The Conceptual Architecture of Bitcoin Core

Authors

Devin Pereira, Kanchan Shrestha, Alex Susilo, Adlai Bridson-Boyczuk, Cyrus Fung, Isaiah Wuthrich

Abstract

This report discusses the conceptual architecture of Bitcoin Core, which is the reference implementation of the Bitcoin network, a cryptocurrency that allows for trustless transactions. It includes an architectural analysis of Bitcoin Core, a detailed breakdown of Bitcoin Core's components and their interactions, the evolution of the Bitcoin Core system, several use cases that illustrate our proposed conceptual architecture, discussions on concurrency and developer division of responsibilities, and finally limitations and lessons learned from our analysis. We propose that Bitcoin Core can be broken down into 11 main components: Transactions, Block/Blockchain, Validation Engine, Storage Engine, Storage Objects, Wallet, Miner, Mempool, Connection Manager, and Peer Nodes. We observe that these components are connected to each other in a layered manner combined with a peer-to-peer architecture and provide justifications for why our components fit within this view.

Introduction

Since the release of Satoshi Nakamoto's whitepaper in 2008, Bitcoin has gone from relative obscurity to a highly liquid asset with a total worth of billions of dollars. In 2013, the price of a single Bitcoin was $77; by 2021, that price had grown to $67617, with around 300,000 confirmed transactions on the blockchain per day. Although Bitcoin's validity as a medium of exchange, environmental impact and social impact remain controversial, its influence cannot be understated. Its novelty and appeal stem from its ability to provide a decentralized, transparent, relatively anonymous and secure way to transfer value digitally. This is made possible through the careful design and integration of a wide range of structures, making software architecture essential to understanding how and why Bitcoin works the way that it does. The principles and structures guiding Bitcoin's continuous evolution are also critical to its functionality. Its nature as both a currency and an open-source, decentralized system means that it must remain stable over long periods, while also being adaptable to changes in its regulatory, development and production environment.

In this report, our team explores the conceptual architecture of Bitcoin Core, which is the reference implementation of the Bitcoin protocol. Documentation regarding Bitcoin Core's internal architecture is readily available on the internet, as it is an open-source cryptocurrency. After a thorough analysis, we concluded that Bitcoin Core follows a layered architecture, with each individual node connected to each other in a peer-to-peer fashion. In our analysis, Bitcoin Core is broken down into 11 main components: Transactions, Blocks/Blockchain, Validation Engine, Storage Engine, Stored Objects, Wallet, Miner, Mempool, Connection Manager, Peer Nodes. Each component is then analyzed in terms of its function and interaction with other components, along with justifications on how it fits within our proposed conceptual architecture. To further illustrate how our analysis of Bitcoin Core is appropriate, two use cases are presented: one of a bitcoin transaction and the other of a bitcoin miner. The evolution of the Bitcoin Core system, concurrency, division of responsibilities amongst developers, and external interfaces are also discussed.

Derivation Process

Our derivation process had us creating an organized plan consisting of different stages to help derive the conceptual architecture. We decided on having four stages: first, we would search through the internet for resources related to Bitcoin Core, second, analyze them to find relevant information, third, came up with a draft architecture together, and finally collaborated to then piece our different findings together into one finalized architecture.

When we were in the first stage, we ran into a few websites that were abundant with detailed information. For instance, one of these sources provided a high-level look at the architecture of Bitcoin Core and every component in detail (Cyperphunk's GitBook). As a result, our second stage was very straightforward since most of the necessary information could be easily found with these well-documented sources. This gave us a good basis for us to build our understanding of the high-level architecture with its components and relationships into a draft architecture in the third stage.

For our initial draft, we collectively agreed on the components, relationships, and using a Peer-to-Peer architectural style, but we could not agree on if it should be only P2P or be combined with other styles. A few of us thought of combining it with a Publish-Subscribe style because of components like mining that did incorporate elements of that style, specifically with its communication. A miner needs to listen and communicate with other nodes for it to be working. However, when considering the other aspects of pub-sub, we found mining did not match them and so we dropped the idea.

After some discussion, we decided on incorporating a layered style. This made more sense to us because we realized many of the components belonged to specific functions of Bitcoin Core and thus could be separated into layers. In the end, after finishing our architecture design, we concluded that the overall architectural style being used with Bitcoin Core is Peer-to-Peer style combined with Layered style. The layers can be divided into the application layer, network layer, protocol layer and database layer. The modules present in our architecture are

Transactions (referred to as Tx), Blocks & Blockchain, Miners, Mempool, Validation Engine, Storage Engine, Headers, Blocks, Coins, Wallet, Peer Discovery, Peer Nodes, and Connection Manager.

