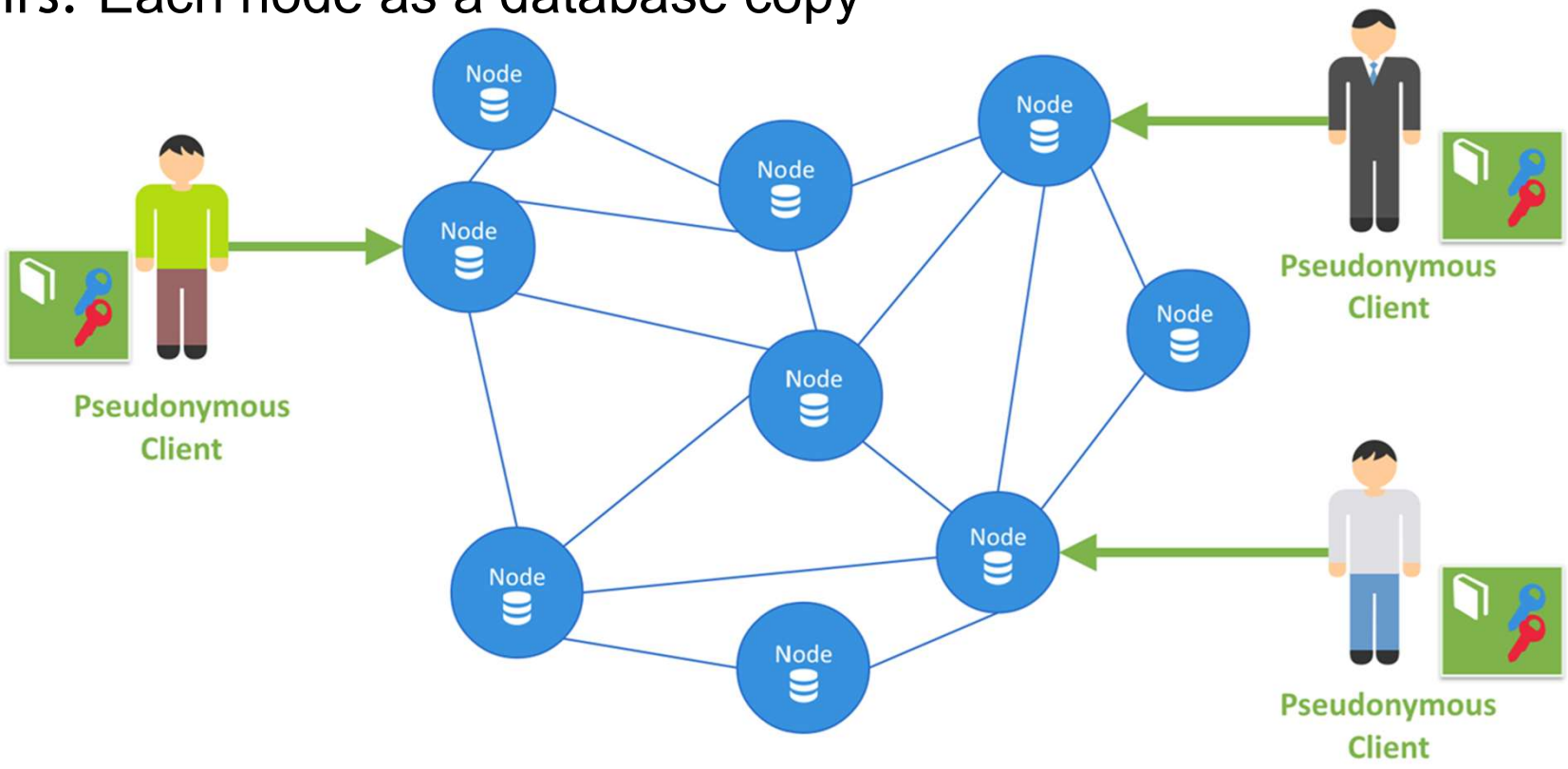
Module 5: Bitcoin Networks

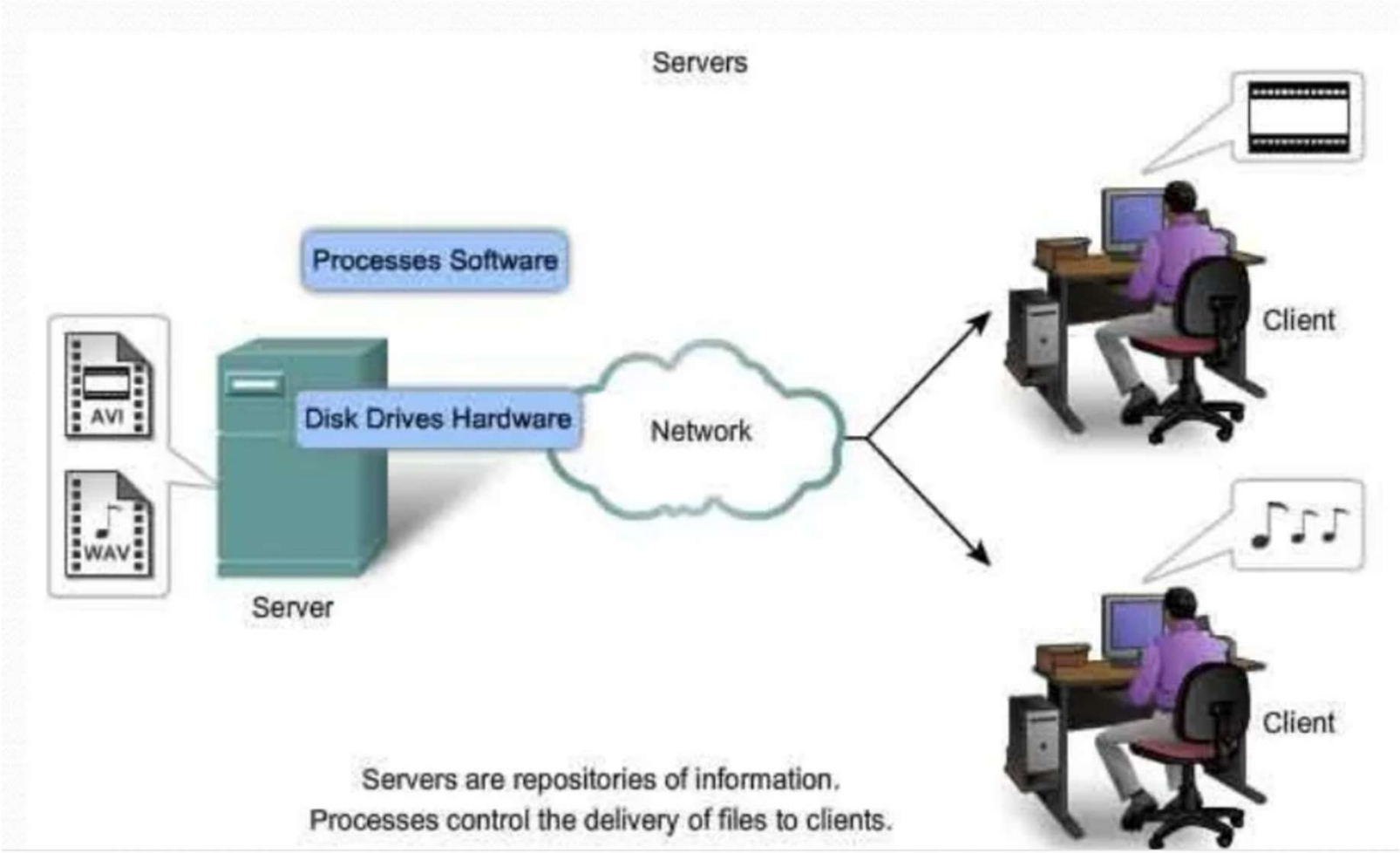
Department of Computer Engineering, VESIT, Mumbai

Blockchain Structure

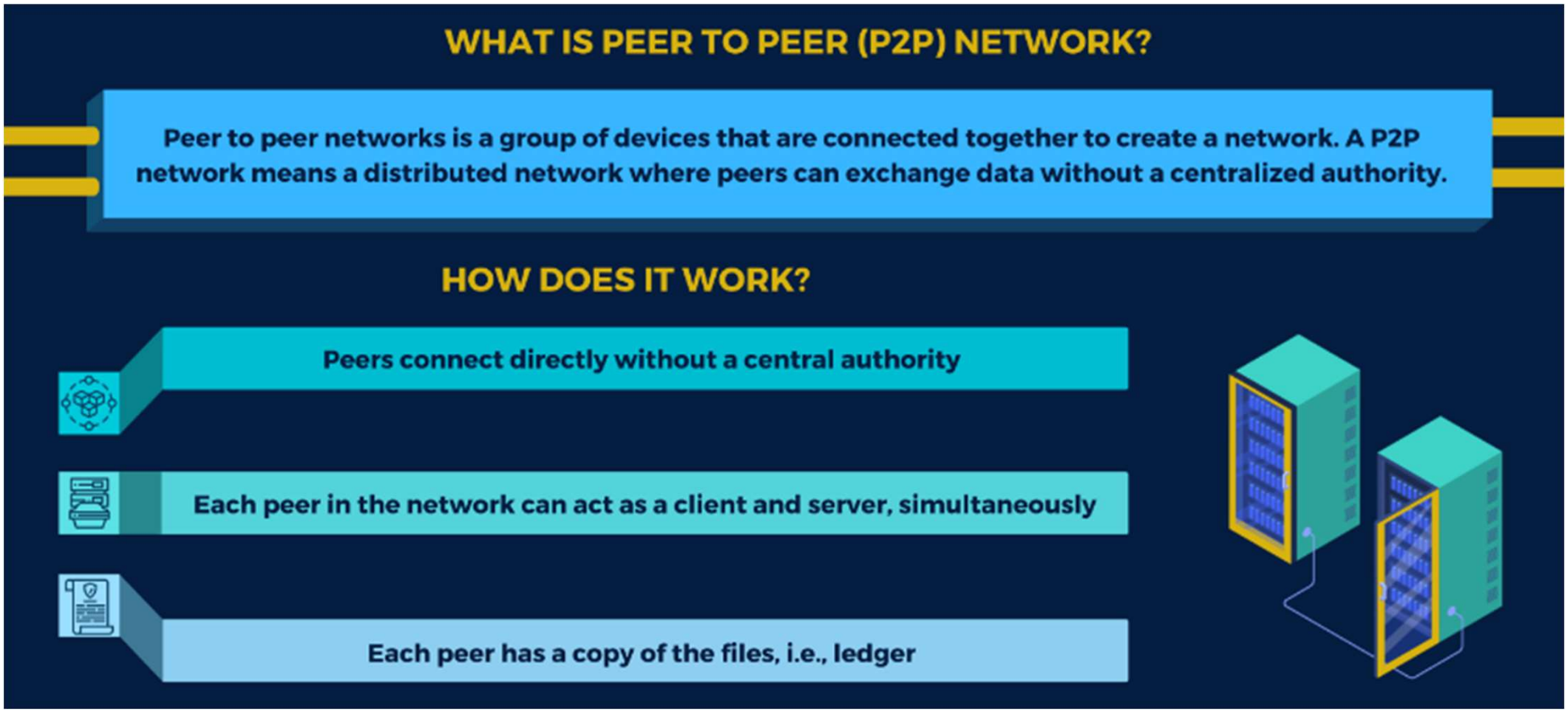
Peer-to-peer Architecture with pseudonymous client bearing key pairs. Each node as a database copy



