Principals and Practice of Cryptocurrencies

Cornell CS 5437, Spring 2016

The Bitcoin-Core Client
Overview

• Specific – we are talking about a single implementation of a specific protocol
  • The reference client
• General – similar data structures appear in any similar protocol implementation
• Inaccurate – details change between versions

• Mastering Bitcoin, chapters 3 and 6
Node Roles

- Propagation
  - Transactions
  - Blocks
- Validation
- Mining

Full node

Miner
Data structures - Transaction

The transaction data type describes a single transaction, either accepted or not.

- A vector of inputs
- A vector of outputs
- Version
- Lock time
Data structures – UTXO Set

Inputs
Spent outputs
Unspent outputs
Data structures – The mempool

The memory pools contains transactions that were not placed in a block (yet).

• Only with valid inputs (possibly still in mempool)
• Limited size (soon)
## Data structures – Block

<table>
<thead>
<tr>
<th>Size [byte]</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Magic number: 0xD9B4BEF9</td>
</tr>
<tr>
<td>4</td>
<td>Block size [bytes]</td>
</tr>
<tr>
<td>80</td>
<td><strong>Header</strong></td>
</tr>
<tr>
<td>1-9</td>
<td>Transaction count</td>
</tr>
<tr>
<td>?</td>
<td>Transactions</td>
</tr>
</tbody>
</table>

*primitives/block.h*
## Data structures – Block Header

<table>
<thead>
<tr>
<th>Size</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>[byte]</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Version</td>
</tr>
<tr>
<td>32</td>
<td>Hash (SHA256$^2$) of previous block header</td>
</tr>
<tr>
<td>32</td>
<td>Hash (Merkle root) of blocks’ transactions</td>
</tr>
<tr>
<td>4</td>
<td>UNIX timestamp</td>
</tr>
<tr>
<td>4</td>
<td>Proof-of-Work target</td>
</tr>
<tr>
<td>4</td>
<td>Nonce</td>
</tr>
</tbody>
</table>

A block is legal if the hash (SHA256$^2$) of its header is small enough, as specified by the target field.

`primitives/block.h`
Data Structures – Block Index

Block Index is an internal data structure that connects the blocks to form a Blockchain.
Data Structures – Chain

CChain is a full chain, as a vector for fast access

Often instantiated as ActiveChain with the active chain
Useful Genesis and Tip methods
Chain Reorganization

When a client learns on a better chain, it reorganizes the Blockchain.

This is done in steps:
1. Remove block 3 (new head: 2)
2. Add block 3’ (new head: 3’)
3. Add block 4’ (new head: 4’)

chain.h
Wallet

The client serves as a wallet, maintaining the user’s funds.

- **Address**: a public-private key pair
- Aggregated into **accounts**
- Balance calculated by going through blockchain
- Address generation takes time (crypto operations), so done in advance to fill a **pool**
- New address for every output
- Transactions accumulate funds and keep change

Send 2.5 BTC to Bob
1. Version exchange (version, verack)
   - Node version and basic state
2. Network maintenance
   - Address exchange (addr, getaddr)
   - Link maintenance (ping, pong)
   - Message rejection (reject)
   - Message filtering
   - Alert
3. Data exchange
   - Publish what blocks/transactions a client has (inv)
   - Ask for blocks/transactions/inv (getdata, getblocks, getheaders, mempool)
   - Send data (block, tx, headers)
Modes of Operation

1. **Mainnet**
   Bitcoin’s actual network. Not for experiments. Expensive mining, expensive transactions, huge Blockchain.

2. **Testnet**
   For experiments. Large network, but no value to coins. Low difficulty, reset.

3. **Regtest**
   For basic testing, free mining. No network – connects nowhere.
Modes of Operation

1. Mainnet
2. Testnet
3. Regtest

What’s the difference?

