are

- bf.add(x) includes x in the Bloom filter.
- bf.elem(x) checks membership of x. If it returns False, then x has never been added to the filter. If x has been added, then it returns True.

Remark

- If bf.elem(x) returns True, x may or may not have been added.
- A negative result is definitive.

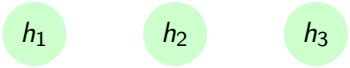
Implementation of a Bloom Filter

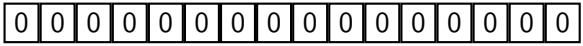
- bitvector of size m, initialized to all 0
- k different independent hash functions $h_i : A \to \{0, \ldots, m-1\}$
- to add x
  - compute $y_i = h_i(x)$ for all k hash functions
  - set bits $y_1, \ldots, y_k$
- to check (potential) presence of x
  - compute $y_i = h_i(x)$ for all k hash functions
  - return False if any of the bits y_i is 0
  - otherwise return True
- the false positive rate

$$\left(1 - e^{-kn/m}\right)^k$$

depends on the number n of elements that have been added to the filter

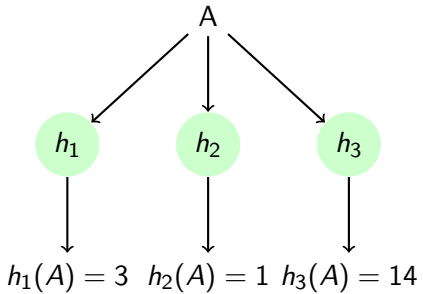
Example

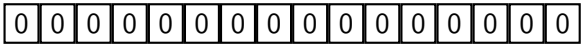




- Bloom filter with three hash functions

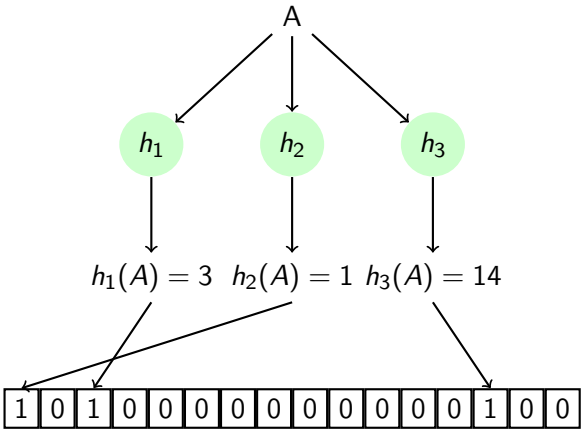
Example





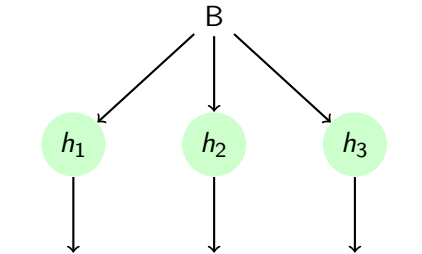
- Bloom filter with three hash functions
- add A

Example



- Bloom filter with three hash functions
- add A

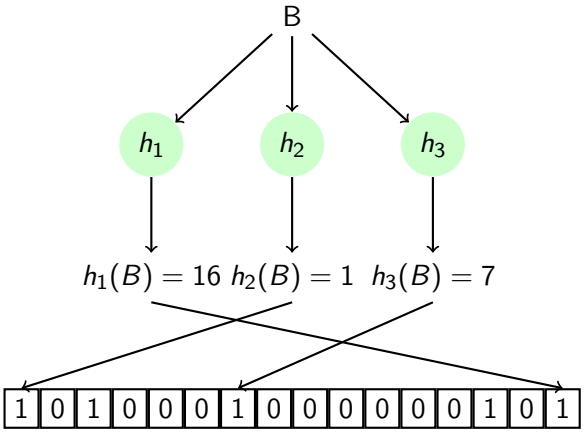
Example



1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0

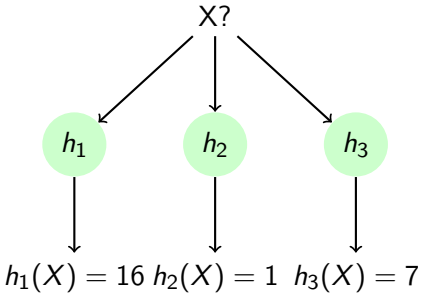
- Bloom filter with three hash functions
- add A
- add B

Example



- Bloom filter with three hash functions
- add A
- add B

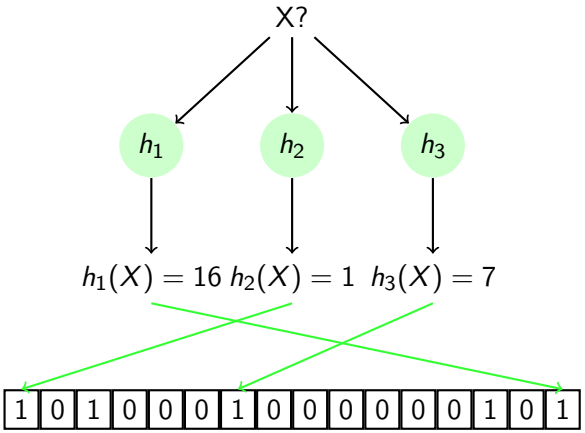
Example



1	0	1	0	0	0	1	0	0	0	0	0	0	1	0	1

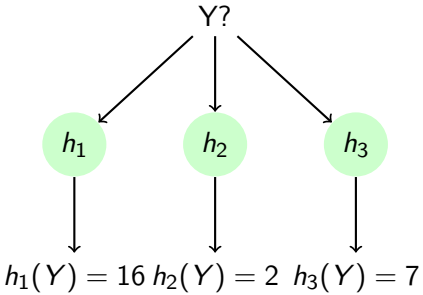
- Bloom filter with three hash functions
- add A
- add B
- check for X

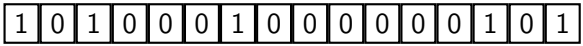
Example



- Bloom filter with three hash functions
- add A
- add B
- check for X
- may be an element

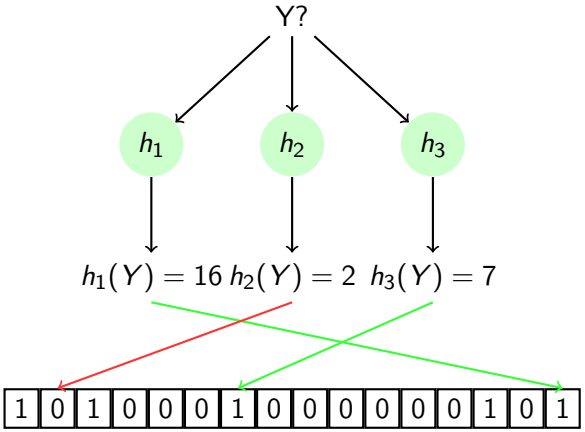
Example





- Bloom filter with three hash functions
- add A
- add B
- check for X
- may be an element
- check for Y

Example



- Bloom filter with three hash functions
- add A
- add B
- check for X
- may be an element
- check for Y
- definitely not an element

Pools

- (in main memory of the node)
- Every node maintains a **transaction pool** of unconfirmed, validated transactions
- These are propagated to the neighbors and may be different on each node
- Every node maintains a pool of UTXOs (confirmed, valid outputs)
- They represent the consensus of the longest chain

Thanks!