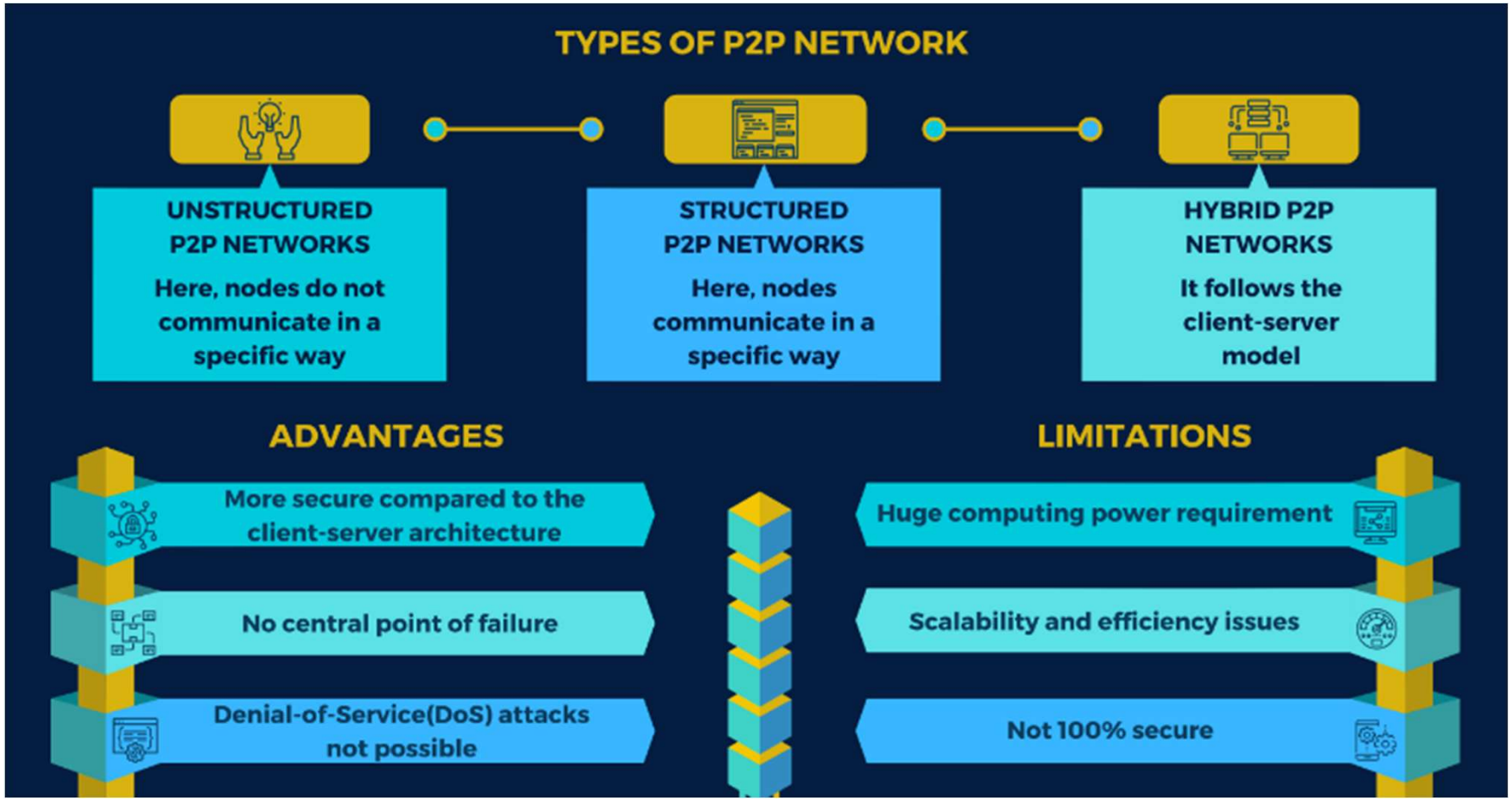
Peer to Peer Network Architecture



Peer to Peer Network Architecture



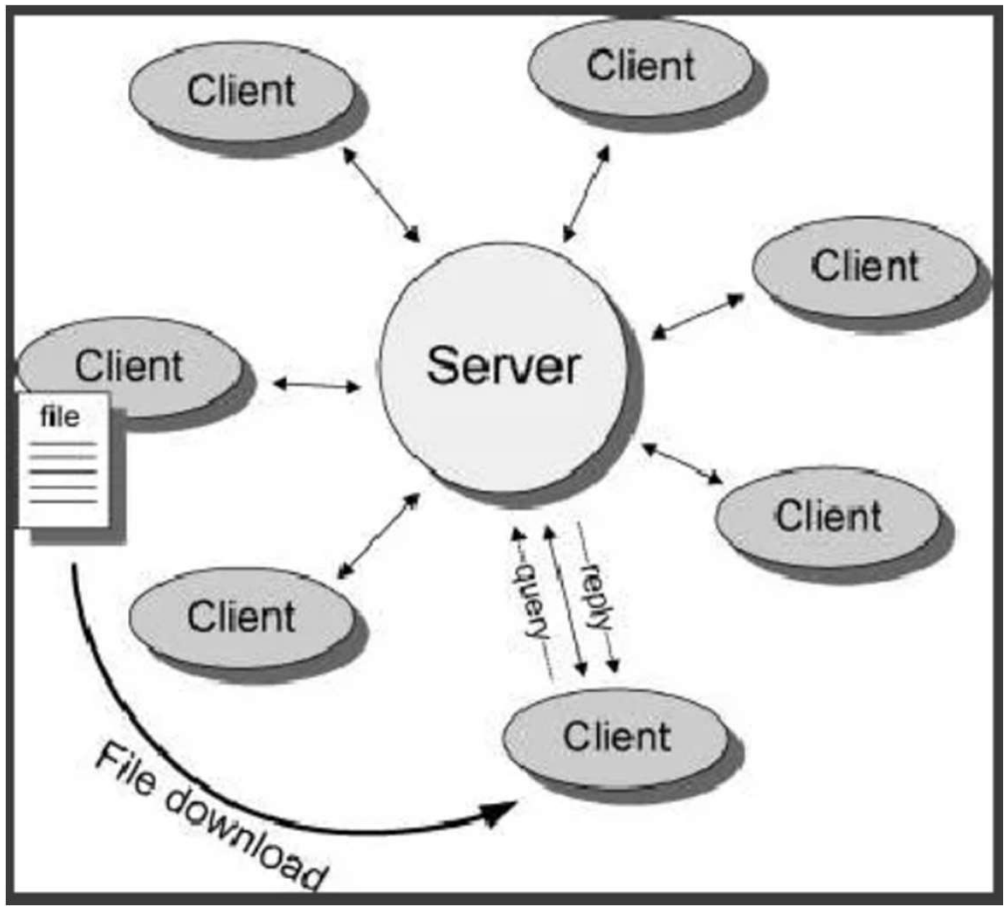
Peer to Peer Network Architecture



Types of P2P Network

- Unstructured P2P
- Structured P2P
- Hybrid P2P

Types of P2P Network : Unstructured P2P



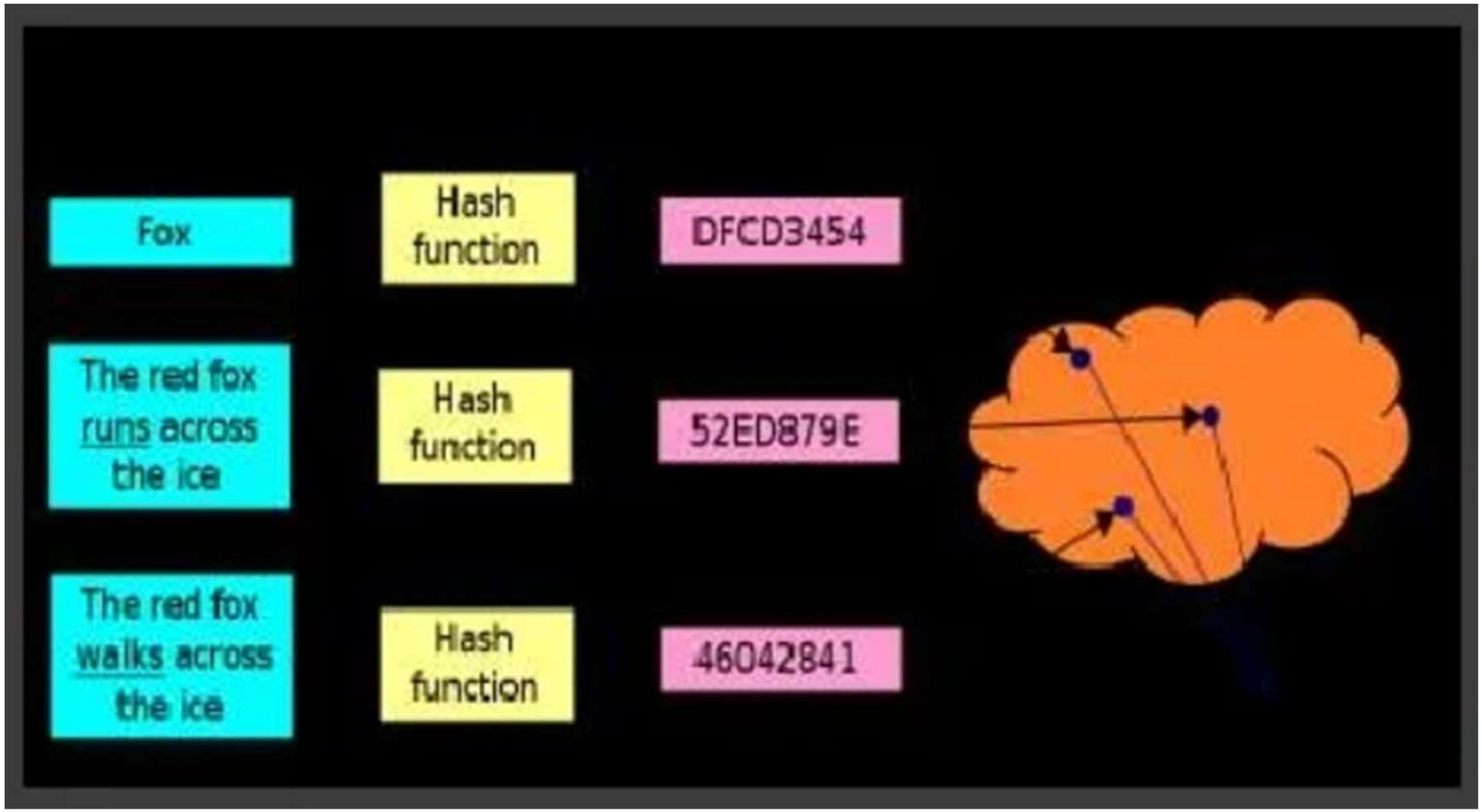
Types of P2P Network : Unstructured P2P

- nodes are not organized in any specific way.
- communication between the nodes is random in nature
- best suited for activities that require a lot of activity.
- **Eg.:** a social platform powered by P2P can utilize it as people can choose to leave or join the network frequently.
- **Drawbacks:** it requires a lot of CPU and memory power to run properly
- The hardware should be able to power the highest number of transactions in the network, which means all the nodes interact with each other at any given time.

Types of P2P Network : Structured P2P

- the nodes do have a way to interact with each other - It is possible because of organized architecture that is used to search for files and to use them efficiently, rather than searching randomly.
- hash functions are used for database lookups.
- structured P2P networks are more efficient and robust.
- **Drawbacks:** They also have some sort of centralization as they are using organized architecture. It also means that they require higher maintenance and setup costs.

Types of P2P Network : Structured P2P



Types of P2P Network : Hybrid P2P

- Combination of the peer to peer architecture and client-server model.
- This is useful for networks where they need a central server with P2P features.
- Hybrid P2P networks are more efficient than the structured and unstructured P2P networks

Bitcoin P2P Network

- Ad-hoc Protocol (runs on TCP Port 8333 for main network and TCP 18333 for testnet.)
- Ad-hoc Network with Random topology
- All nodes are equal
- New nodes can join at any time
- Forget non-responding nodes after 3 hours

DNS Seed - Courtesy: https://github.com/bitcoin/bitcoin/blob/0cda5573405d75d695aba417e8f22f1301ded001/src/chainparams.cpp#L116

Bitcoin P2P Network

- Peer-to-peer (P2P) network where nodes perform transactions.
- They verify and propagate transactions and blocks.
- Nodes called miners also produce blocks.

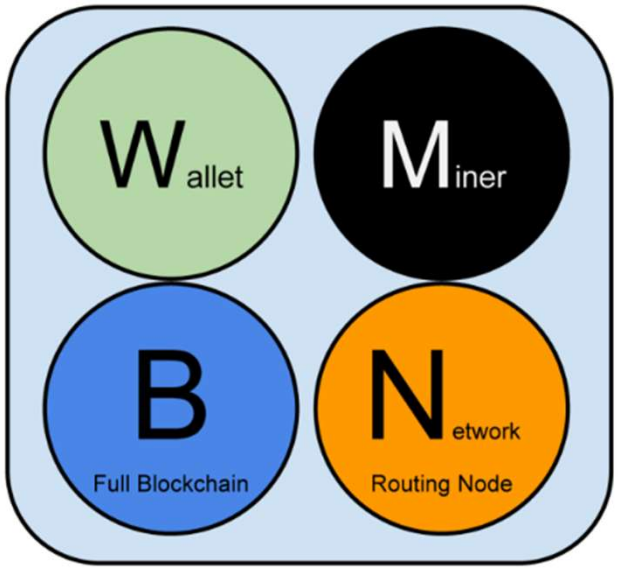
Bitcoin P2P Network

- A Bitcoin network is identified by its magic value.
- Magic values are used to indicate the message's origin network.

Network	Magic value (in hex)
main	0xD9B4BEF9
testnet	0xDAB5BFFA
testnet3	0x0709110B

Bitcoin Node Functions

- A full Bitcoin node performs four functions.
  - Wallet,
  - Miner,
  - (Full) Blockchain, and
  - Network routing.



- The latest version of the Bitcoin protocol is 70015, which was introduced with Bitcoin Core client 0.19.0.1.

Bitcoin Protocol Messages - Network Discovery

- Version: This is the first message that a node sends out to the network, advertising its version and block count. The remote node then replies with the same information and the connection is then established.
- Verack: This is the response of the version message accepting the connection request.
- Inv: Used by nodes to advertise their knowledge of blocks and transactions.
- Getdata: This is a response to inv, requesting a single block or transaction identified by its hash.

Bitcoin Protocol Messages - Network Discovery

- Getblocks: This returns an inv packet containing the list of all blocks starting after the last known hash or 500 blocks.
- Getheaders: used to request block headers in a specified range.
- Tx: used to send a transaction as a response to the getdata protocol message.
- Block: This sends a block in response to the getdata protocol message.
- Headers: This packet returns up to 2,000 block headers as a reply to the getheaders request.

Bitcoin Protocol Messages - Network Discovery

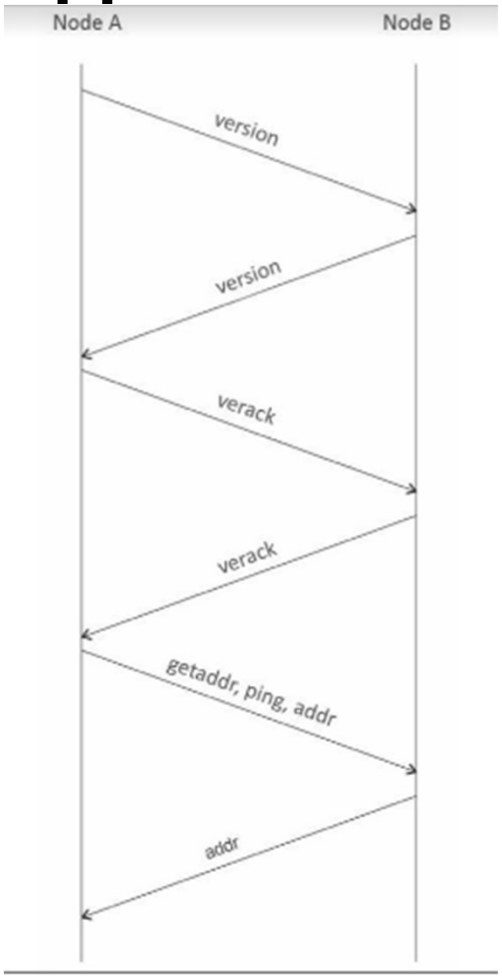
- Getaddr: This is sent as a request to get information about known peers.
- Addr: This provides information about nodes on the network. It contains the number of addresses and address list in the form of an IP address and port number.
- Ping: This message is used to confirm if the TCP/IP network connection is active.
- Pong: This message is the response to a ping message confirming that the network connection is live.

Network Discovery

- **nVersion** - The bitcoin P2P protocol version the client “speaks” (e.g., 70002)
- **nLocalServices** - A list of local services supported by the node, currently just NODE_NETWORK
- **nTime** - The current time
- **addrYou** - The IP address of the remote node as seen from this node
- **addrMe** - The 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
- (e.g., /Satoshi:0.9.2.1/)
- **BestHeight** - The block height of this node’s blockchain

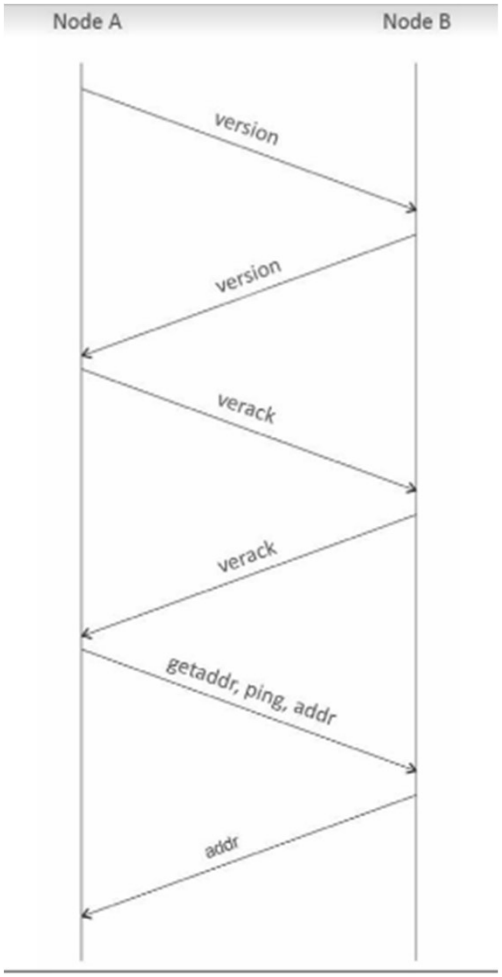
Node discovery Protocol - Blocks-First Approach

- Network protocol sequence diagram shows communication between two Bitcoin nodes during initial connectivity.
- First, Node A starts the connection by sending a version message that contains the version number and current time to the remote peer, Node B.
- Node B then responds with its own version message containing the version number and current time.
- Node A and Node B then exchange a verack message, indicating that the connection has been successfully established.
- After this connection is successful, the peers can exchange getaddr and addr messages to discover other peers on the network.



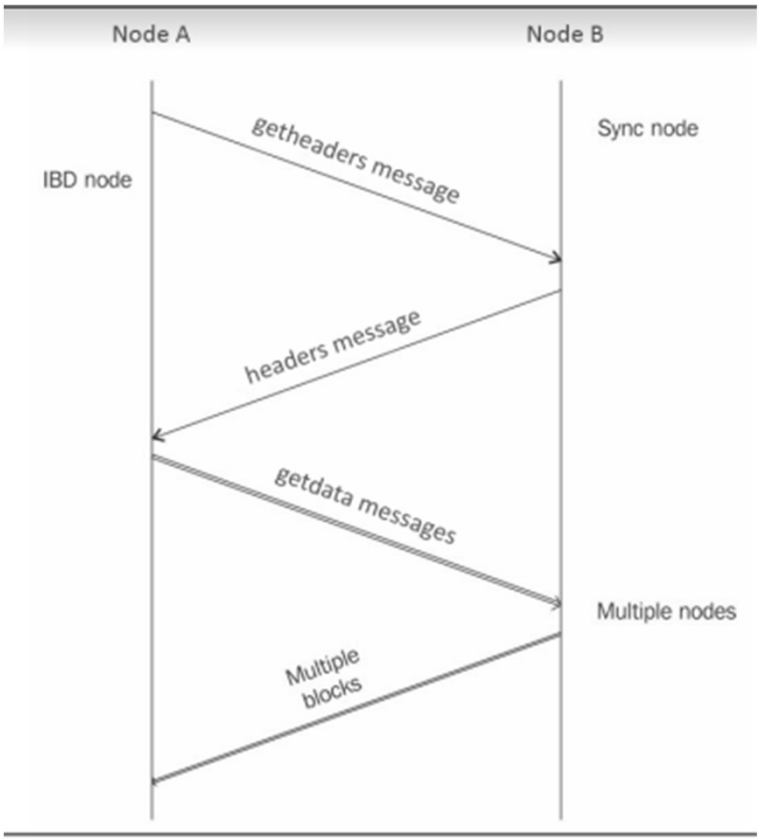
Blocks First Approach

- Now, the block download can begin.
- If the node already has all the blocks fully synchronized, then it listens for new blocks using the inv protocol message;
- otherwise, it first checks whether it has a response to inv messages and has inventories already.
- If it does, then it requests the blocks using the getdata protocol message; if not, then it requests inventories using the getblocks message. This method was used until version 0.9.3.



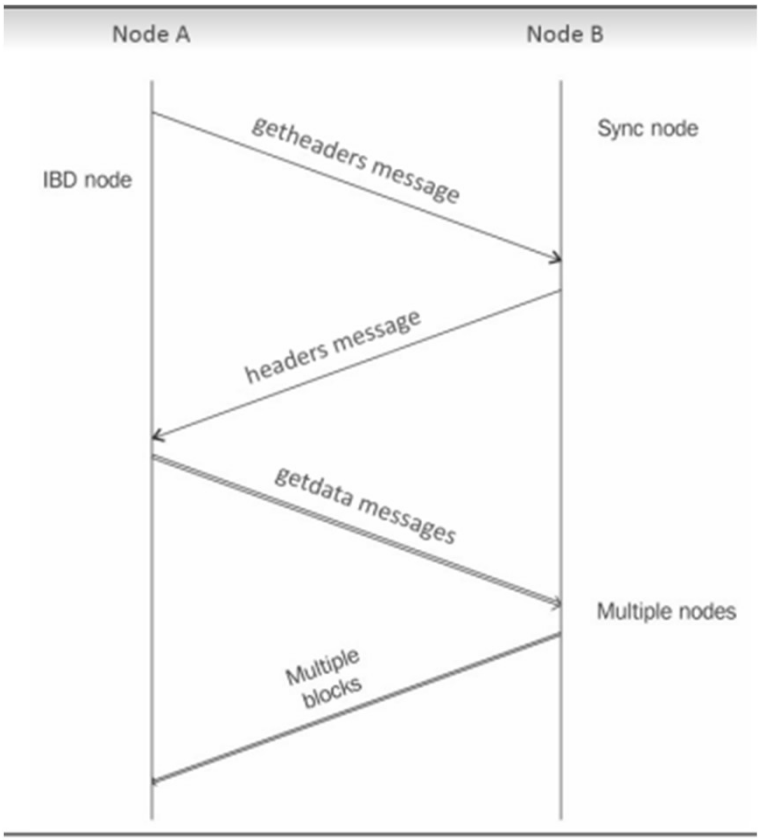
Headers First Approach - Bitcoin core client 0.10.0

- The initial block download method named headers-first was introduced.
- This resulted in major performance improvement, and blockchain synchronization that used to take days to complete started taking only a few hours.
- The core idea is that the new node first asks peers for block headers and then validates them. Once this is completed, blocks are requested in parallel from all available peers.
- This happens because the blueprint of the complete chain is already downloaded in the form of the block header chain.