• network level
  • TCP port
  • magic number
• Consensus level
  • Genesis block
  • addresses
• Difficulty
  • Mainnet difficulty updates every 2 weeks
  • Testnet difficulty has an auto-reset
  • Regtest difficulty doesn’t update
Interacting with the Client – Bootstrapping

- Data directory
  - Blocks
  - Blockchain data (block index)
  - Wallet
  - Configuration file

- Initialization arguments
  - Configuration file

Use multiple local clients with different directories and carefully planned config files to run a regtest network on a single machine
Interacting with the Client – Bootstrapping

• Configuration file

testnet=0
addnode=69.164.218.197 # Also look for this node
cconnect=10.0.0.1:8333 # Only look for this node
maxconnections=125 # incoming + outgoing

server=1 # Accept RPC
rpcuser=myName
rpcpassword=CHOOSE_SMART!
rpcallowip=10.1.1.34
rpcport=8332
Interacting with the Client – RPC

- `sendtoaddress (...)`
- `sendfrom (...)`: Send funds from account to address
- `createrawtransaction (tx details)`
- `getaddressesbyaccount (account)`
- `getbalance`: in all accounts
- `getbestblockhash`: hash of chain head
- `getblockcount`: length of main chain
- `getblockhash (index)`
- `getblock (block hash)`
- `getrawmempool`: Get transaction IDs in mempool
- `gettransaction (tx ID) (index appropriately for all txns)`
- `setgenerate (generate, procLimit)`: procLimit is number of processors to use, or number of blocks to generate in regtest

getblock response (not real)

```json
{
"hash": "0000000000fe549a89848c76070d4132872cfb6efe5315d01d7ef77e4900f2d39",
"confirmations": 88029,
"size": 189,
"height": 227252,
"version": 2,
"merkleroot": "c738fb8e22750b6d3511ed0049a96558b...46f3f77771ec825b22d6a6f4a",
"tx": ["c738fb8e22750b6d3511ed0049a96558b0bc57046f3f77771ec825b22d6a6f4a"],
"time": 1398824312,
"nonce": 1883462912,
"bits": "030a2b4a",
"difficulty": 120,033,340,651.24,
"chainwork": "00000...00000000000000000000000000000000083ada4a4009841a",
"previousblockhash": "000000000c7f4990e6ebf71ad7e21a4713...05b3998d7a814c011df",
"nextblockhash": "00000000afe1928529ac766f1237657819a11cfc...f119e868ed5b6188"
}
```
getrawtransaction response (testnet)

```json
{
    "hex" : "0100000001268a9ad7bfb21d3c086f0ff28f73a064964aa069ebb69a9e4...,"
    "txid" : "ef7c0cbf6ba5af68d2ea239bba709b26ff7b0b669839a63bb01c2cb8e8...,"
    "version" : 1,
    "locktime" : 0,
    "vin" : [...]
    "vout" : [...]
    "blockhash" : "00000000103e0091b7d27e5dc744a305108f0c752be249893c749...,"
    "confirmations" : 88192,
    "time" : 1398734825,
    "blocktime" : 1398734825
}
```
getrawtransaction response (testnet)

"vin" : [{
    "txid" : "d7c7557e5ca87d439e9ab6eb69a04a9664a0738ff20f6f083c1db2...",
    "vout" : 0,
    "scriptSig" : ...,  
    "sequence" : 4294967295
}]

"vout" : [{
    "value" : 0.39890000,  
    "n" : 0,
    "scriptPubKey" : ...
}]}
Interacting with the Client – RPC

• Directly: CLI with bitcoin-cli executable

> bitcoin-cli getrawtransaction a9d4599e15b53f3eb531608dddb31f48c...

{  
    "hex" : "0100000001344630cbff61fbc362f7e1ff2f11a344c29326e4ee9...",  
    "txid" : "a9d4599e15b53f3eb531608dddb31...",  
    "version" : 1,  
    "locktime" : 0,  

• JSON over HTTP
  • Better with a wrapper