Conceptual Architecture for Bitcoin Core

by
Gabriel Lemieux,
Jack Taylor,
Jack Fallows,
Jiacheng Liu,
Maninderpal Badhan,
Nil Shah

Abstract:

Bitcoin Core is an implementation of the Bitcoin Protocol created by Satoshi Nakamoto. By the end of this paper, it should be evident why Bitcoin was revolutionary and successful. The insight that Bitcoin has offered will propel a new form of currency that is decentralized and fully digital. We will explain the components involved in the Bitcoin Core software from a purely conceptual viewpoint, along with some examples to help illustrate its use.

Introduction and Overview:

Before the invention of money, people would trade goods with each other as a form of commerce; for example, a farmer would trade eggs with a fisherman in exchange for some fish. As time went on, people began using coins made from precious minerals to pay for goods. The way we represent and exchange value within our society has changed many times due to continuous technological advances and increased **globalization**. The advent of the internet and e-commerce has led to a high demand for the secure purchasing of goods online, a situation where the **identities** and **trustworthiness** of two participating individuals are hard to verify. E-commerce has also necessitated the exchange of currencies from different countries as trade is no longer defined by geographic location. To address these issues with the current currency system we use third parties such as financial institutions to verify and validate our transactions. This approach to e-commerce is referred to as a **centralized system** where there is one point of control. This centralized system works quite well, however, it is not foolproof and still allows entities to reverse transactions and occasionally perform dishonest transactions. This also places all authority under one entity whose interests may be hard to gauge. The **Bitcoin Core** system we will be outlining in this document removes the need for a third party to verify transactions and renders transactions irreversible.

Bitcoin was first announced in 2008 when the creator of Bitcoin, Satoshi Nakamoto, released a paper named “Bitcoin: A Peer-to-Peer Electronic Cash System”. The paper explained how a **peer-to-peer system** could be used to make electronic transactions without relying on trust like in centralized baking systems. In 2009 the **Bitcoin network** was released with the

creator mining the first block of bitcoin, which was named the **genesis block** of bitcoin (Nakamoto). Bitcoin is an **open-source** system which means that anyone is able to view the code and contribute to it. After Satoshi's departure in 2011, many active contributors changed the system's name to Bitcoin Core to differentiate itself from the original network. Bitcoin Core is managed by contributors all around the world who volunteer their free time to keep the system running and updated. To this day there have been a total of 900 contributors with its latest version being Bitcoin Core 24.0.1.