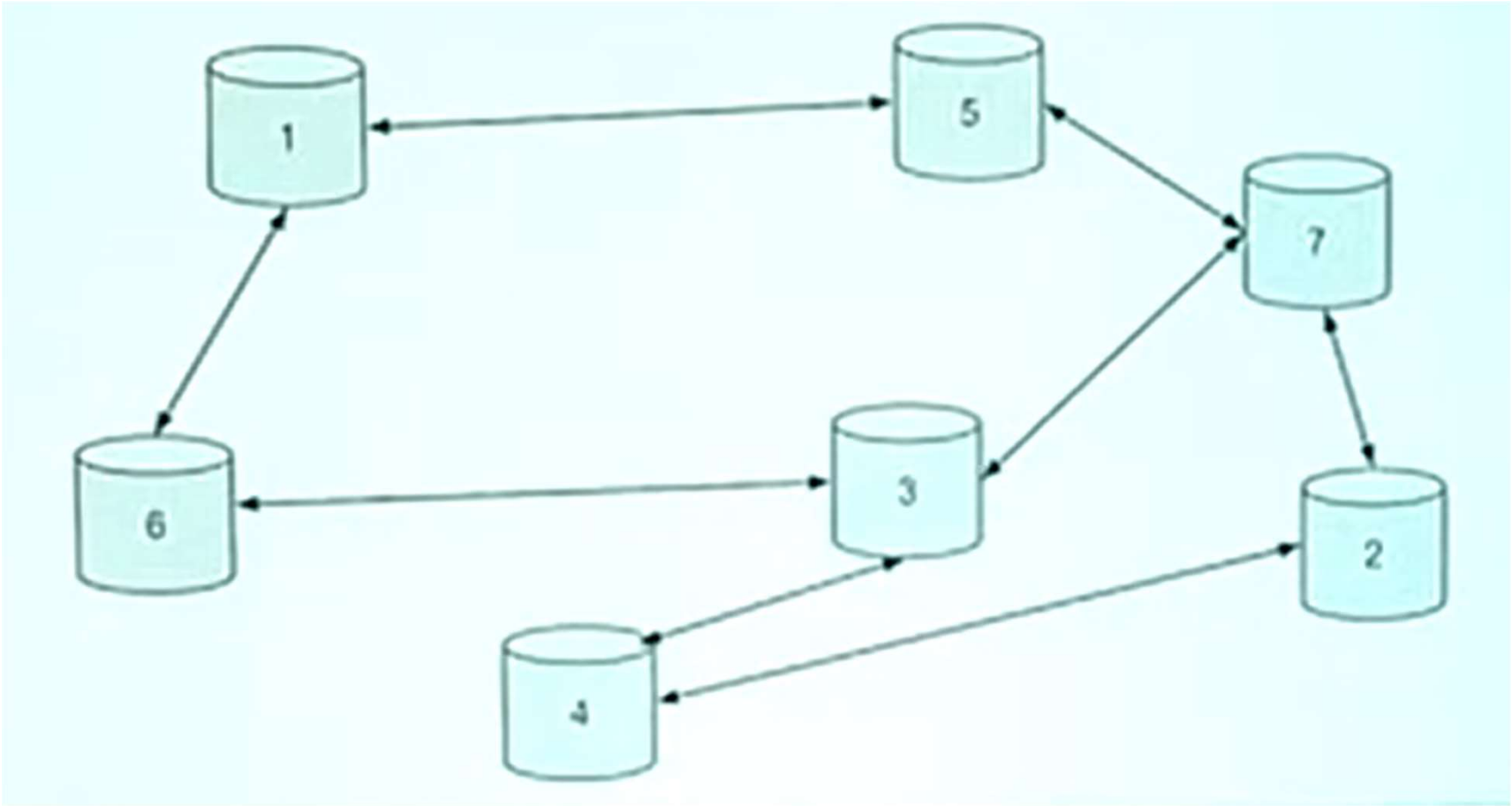


Headers First Approach - Bitcoin core client 0.10.0

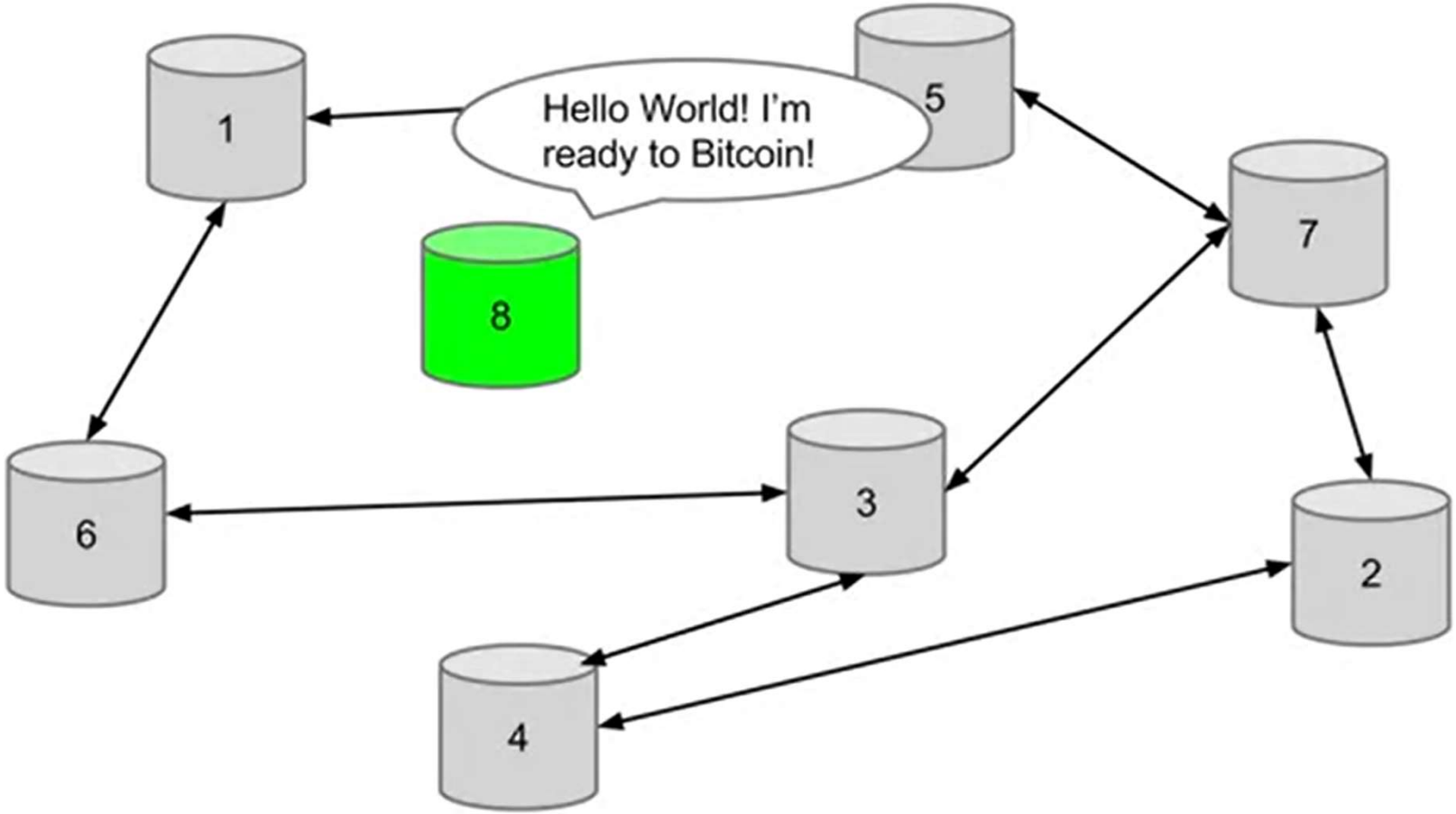
- In this method, when the client starts up, it checks whether the blockchain is fully synchronized
- if the header chain is already synchronized; if not, which is the case the first time the client starts up, it requests headers from other peers using the getheaders message.
- If the blockchain is fully synchronized, it listens for new blocks via inv messages, and if it already has a fully synchronized header chain, then it requests blocks using getdata protocol messages.
- The node also checks whether the header chain has more headers than blocks, and then it requests blocks by issuing the getdata protocol message:



