# Ch8. The Bitcoin Network

Mastering Bitcoin of O'Reilly

## P2P Network Architecture

- Bitcoin is constructed on a P2P network architecture on top of the Internet.
- P2P networks are inherently resilient, decentralized, and open.
- Today's Internet architecture is more hierarchical, but the IP still retains its flat-topology essence.

### Bitcoin Network

- The term *bitcoin network* refers to the collections of nodes running the bitcoin P2P protocol.
- There are some other protocols such as Stratum that are used for mining and lightweight or mobile wallets. > Provided by gateway routing servers that access the bitcoin network using the bitcoin P2P protocol and then extend that network to nodes running other protocols.
- The term *extended bitcoin network* refers to the overall network that includes the bitcoin P2P protocol, pool-mining protocols, the Stratum protocol, and any other related protocols connecting the components of the bitcoin system.

## Node Types and Roles

- A bitcoin node is a collection of functions: routing, the BC DB, mining, and wallet services.

### The Ns: Network Routing

- All nodes include the routing function to participate in the network and might include other functionality.
- All nodes validate and propagate TXs and blocks, and discover and maintain connections to peers.

### The Bs: Full BC

- Some nodes, called full nodes, also maintain a complete and up-to-date copy of the BC.
  - Autonomously and authoritatively verify any TX without external reference.
- While some nodes maintain only subset of the blockchain and verify any TX using a method called *simplified payment verification* (SPV).
  - They're called SPV nodes or lightweight nodes.
  - Not drawn with the "B" circle.

### The Ms: Miner

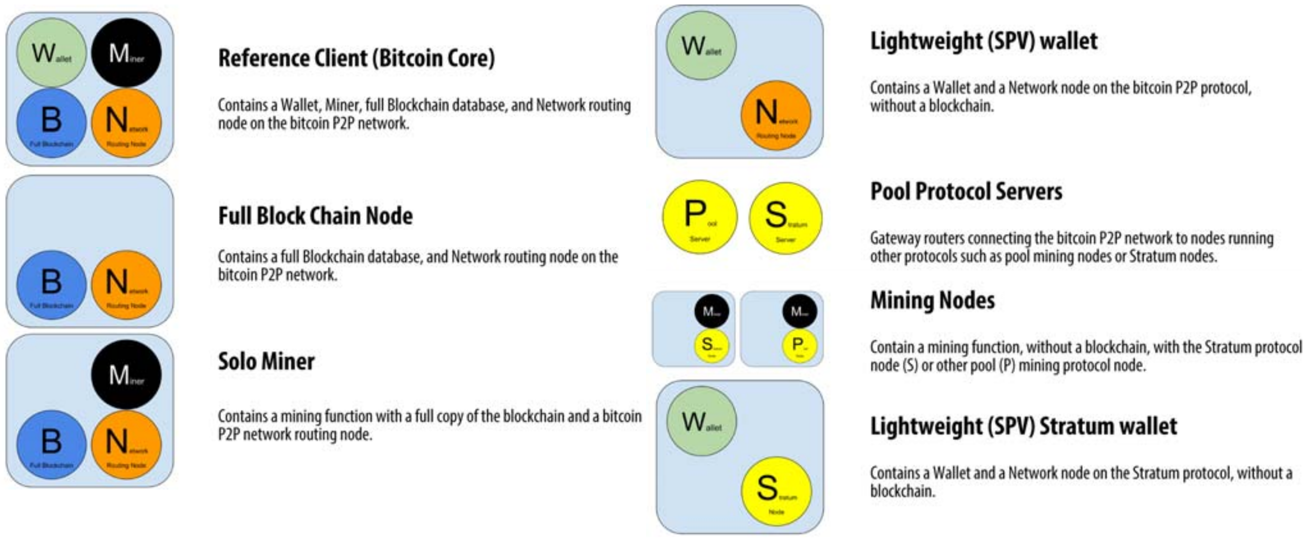
- Mining nodes compete to create new blocks (PoW).
- Some of which are also full nodes, while others depend on pool servers to maintain a full node.