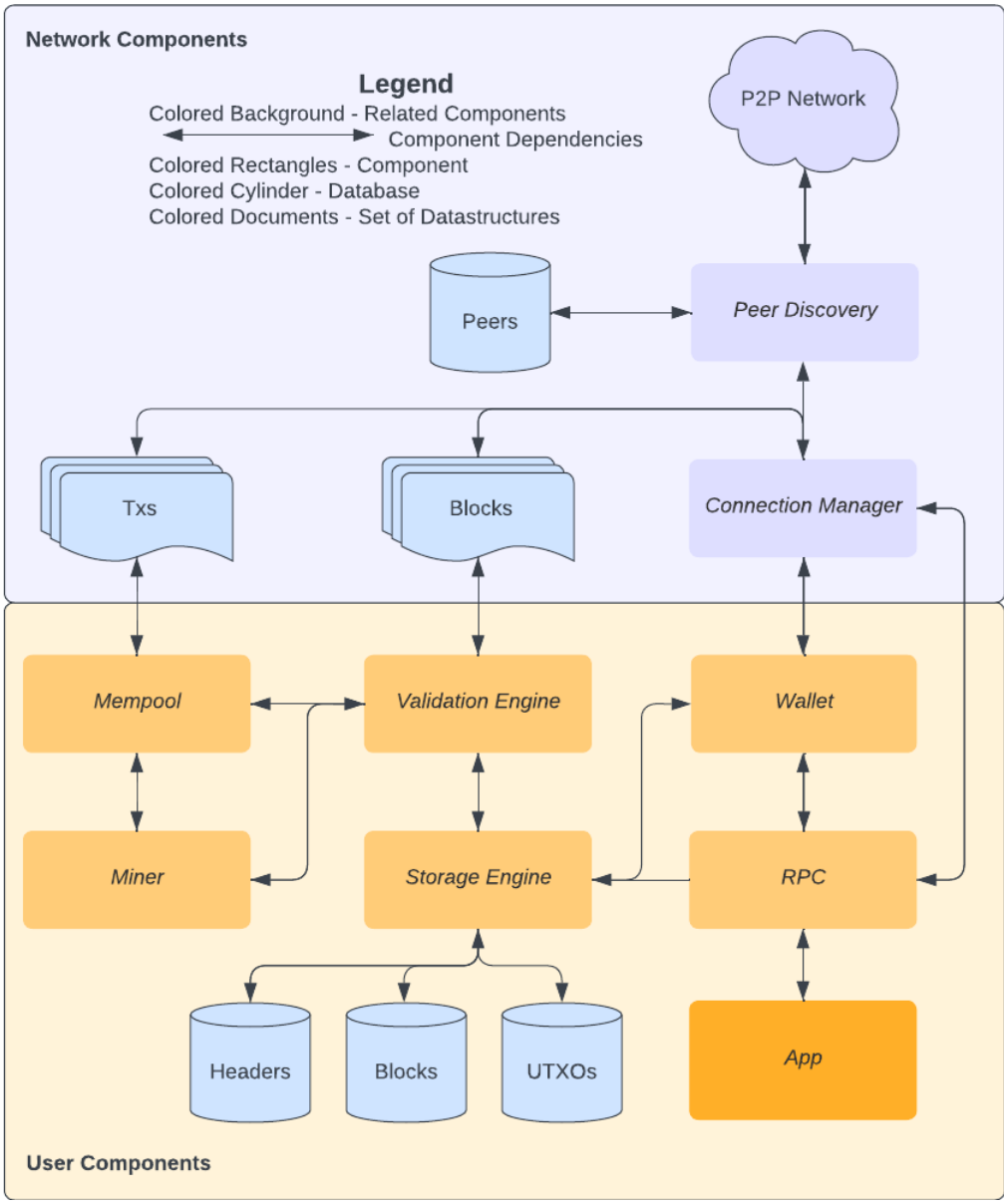
Bitcoin Core is a **decentralized** system that, unlike centralized systems, does not have a central point of power that controls the validation of transactions. Instead, it utilizes its **user nodes** to validate the transactions themselves utilizing a **peer-to-peer network**, where each individual full node in the network validates the blockchain and all its transactions using a **proof-of-work algorithm**. Bitcoin coins have no physical form and are therefore fully digital. Users have **keys** that prove their ownership over the coins. The keys are also used to sign transactions to transfer the coin to a new owner. As long as you own the key you are able to transfer that coin; this is the only requirement to spend coins. Keys are kept in the owner's **bitcoin wallet**. If the coins have no physical version and no central power "printing" more, how do new coins get created? This happens through a process called **mining** where miners compete with each other to find valid solutions to the proof-of-work algorithm that validates the current block while simultaneously minting new currency.

This report will explain the architecture of Bitcoin Core, beginning with an explanation of the various components and subsystems involved and their interaction with each other. Diagrams will be included throughout to illustrate the structure of the system and clarify particularly confusing ideas. Use cases will follow which illustrate common tasks that users perform and detail the component-level interactions involved using sequence diagrams. A data dictionary will be included to define key terms used across the report as well as naming conventions that will explain the various acronyms used in the report. This report should leave the user with a complete conceptual picture of how the bitcoin core system and its various components function, and how they are used when performing common user tasks.

Derivations:

To derive the Conceptual Architecture for Bitcoin Core our group consulted the GitHub book outlining major components of the software (Cypherpunks-Core), while also independently compiling references. Once we had identified a main set of components we grouped them and assigned each group member a specific grouping to work on. We tried to place a focus on common Bitcoin use cases and which components are involved, to determine which sections needed the most attention. We recognize that as an Open Source

Conceptual Architecture:



Peer-to-Peer Network

Bitcoin uses a Peer-to-Peer architectural network style which means that all computers that are found in the network are considered peers and are all equal to one another, with no one peer being more important or special than any other. There is no hierarchy, no server or central service in the network, only equal nodes. The nodes in the network will therefore receive and provide information across the network. Although all nodes are equal they can have different jobs inside the network. A node has four functionalities: mining, routing, blockchain database, and wallet services.

Wallets

As Bitcoin is meant to be a replacement for traditional fiat money, one of its core features is the wallet. The wallet allows users to store, send, and receive Bitcoins as well as issuing public and private keys to facilitate these transactions (Bitcoin Developer). A distinction should be made between wallet programs and wallet files. Wallet programs are programs that can be downloaded by users to allow them to send and receive bitcoins. Wallet files on the other hand at their core are simply just a file containing a collection of private keys (although these don't have to be digital) (Bitcoin Wallet).