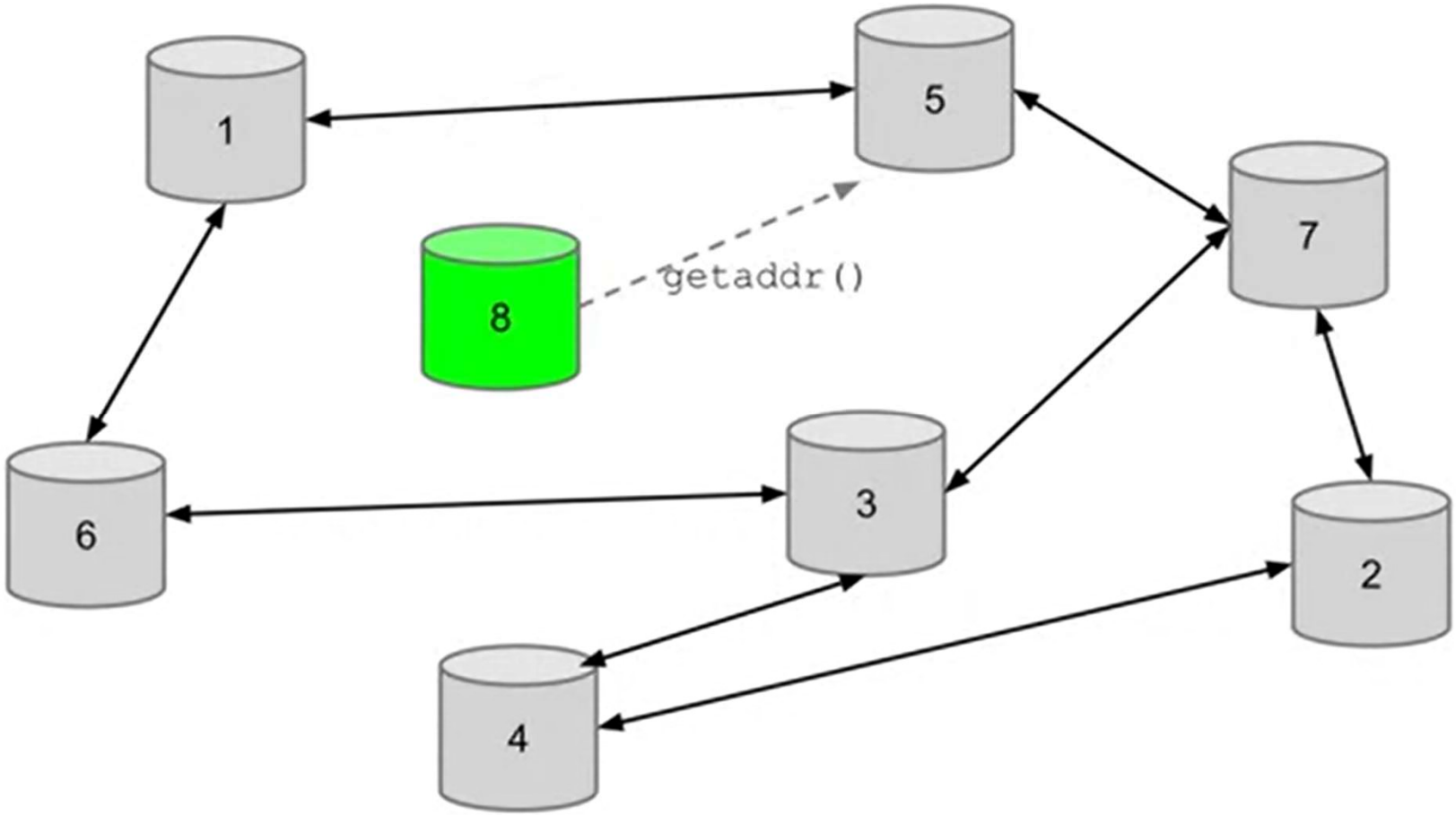
Joining the Bitcoin P2P Network



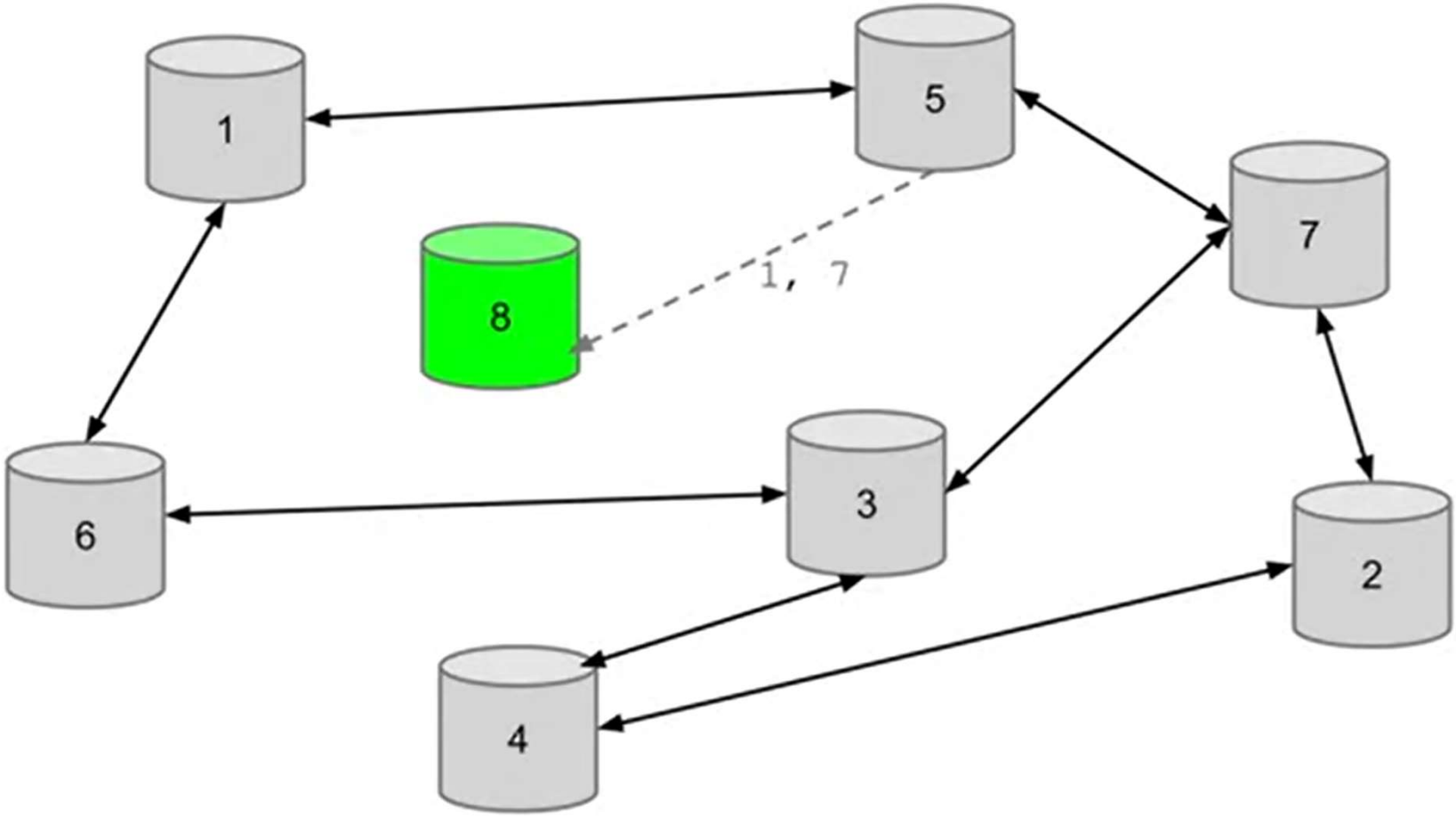
Joining the Bitcoin P2P Network



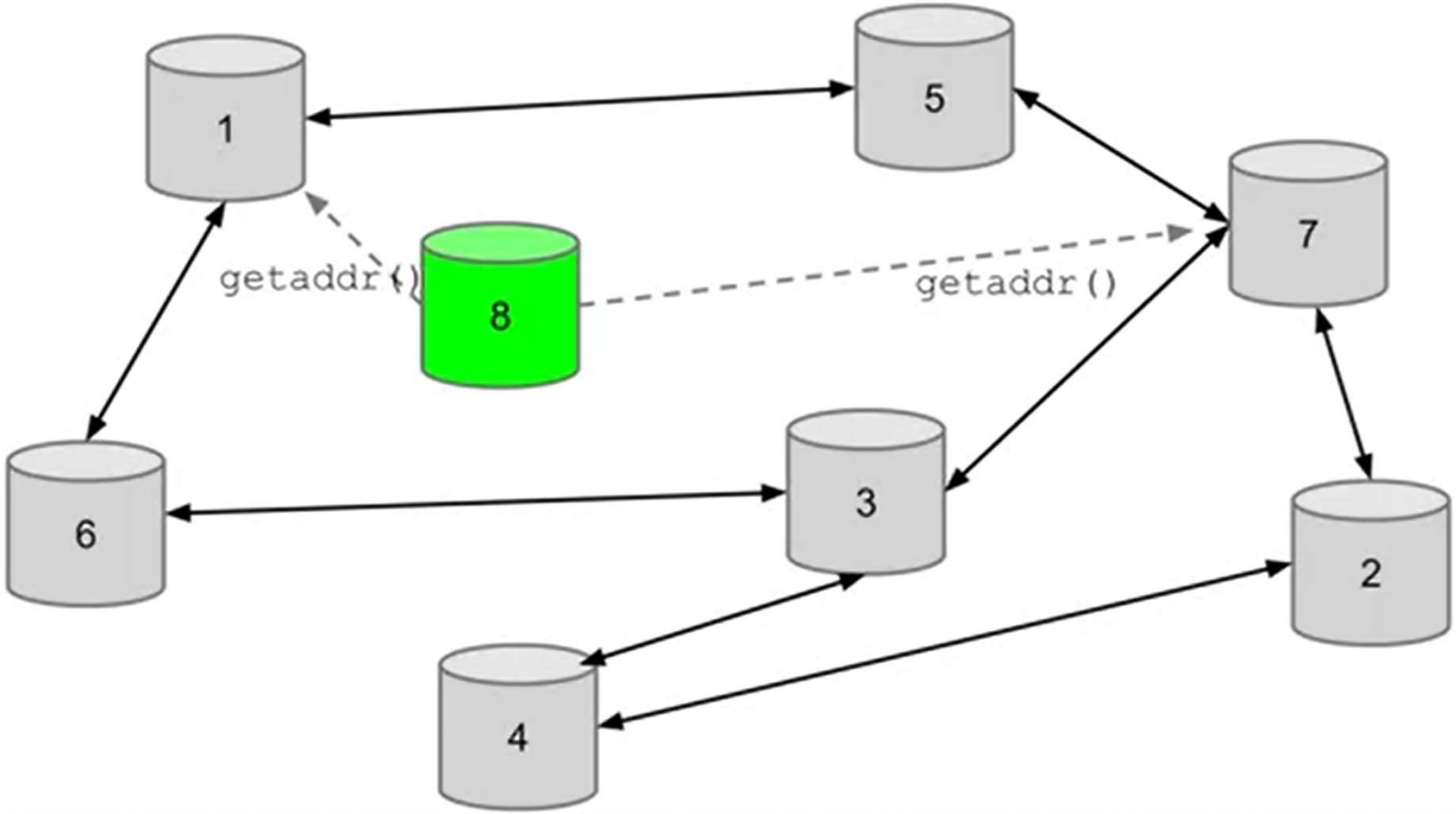
Joining the Bitcoin P2P Network



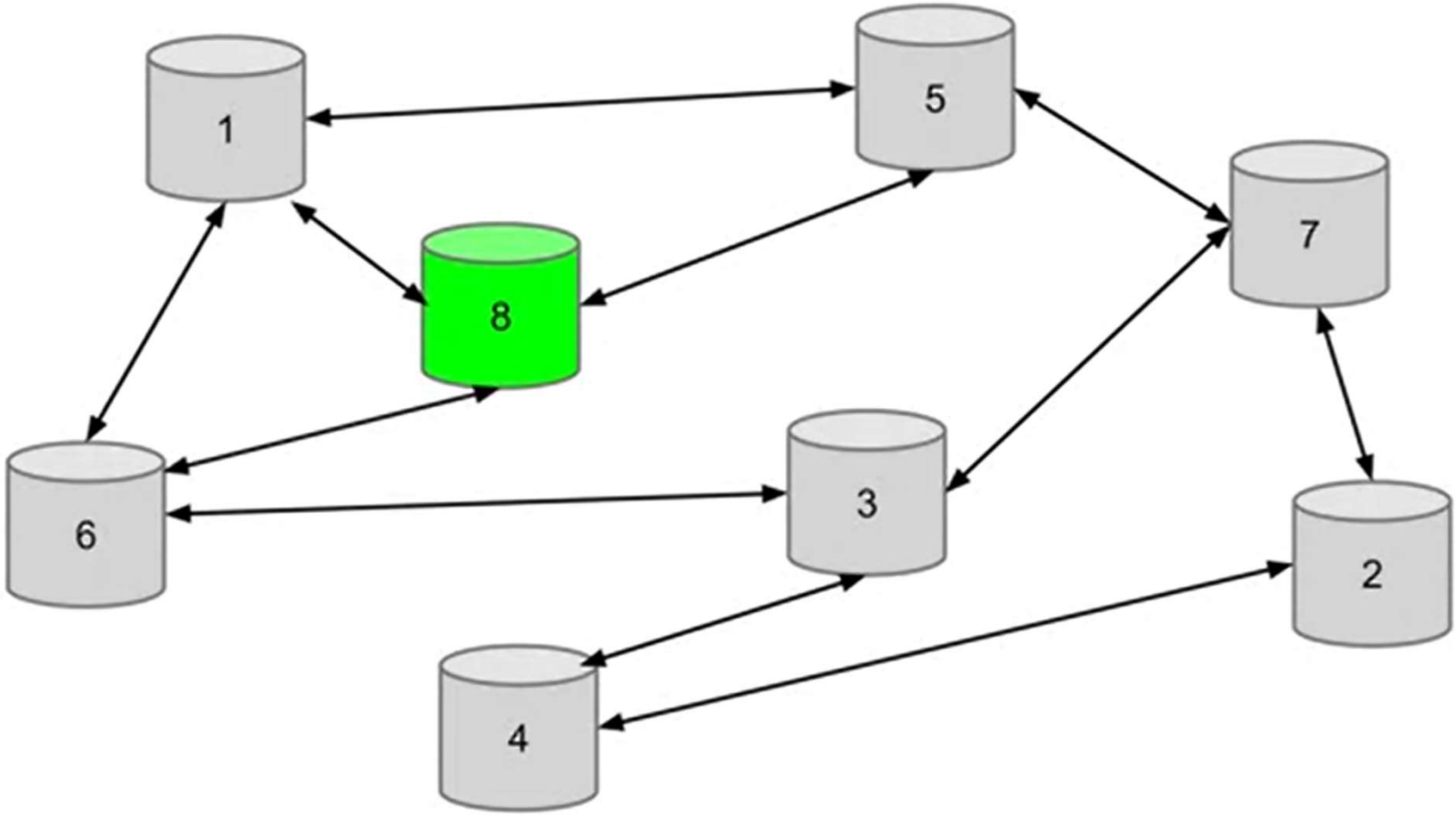
Joining the Bitcoin P2P Network



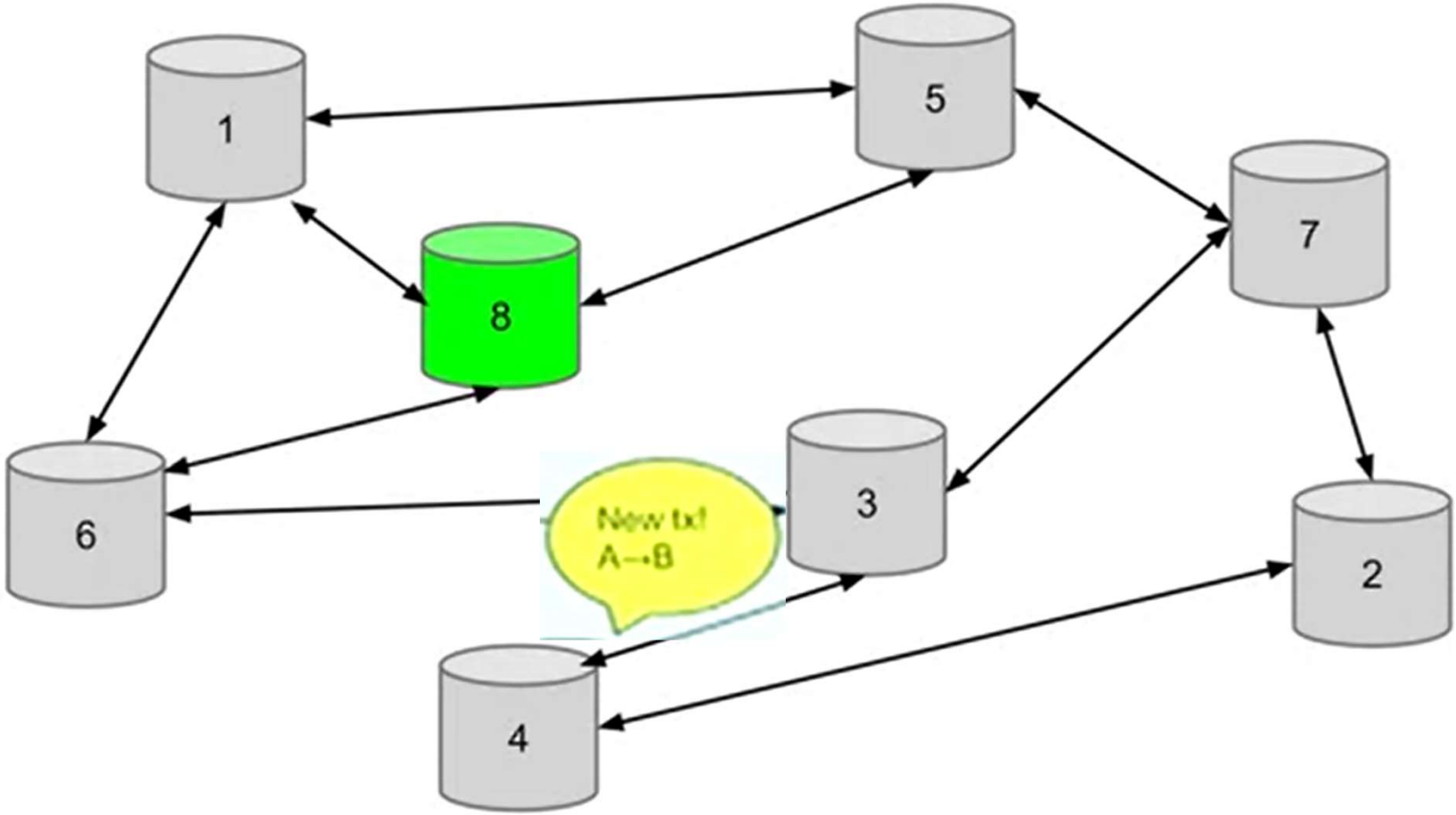
Joining the Bitcoin P2P Network



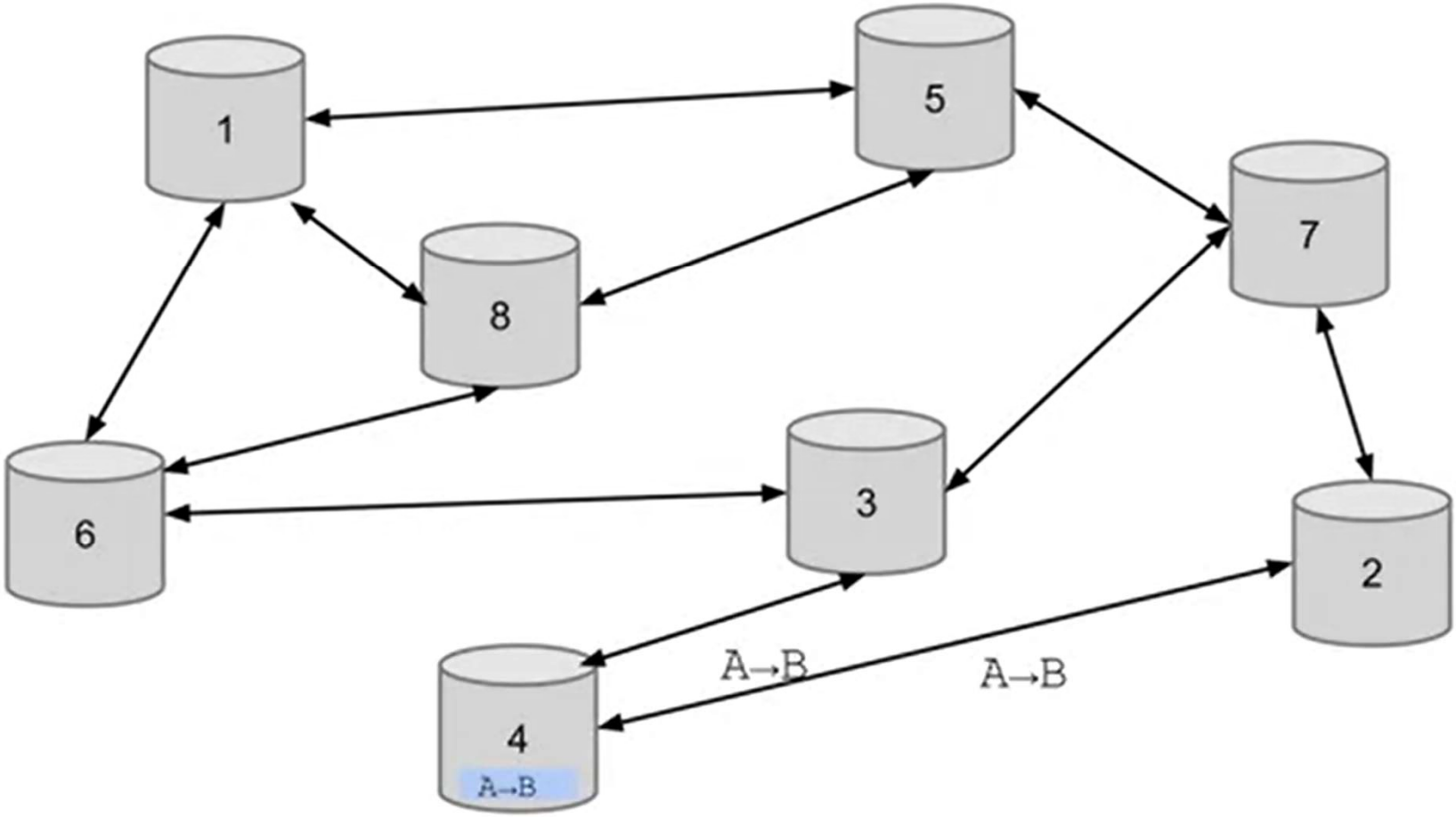
Joining the Bitcoin P2P Network



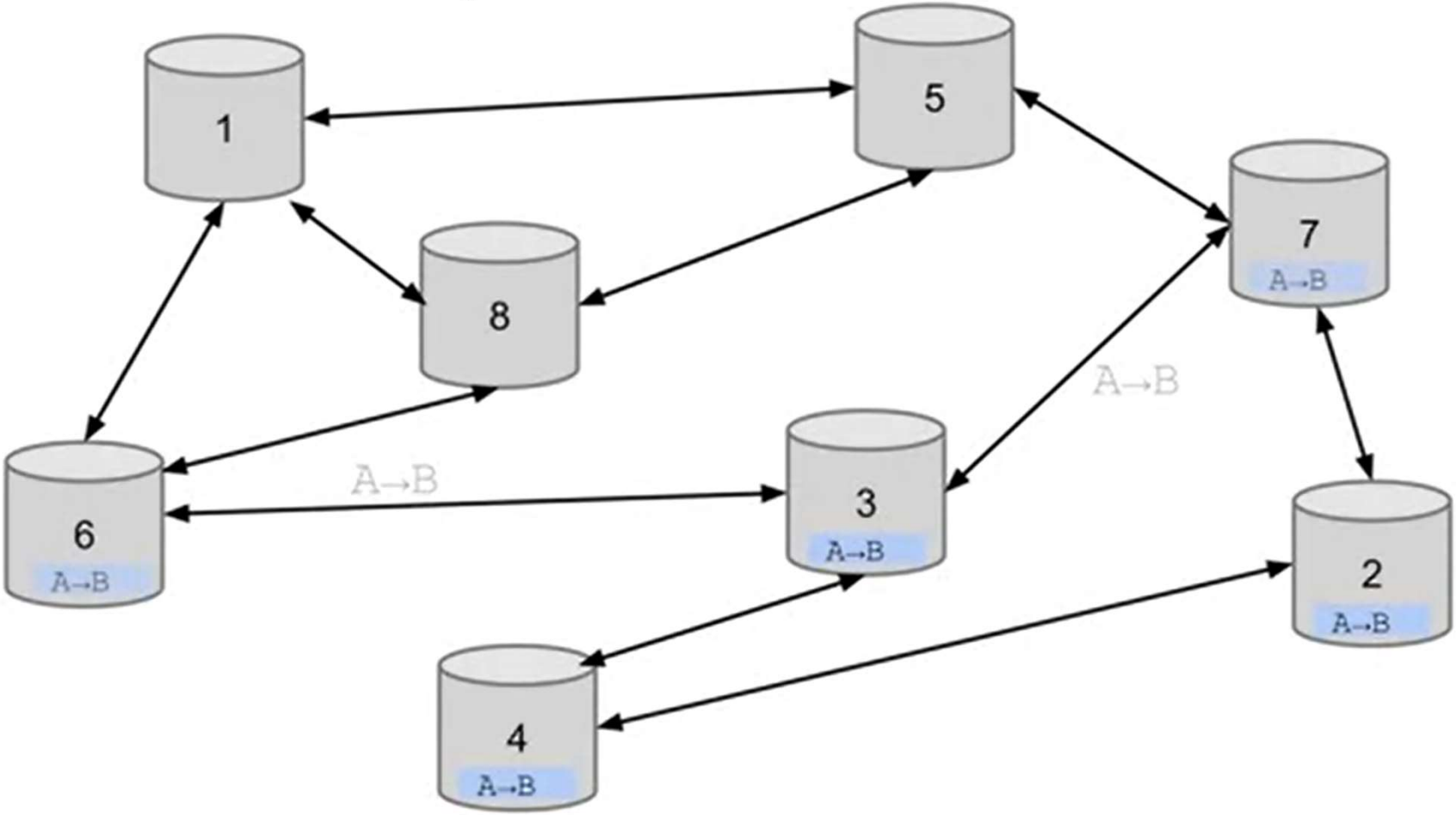
Transaction Propagation - Flooding



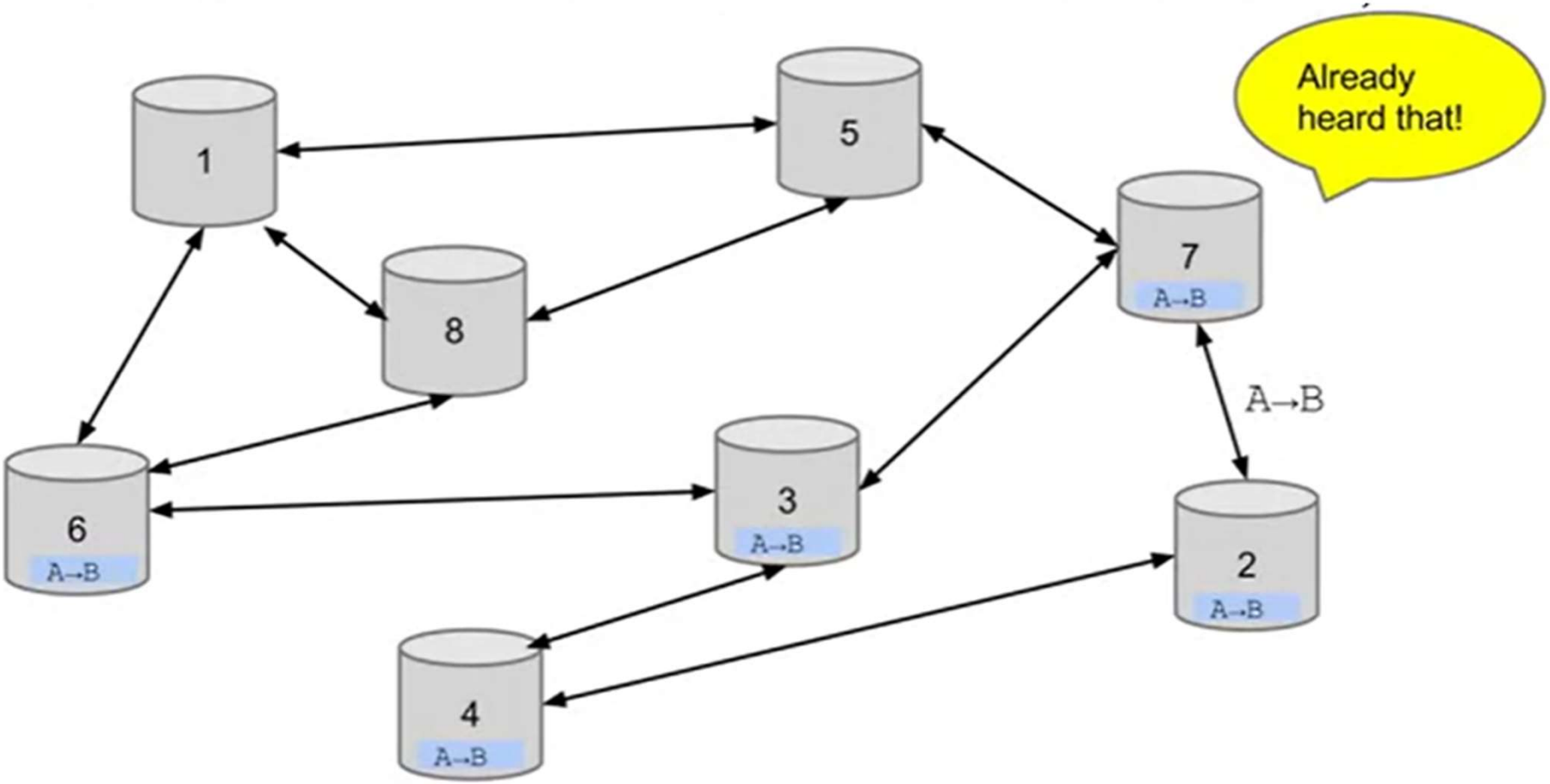
Transaction Propagation - Flooding



Transaction Propagation - Flooding



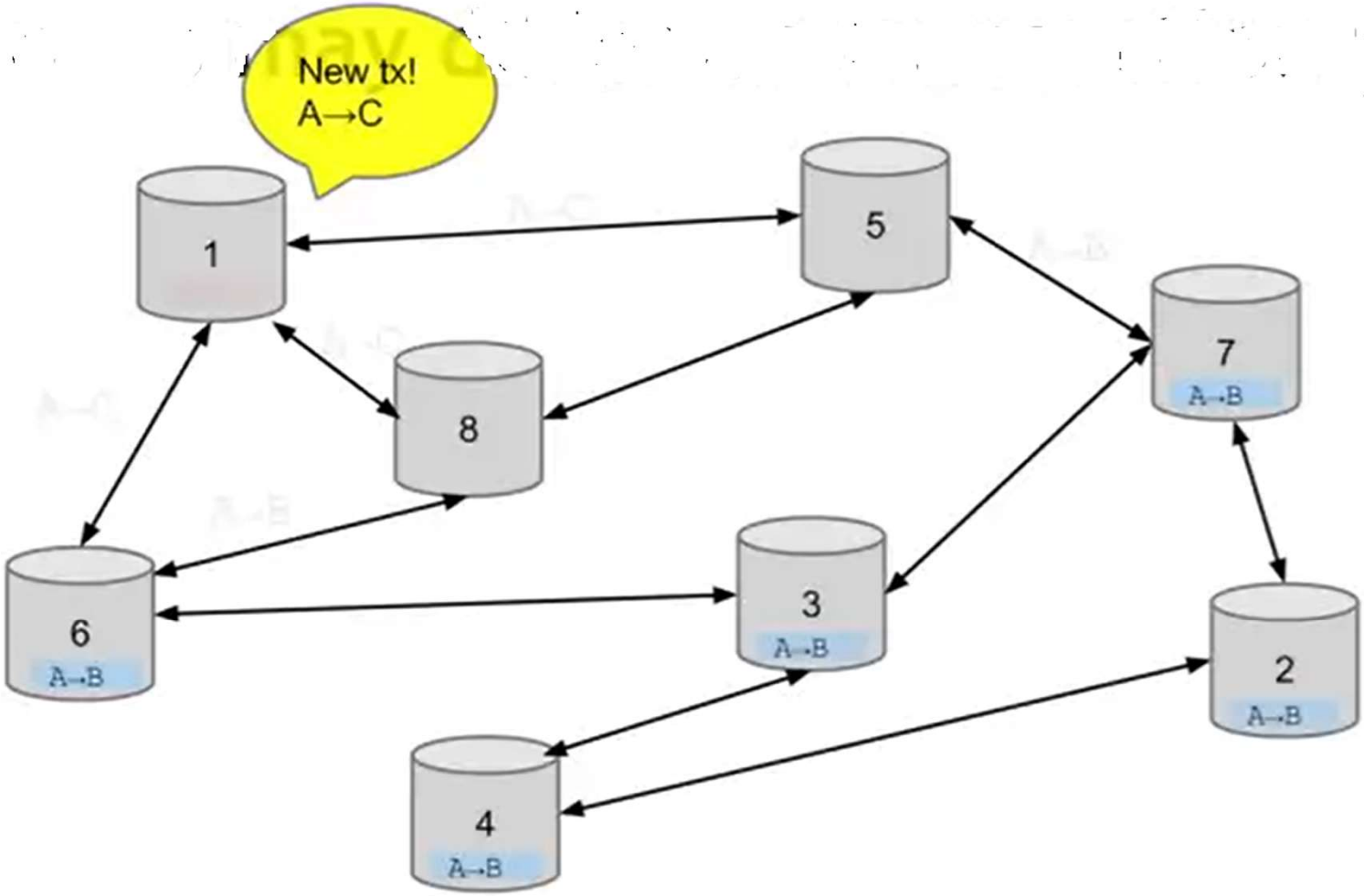
Transaction Propagation - Flooding



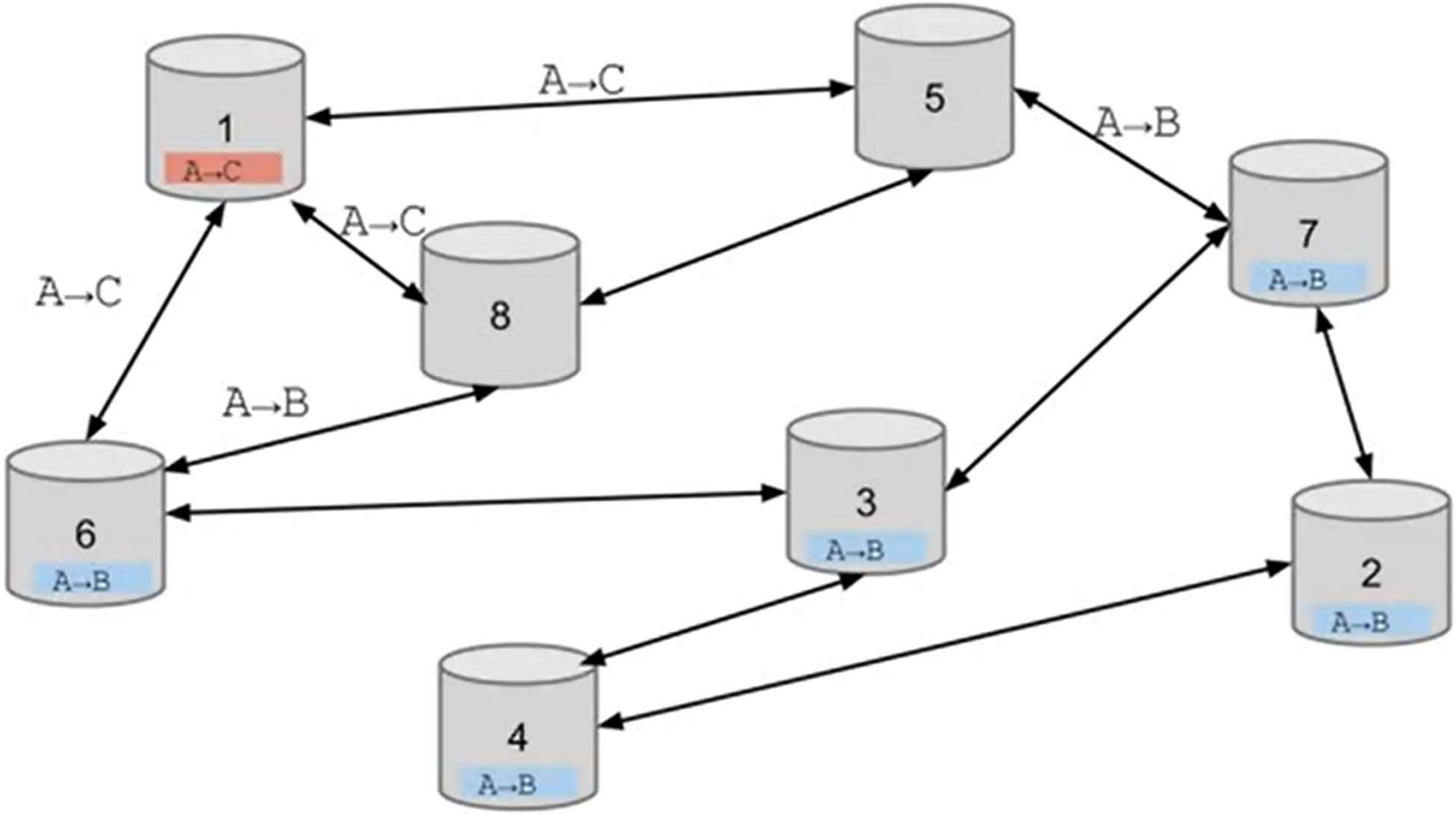
Relay a proposed transaction?

- Transaction valid with current blockchain
- Script (default) matches a whitelist - Avoid Unusual Scripts