Architecture

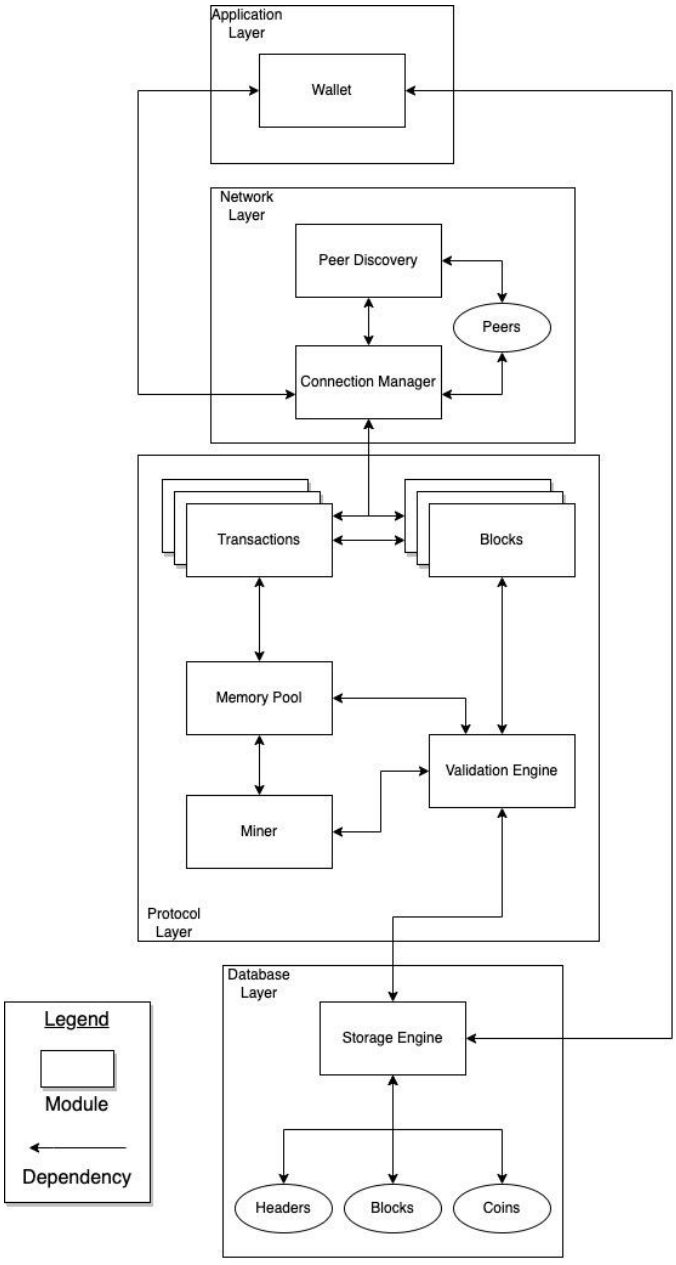

Figure 1. The Peer-to-Peer + Layered architecture of Bitcoin Core

Transactions

A transaction in Bitcoin represents a transfer of a certain amount of Bitcoin from one address to another and is part of the protocol layer of Bitcoin Core's layered architecture. Each transaction may contain multiple inputs and outputs, where inputs represent debits against an address and outputs represent credits to an address. The difference between the inputs and outputs is the fee associated with that transaction. The transaction also contains proof of ownership for each of the bitcoins in the inputs, meaning that the input for the current transaction is reliant on the outputs of prior transactions.

The core structure of a transaction is based on the unspent transaction output (UTXO), which is a discrete amount of BTC that an address has received but not yet spent. Inputs to transactions consume existing UTXOs, and outputs from transactions create new UTXOs that are recognized by the whole bitcoin network and are available for the owner to spend in future transactions. UTXOs are combined and split to satisfy transaction amounts. Consider a case where Alice has received two transactions of 0.7 BTC and 0.5 BTC, creating two UTXOs with the respective values, and wants to send 1 BTC to Bob. The transaction can then take those two UTXOs as inputs with a combined value of 1.2 BTC, with an output of 0.1999 BTC returned to Alice as "change" minus the transaction fee.

When a user initiates a transaction, it is broadcast to a node and sits in the node's mempool awaiting validation. The node validates the transaction and passes it on to its neighboring nodes if it is valid, synchronously returning either a rejection or acceptance message to the initiator of the transaction. This process continues until the transaction has propagated through the entire Bitcoin peer-to-peer network, making it possible for the vast majority of participants in the Bitcoin network to have the same information about the system state. Once a mining node has validated the pending transaction using its validation engine, it becomes part of a block and is added to the blockchain, thereby confirming it and ensuring that everyone in the Bitcoin network agrees that the specified amount of Bitcoin transacted has been removed from one address and added to another.