Sending and receiving money are the two essential features that a wallet program may implement. A wallet program has to distribute public keys to peers to receive Bitcoins and needs to sign transactions using a key to send Bitcoins (Wallets). The wallet may need to connect to the P2P network to receive information from the blockchain or to broadcast new transactions (distributing keys and signing transactions don't interact with the P2P network). Thus the three main sub-components of a wallet program are the public key distributor, the signer, and a networked program (Bitcoin Wallet). Note that although many popular wallets implement all three features in a single program, some users may choose to use different components for the sub-components to increase security (keeping the components that sign and distribute to the P2P network separate from the private key file to prevent any breaches of security).

At the core of most wallet programs exists the wallet file, where the randomly generated private keys are stored. This process has no communication with the Bitcoin network and is usually done/stored in a secure environment. Most wallet programs start off with generating a random seed mnemonic phrase chosen from 2048 words that generates the user's private key (Bitcoin Briefly). This private key is then transformed using elliptic curve multiplication into a public key, which is then hashed to get a wallet address (Wallets) (O'Reilly). Note that there exist wallets (HD wallets specifically) that will generate a new private-public key pair from your seed for every transaction (Bitcoin Briefly). The public key distributor component will then use the network component to communicate with the P2P network to distribute the public keys for use in transactions. The network component also monitors for output sent to the public key/wallet address, for which it will then use the signing component to sign transactions, and then broadcast the transaction to the P2P network (Bitcoin Developer).

Blocks

Blocks are considered the base and foundation of the Bitcoin Core architecture. The structure of a block is split into four components: block size, block header, transaction counter, and transactions. A block header is a container of data which consists of three components. Firstly, it contains a reference to the previous block hash (parent block), which is generated using the SHA-256 hash algorithm. This allows blocks to be linked together creating a "blockchain". Furthermore, it consists of metadata needed to mine the block. It contains the difficulty target needed for the proof-of-work algorithm, the nonce and the creation time of the block. Lastly, it contains a merkle tree root which effectively encodes every transaction in the block.

Only blocks with proof of work are allowed to be in the blockchain and this occurs when miners collect transactions into a block, find the proof of work to validate the block and then send out the validated block.

Blockchain

The fundamental system for transaction validation and storage is the blockchain. It is implemented as a linked list that connects each new block to the blocks before it through backwards linking. The identification process uses SHA256 hashing, and the header of a block contains the hash of the previous block in the chain (Cypherpunks-Core).

A node adds a block to the head of the chain by looking for this "previous block hash" when the block is received from the network. The node will check if the hash is a known part of the blockchain, and if it is, the new block is added to the blockchain as a "child" block. Because of this, there are temporary situations where one block on the chain has two children (known as a "blockchain fork"). These situations are resolved when one child chain becomes longer than the other, as described later in this section.

The blockchain is integral in the integrity of Bitcoin as a currency, as the Bitcoin protocol must verify that no individual currency unit gets spent more than once. Rather than using a centralized mint, a feature that contradicts the **peer-to-peer architecture style**, Bitcoin implements a **proof-of-work** system, allowing for the features of a timestamp server to be integrated into a P2P architecture.

A timestamp server "works by taking a hash of a block of items to be timestamped and widely publishing the hash." (Nakamoto 2) The creation of the hash verifies the existence of the item at a given point in time, and this can be used to sequentially order transactions in blocks. The hash of each new element includes the hash of the previous element in order to form the blockchain.

The obvious downside of this approach is that a central server must exist with the capacity to broadcast these hashes. Thus, in order to incorporate this technology into a peer-to-peer architecture, some other system must be used. The solution used in Bitcoin is the proof-of-work system; in order to verify a transaction and add it to the blockchain, processing power must be expended to find a hash with a suitable number of leading zeros. This is done by incrementing a nonce, an additional value that modifies the block's hash. Once a CPU finds a suitable nonce to create a hash, other nodes in the peer-to-peer network are notified.

To run the network with proof-of-work incorporated, a general sequence of events must be followed. Nakamoto outlines this in his 2008 paper, stating that first new transactions must be broadcast to all nodes. After this broadcast, nodes collect transactions into blocks and find proof of work for them. This block is then broadcast to and verified by other nodes, which then express that verification by appending it to the chain (Nakamoto). In general, the longest branch of the blockchain is more "honest", as it represents a greater amount of CPU work, and ties regarding which branch to work on next are broken based on this.