- Haven’t seen before - Avoid Infinite loops
- Doesn’t conflict with others I’ve relayed - Avoid double spends

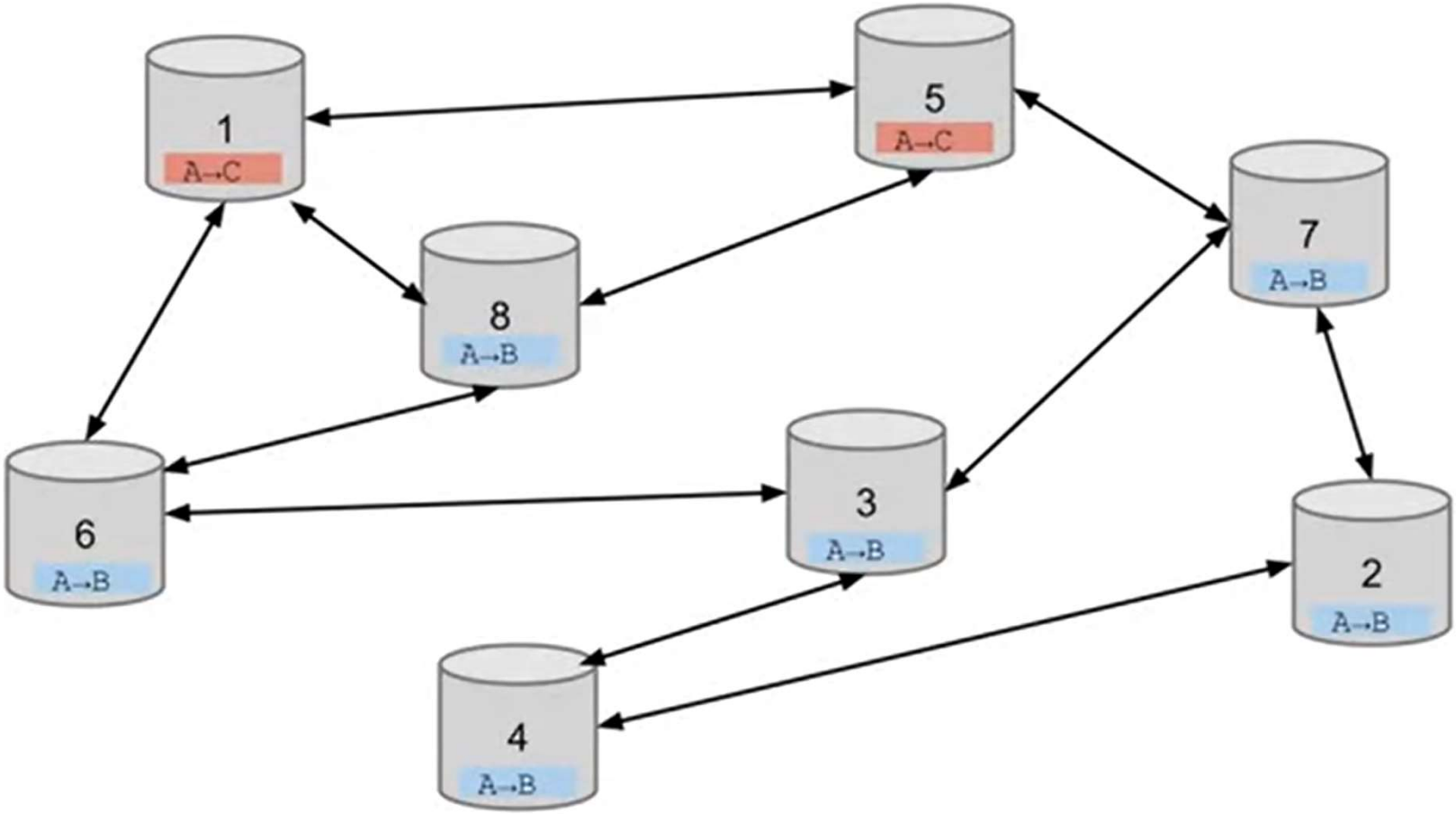
Nodes may differ on transaction pool



Nodes may differ on transaction pool



Nodes may differ on transaction pool



Race Conditions

Transactions or blocks may *conflict*

- Default behavior: accept what you hear first
- Network position matters
- Miners may implement other logic!

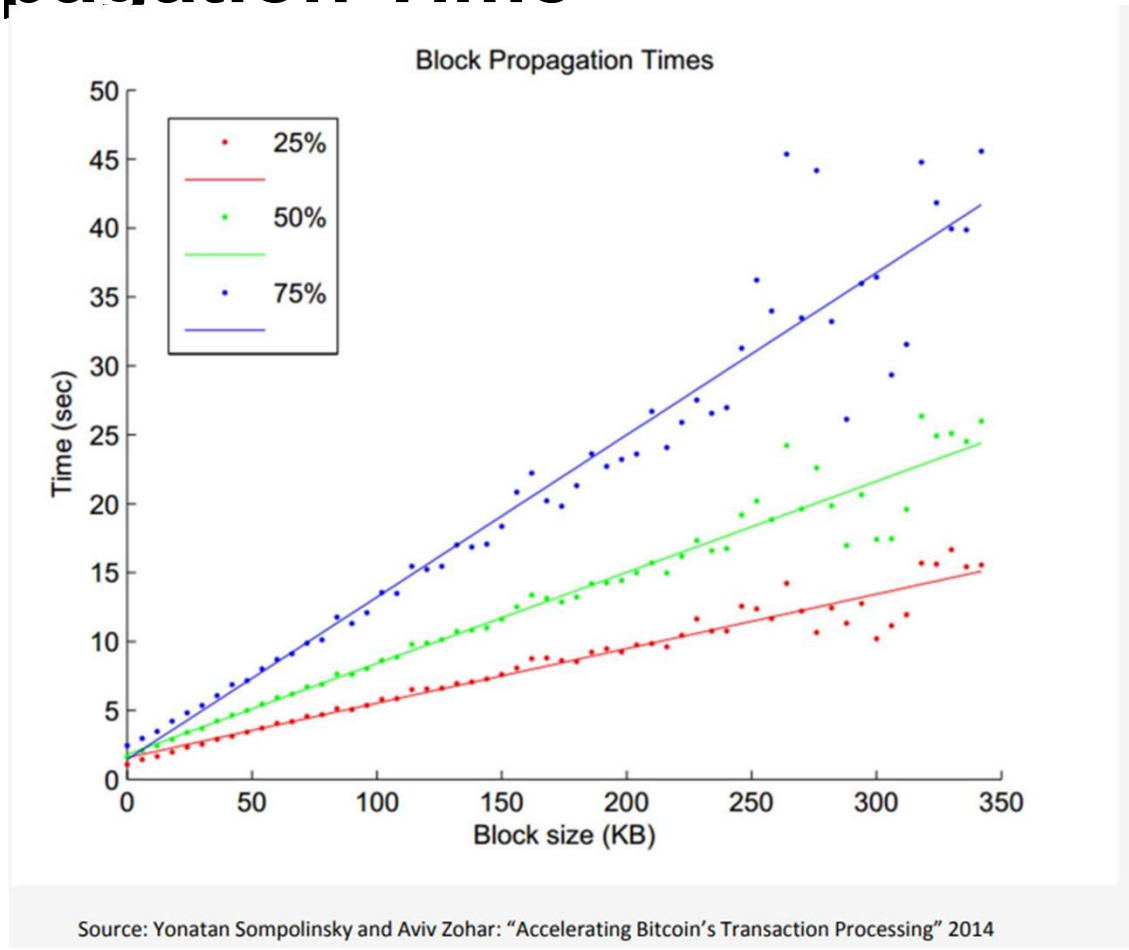
Block propagation nearly identical

Relay a new block when you hear it if:

- Block meets the hash target
- Block has all valid transactions
  - Run *all* scripts, even if you wouldn't relay
- Block builds on current longest chain
  - Avoid forks

Sanity check
Also may be ignored...

Block Propagation Time



Source: Yonatan Sompolinsky and Aviv Zohar: "Accelerating Bitcoin's Transaction Processing" 2014

How big is the network?

