Conceptual Architecture Analysis of Bitcoin Core

CISC 322
Friday, February 17th 2023

Group 39: Yash
Mercy Doan (mercy.doan@queensu.ca)
Yash Patel (19yp29@queensu.ca)
Cain Susko (20cjbs@queensu.ca)
Alice Slabosz (19aas15@queensu.ca)
Shauna Tuinstra (19st58@queensu.ca)
Samuel Lownie (19syl@queensu.ca)

Table of Contents

Abstract

Bitcoin Core is an open-source reference implementation of the bitcoin system and is the authoritative reference on how each part of the technology should be implemented. In this report, we seek to present a description of the conceptual architecture of Bitcoin Core as a system. To do this, we will examine the system's purpose, subsystems, and proposed evolution. Bitcoin Core uses a peer-to-peer architectural style to provide an electronic payment system without the need for trusted third parties. Every peer in the network has access to the public ledger of transactions, called the blockchain, to verify payments and prevent double spending. The overarching goals for developing Bitcoin Core are to provide an open-source, directly peer-to-peer, and private system for electronic payment.

Our examination of the conceptual architecture of Bitcoin Core led us to derive that the main architecture style used is the peer-to-peer style with the use of layers to divide the user interface, local system, and peer network. It includes three main divisions of submodules and components: the network and the nodes within it, transactions and the blockchain, and user wallets and interfaces. Each of these components are examined in this report and their roles in system function, data flow, and evolution are presented to demonstrate a full view of how the Bitcoin Core system is to operate in order to achieve its primary goals.

Introduction and Overview

In the digital age, electronic payment has become increasingly popular for use in e-commerce and sending money between friends. All of these transactions rely on a trusted third party, often a bank, to verify payment and receipt of payment. These payments are also fundamentally non-private and always reversible. Banks must constantly deal with payment disputes and negotiate reversals when demanded by their customers. Bitcoin, however, leverages computation to build confidence in transactions. The definition of an electronic coin is its transaction history that is too computationally expensive for a payment to be reasonably reversed or faked.

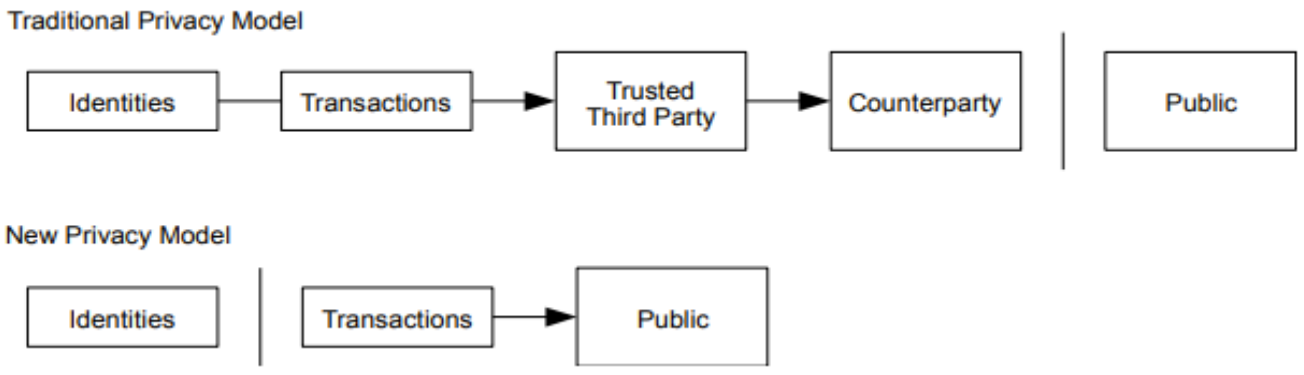

Figure 1: Diagram showing the Bitcoin privacy model versus traditional electronic payments.

