Table of contents

- Network architecture
- Node types
- Network discovery
- Full node
- Lightweight node

Network architecture

- Bitcoin network
    - Peer-to-Peer (P2P) network
        - No server/connection hierarchy/centralized node
        - All peers provide and consume services at same
    - Flat mesh topology
    - Originally, Bitcoin network = A collection of nodes running Bitcoin P2P protocol

Network architecture

- Bitcoin network
  - Additional protocols (e.g., Stratum) came to support lightweight/mobile wallets
    - Enabled by gateway routing servers
      - Bridge pool miners to Bitcoin P2P
  - Extended Bitcoin Network = Bitcoin P2P protocol + pool mining protocols + Stratum protocol

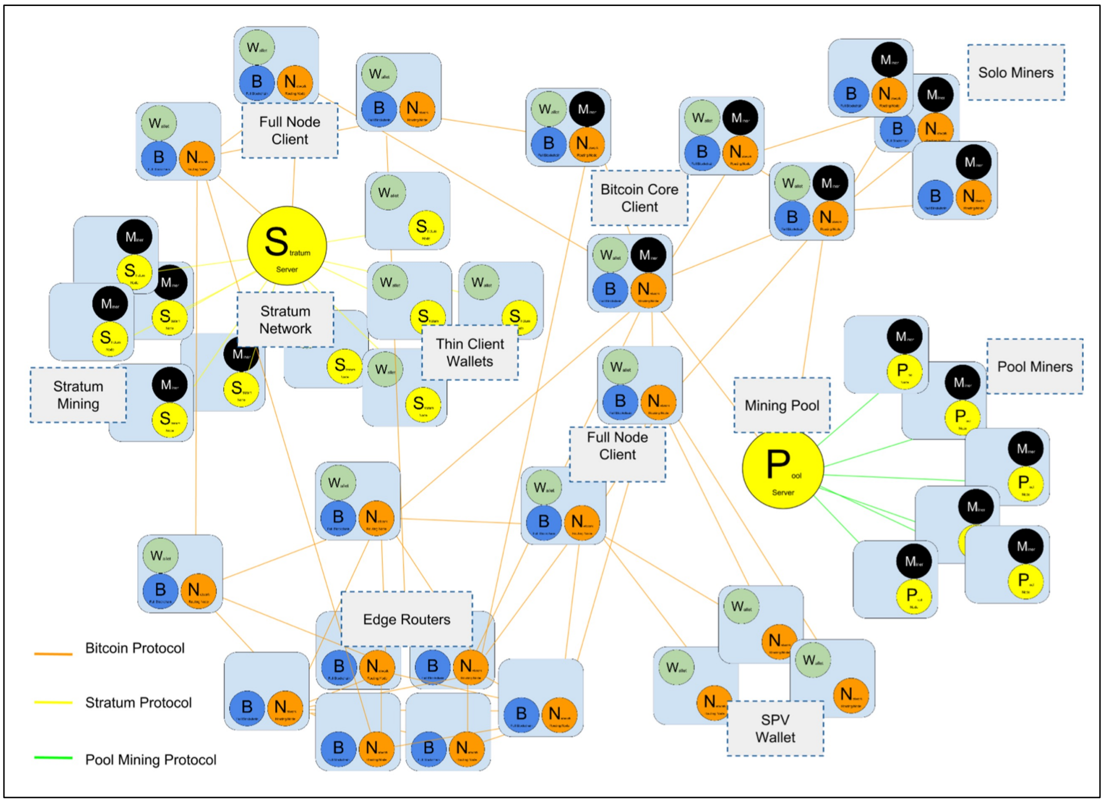
Node types

Node	Functions	Notation

Reference client (Bitcoin Core or Satoshi Client)	W+M+B+N	W = Wallet
		M = Miner
		B = Full blockchain
		N = Network/routing node
		P/S = Pool_mining/Stratum

Full node client	W+B+N	
Full blockchain nodes	B+N	
Solo miner	M+B+N	
Lightweight (SPV) wallet	W+N	
Pool protocol servers	P/S server (gateway routers)	
Mining nodes	M+P/S node	
Lightweight (SPV) Stratum wallet	W+S node	

Node types



Network discovery

- A booting up node must discover & connect with at least one existing Bitcoin node on network
  - Geographic location of nodes is irrelevant
  - Existing nodes are selected at random
- Establish a TCP connection to a known peer
  - On port 8333 (Bitcoin "known" port) or other provided port
- After connecting, start a "handshake" via *version* message

Network discovery

- The *version* message includes
  - *PROTOCOL_VERSION,* constant defining protocol version of initiator
  - *nLocalServices,* list of services supported by initiator, NODE_NETWORK for now
  - *nTime,* current time
  - *addrYou,* IP address of remote node as seen by initiator
  - *addrMe,* IP address of initiator, as discovered by initiator
  - *subver,* subversion showing type of software running on initiator
  - *BestHeight,* block height of initiator's blockchain

- Remote node responds with *verack* and establishes a connection
  - Optionally, sends its own *version* message

Network discovery

- How to find peers?
  - No special nodes!