- Impossible to measure exactly
- Estimates upto 1 Million IP addresses / Month
- Only about 5-10k “Full nodes”
  - Permanently connected
  - Fully-Validate
- **This number may be dropping.**

Bitcoin full node distribution

RANK	COUNTRY	NODES
1	United States	2466 (24.32%)
2	Germany	1936 (19.09%)
3	France	674 (6.65%)
4	Netherlands	484 (4.77%)
5	China	402 (3.96%)
6	Canada	402 (3.96%)
7	United Kingdom	351 (3.46%)
8	Singapore	312 (3.08%)
9	Russian Federation	270 (2.66%)
10	Japan	248 (2.45%)
	More (102)	

TOTAL 10140 NODES AT 5 PM ON JAN 23, 2019

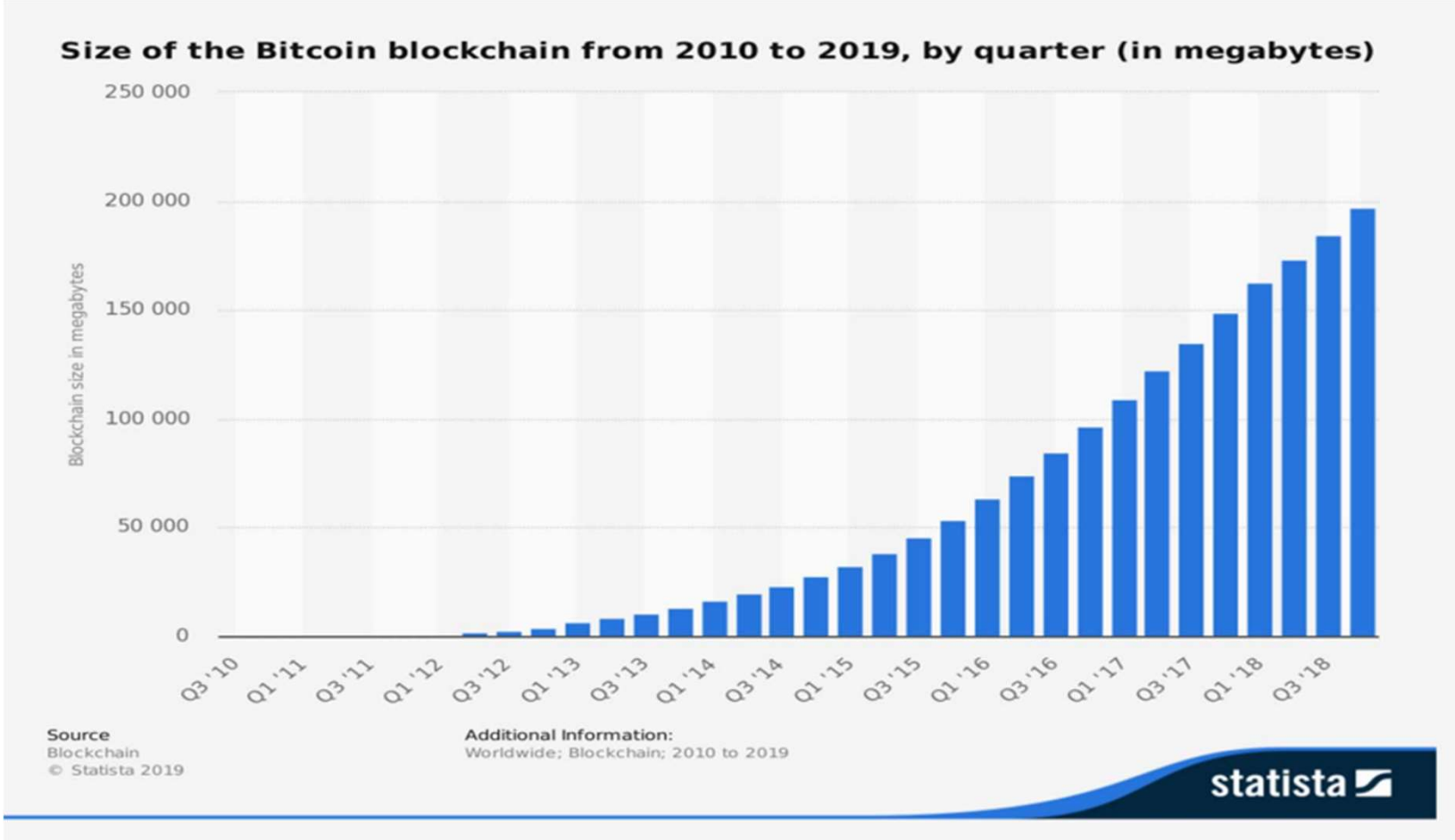
Nodes on the Bitcoin P2P Network

- Two main **types of nodes** :
  - Full nodes and
  - Simple Payment Verification (SPV) nodes.
- **Full nodes:** are implementations of Bitcoin Core clients performing the wallet, miner, full blockchain storage, and network routing functions.
- However, it is not necessary for all nodes in a Bitcoin network to perform all these functions.
- **SPV nodes or lightweight clients** : perform only wallet and network routing functionality.

Thick Client / Fully-Validating Nodes

- Permanently Connected
- Store entire blockchain
- Hear and forward every node / transaction

Fully-Validating Nodes: Storage Costs



Thin / SPV (Not Fully-validating Nodes)

- A simplified payment verification (SPV) method is used to **allow them to operate without storing the full blockchain**.
- SPV node is becoming the **most common form of bitcoin node**, especially for bitcoin wallets.
- SPV nodes **download only the block headers and do not download the transactions** included in each block. The resulting chain of blocks, without transactions, is **1,000 times smaller than the full blockchain**.

Thin / SPV (Not Fully-validating Nodes)

- A full blockchain node verifies a transaction by **checking the entire chain of thousands of blocks** below it in order to guarantee that the UTXO is not spent,
- whereas an SPV node will **verify the chain of all blocks (but not all transactions)** and link that chain to the transaction of interest.
- **Eg.:** when examining a transaction in block 300,000, a full node links all 300,000 blocks down to the genesis block and builds a full database of UTXO, establishing the validity of the transaction by confirming that the UTXO remains unspent.

Thin / SPV (Not Fully-validating Nodes)

- An SPV node **cannot validate whether the UTXO is unspent**. Instead, the SPV node will establish a **link between the transaction and the block that contains it,** using a merkle path.
- The SPV node establishes the existence of a transaction in a block by **requesting a merkle path proof and by validating the Proof-of-Work** in the chain of blocks. However, a transaction's existence can be "hidden" from an SPV node.
- This vulnerability can be used in a **denial-of-service attack or for a double-spending attack** against SPV nodes.

Thin / SPV (Not Fully-validating Nodes)

- To defend against such attack, an SPV node needs to **connect randomly to several nodes**, to increase the probability that it is in **contact with at least one honest node**.
- This need to randomly connect means that SPV nodes also are **vulnerable to network partitioning attacks or Sybil attacks**, where they are connected to fake nodes or fake networks and d**o not have access to honest nodes** or the real bitcoin network.

Thin / SPV (Not Fully-validating Nodes)

- **Idea:** Don’t Store everything
- Store block headers only
- Request transactions as needed - To verify incoming payment
- Trust fully-validating nodes
- 1000x cost savings! (200 GB->200MB)

Nodes on the Bitcoin P2P Network

- **Pool Protocol Servers** : Non-standard, but heavily used nodes.
  - make use of alternative protocols, such as the stratum protocol
  - These nodes are used in mining pools.
  - Nodes that only compute hashes use the stratum protocol to submit their solutions to the mining pool
- **Mining nodes:** Perform only mining functions.
- It is possible to run SPV software that runs a wallet and network routing function without a blockchain.
- **SPV** clients only download the headers of the blocks while syncing with the network

Courtesy: https://en.bitcoin.it/wiki/Stratum_mining_protocol

Stratum Protocol

- It is a line-based protocol that makes use of plain TCP sockets and human-readable JSON-RPC to operate and communicate between nodes.
- Stratum is commonly used to connect to mining pools.

Courtesy: https://en.bitcoin.it/wiki/Stratum_mining_protocol

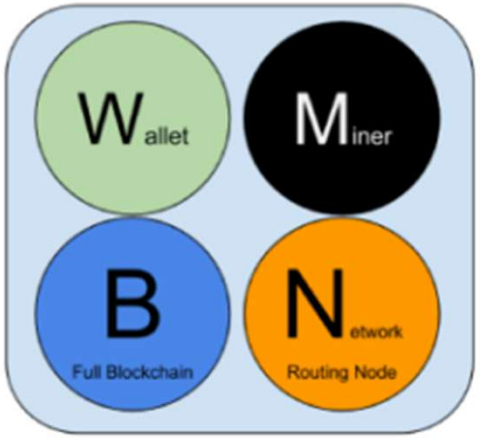
Extended Bitcoin Network

- The main bitcoin network, running the bitcoin P2P protocol, consists of between **5,000 and 8,000 listening nodes** running various versions of the bitcoin reference client (Bitcoin Core) and
- A few hundred nodes running various other implementations of the bitcoin P2P protocol, such as **Bitcoin Classic, Bitcoin Unlimited, BitcoinJ, Lib-bitcoin, btcd, and bcoin.**
- A small percentage of the nodes on the bitcoin P2P network are also **mining nodes**, competing in the mining process, validating transactions, and creating new blocks.

Extended Bitcoin Network

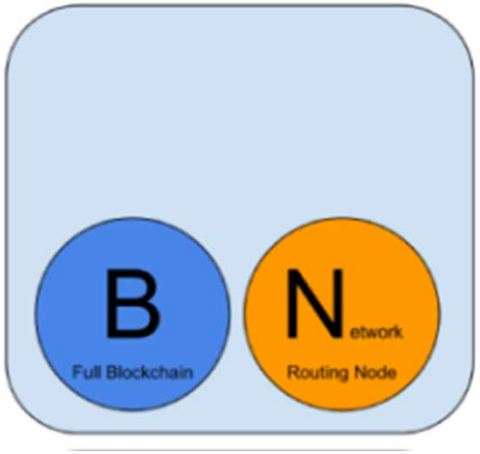
- These nodes **act as network edge routers**, allowing various other services (exchanges, wallets, block explorers, merchant payment processing) to be built on top.
- The extended bitcoin network includes the **network running the bitcoin P2P protocol, as well as nodes running specialized protocols**.
- There are a **number of pool servers and protocol gateways** that connect nodes running other protocols.
- These other protocol nodes are mostly pool mining nodes & lightweight wallet clients, which do not **carry a full copy of the blockchain**.

Extended Bitcoin Network



Reference Client (Bitcoin Core)

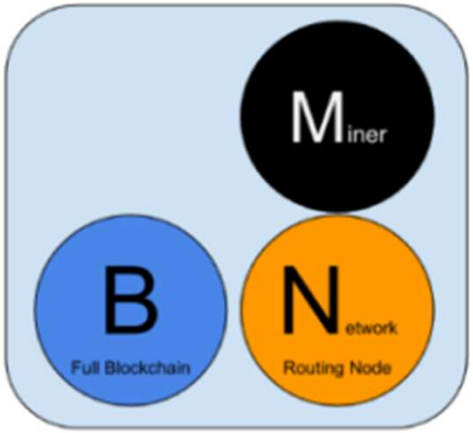
Contains a Wallet, Miner, full Blockchain database, and Network routing node on the bitcoin P2P network.



Full Block Chain Node

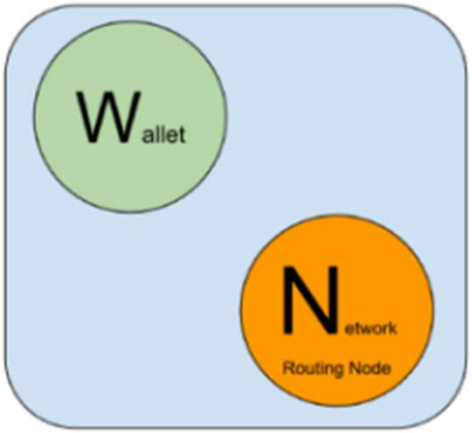
Contains a full Blockchain database, and Network routing node on the bitcoin P2P network.

Extended Bitcoin Network



Solo Miner

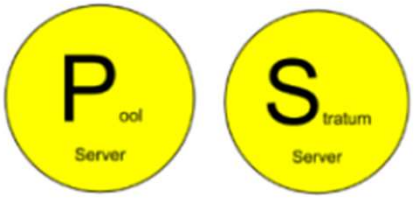
Contains a mining function with a full copy of the blockchain and a bitcoin P2P network routing node.



Lightweight (SPV) wallet

Contains a Wallet and a Network node on the bitcoin P2P protocol, without a blockchain.

Extended Bitcoin Network

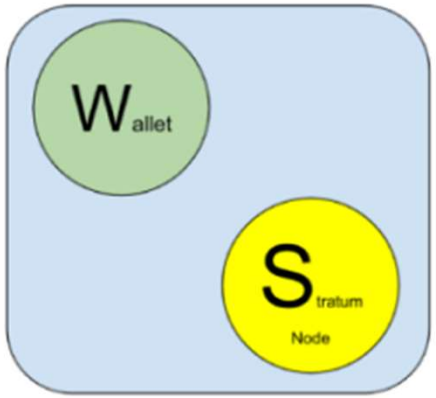


Pool Protocol Servers

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

Mining Nodes

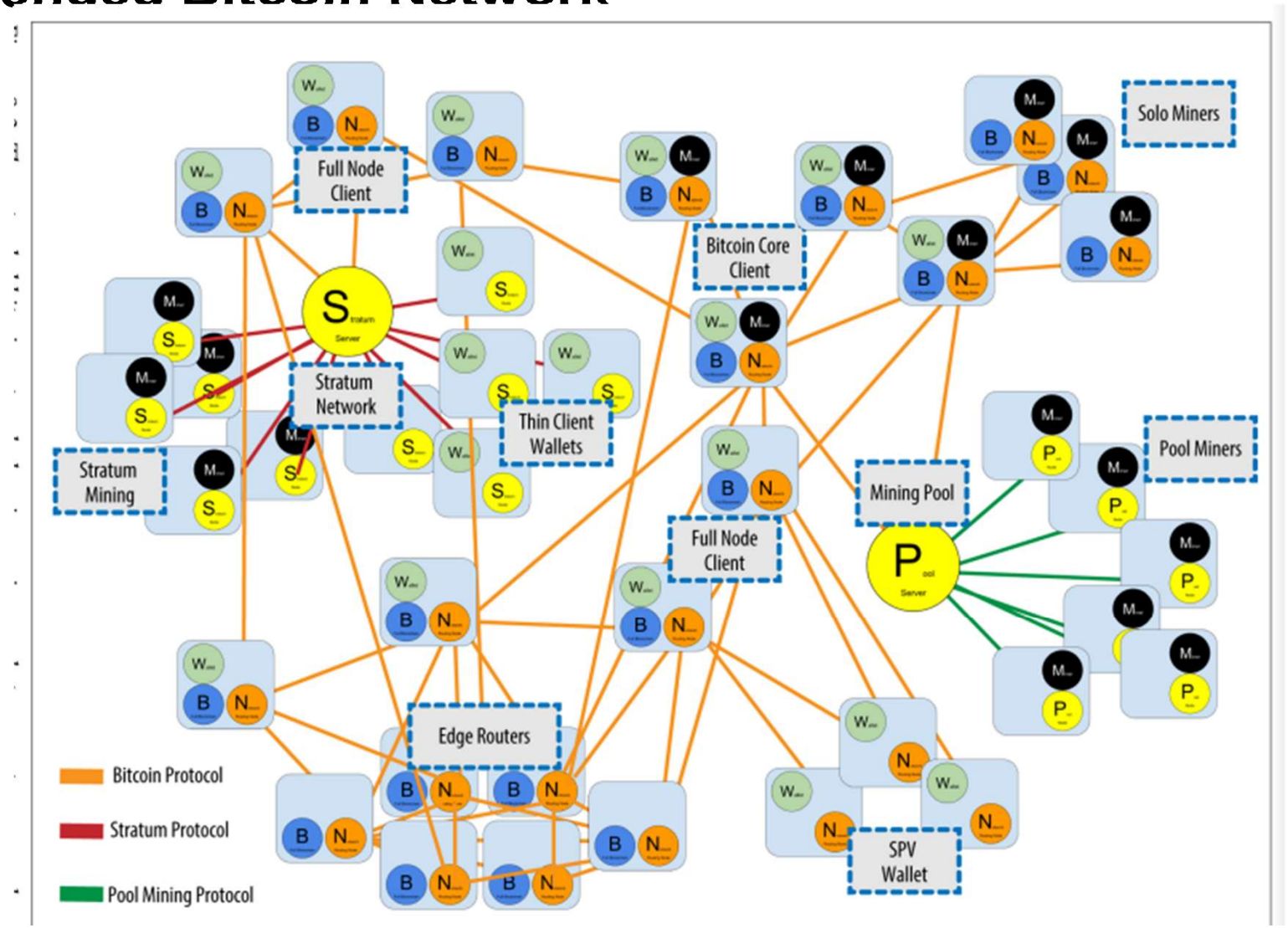
Contain a mining function, without a blockchain, with the Stratum protocol node (S) or other pool (P) mining protocol node.



Lightweight (SPV) Stratum wallet

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

Extended Bitcoin Network



Bitcoin Relay Networks - Why?

- Bitcoin miners are engaged in a **time-sensitive competition** to solve the Proof-of-Work problem and extend the blockchain.
- While participating in this competition, bitcoin miners must **minimize the time between the propagation of a winning block and the beginning of the next round of competition**.
- In mining, network latency is directly related to profit margins.
- A **Bitcoin Relay Network** is a **network that attempts to minimize the latency** in the transmission of blocks between miners.

Bitcoin Relay Networks

- The network consisted of several specialized nodes **hosted on the Amazon Web Services infrastructure** and served to connect the majority of miners and mining pools.
- The original Bitcoin Relay Network was replaced in 2016 with the introduction of the **Fast Internet Bitcoin Relay Engine or FIBRE**, created by core developer Matt Corallo.
- FIBRE is a **UDP-based relay network** that relays blocks within a network of nodes.
- FIBRE implements **compact block optimization** to further reduce the amount of data transmitted and the network latency.

Bloom Filters

- A bitcoin feature, called Bloom filters to **address the privacy risks of SPV nodes**.
- A bloom filter is a **probabilistic search filter**, a way to describe a desired pattern without specifying it exactly.
- Bloom filters offer an **efficient way to express a search pattern** while protecting privacy. They are used by SPV nodes to ask their peers for transactions matching a specific pattern, without revealing exactly which addresses, keys, or transactions they are searching for.

Bloom Filters

- Bloom filters serve this function by **allowing an SPV node to specify a search pattern for transactions** that can be tuned toward precision or privacy.
- A **more specific bloom filter will produce accurate results**, but at the expense of revealing what patterns the SPV node is interested in, thus revealing the addresses owned by the user's wallet.
- A **less specific bloom filter will produce more data about more transactions, many irrelevant to the node**, but will allow the node to maintain better privacy.

Bloom Filters

- Bloom filters are implemented as a variable-size array of **N binary digits (a bit field)** and a **variable number of M hash functions**.
- The hash functions are designed to always **produce an output that is between 1 and N**, corresponding to the array of binary digits.
- The hash functions are generated deterministically, so that any node implementing a bloom filter will always use the same hash functions and get the same results for a specific input.
- By choosing different length (N) bloom filters and a different number (M) of hash functions, the **bloom filter can be tuned, varying the level of accuracy and therefore privacy**.