Since the initial paper was published in 2008 and the first implementations of a Bitcoin platform emerged, thousands of other cryptocurrencies were created following the basic principles of Bitcoin. In the following decade, cryptocurrency grew to enormous economic status with a market cap high of $3 trillion. Bitcoin specifically reached $1 trillion in total market cap at its height. The purpose of this report is to provide an insight into the structure of this economically and socially important system. By focusing on the conceptual architecture

of Bitcoin Core specifically, we will provide a detailed, while still remaining general, description of how the organization of cryptocurrency systems works to provide purely peer-to-peer, private transactions. More specifically, this report will examine the overall architecture styles of Bitcoin Core, its support for system evolution, concurrency, data flow and control, and division of development responsibilities.

Conceptual Architecture

About

Bitcoin Core provides a purely peer-to-peer electronic payment system. The peer-to-peer network is formed by nodes, where a node refers to a computer running the Bitcoin software. There are three main types of nodes that make this system work: full nodes, miner nodes, and light nodes. Full nodes store a full copy of the accepted blockchain and are able to fully verify transactions according to Bitcoin protocols and rules. These nodes pick up proposed transactions and verify them. If a transaction is valid, the node will broadcast it to the other nodes it is connected to. Once a sufficient number of nodes are in agreement on the transaction's validity, it is added to a pool of valid transactions.

Nodes

Miner nodes pick out transactions from this pool of valid transactions. Then the miner node packages them into blocks. These nodes run a special version of the Bitcoin software that contains the rules for creating blocks. Once a block is created, the miner node then broadcasts it to other nodes. Full nodes pick up the proposed blocks and verify them. If a block is valid, the full node adds it to its own copy of the blockchain before broadcasting the addition to other nodes. Once enough full nodes validate the block, consensus is reached and the transaction is added to the official public ledger of transactions. Miners are rewarded when blocks they create are added to the blockchain to create incentive as mining is expensive but crucial for the network to work.

Light nodes or SPV nodes, are nodes that maintain only a subset of the blockchain and can verify transactions using the simplified payment verification method, which is where they get their name from. These nodes are designed to run on power and space constrained devices such as smartphones, tablets, or embedded systems. A light node only downloads the block headers, not the transactions in each block, this results in a chain that is 1000 times smaller than the full blockchain. Though they can perform transactions like a full node, a light node performs it slightly differently by referencing the depth in the blockchain rather than the height. A light node will verify the chain of all blocks and link that chain to the transaction of interest.

When examining the system from different viewpoints, two main architectural styles stand out. An operational view highlights the peer-to-peer architecture of the network supporting Bitcoin. On the other hand, using a functional viewpoint shows the layered style of the system. The top layer consists of the user interface, mostly cell phone apps or websites that provide users with graphical, easy to use ways to transfer and request Bitcoin. The second layer consists of the local version of the Bitcoin software. This could be a full node, miner node, or light mode. This layer manages a user's wallet, information, and, depending on the type of node, a copy of the blockchain. The final layer is the connection layer. This is where the peer-to-peer network resides. This layer handles the formatting, sending, and receiving of messages across the network.

Blockchain and Blocks

The blockchain is a back-linked list of blocks and transactions, and can be thought of as a stack. Blocks are placed at the top of the blockchain which allows for an unlimited number of blocks on the blockchain. Each block in the chain also references the previous block, known as the parent block through a reference variable in its header. Put simply, each block contains the hash of its parent inside its own header and can be used to link back to the first block ever created, known as the genesis block.

A block is a container data structure that schedules transactions for inclusion in the public ledger, the blockchain. A block is made up of two components: a header and a body. The header of a block contains metadata relevant to the operations of the block while the body contains all transaction data and on average, contains more than 1900 transactions. The header is composed of five sets of metadata: the reference to the previous block hash, the difficulty, timestamp, nonce and merkle tree root.