Peers

Peers, or full nodes, validate transactions and blocks and accept transactions and blocks from other peers as well. They can then validate the accepted transactions and blocks and pass them to other peers in their network. This is what allows Bitcoin Core to create a decentralized service.

Peer Discovery

In order for full nodes to verify and relay transactions to other full nodes, the program needs to discover the IP addresses of these full nodes. For peers to discover each other, Bitcoin Core queries DNS seeds, which are hardcoded into the program. DNS seeds are managed by Bitcoin community members and full nodes are added to them through dynamic DNS seed servers or manually through static DNS seeds. Once connected to a network, peers can send address messages to other peers that are also connected, using their IP addresses and port numbers. Records of previously connected peers are kept in a Bitcoin Core database, allowing the known peers to connect on later startups without having to use the DNS seeds again. This method of peer discovery is fully decentralized.

Miner

Miners are responsible for adding new blocks to the blockchain as well as ensuring the security of the system in a decentralized manner. They do this by maintaining a local copy of the blockchain and listening for all new unconfirmed transactions broadcast by full nodes. Miners must construct a candidate block before they begin mining the new block to be added. Construction of a candidate block requires creating a block header, filling out the metadata for the header, and aggregating unconfirmed transactions into a Merkle tree whose root is hashed and also added to the header (Cypherpunks-Core).

Once the candidate block is created mining can commence. This requires finding a solution to the Proof-of-Work algorithm that makes the candidate block valid. The calculation of this solution involves repeatedly hashing the block header while changing single parameters until the result of the hash matches a target value. The hash function used for Bitcoin is SHA256, and the target value is related to the current block number (Cypherpunks-Core).

When a Miner finds a valid solution to the current block it is broadcast across the network, the solution is validated by other nodes, and it is added to each node's local copy of the blockchain. New Bitcoin is minted upon a successful solution and given to the Miner along with transaction fees which represent a small margin between inputs and outputs across all transactions. These incentives are only given to a single Miner who broadcasts the solution, However, it’s common for Miner Pools to split the reward for more consistent smaller payouts.

Miners play an extremely critical role in the overall operation of the network and its security in addition to being the means for new currency to enter circulation.

Connection Manager

To connect to another peer, a “version” message is sent to the peer containing the version number, block, and current time of the full node. The peer will then respond with its own message containing its own corresponding data. Until both “version: messages have been received, no other messages can be sent to the other peer. Once the peers both receive a “version” message, they each send a “verack” message, which acknowledges that the “version” messages were received and indicates that the connection has been made. When connected, “addr” messages can be sent to gather additional peers.

By default, connections are maintained by sending a message to peers after 30 minutes of inactivity. After 90 minutes of inactivity, if no messages were received, it is assumed that the connection is closed.

RPC

RPC is an acronym for Remote Procedure Call, which is a software communication protocol that allows a client program to call procedures from a host program while each is hosted on a different network. Once the procedure is sent by the client, the server executes the procedure and sends back a response. RPCs create a trouble-free method to send and receive information from the blockchain. The blockchain uses a JSON-RPC interface that creates access to API functions that allow interaction with the blockchains. Some examples of procedures are:

- creating a transaction
- getting the hash of a block
- getting mining-related information
- getting information about a wallet
- getting difficulty of proof of work

The server the client connects via RPCs is called an RPC node, these nodes run blockchain client software and let clients get access to the blockchain data and respond to the sent RPC procedures. RPC endpoints are locations in the network where RPC procedures are sent to access data on the blockchain. All RPC nodes have RPC endpoints (Alchemy).

There are three types of RPC endpoints. Public RPC Endpoints can be used by anyone at any time and they are free. shared and rate limited which makes it near impossible for applications to run on (Alchemy). Private RPC endpoints are used by Decentralized applications (Alchemy). Decentralized applications need information from the blockchain to exist, a private RPC endpoint lets this happen with extremely fast and reliable RPc procedure calls as there is a lack of crowding. Alternative RPC endpoints are also used by decentralized applications as a backup RPC endpoint in case the primary endpoint becomes available (Alchemy). This allows for a happy user experience as less downtime will occur.