Bloom Filters

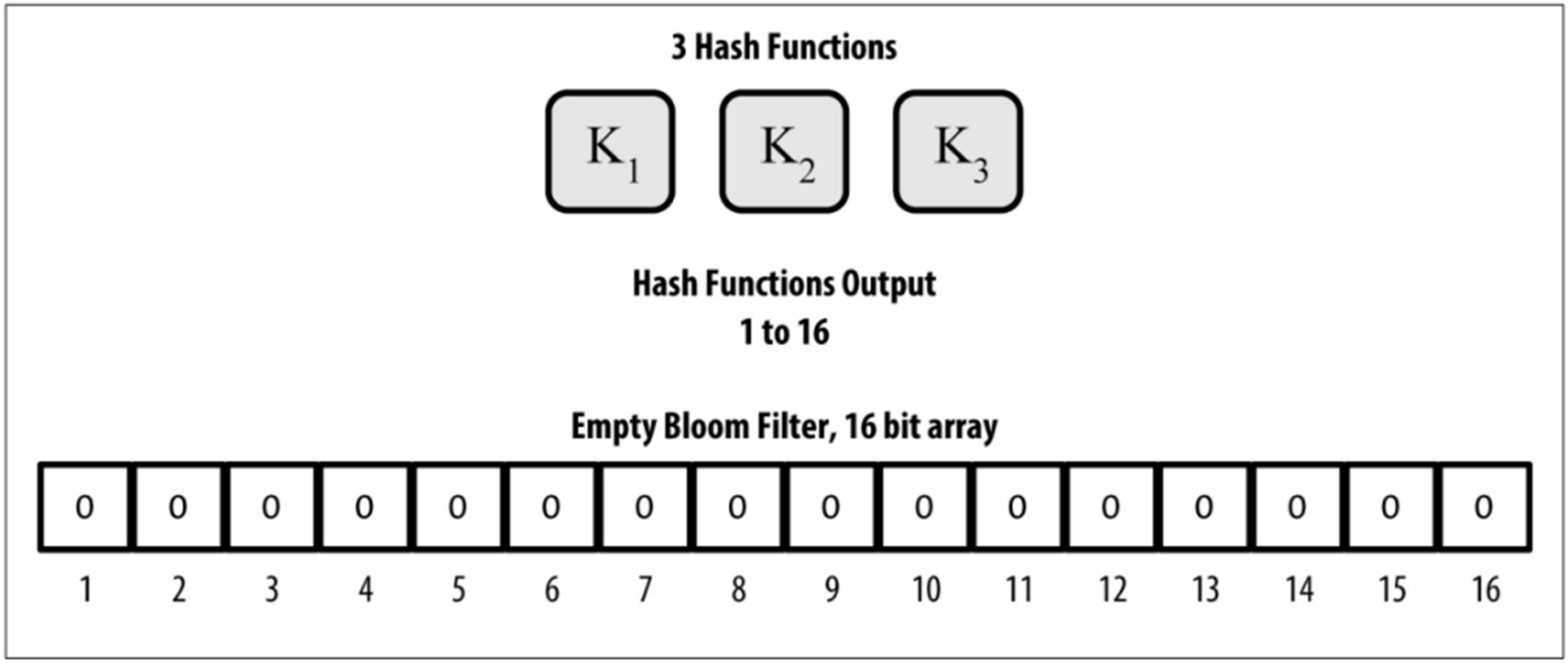


Figure 8-8. An example of a simplistic bloom filter, with a 16-bit field and three hash functions

Bloom Filters

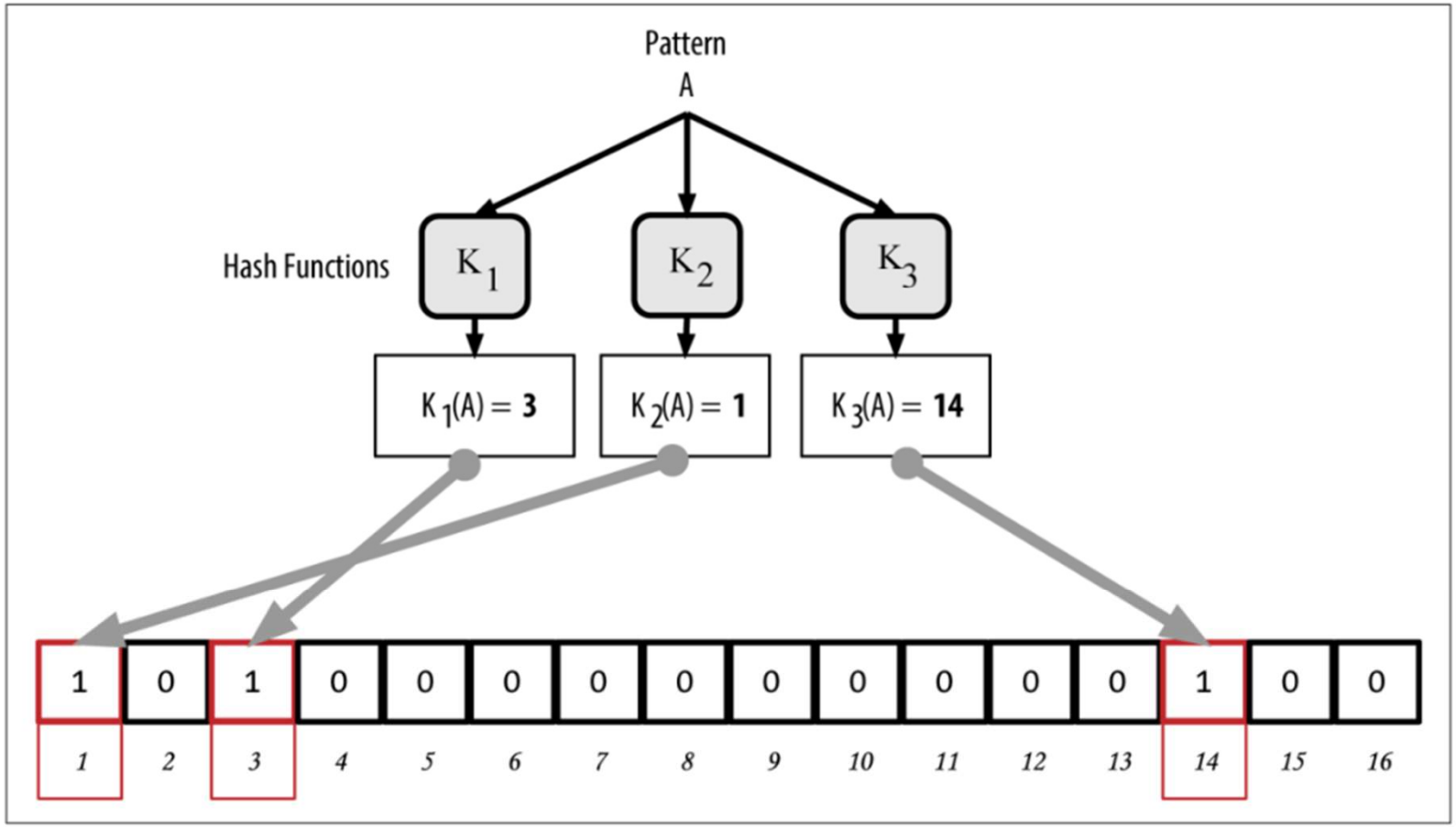


Figure 8-9. Adding a pattern "A" to our simple bloom filter

Bloom Filters

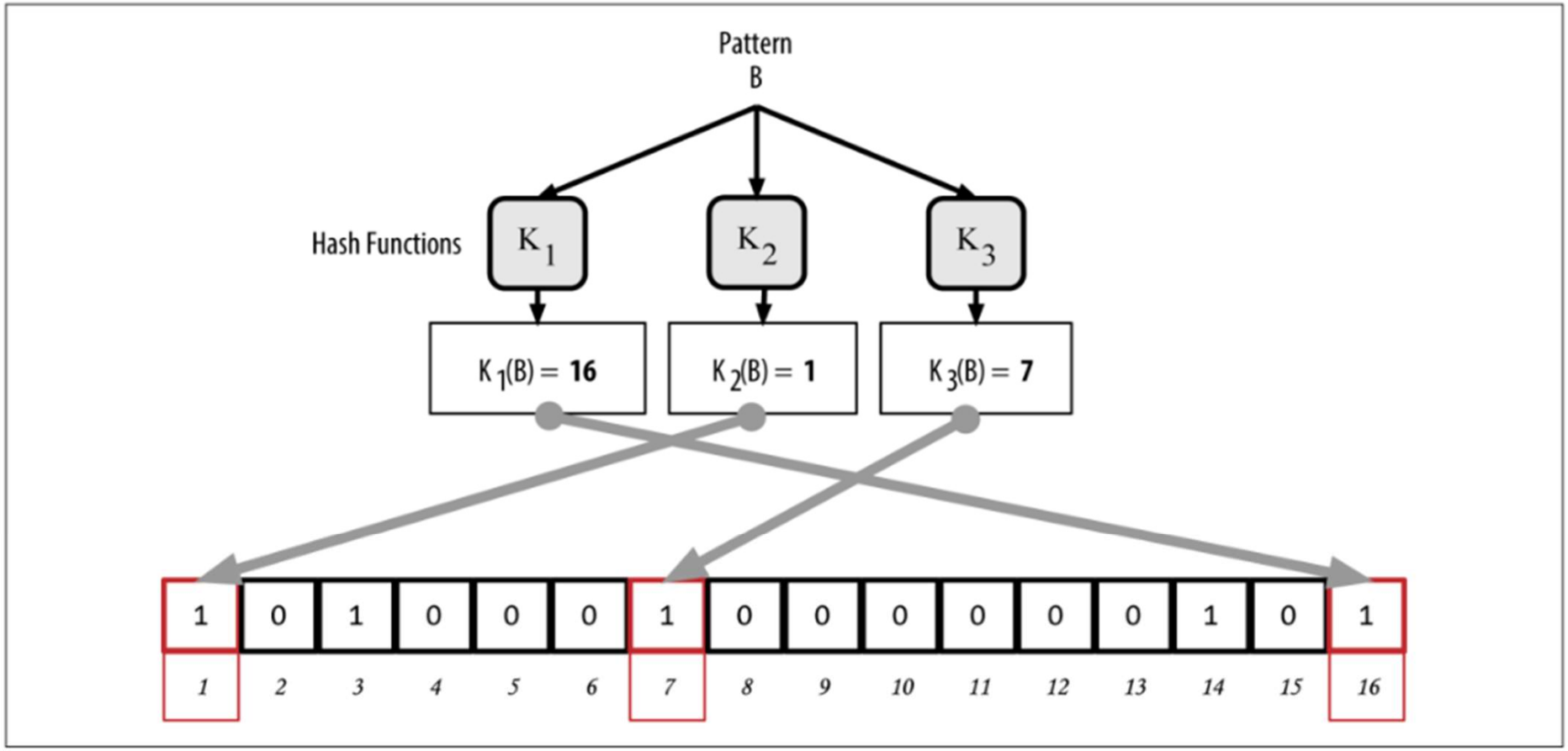


Figure 8-10. Adding a second pattern "B" to our simple bloom filter

Bloom Filters

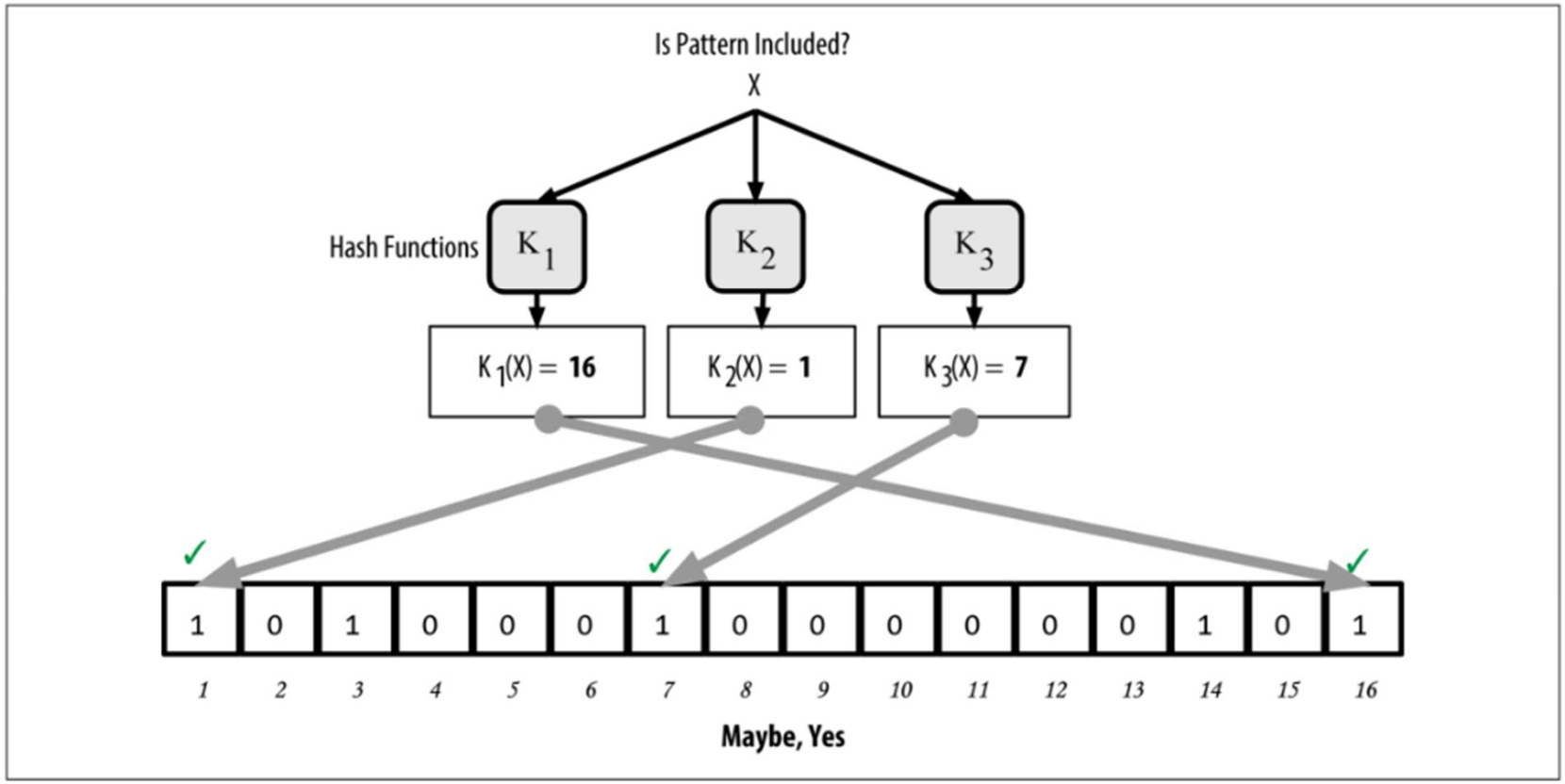


Figure 8-11. Testing the existence of pattern "X" in the bloom filter. The result is a probabilistic positive match, meaning "Maybe."

Bloom Filters

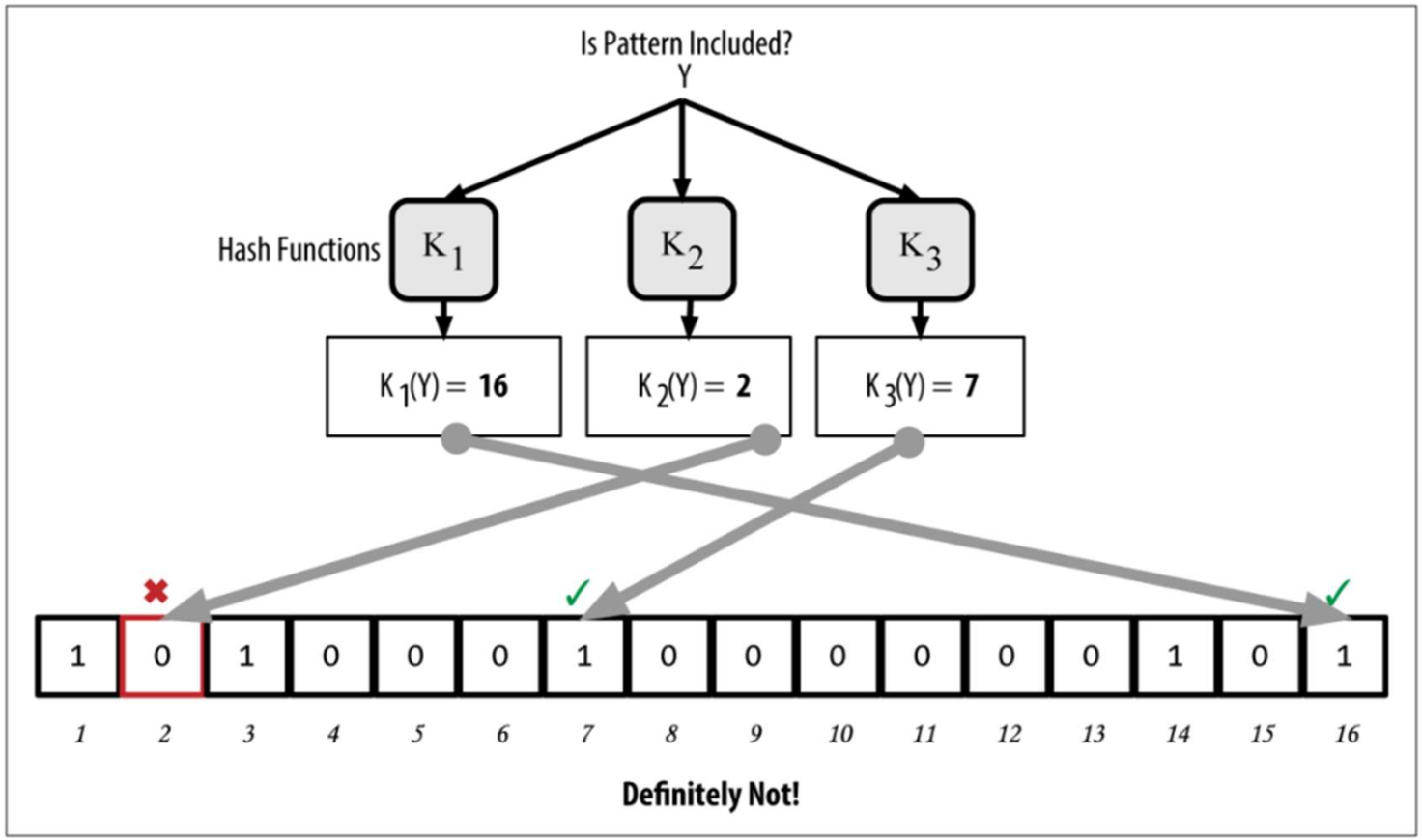


Figure 8-12. Testing the existence of pattern "Y" in the bloom filter. The result is a definitive negative match, meaning "Definitely Not!"