### The Ws: Wallet

- Increasing, many user wallets, especially those running on resource-constrained devices, are SPV nodes.

### Others

- 
- There are servers and nodes running other protocols, such as specialized mining pool protocols and lightweight client-access protocols.




## The Extended Bitcoin Network

- The main bitcoin network consists of 5000+ listening nodes running various versions of the bitcoin reference client (Bitcoin Core).
  - Other few hundreds running various other implementations of the bitcoin P2P protocol.
- A small percentage of the nodes on the bitcoin P2P network are also mining nodes.
- Large companies interface with the bitcoin network by running full-node clients based on the Bitcoin Core client, but without mining or wallet functions.
  - These nodes act as network edge routers, allowing other services to be built on top.
- Attached to the main bitcoin P2P network are a number of pool servers and protocol gateways that connect nodes running other protocols.
  - These nodes are mostly pool mining nodes and lightweight wallet clients.

## Bitcoin Relay Networks

- In mining, network latency is directly related to profit margins.
- Relay networks appears.
  - NOT replacements for P2P network.
  - Overlaying networks providing additional connectivity between nodes with specialized needs.
  - Like freeways and rural roads.

- A *Bitcoin Relay Network* is a network that attempts to minimize the latency in the transmission of blocks between miners.
  - The original *Bitcoin Relay Network* consist of several specialized nodes hosted on the Internet infrastructure around the world and served to connect the majority of miners and mining pools.
  - Next original, Fast Internet Bitcoin Relay Engine (*FIBRE*) is a UDP-based relay network hat relays blocks within a network of nodes.
  - *Falcon* by Cornell University, uses "cut-through-routing", propagates parts of blocks upon receiving.

## Network Discovery

In this section, everything is under Bitcoin P2P network if not specified.

### Protocol

- When a new node boots up, it must discover other nodes on the network in order to participate.
  - Must find at least one existing node on the network and connect to it.
  - Any existing nodes can be selected at random.
  - (Port 8333 is generally known as the one used by bitcoin)
- Upon establishing a connection, the node will start a *handshake* by trasmitting a version message including:
  - `nVersion`: P2P protocol version the client "speaks".
  - `nLocalServices`: a list of local services supported by the node, currently just `NODE_NETWORK`.
  - `nTime`: current time.
  - `addrYou`: IP address of the remote node as seen from this node.
  - `addrMe`: IP address of the local node, as discovered by the local node.
  - `subver`: a sub-version showing the type of software running on this node.
  - `BestHeight`: block height of this node's BC.

#### Finding New Peer

- The receiving local peer will check if `nVersion` is compatible.
  - If so, it will acknowledge the version message and establish a connection by sending a `verack`.
- Methods
  - DNS query
    - Using *DNS seeds*: DNS servers providing a list of nodes.
    - Some DNSs provide static list, others return a random subset from a list.
    - Bitcoin Core client contains the smaes of 5 different DNS seeds.
    - Diversity of ownership and implementation of different seeds offers a high level of reliability for the bootstrapping.
  - Given IP address
    - Establish connections through further introductions.
    - After the initial seed node is used to form introductions, client will disconnect form it and use the newly discovered peers.

### Address Exchange

- Once connection(s) are established, the new node will send an `addr` message to make itself known and better connected.

- Additionally, it can send `getaddr` to the neighbors, asking them to return a list of other peers.

### Path

- A node must connect to a few different peers in order to establish diverse paths into the bitcoin network.
  - Only 1 connection is needed to bootstrap, for introduction.
  - Unnecessary and wasteful of network resources to connect too many nodes.
  - A most-recent-success is remembered for next bootstrap.
- If there's not traffic on a connection, nodes will periodically send a message to maintain the connection.
  - Disconnection timeout: 90 minutes.