Blocks & Blockchain

Transactions are aggregated into data structures known as blocks, which are composed of more than 1900 transactions on average. The header of a block contains a reference to the hash of the previous block, mining information, and a cryptographic summary of all its constituent transactions using a Merkle tree. Due to the avalanche effect exhibited by hash functions, where a single change in the input data creates a totally different hash, the current block's hash is cryptographically linked to the previous block's hash. Changing the hash of a single block would require one to recalculate the hashes of all its ancestors, so the vast amount of computing power that would be required to recalculate all the blocks makes the history of Bitcoin functionally immutable. Each block is therefore able to verify its predecessor all the way back to the first block ever minted (the "genesis block").

The structure created through the sequential linking of blocks is referred to as the *blockchain.* The blockchain forms the network layer of Bitcoin Core's layered architecture, with

a copy stored in every node. Blocks are added to the blockchain by miners, who verify the transactions and receive newly minted Bitcoin and transaction fees as an incentive. The blockchain forms the network layer of Bitcoin Core's layered architecture.

Validation & Storage Engines

The validation engine is responsible for certifying that transactions and blocks are valid, as well as complying with Bitcoin's protocol. Initially new transactions are verified prior to being added to a block. Each transaction must verify that the inputs are valid, the transaction meets the minimum fee requirement and that it follows the rules of Bitcoin protocol. It then verifies that each new block created has valid proof-of-work and once again adheres to the Bitcoin consensus rules. If the block is successfully verified, it is added to the node's local copy of the blockchain.

The storage engine's main responsibility is maintaining a copy of the blockchain and keeping it up to date with new transactions and blocks that are added. To do this it must store the blockchain on the disk, and concurrently update it with transactions and blocks as it receives them from other nodes on the network. Due to the nature of a blockchain growing large over time, the storage engine is efficient and scalable. To meet this requirement Bitcoin Core uses LevelDB as the database due to its high read and write performance and ability to handle large datasets. The storage engine is also responsible for maintaining other data structures used by Bitcoin core. For example, the UTXO database which tracks all the unspent transaction outputs in the blockchain. Together, the validation and storage engines work to guarantee the integrity of the Bitcoin network.

Wallet

A wallet is an application that serves as the primary user interface. It controls access to a user's money, tracks balance, manages keys and addresses, and allows users to create and sign transactions. However, it is important to know that a bitcoin wallet does not contain bitcoin, and only contains keys. Therefore, a bitcoin wallet is more like a keychain consisting of private/public keys. Users can sign transactions with the keys, and thus prove they own the output of their transactions (their coins). Coins are stored on the blockchain as transaction outputs.

There are two types of wallets, depending on whether a wallet's keys are related to each other or not. The first is known as a "nondeterministic" wallet where each key is independently generated from a random number and is not related to another key. This has the disadvantage of requiring the wallet to be backed up often due to copies being required for all random keys present, otherwise, the funds they control can be permanently lost. However, the result is that address reuse occurs, reducing privacy by associating several transactions and addresses with each other. Due to these reasons, nondeterministic wallets are being phased out in favour of "deterministic" wallets, the second type.

A deterministic wallet, also known as a “seeded” wallet, contains private keys that all stem from a one master key known as the “seed”. The seed is a randomly generated number combined with other data to create the private keys. This data allows the seed to be sufficient to recover all keys of the wallet, and so only a single backup at the time the wallet is created is necessary. The seed also allows for easy migration of the user keys between different wallet implementations. There exist further derivatives of deterministic wallets, such as “hierarchical deterministic” wallets that use a tree structure to contain the keys, with the seed at the root.

Miners & the Mempool

All full-node clients are constantly mining as it is the process of adding blocks to the blockchain and creating new bitcoins. In this process, miners receive recent transactions which they then verify by ensuring that the transaction satisfies a list of criteria. If the transaction is deemed valid, then it is added to the mempool, which contains all transactions that have been validated by the miner but not yet confirmed by other nodes.

While validating transactions, all miners are also attempting to solve a complex problem based on a cryptographic hash algorithm. This problem is called the Proof-of-Work and is solved by a miner every 10 minutes on average. The miner that solves the problem gains the right to create and transmit the newest block in the blockchain. This block will consist of all the verified transactions in the mempool as well as the Proof-of-Work. Once the new block is sent to the other nodes, a new block begins to be mined. This new block is called the candidate block as it is missing a solution to the Proof-of-Work. When the other miners receive the new block, they will compare the block with their mempool and remove transactions that were included in the new block and will start to work on their own candidate block.

As a reward for their work, the first person to mine the block receives the transaction fees from the transactions in the block and a sum of bitcoin. The amount of bitcoin rewarded started at 50 bitcoin per block and it diminishes by 50% every 210,000 blocks or approximately 4 years, resulting in the current rate of 6.25 bitcoin per block. This means that after the year 2140, mining will reward no more bitcoin and will only supply the transaction fees.