How SPV nodes use Bloom Filters?

- An SPV node will **initialize a bloom filter as "empty"**; in that state the bloom filter will not match any patterns.
- The **SPV node will then make a list of all the addresses, keys, and hashes** that it is interested in.
- It will do this by extracting the public key hash and script hash and transaction IDs from any UTXO controlled by its wallet.
- The **SPV node then adds each of these to the bloom filter**, so that the bloom filter will "match" if these patterns are present in a transaction, without revealing the patterns themselves.

How SPV nodes use Bloom Filters?

- The SPV node will then **send a filterload message to the peer**, containing the bloom filter to use on the connection. On the peer, bloom filters are checked against each incoming transaction.
- The **full node checks several parts of the transaction** against the bloom filter, looking for a match including:
    - The transaction ID
    - The data components from the locking scripts of each of the transaction outputs (every key and hash in the script)
    - Each of the transaction inputs
    - Each of the input signature data components (or witness scripts)

How SPV nodes use Bloom Filters?

- By checking against all these components, bloom filters can be used to match public key hashes, scripts, **OP_RETURN values, public keys in signatures,** or any future component of a smart contract or complex script.
- After a filter is established, the **peer will then test each transaction's outputs against the bloom filter.** Only transactions that match the filter are sent to the node.
- In response to a getdata message from the node, peers will **send a merkleblock message** that contains only block headers for blocks matching the filter and a merkle path for each matching transaction.
- The peer will then also **send tx messages containing the transactions** matched by the filter.

SPV Nodes and Privacy

- Nodes that implement **SPV have weaker privacy than a full node**. A full node receives all transactions and therefore reveals no information about whether it is using some address in its wallet.
- An **SPV node receives a filtered list of transactions** related to the addresses that are in its wallet. As a result, it reduces the privacy of the owner.
- **Bloom filters** are a way to reduce the loss of privacy.
- However, even with bloom filters, an adversary **monitoring the traffic of an SPV client or connected to it** directly as a node in the P2P network can collect enough information over time to learn the addresses in the wallet of the SPV client.

Encrypted and Authenticated Connections

- To **increase the privacy and security of the bitcoin P2P network**, there are two solutions that provide encryption of the communications:
  - **Tor Transport and**
  - **P2P Authentication and Encryption with BIP-150/151.**
- **Tor Transport:**
  - The **Onion Routing network, is a software project** and network that offers encryption and encapsulation of data through randomized network paths that offer anonymity, untraceability and privacy.
  - Bitcoin Core offers several configuration options that **allow you to run a bitcoin node with its traffic transported** over the Tor network. Bitcoin Core can also offer a **Tor hidden service** allowing other Tor nodes to connect to your node directly over Tor.

Encrypted and Authenticated Connections

P2P Authentication and Encryption with BIP-150/151:

- Two Bitcoin Improvement Proposals, **BIP-150 and BIP-151**
- Two BIPs define optional services that may be offered by compatible bitcoin nodes.
- BIP-151 enables negotiated encryption for all communications between two nodes that support BIP-151.
- BIP-150 offers optional peer authentication that allows nodes to authenticate each other's identity using ECDSA and private keys.
- BIP-150 requires that prior to authentication the two nodes have established encrypted communications as per BIP-151

Encrypted and Authenticated Connections

P2P Authentication and Encryption with BIP-150/151:

- BIP-150 and BIP-151 **allow users to run SPV clients** that connect to a trusted full node, using encryption and authentication to protect the privacy of the SPV client.
- Authentication can be u**sed to create networks of trusted bitcoin nodes** and **prevent Man-in-the-Middle attacks**.
- Finally, P2P encryption, if deployed broadly, would **strengthen the resistance of bitcoin to traffic analysis and privacy-eroding surveillance**, especially in totalitarian countries where internet use is heavily controlled and monitored.

Transaction Pool

- Almost every node on the bitcoin network maintains a temporary list of unconfirmed transactions called the **memory pool, mempool, or transaction pool.**
- Nodes use this **pool to keep track of transactions** that are known to the network but are not yet included in the blockchain.
- For example, a **wallet node will use the transaction pool to track incoming payments** to the user's wallet that have been received on the network but are not yet confirmed.
- As transactions are received and verified, they are added to the transaction pool and relayed to the neighboring nodes to propagate on the network.

Transaction Pool

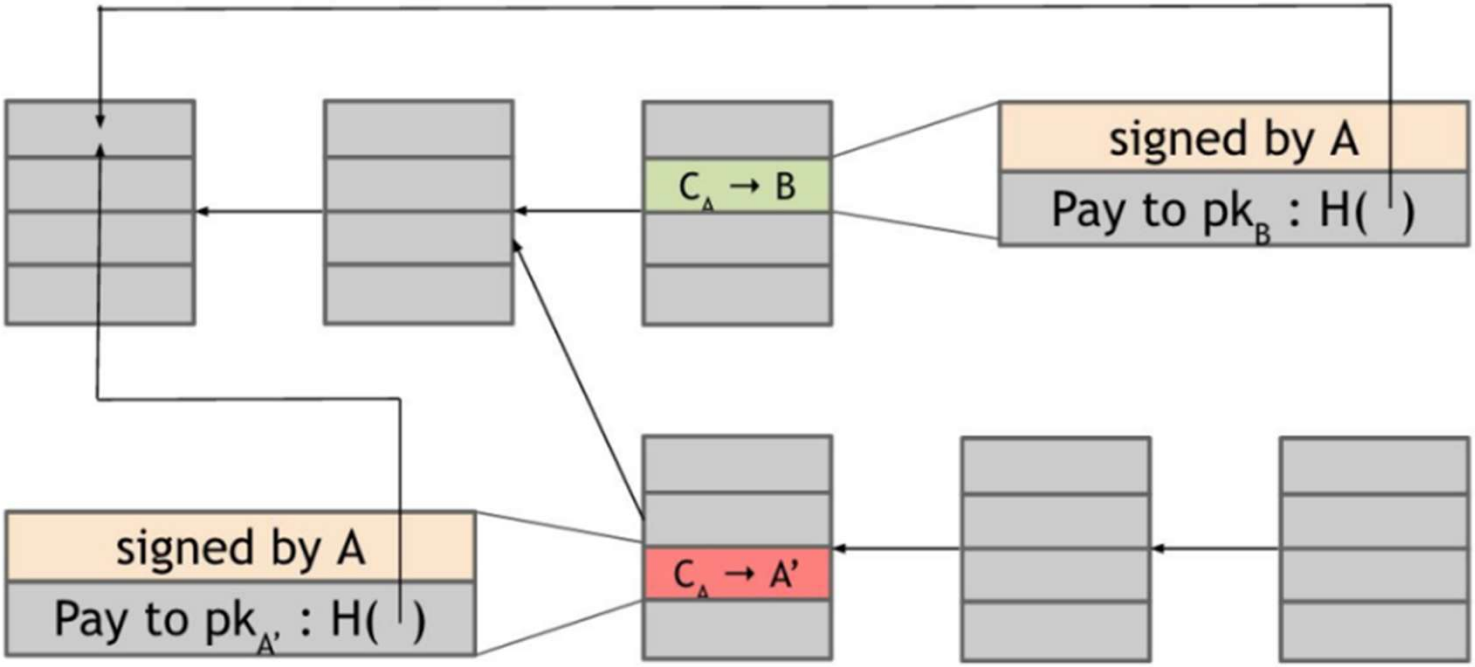
- Maintain a **separate pool of orphaned transactions**. If a transaction's inputs refer to a transaction that is not yet known, such as a missing parent, the **orphan transaction will be stored temporarily in the orphan pool until the parent transaction arrives**.
- Both the transaction pool and orphan pool (where implemented) are **stored in local memory and are not saved on persistent storage**; rather, they are dynamically populated from incoming network messages. When a node starts, both pools are empty and are gradually populated with new transactions received on the network.

Transaction Pool

- **Bitcoin client also maintain a UTXO database or pool**, which is the set of all unspent outputs on the blockchain. Although the name “UTXO pool” sounds similar to the transaction pool, it represents a different set of data.
- Unlike the transaction and orphan pools, the **UTXO pool is not initialized empty but instead contains millions of entries of unspent transaction outputs**, everything that is unspent from all the way back to the genesis block. The UTXO pool may be housed in local memory or as an indexed database table on persistent storage.

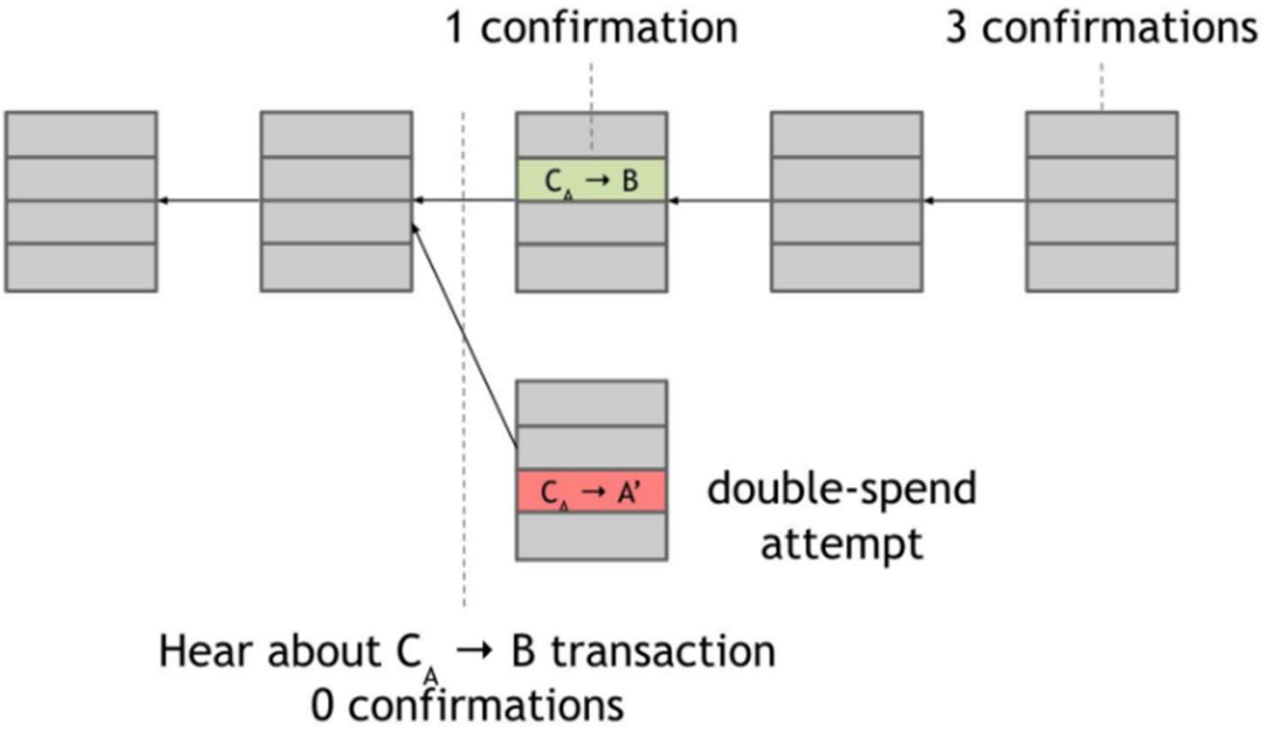
Can we give nodes an incentive for behaving honestly?

- A Double Spend Attempt



Can we give nodes an incentive for behaving honestly?

- Bob the merchant's View



Incentive based Engineering

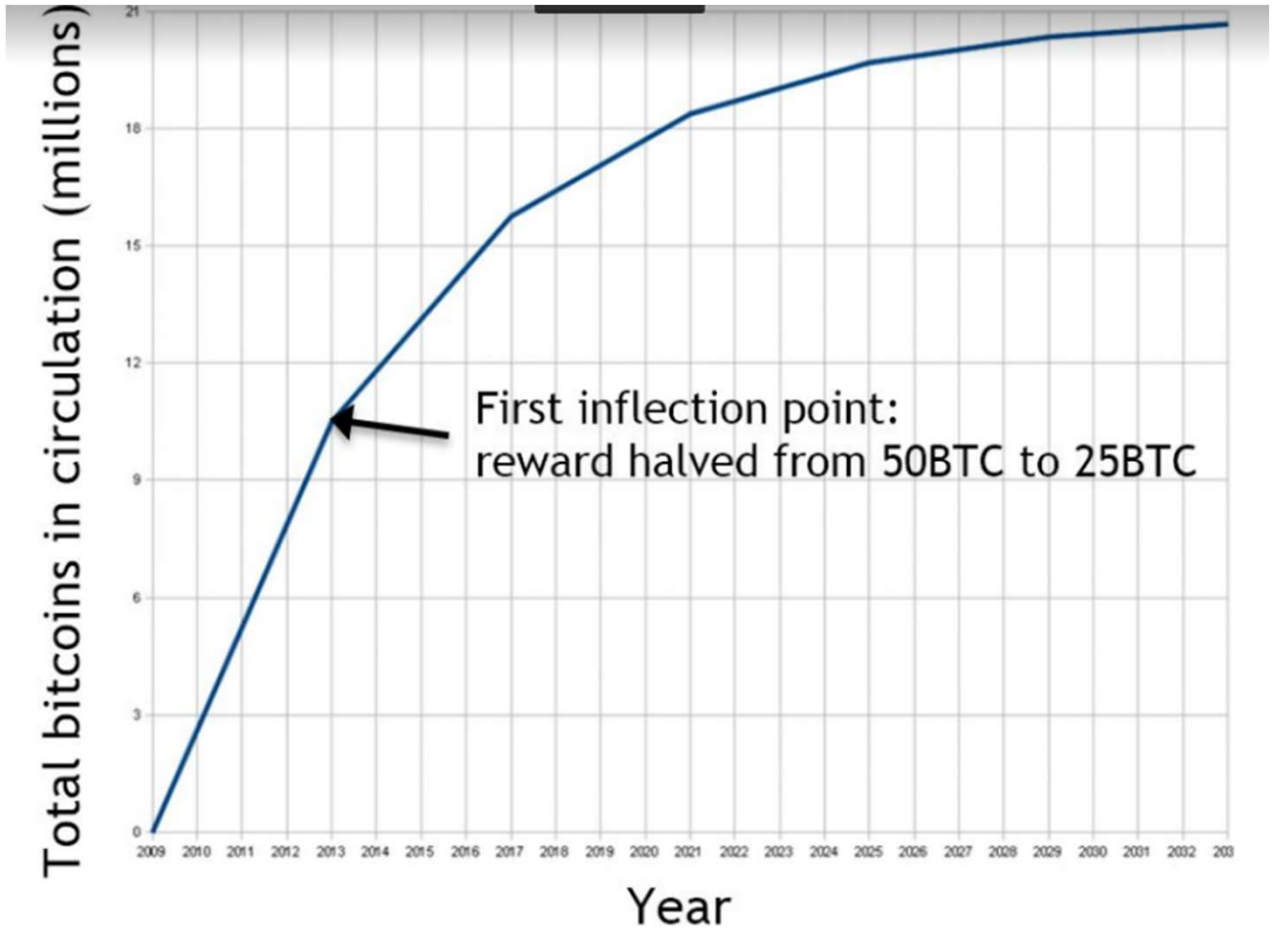
- At a particular time, the **miner has a large number of transactions** that need to be processed.
- Also, a **sender can send only some necessary information** about the transaction since the maximum block size is already predefined by the system. Hence, to make the work of Miner easier by sending only relevant information to make the entire process faster.
- For **incentivizing the Miner** for his correct decisions during the transactions, the **sender generally sends some incentives (rewards) in form of the cryptocurrency**, in which the transaction takes place. Generally, a small percentage of the whole transaction that takes place.
- For Example – For a $250 Bitcoin transaction, the sender may send the incentive to the Miner of about $15 in Bitcoin

Incentive Mechanisms in Bitcoin

1. Block Rewards:

- A Block Reward is basically a reward given to a Bitcoin Miner for every Block for which he has solved the complex mathematical algorithm, and thus, the transaction records are added to the Blockchain.
- Halving Process is associated with Bitcoin that the Block Reward just halves itself once every 2016 Blocks are mined, every 4 years.
- Bitcoin every 4 years-
  - **2012-** 25.00 BTC.
  - **2016-** 12.50 BTC.
  - **2020-** 6.25 BTC

The block reward is cut in half every four years limiting the total supply of bitcoins to 21 million



Incentive Mechanisms in Bitcoin

2. Transaction Fee / Incentive Pools:

- The creator of any transaction can choose to make the total value of the transaction outputs less than the total value of its inputs.
- Whoever creates the block that first puts that transaction into the blockchain gets to collect the difference, which acts a transaction fee.
- So if you're a node that's creating a block that contains, say, 200 transactions, then the sum of all those 200 transaction fees is paid to the address that you put into that block.

Incentive Mechanisms in Bitcoin

2. Transaction Fee / Incentive Pools:

An Incentive Pool for Blockchain works in the following ways :

- A group of Miners tends to form an Incentive Pool so that, they can create blocks together for validating transactions, and then share the Incentives among all the Miners with equality.
- When a Pool Miner finds out the solution of the block header, he cannot mess with that transaction without removing the validation from it.

Cost of Mining in Bitcoin

If

mining reward > mining cost

then miner profits

where

mining reward = block reward + tx fees

mining cost = hardware cost + operating costs (electricity, cooling, etc.)

References

- Incentive based Engineering (Author - Princeton) : Link
- https://www.geeksforgeeks.org/blockchain-incentives-to-miners/
- Peer-to-Peer Network Architecture (Mastering Bitcoin - 2nd Edition) : Link