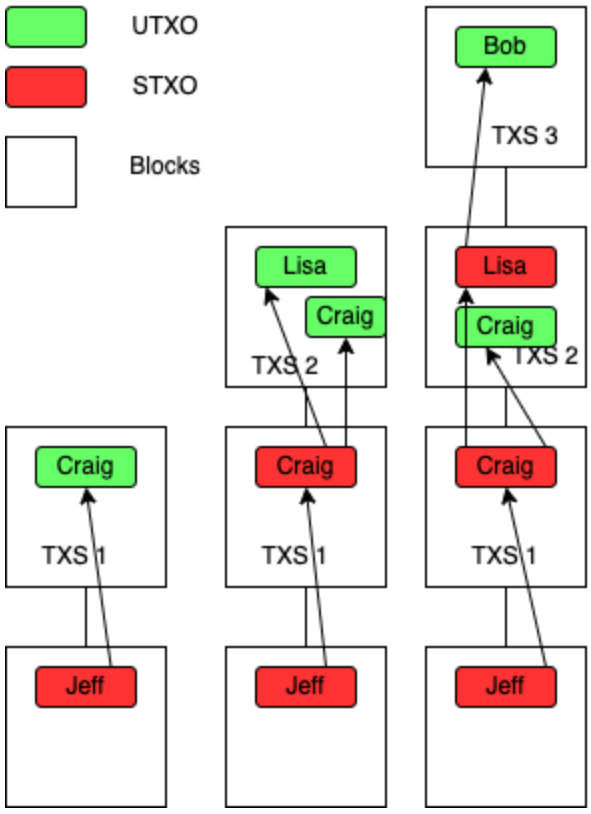
Transactions

Transactions (TXs) are one of if not the most important component of the Bitcoin Core system. All the other components are there to ensure transactions function properly as required by the system which means they need to be created, propagated on the system, validated, and added to the blockchain. Transactions are a data structure that encodes the transfer of value between Bitcoin users. Transactions do not directly contain the sender or receiver's addresses, nor coins or balances of accounts.

Transactions contain an output and in most situations an input. The output of a transaction is an invisible chunk of bitcoin on the blockchain which has been validated by the whole network. These are called UTXO or STXO (UTXO if they haven't been spent and STXO if they have) and they stay on the blockchain. The balance of a user in their wallet is the sum of all UTXO owned by that user on the blockchain. These UTXO are spread out over multiple different transactions and blocks on the chain. UTXO cannot be divided which means when used as input in a transaction they must be used completely. If that amount is larger than the required amount, the sender will receive change back as a new UTXO. Using dollars as an example, if a good costs you $2 but you only have a $5 bill you will give the $5 bill in its entirety to the cashier who will then give you back $3 in change.

The input of a transaction is the UTXO owned by the sender that will make up the value needed for the transaction. The input can either be one UTXO greater or equal to the amount required or multiple UTXO greater or equal to the amount. Once a transaction has been broadcasted on the network every validating node will retrieve the UTXO used as input to validate the transaction (Sinkevicius). As mentioned earlier not all transactions have input, these are called coinbase transactions. These transactions will always be the first transaction in a block. This transaction happens when a miner successfully mines new bitcoin coins. Therefore, it does not require a UTXO as input but has an input called coinbase.

Each transaction has a transaction fee. In transactions using a centralized system with a financial institution a transaction fee is paid to the central power (the bank) as a payment for using their services. However, in the Bitcoin system there is no central power so why is a transaction fee necessary? Well, transaction fees are paid to the miners to reward them for securing the network. They are also used as a security system to make it impractical for someone attacking the system to flood it with transactions. Fees are calculated by the wallet by using the size of the transaction in kb.



Here is a Diagram demonstrating TXs on the blockchain. The first transaction is from Jeff to Craig. Craig gets a UTXO and once he makes a transaction to Lisa that UTXO becomes a STXO and the output is a UTXO to Lisa and one to Craig. This demonstrates how the output from previous transactions are used as the input for new transactions as long as they haven't been spent previously.

Mempool

While the central idea of Bitcoin is adding transactions to the blockchain, transactions that are recently broadcasted do not immediately get added to the ledger. Instead, unconfirmed transactions get added to a waiting space, called the Mempool (QuickNode). When a transaction is broadcasted, it gets sent to peers over the Bitcoin network, which then validates the transaction (i.e. valid signatures, etc.) (Binance Academy). The transaction will then get added to a node's mempool. While the use of mempools are common, there is no singular mempool. Instead, each node in the network – a node being a computer that supports the Bitcoin network – has its own mempool. The current size of the mempool on a specific node gives an estimate to the amount of traffic on the network, with a bigger mempool signifying more traffic and thus implying higher transaction fees (Binance Academy). Transactions in the mempool are then taken – usually those with the highest fees first – and added to the ledger (Binance Academy). The mempool thus serves a vital function as a holding space for unconfirmed transactions and gives a view into network traffic, which can be useful in estimating transaction fees.