## Full Nodes

- Full nodes are nodes that maintain a full BC with all TXs.
  - In the early year, all nodes are full nodes.
  - (Maybe should be called "full BC nodes" for accuracy.)
  - Later, *lightweight client*s are introduced as new forms of Bitcoin clients.
- Traits
  - Maintain a complete and up-to-date copy of Bitcoin BC, from the genesis block to the latest known block.
  - Can independently and authoritatively verify any TX without any other nodes or sources.

## Exchanging "Inventory"

- The first thing a full node will do once it connects to peers is try to construct a complete BC.
  - A new full node only knows the genesis block, and it will download all the blocks afterwards.
- Syncing
  1. Check `version` message: contains `BestHeight`, a node's current BC height.
  2. Receive `version` messages from peers, compare to its own BC.
  3. Peered nodes exchange `getblocks` messages: contains the hash of the top block on their BC.
  - The peer that has the longer BC can identify which blocks the other node needs.
    - Then share the first 500 blocks' hash using an `inv`(inventory) message
    - Other nodes will retrieve them using the hashes from the `inv` message.

## Simplified Payment Verification (SPV) Nodes

- Most common form of bitcoin node, especially bitcoin wallets.

- Only save the block headers.
  - Thus about 1000 times smaller than the full BC.
  - But cannot picture all UTXOs.
- TX verification
  - Verification of TXs relies on peers to provide partial views of relevant parts.
  - Referencing TX *depth* instead of *height*. (merkle path)
  - In most cases, 6 block ahead is stable enough for Bitcoin, and show that it was not a double-spend.
- An SPV node knows if a block (a TX) exists.
  - However, it cannot verify it, due to the lack of full record.
  - Thus, SPV nodes usually connect randomly to several nodes, to increase the contact chance to at least one honest node.
    - Making SPV nodes vulnerable to ++network partitioning attacks++ or ++Sybil attacks++.
- For most practical purposes, well-connected SPC nodes are secure enough, striking a balance between resource needs, practicality, and security.
  - However, running a full BC node is the safest after all.
- SPV nodes use a `getheader` message instead of `getblocks`.
  - Responding peer will send up to 2000 block headers using single `headers` message.
  - Otherwise the same as that used by a full node to retrieve full blocks.
  - Any TXs of interest are retrieved using a `getdata` request.

- Privacy issue
  - SPV nodes need to retrieve specific TXs to selectively verify them.
  - The request for specific data can inadvertently reveal the addresses.\

## Bloom Filters

- A feature to address the privacy risks of SPV nodes.

- A probabilistic search filter, a way to describe a desired pattern without specifying it exactly.
  - Allow specifications to search patterns for TXs that can be tuned toward precision or privacy.
  - More specific bloom filter will produce accurate results, but at the expense of privacy.

### Implementation

- Foundation
  - A variable-size $N$-bit array indexed 1 to $N$.
  - A variable number of $M$ hash functions ranging [ 1, $N$ ].
  - Different $N$ and $M$ result in variable level of accuracy and privacy.

- Pattern recording
  - Let $P_i$ be the original filter pattern.
  - For each hash function $H_i$ , set Array[ $H_j$( $P_i$ ) ] to 1. (That is, to set 1 to its corresponding index.)
    - If a bit had already been set to 1, it is not changed.
    - If more patterns record on overlapping bits, the accuracy decreases.
    - Thus, a larger bit array and more hash functions can record more patterns with higher accuracy.
- Pattern matching
  - A pattern $Q$ is hashed by each hash function to match the bloom filter.
  - $Q$ is hashed by each hash function and tested against the bit array.
  - If all bits indexed are set to 1, then $Q$ is *probably* recorded in the filter. (Due to the probable overlapping.)
    - Thus, a (positive) match is a "Maybe, yes."
    - Similarly, a mismatch (negative match) is a "Definitely Not."