Peer Discovery & Connection Manager

With Bitcoin Core following a P2P architectural style, one of the most critical components of the network’s functionality is peer discovery. With the software’s decentralized style, nodes need to be able to discover and connect with others in order to participate in the network. In order to achieve this, Bitcoin Core uses a three-pronged combination of methods: DNS seeds, hardcoded IP addresses, and PEX. To start, DNS seeds are a set of domain names that serve as an initial list of nodes that can be contacted to bootstrap the P2P network. Next, Hardcoded IP addresses are included to be used as a fallback in case the DNS seeds are unavailable. Finally, PEX allows nodes to share information about their connected peers with other nodes in the network, increasing the number of available peers and resiliency to failure.

This approach to peer discovery ensures that Bitcoin Core can always find new nodes to connect to, which maintains the integrity and availability of the Bitcoin Network as a whole.

While the ability to discover and connect with peers is important, without something to manage these various connections, what is left is nothing but an intertwined mess. Hence, the Connection Manager is designed to handle numerous connections with different peers, with the ability to optimize the number of connections based on available resources. The manager also plays an important role in security. It is able to enforce rules for incoming and outgoing connections using methods such as filtering incoming connections to only allow connections from whitelisted IP addresses or limiting the number of outgoing connections to prevent denial-of-service attacks. Another critical feature of the Connection Manager is its ability to handle network partitioning, determining which side of the split to connect to and detect when the network has been rejoined. All-in-all the Connection Manager in Bitcoin Core is an essential component of the network, enabling the establishment and maintenance of connections with other nodes, optimizing network resources, and providing critical security.

Peers

Peers are represented as nodes in the bitcoin network and are equal but can have different roles depending on the functionality they support. A bitcoin node consists of up to four functions: routing, the blockchain database, mining, and wallet services. Bitcoin Core client nodes feature all four and are classified as full nodes. This type of node maintains a complete and up-to-date copy of the blockchain, enabling them to autonomously and authoritatively verify any transaction without external reference. Some nodes may contain a subset of the blockchain and verify transactions through simplified payment verification (SPV). These are known as SPV or lightweight nodes. There are also mining nodes, which can be full nodes. User wallets may be a part of a full node, or with an SPV node. Furthermore, there are servers and protocol nodes.

Evolution

On the public website, safe releases of Bitcoin Core are available for download by prospective peers. Users are cautioned against directly cloning the repository as they may install an unstable version. All safe versions of the software are made available for users of any popular operating system. Although there is no official documentation for Bitcoin Core, informative tutorials are provided for the stakeholders.

Updates to the Bitcoin protocol, which Bitcoin Core must invariably adopt, are documented in a Bitcoin Improvement Proposal (BIP). If a BIP is widely accepted by the community, the BIP editor will add it as a draft to the project’s repository. For the BIP to be considered accepted, a reference implementation of the changes must be produced.

The backwards compatibility section of a BIP is noteworthy. The initiator of the BIP (called champion) must discuss their approach to addressing backwards incompatibilities should there be any. The effect of implementing a feature is usually described in terms of the impact it has upon users of the protocol or miners. BIPs which impose backwards incompatible changes are called hard forks. In the history of the Bitcoin protocol, no BIPs involving hard forks have been accepted. Backwards compatible soft forks are preferred.

The developers came up with a workaround for making significant alterations to the protocol without employing hard forks. Modifications are implemented as additional features which become the norm over time. For example, the introduction of the Segregated Witness (SegWit) feature aimed to reduce the size of each transaction, effectively increasing the number of transactions possible across the network. Without mandating the upgrade to SegWit, the developers made it possible to send transactions to SegWit addresses.

The disinclination towards hard forks is justifiable. There is a significant risk for the Bitcoin network. Many peers simultaneously becoming inactive due to incompatible software poses a security threat. When a miner proposes to add a new block to the blockchain, the network of peers must reach a consensus to accept the block. If a malicious entity establishes control of just over 50% of the network's computational power because the size of the network is reduced, it is possible for them to exploit the blockchain. For Bitcoin this means that they could lower the difficulty to mine blocks and obtain large financial rewards almost instantaneously. Otherwise, the 51% attack is merely a theoretical consideration.

Concurrency

There can be over 10 thousand nodes connected at any time in the Bitcoin network. A high degree of concurrency is required to maintain ease of use and system security, which Bitcoin Core satisfies by utilizing a multi-threaded architecture.

Once a transaction is made by the payer, it is important that the payee receives the incoming transaction, and it is deemed irreversible within a reasonable timeframe. If transactions take too long to transmit, the system will be cumbersome to use. On the other hand, if new blocks are not formed and transmitted efficiently, the integrity of the transactions are at risk as a transaction is only considered irrevocable after six confirmations. Bitcoin Core aims to solve both issues simultaneously through multithreading. The constant receiving, validating and then sending of recent transactions is necessary to transmit transactions as fast as possible. Simultaneously, the mining component is about racing to solve the Proof-of-Work problem and add a block to the blockchain. The other nodes will then confirm the new block, increasing the security of the transaction and the blockchain as a whole. The system never pauses mining nor transmitting transactions as both tasks are essential to the protocol.