Validation Engine

The validation engine checks each block of any transaction it receives and validates it, determining whether it can trust the block without needing to trust the miner that created it. Bitcoin Core handles validation using proof-of-work, where cryptographic problems must be solved for each block to validate them before they are added to the blockchain. To validate the block, the SHA-256 hash of the previous block is taken and hashed to generate a name that is then used to solve the current problem. If the solution is within an accepted range of characters, then it is accepted as correct and the block is validated.

Storage Engine

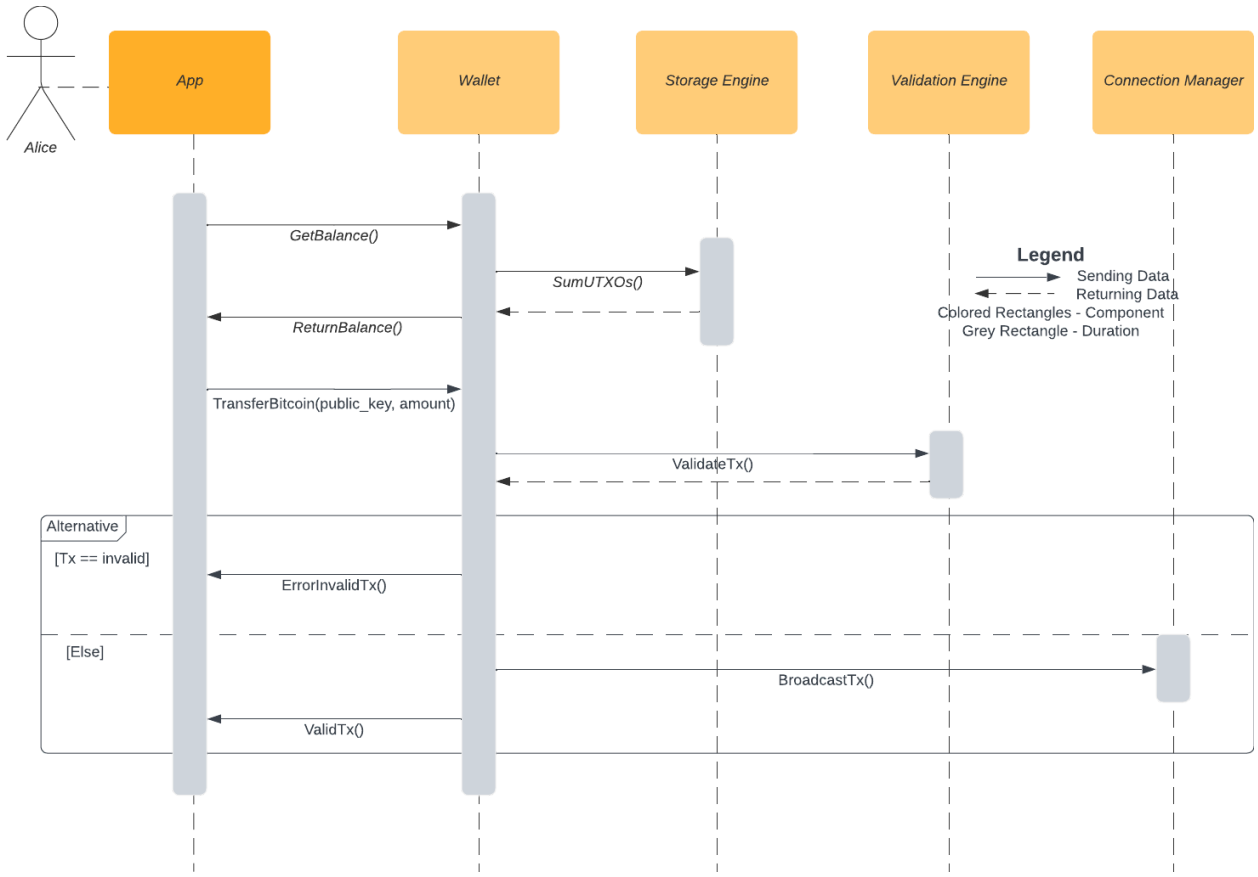
The storage engine provides an interface between a Bitcoin wallet and its contents. The contents are stored as a database of UTXOs; when a transaction occurs that involves spending bitcoin, the storage engine performs a search of the UTXO set to find a sufficient quantity of unspent Bitcoin and transfers that currency to the wallet in order to complete a transaction (Cypherpunks-Core).