  - Seed nodes
    - Long running stable nodes are listed in clients
    - Not necessary to connect to seed nodes
      - Use them to discover other nodes

  - Give an IP address to booting node

Network discovery

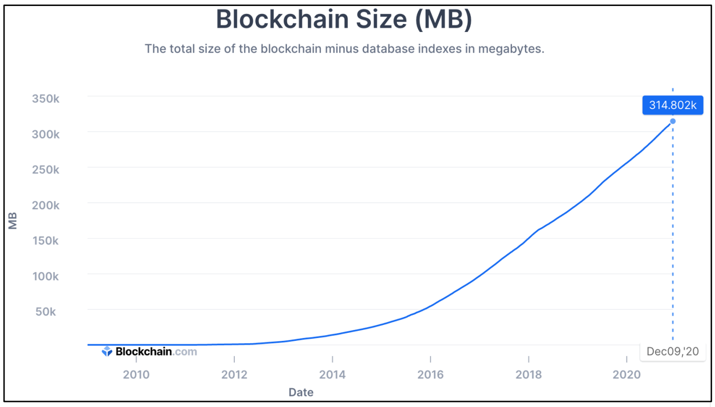
- How to find peers?
  - After one or more connections are established
    - New node sends *addr* message (own IP address) to neighbors
    - Neighbors gossip *addr* message to their neighbors
      - Announcement, visibility
    - New node can also send *getaddr* message to neighbors
      - To get a list of IP addresses of other peers

Full node

- Full node
  - Maintains a complete and up-to-date copy of blockchain with all Tx
  - Independently builds and verifies each block
    - Starting from the genesis block
    - Up to the latest known block in network
    - Keep on building

Full node

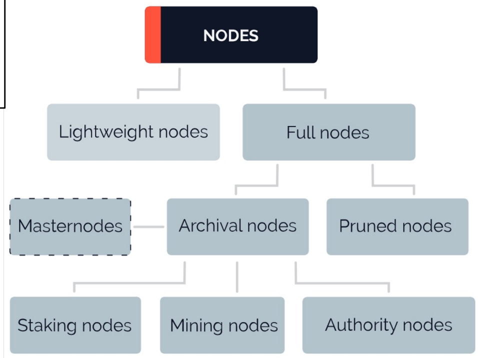
- Full node
  - GB of persistent storage required



Blockchain Size (MB)

The total size of the blockchain minus database indexes in megabytes.

  - Originally, each node was a full node
    - New forms of clients have emerged



NODES

Lightweight nodes

Full nodes

Masternodes

Archival nodes

Pruned nodes

Staking nodes

Mining nodes

Authority nodes

Lightweight node

- Simplified Payment Verification (SPV)
  - Many devices (e.g., smartphones) are space-/power-constrained
    - Can't store full blockchain
  - SPV enables such devices to work without storing full blockchain
  - Such clients are called SPV/lightweight clients
    - Technique especially used by wallets
  - Different verifications approach
    - Relies on peers for relevant info

Lightweight node

- Simplified Payment Verification (SPV)
    - Verifies Tx by reference to their "depth"
        - Instead of their "height" as done by full node

    - When checking a Tx in block #300,000
        - Full node links all 300,000 blocks down to genesis block
        - SPV client verifies it by establishing its depth, say, till block #300,006
            - Since 6 further blocks are built over it (Tx is 6 blocks deep)
                - It isn't a double spend

    - SPV clients
        - Use merkle/authentication path
        - Don’t have all Tx/UTXO record
        - Don't download full blocks, just block headers

Lightweight node

- Simplified Payment Verification (SPV)
    - SPV client is interested in incoming Tx to addresses in its wallet
        - It establishes a bloom filter on its connections to peers
            - Limit received Tx only to addresses of interest

        - When a peer sees a Tx matching bloom filter
            - Sends a *merkleblock* message, which includes
                - Block header (only header info not entire block)
                - Merkle path of Tx of interest to merkle root in that block

        - As a proof of Tx recording in blockchain, SPV node uses
            - Merkle path, proof of inclusion of Tx in block
            - Block header, linking the block to blockchain

Lightweight node

- Simplified Payment Verification (SPV)
  - SPV node consumes <1 KB of data for block header + merkle path
    - Thousand times less than a full block size (=1 MB)
  - SPV node can
    - Definitely prove that Tx exists
    - But, can't verify that a double spend of same UTXO doesn’t exist
  - SPV are vulnerable to such double spending attacks
    - To defend, an SPV node connects randomly to several nodes

Lightweight node

- Simplified Payment Verification (SPV)
  - SPV nodes retrieve specific/selective Tx
    - Privacy risk
      - Bloom filters
        - Receive a subset of Tx
        - Without revealing precisely addresses of interest
  - Bloom filters
    - Probabilistic data structure
    - An element either is definitely not in the set or may be in the set
      - Correct about actuals/positives (TP=1, FN=0)
      - May be incorrect about fakes/negatives (FP exists)