Bitcoin Core consists of many subsystems that work together and as such, the synchronization and concurrency can be analyzed furthermore. The system is constantly receiving and transmitting data, so the peer discovery system is always searching for new nodes

to connect to. The validation engine relies on the connection manager to supply incoming transactions and then begins the process of validating said transactions. If the transaction is deemed valid, it is sent back to the connection manager and transmitted to the connected nodes. Since new transactions are constantly being made, it is essential the connection manager and validation engine work together to efficiently validate and transmit valid transactions while also storing them in the mempool. At the same time, the miner is attempting to find a solution to the Proof-of-Work so that once the solution is found, the transactions in the mempool along with the Proof-of-Work solution can be sent out to other nodes in the form of a new block. The validation and mining processes are equally important and thus the two subsystems work concurrently. If another miner creates the block first, the storage engine relies on the validation engine to first confirm the block and then stores the block on the system. The miner then moves on to forming the next block. Additionally, the users can send and receive bitcoin at any point using the UI of the Bitcoin Core app. When sending bitcoin, a transaction is made and then shared to other nodes using the connection manager and when receiving bitcoin, the connection manager will send the transaction to the validation engine to begin validating, while also checking the keys in the user's wallet to see if they are the payee.

Division of Responsibilities for Developers

Based on the conceptual architecture of Bitcoin Core, it can be inferred that the developers responsible for the implementation of the system were not necessarily divided into specific teams. As established earlier, the overall software architecture of the system follows a P2P architectural style, where each node in the network acts as both a client and server. This decentralized nature allows for a more loosely organized form of collaboration, enabling informal conventions to ensure efficient and effective deployment.

In terms of the division of work and responsibilities, participating developers have various “focuses” on the Bitcoin project. This division of focuses is essential to the project, ensuring the security, stability, and continued development of the Bitcoin network. Though the focuses developers have can vary, the range of focuses includes software development, testing and quality assurance, or even community management. However, it is worth noting that these responsibilities may not be exclusive to a single focus as there may be an overlap. Furthermore, developers may participate in multiple areas of the project depending on their skills and interests.

In addition to the various focuses, work is also divided based on the priority hierarchy of modules. Essential modules, found at the top of the hierarchy, would be developed first and foremost before other modules that could be found lower in the hierarchy, being considered less important and put into the backlog for the future. For example, the “consensus rules” are considered to be the one of the most essential parts of the Bitcoin Core system as the module determines the rules by which the network operates. Thus, developers who have this module as their particular focus are given a high priority in terms of development. Similarly, developers who focus on the network layer may be given similar treatment, as they ensure that the network is secure, efficient, and reliable.

Despite developers possessing certain focuses on the project, that does not mean their knowledge should be limited to said focuses alone. All developers should have a fundamental understanding of the system, recognizing the architecture and how all the modules are used and interact with each other. For instance, all developers should have a general understanding of the Bitcoin protocol. This protocol is standardized and enforced throughout the system, ensuring a secure and high-performing solution for managing transactions and issuing bitcoins in a decentralized manner. Moreover, developers should also follow the basics of adhering to the established coding standards and provide clear and concise documentation for every contribution they make. This in combination with a sound understanding of the system's architecture ensures that their contributions integrate seamlessly with the rest of the system.

External Interfaces

Bitcoin Core offers both a command line interface and a graphical user interface, the latter of which can built optionally by nodes. As expected, the GUI provides less features than accessible via the CLI API. However, it is a simple alternative to performing certain tasks which would otherwise require entering multiple commands in the CLI. There are a variety of authenticated API calls a user can make to obtain metrics related to the blockchain, mining process, network activity, their node performance, etc.

The Bitcoin Core software is also capable of supporting multiple lightweight wallets at the same time, each with similar security and privacy to its built-in wallet dependent on the rigor of their implementation. This functionality of enabling the operations of these external wallets, sometimes called partial nodes, is crucial to the widespread use of the Bitcoin network. The general public is not in possession of the level of technical expertise necessary to install and maintain a full node (or peer). This feature is crucial for individuals seeking only to transact.

Another interface made available is the unauthenticated REST interface. It allows developers to produce their own derivative applications using public data obtained from Bitcoin Core and the Bitcoin network. Data related to transactions, blocks, the memory pool, and many other functional units are made available at different endpoints.