## Application of Bloom Filters on SPV Nodes

- Bloom filters are used to filter the TXs (and the containing block) that and SPV nodes receives.

- Steps
  - Requesting node
    - All bits in bloom filter set to 0.
    - List owning addresses, keys and hashes.
      - Extracting PKH, SH and TX IDs from any UTXO in the node.
    - Send a `filterload` message to peers, containing the bloom filter.
  - Peers
    - Check matching:
      - TX ID
      - Data components from the locking scripts of the TX outputs
      - TX inputs
      - Input signature data components (or witness scripts)
    - Send back probably matching TXs.
      - Only a `merkleblock` message containing only block headers is sent.
  - Back to requesting node
    - Discard false positives.
    - Update UTXO and wallet balance.
    - Modify the bloom filter for future matching.
- Filter modification
  - Send `filteradd` message to interactively add patterns to the filter.
  - Send `filterclear` message to clear the filter.
  - Pattern removal is not possible, it must clear and resend the new bloom filter.

## SPV Nodes and Privacy

- SPV nodes have weaker privacy than full nodes.
- Bloom filters are a way to reduce the loss of privacy. However, even with bloom filters, some methods can still collect enough information over time to learn the addresses of the SPV node.

## Encrypted and Authenticated Connections

- The original implementation of Bitcoin communicates entirely in the clear.
  - Not a major concern for full nodes, but a big problem for SPV nodes.
- Solution:
  - *Tor Transport*
  - *P2P Authentication and Encryption* with BIP-150/151.

### Tor Transport

- Tor stands for *The Onion Routing network*.
- Offering encryption and encapsulation of data through randomized network paths that offer anonymity, untraceability and privacy.
- Bitcoin Core supports Tor.

### Peer-to-Peer Authentication and Encryption

- Two Bitcoin Improvement Proposals, BIP-150 and BIP-151, add support for P2P authentication and encryption in the network.
- BIP-150 (Peer authentication)
  - Optional peer authentication
  - Using ECDSA and private keys
  - Requires BIP-151 communications.
- BIP-151 (Peer-to-peer communication encryption)
  - Enabling negotiated encryption for all communications between nodes supporting BIP-151.
- Benefits
  - Prevent man-in-the-middle attack.
  - Strengthen resistance of bitcoin to traffic analysis and privacy-eroding surveillance.

## Transaction Pools

- Memory pool: a per-node temporary list of unconfirmed TXs.
  - Keep track TXs that are known to the network but are not yet on the BC.
  - Wallet nodes use TX pool to track (unconfirmed) incoming payments.
- Upon reception and verification, TXs are added to the TX pool and relayed to the neighbors.

- Some also maintain a separate pool of orphaned TXs.
  - When a TX is added to the pool, the orphan pool is checked if there are any of its children.
  - Any matching orphans are then validated, removed from the orphan pool, and added to the TX pool.
    - Recursively find in the orphan pool, until there is no orphans or no descendants are found.
- Both TX pool and orphan pool (if implemented) are in local memory.
  - Empty on node startup.
  - Gradually populated with new TXs received.
- Some maintain UTXO DB or pool.
  - Not initialized empty but instead contains millions of entries of all UTXO.
  - May be a pool on local memory or an indexed DB on mass storage.
- Differences
  - TX and orphan pools represent a single node's local perspective.
    - Might vary significantly from one node to another.
    - Depending on when the node starts.
    - Only contains unconfirmed outputs.
  - UXTO pool represents the emergent consensus of the network.
    - Vary little between nodes.
    - Only contains confirmed outputs.

## Table of Terms (Lexicographical order)

| Abbr. | Term |
|---|---|
| BC | blockchain |
| BIP | Bitcoin improvement proposals |
| DB | database |
| DNS | domain name server |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| P2P | peer-to-peer |
| PKH | public key hash |
| PoW | proof of work |
| SH | script hash |
| SPV | simplified payment verification |
| TX | transaction |
| UTXO | unspent transaction output |