The storage engine also stores the headers of the blocks that the user's CPU (representing a node in the P2P network) has encountered in the blockchain. The purpose of this is to support the validation engine to verify the identity of a new block being added to the chain. The validation engine passes the header of a new block to the storage engine; the storage engine then parses the header to find the header of the parent block and checks the parent's header in its database to see if the parent is a familiar block. If a valid match is found, the storage engine passes this information to the validation engine so the block may be appended to the chain. This is essential in the blockchain validation process that legitimizes Bitcoin.

Use Cases:

1. **Wallet Management**

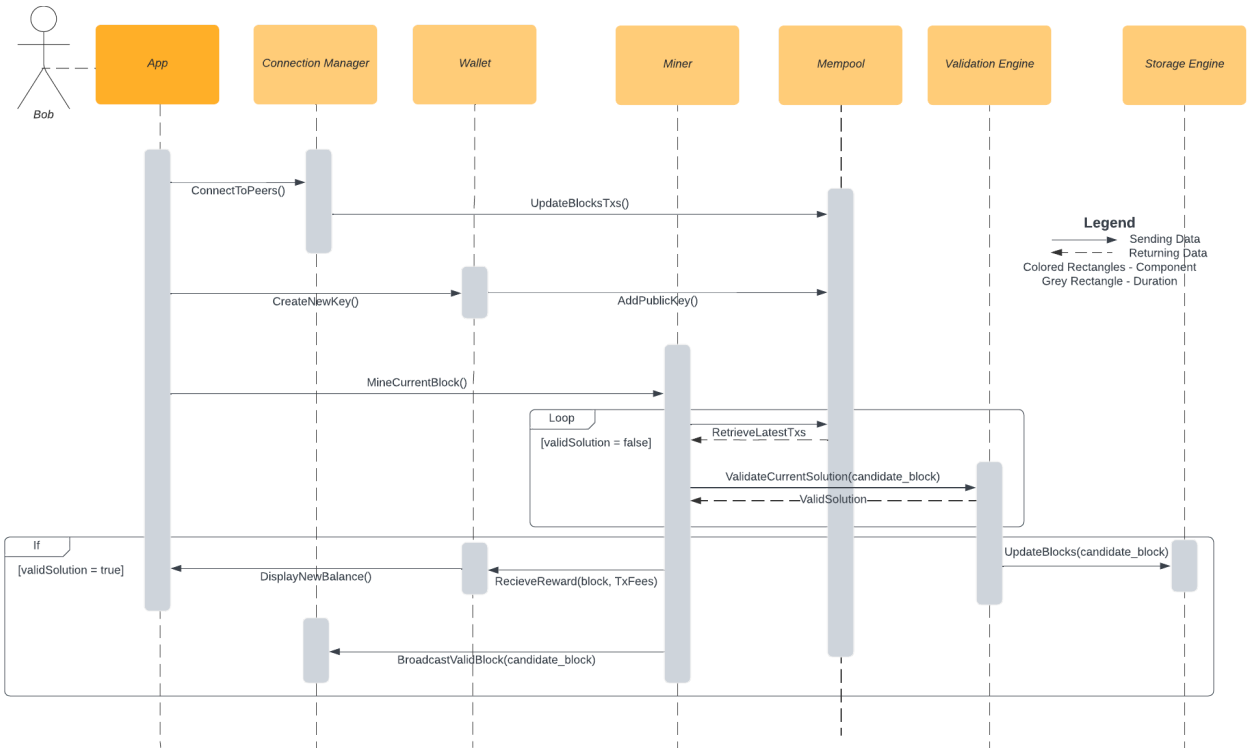
Alice is a Bitcoin user who has been using Bitcoin Core for her wallet software. She wants to check her wallet balance and then transfer some bitcoin to her friend Bob's account. Note that all calls from the app to components go through the RPC but for simplicity it was excluded.



2. Mining

Bob wants to mine bitcoin using the Bitcoin Core software, so he sets up a fresh installation by following online tutorials and connects to the Bitcoin network. He begins by creating a new key to receive bitcoin and then starts mining. When he successfully creates a new block, he is rewarded with bitcoin.

I excluded some components that would have further interactions with the Connection Manager as they would make the diagram too busy. I also excluded a larger loop and conditional that would handle the case of a new block being broadcast during the mining process. This diagram assumes that the current block is not solved and would require you to begin again if a new block was broadcast. The focus was on conveying the reward dependent on the candidate block being valid.



Data Dictionary:

Block - A block is a collection of bitcoin transactions that have been validated and recorded on the blockchain. Each block contains the transactions, a header with other metadata, and a reference to the previous block in the chain.