Use Cases / Diagrams

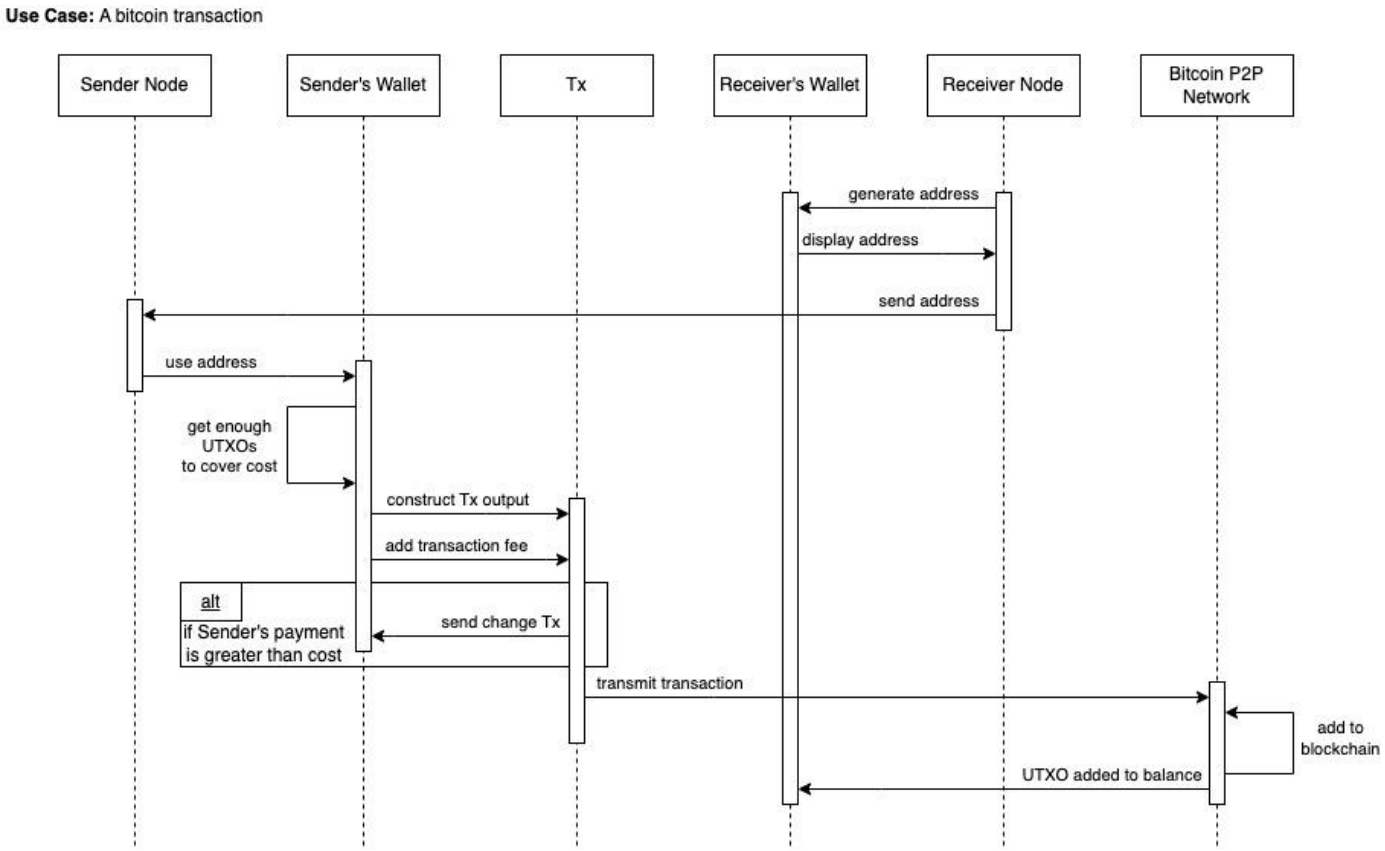

Figure 2. The use case of a bitcoin transaction

The first use case is when bitcoin is used in a transaction. This would be one of the most common use cases. To start with, the Receiver must first generate the address, a private/public key pair, before the sender can create the transaction. The private key data is copied and turned into a pubkey (public key) by the wallet. This key is cryptographically hashed for security purposes, then is sent from the Receiver to the Sender, encoded as a Bitcoin address (there are multiple ways for it to be transmitted and it can also be encoded further, such as into a QR code containing a bitcoin URL). After the Sender obtains the address and decodes it back into a normal hash, they can start to create the transaction. However, the Sender's wallet must first check to see if it has sufficient funds. It will find the smallest number of UTXOs necessary to cover the transaction cost. Once it has found its UTXOs, it will create a standard P2PKH transaction output, containing instructions allowing any user with proof of having the private key associated with the receiver's hashed pubkey to spend the output. Along with this, the wallet will add a transaction fee (determined differently depending on the wallet application), so miner nodes are incentivized to look at the transaction. If the Sender's gathered UTXOs are higher than the transaction cost, a second transaction output is derived from the original transaction containing the change and is sent to the Sender's wallet. Finally, the transaction is transmitted to the Bitcoin P2P Network, where it will be added to the blockchain, allowing the Receiver to spend their bitcoin.