Wallets, Keys, and Transactions

A wallet is an application that serves as the primary user interface, controls access to the user's money, manages keys and addresses, tracking their balance, and creating and signing transactions. Despite what its name implies, a wallet does not contain bitcoin but rather keys to the "coins" on the network. There are two types of wallets: nondeterministic and deterministic. In a nondeterministic wallet, the keys are independently generated from a random number and have no relation to each other. The issue with this is that each key must be backed up as if control is lost on a randomly-generated key, it becomes accessible and so do the funds associated with it. Bitcoin Core itself uses a nondeterministic, but this is meant for testing purposes only and not commercial use.

A deterministic or "seeded" wallet contains keys which are derived from a common master key known as a seed. This seed is a randomly generated number that can be used to create and recover any key generated from it, and because of this, the seed is the only thing that needs to be backed up, resulting in a more efficient system compared to a nondeterministic wallet.

Transactions are considered to be the most important part of the bitcoin system as without them, bitcoin is just bits of information. The components discussed thus far were designed to ensure that transactions could be created, propagated on the network, validated, and added to the blockchain. What's interesting about transactions is that at a software-level, there are no coins, senders, recipients, balances, accounts, or addresses. Instead, the core of a transaction is what's called a transaction output, which are chunks of bitcoin currency.

Full nodes track all available and spendable outputs which are also known as unspent transaction outputs or UTXOs. When a user's wallet 'receives' bitcoin, it means that the wallet has detected a UTXO that can be spent with one of the keys they have in their wallet. Thus one could think of a user's 'balance' as the sum of all UTXO that their wallet could spend. To create a transaction, the wallet selects from the UTXO it controls and creates one input pointing to it and unlocks it with an unlocking script.