Block Header - Contains information about a given block and its transactions. Block Headers are hashed repeatedly during proof of work.

Blockchain - the decentralized, immutable, public ledger of all bitcoin transactions, stored in cryptographically linked blocks of transactions.

Decentralized applications - application that runs on a decentralized network instead of a single server

Genesis Block - The first block in the Bitcoin blockchain.

Merkle Tree - also known as binary hash trees, a Merkle Tree is a tree that serves as a summary of all the transactions in a block by providing hashes for different blocks of data at its leaf nodes.

Minting - Minting is the process of creating more Bitcoin during the addition of each new block.

Mining - Validation of pending transactions within the Bitcoin network.

Miner Pools - A network of distributed miners who split rewards for a successful block solution based on computational resources provided.

Nonce - a value that is set so the hash of a block will have a number of leading zeros

Proof of Work - A form of cryptographic proof used to show that a given amount of computational effort has been used to reach a solution. The proof is easily verifiable by

independent parties once it is presented. In the case of Bitcoin, Proof of Work means that a Miner has presented valid parameters to SHA256 that solve the current Block.
Full Node - A program that fully validates transactions and blocks. Almost all full nodes also support the network by accepting transactions and blocks from other full nodes, validating those transactions and blocks, and then relaying them to further full nodes.
DNS seed - A DNS server which returns IP addresses of full nodes on the Bitcoin network to assist in peer discovery.
SHA256 - The hash function used in Bitcoin Core's mining process. SHA256 always produces a 256-bit output regardless of input size. This can be thought of as a way to create unique fingerprints for every input.

Naming Conventions:

UTXO - unspent transaction output
STXO - Spent transaction output
TXs - Transactions
P2P - Peer-to-peer

Conclusions:

In conclusion, to obtain a deeper understanding of the Bitcoin Core system we analyzed each component to gain a better conceptual understanding of how decentralized P2P systems work. We ran through common use cases to determine which components depend on each other in order to construct our box and arrow diagram which provides a high-level overview of how the system works. We identified that the backbone of the whole system relies on the blockchain in conjunction with distributed Miners and gained a much deeper understanding of Bitcoin Core's structure as a whole.

Lessons Learned:

- Completing mockup diagrams based on preliminary readings would have gone a long way in setting a stronger direction for both, the sections included in this document, and the division of work. We relied on a diagram that didn't give the full picture initially and wasted some time being somewhat unfocused in our direction.
- As with any project starting earlier would have led to a much more cohesive document. We gave a few days for reading of material but most sections were both researched and written about at the same time, which can lead to a lack of knowledge of how different sections or component descriptions inter-relate.

References:

"4. Keys, Addresses, Wallets - Mastering Bitcoin [Book]." *Www.oreilly.com*, www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch04.html. Accessed February 19, 2023.

Binance Academy. (n.d.). *Mempool*. Binance Academy. academy.binance.com/en/glossary/mempool. Accessed February 19, 2023.

"Developer Guides — Bitcoin." *Developer.bitcoin.org*, developer.bitcoin.org/devguide/index.html#. Accessed 19 February 2023.

How to access Bitcoin Mempool. QuickNode. (n.d.). www.quicknode.com/guides/web3-sdks/how-to-access-bitcoin-mempool. Accessed February 19, 2023.

"Introduction · GitBook." *Cypherpunks-Core.github.io*, cypherpunks-core.github.io/bitcoinbook/. Accessed 19 Feb. 2023.

Nakamoto, Satoshi. "Bitcoin: A Peer-To-Peer Electronic Cash System." 31 Oct. 2008.

Sinkevicius, Arnas. "How Do Bitcoin Nodes Validate Transactions in Blockchain?" *Coinmonks*, 5 June 2022, medium.com/coinmonks/how-do-bitcoin-nodes-validate-transactions-in-blockchain-7ec0603a0140. Accessed 19 Feb. 2023.

"The Ultimate Guide to Bitcoin Wallets, Seeds, Private Keys, Public Keys, and Addresses." *Bitcoin Briefly*, 20 Jan. 2021,

bitcoinbriefly.com/ultimate-guide-to-bitcoin-wallets-seeds-private-keys-public-keys-and-addresses/. Accessed 19 Feb. 2023.

"Wallets — Bitcoin." *Developer.bitcoin.org*, developer.bitcoin.org/devguide/wallets.html. Accessed February 18, 2023.

"What Is an RPC Node?" *Www.alchemy.com*, www.alchemy.com/overviews/rpc-node. Accessed 19 Feb. 2023.