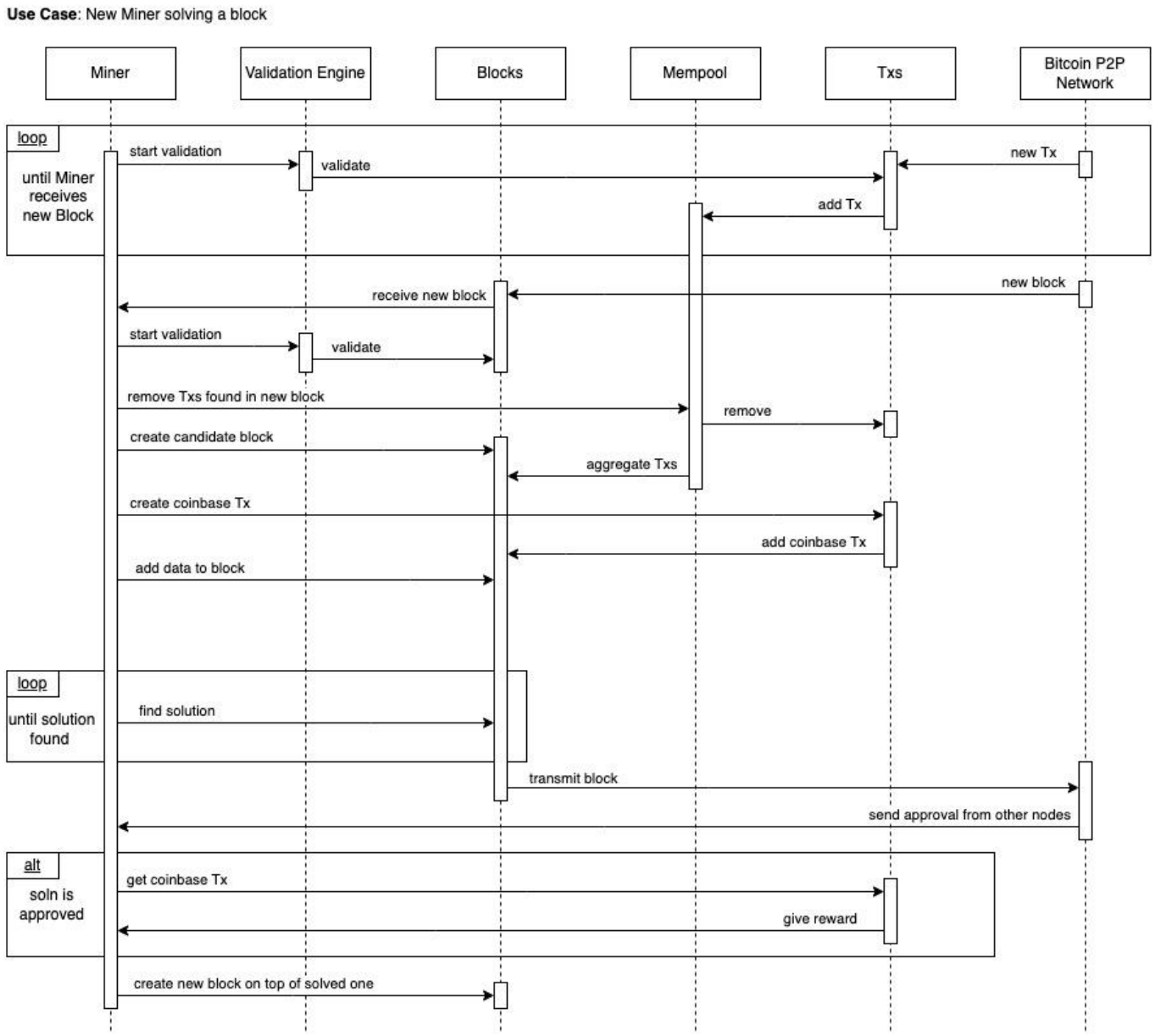

Figure 3. Use case of a new miner solving a block

In this scenario, a new miner joins the bitcoin network and solves a block. When it joins, it listens to the network, looking for unconfirmed transactions. It validates and places these into the memory pool. The arrival of a new block signals the node to stop working on the current block because another node has finished solving it. It will validate this new block, and after verifying, it removes the transactions from the memory pool that are found in the block. Now, the miner will start working on a new block. This process begins with the creation of a *candidate* block. The remaining transactions from the memory pool are added in, along with the *coinbase* transaction, containing the reward for the miner. The reward is calculated by summing the fees of all transactions in the block with the reward of the block height. Finally, the block is given data to its header, containing data such as the timestamp and the hash of the previous block of the chain. Now, the miner will begin attempting to find the Proof-of-Work algorithm and solve

the block. If the miner finds the solution, then it will transmit the block to the Bitcoin P2P Network to be validated by its peers, and if validated, the miner's reward is given to be used. Otherwise, the miner will lose the reward, and will have wasted electricity for nothing. In either case, after it is done, it will create a new block on top of the solved one, extending the blockchain, and continue mining.