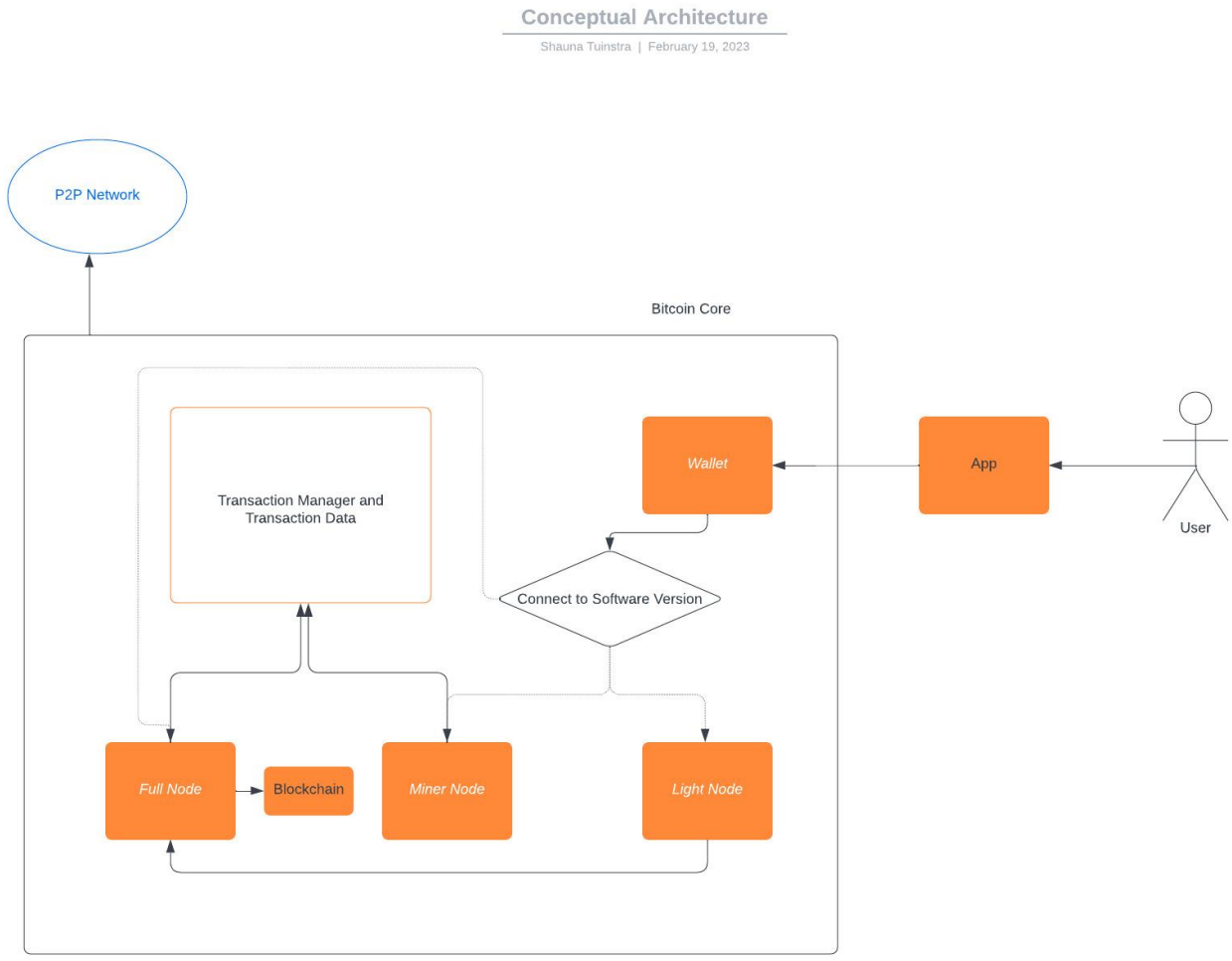

Figure 2: Overview of conceptual architecture demonstrating relationships between submodules

System Evolution

Bitcoin Core has been in development for more than 10 years. Currently, Bitcoin Core supports the blockchain, transactions, contracts, wallets, payment processing, mining, and a P2P network [5]. The scope of our system evolution analysis will be from version 12, released in 2016, to version 24, which is the most recently released version at the time of writing. There are usually about 2 version updates per year. The analysis will go over the evolution of fee handling, wallet implementation, GUI changes, security updates, and future steps.

Fee Handling

According to the Bitcoin Core website, transaction fees (hereafter referred to as just 'fees') are defined as 'the amount remaining when the value of all outputs in a transaction are subtracted from all inputs in a transaction; the fee is paid to the miner who includes that transaction in a block.'

In version 12 (hereafter referred to as v12), a major issue that arose was that as the number of transactions increased, low fee transactions may not be included in blocks. This caused them to be stuck on the network and

possibly never being confirmed. V12 introduced a replace-by-fee that users could opt into. This would replace the user's transaction with a newer transaction by including a higher fee, and allow miners to earn more income. If miners choose this option, the transactions will be flagged as replace-by-fee. Miners will decide which transactions to include in blocks and why.

In v13, replace-by-fees were improved with the addition of the 'child pays for parent' policy, which allowed miners to pick the most profitable set of transactions. V14 allows users to prioritize their own transactions by manually including higher fees for more urgent transactions. V15 implements a toggle for the replace-by-fee so that users can easily use it in subsequent transactions. In v16, replace-by-fee became the norm for transactions, although users could still opt out of it. In v23, fee estimation is improved by taking replace-by-fee transactions into account.

Wallet Implementation

In v12, blockchain pruning was introduced for wallet users. To reduce unnecessary space usage in a user's node, wallet users could remove data that their node had already verified. In v14, users can specify where they want to prune their blockchain.

V15 allows client and RPC users to create several wallets with separate Bitcoin addresses, private keys, and funds. V16 saw the implementation of Segregated Witness (SegWit) in the wallet, called 'the biggest protocol upgrade ever'. SegWit introduced a new block data structure for upgraded nodes and replaced the block size limit with a block "weight" limit, increasing transaction capacity on the network. V16 allowed wallet users to generate SegWit addresses for receiving payments, providing added block space and allowing wallet users to pay lower fees than non-SegWit transactions. This update came with a new address format, bech32.