Limitations & Lessons Learned

Researching the conceptual architecture of Bitcoin Core, there were a few limitations that the team faced. One of the most prevalent limitations was the fact that Bitcoin Core has no official documentation. Though general information was available, sources were scattered throughout the internet, some being more reliable than others. Thus, the team scoured through countless websites, filtered out the ones deemed "unreliable", and combined the findings to create a conceptual architecture report. However, seeing as none of the sources are official documents, it is difficult to say that the findings are completely reflective of the software's true conceptual architecture. Furthermore, it is even less likely that this conceptual architecture aligns with the concrete architecture due to how complex the system is. A more specific limitation the team faced was the ability to test the use cases and sequence diagrams. In order to validate these, one of the members had to have either a wallet and bitcoin or a mining rig, which none of them did. Thus, the team was forced to sample multiple examples online and infer the information. However, limitations are not the only things worth noting.

Despite the limitations stumbled upon, the team also encountered many positive lessons. The first lesson learned was the importance of working backwards from the deadline. In order to be successful, distributing the workload and having open communication was key. Scheduling bi-weekly meetings proved most effective, not only for monitoring progress, but also for collaboration in terms of sharing research, ideas, and determining the next steps. The work was cut up into stages, initially dividing the different questions amongst members to research and familiarize themselves before splitting up and working independently on their respective sections. In addition, the team was also able to learn about the broader context in which Bitcoin Core operates. Not only is Bitcoin Core just a piece of technology, but also a social and economic phenomenon that is impacting industries and communities all over the world. Moving forward, the team will continue to research and take detailed notes on the broader context of Bitcoin Core, including regulatory social impacts, economic impacts, and the potential for future developments and applications.

Data Dictionary

Transaction – A structure representing a transfer of value from one Bitcoin address to another

Coinbase Transactions – The first transaction of every new block will contain this. It holds the reward for the miner should they solve the block. The reward total is calculated by combining the transaction fees of all transactions in the block together with the reward of the current block height. The block height reward starts off at 50 bitcoin and is halved every 210,000 blocks. Currently, it is at 6.25 bitcoin.

Block – A structure that contains around 1900 transactions and the information necessary for miners

UTXO – An output of a Bitcoin transaction that represents a discrete amount of Bitcoin credited to an address

Blockchain – New blocks reference a “parent” block and thus act as a chain.

Component – A collection of related procedures

Node – A participant in the Bitcoin P2P network, which could be an instance of Bitcoin Core

Peer-to-peer style – An architectural style consisting of equally privileged, independent nodes that can all connect directly to each other

Layered style – An architectural style where components are organized into hierarchical layers, each providing a service to the layer above and serving as a client to the underlying layer

Naming Conventions

P2P – Peer-to-peer

TX - Transaction

UI – User Interface

DNS Seeds – Domain Name System Seeds

PEX – Peer Exchange

SPV – Simplified Payment Verification

UTXO – Unspent Transaction Output

P2PKH – Pay-to-Public-Key-Hash

Conclusions

In conclusion, we were able to analyze the conceptual architecture and determine and break down Bitcoin core into 11 main components organized into a layered architecture with a peer-to-peer network of nodes. Throughout this report, we were able to document the purpose, functionality and interactions of these components. This included how they interact with users using the Bitcoin Core software, other components within the system, as well as the developers modifying the system itself. Furthermore, we discussed the relevance of concurrency and where it is seen in a P2P network such as this. Finally, we researched the methods of evolution seen by the system and the division of responsibilities among Bitcoin Core developers.

Bitcoin and its systems have been revolutionary in the world of blockchain technology and decentralized cryptocurrencies. Being open-source and available for all developers to explore and possibly innovate upon reinforces the importance and influence of this software.

Moving forward we are looking forward to expanding on the conceptual architecture and continuing to delve deeper into specifics of how these components function in the succeeding report focused on Bitcoin Core's concrete architecture.

References (MLA)

Asplund, Mikael, et al. "In-Store Payments Using Bitcoin." *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2018, https://doi.org/10.1109/ntms.2018.8328738.

"Chapter 10: 'Mining and Consensus'." *Chapter 10: 'Mining and Consensus' · GitBook*, https://cypherpunks-core.github.io/bitcoinbook/ch10.html.

"Transactions." *Bitcoin*, https://developer.bitcoin.org/devguide/transactions.html.

"What Are Bitcoin Blockchain Nodes? - Bitstamp Learn Center." *Bitstamp*, https://www.bitstamp.net/learn/crypto-101/what-are-bitcoin-blockchain-nodes/.

"Chapter 6: 'Transactions'." *Chapter 6: 'Transactions' · GitBook*, https://cypherpunks-core.github.io/bitcoinbook/ch06.html.