In v17, the coin selection for wallets was improved by using the branch and bound algorithm. Previously, coins in a wallet were stored as distinct chunks, and when a payment was made from a wallet, the amount was added up by combining different existing chunks. If the chunks in a wallet don't add up to exactly the amount required for a transaction, the overflow funds must be accounted for in the transaction. The branch and bound algorithm improves the efficiency of transactions: it prevents new chunks from being added to transactions by calculating each chunk's fee before it is used, and optimally matches different chunks to avoid having to deal with leftover funds.

Descriptor and Watch-Only wallets are implemented by v20, which allow users to categorize wallets by spending condition and allow users to store their private keys offline respectively. In v23, descriptor wallets are now the default wallet to improve backup and recovery for bitcoin funds. V23 can also spot typos in bech32 addresses.

GUI Changes

In v17, the GUI was given a toggle to prune from a user's wallet for casual or non-technical Bitcoin users. V18 allowed users to access multiple wallets from the GUI. In v19, bech32 addresses were set as the default option in the GUI, and payment protocol support was disabled due to low usage.

V20 now allows hardware wallet compatibility in the GUI. V21 then provides full support for hardware wallets - previously, users had to use commands or manual copy and paste code to sign transactions.

Security Updates

In v12, Bitcoin Core automatically connects to the Bitcoin network by using Tor, which anonymizes users to protect their privacy, if the user has installed Tor on their computer. In addition to Tor support, v22 allows connecting through the Invisible Internet Project (I2P), which is a decentralized anonymous P2P network. This gives users more options to protect their privacy. In v23, support for the CJDNS network was added in addition to Tor and I2P to further allow users to protect their privacy.

V12 introduced faster signature validation, which is a cryptographic trick to prove ownership of Bitcoin. V18 allows users to connect a hardware wallet to Bitcoin Core using the Hardware Wallet Interaction tool. Also in v19, Bloom filters, which transfer data from node to node, were discontinued due to privacy issues. They were replaced by compact client side block filtering, which instead filters relevant data.

In v21, when nodes broadcast transactions to the Bitcoin network, it was possible to locate the origin of the transaction due to frequent re-broadcasts. V21 fixed this by making re-broadcasting less frequent, which actually reduces the need to re-broadcast at all.

P2P networks can be attacked. One example is a partitioning attack where an attacker will control large amounts of Bitcoin nodes, and cut off parts of the network. This can lead to double spending. However, if a node in the partitioned section has a connection to an honest node, it will favor legitimate chains over the attacker's chain. In v19, nodes are given two extra outgoing connections. Their only purpose is to relay blocks, not any transactional data, so that every node has a better chance of being connected to honest nodes.

Multi-signature coins are coins that can only be spent using multiple signatures from different private keys. V22 increased the amount of signatures possible from 16 to 20.

Future Steps

The current version of Bitcoin Core, v24, has added changeless transactions, wallet migration support, a menu for the GUI, improvements to P2P communications, and other aspects. For example, V24 introduces Miniscript support, which is a framework for Bitcoin's own programming language, Bitcoin Script. Miniscript allows developers to structure their Bitcoin Scripts more consistently and safely. According to the v24 release notes, 'users can now create a wallet containing a Miniscript script, create addresses for that wallet and fund them with bitcoin'. In future releases, the wallet should support spending from the user's address.

Overall, the system has evolved significantly over time, with old protocols being swapped out for newer ones, infrastructural calculations being improved, etc.. Some significant future steps are the scalability of Bitcoin Core, and its security. Bitcoin Core has had to make many adjustments to accommodate not only a larger user base, but also the users' own hardware limitations. Since it is a P2P network, it is self-sustaining, but it is still important to keep scalability in mind. Having more users creates more data, and although there are already solutions in place to reduce data storage like pruning, the system may require even more efficient pruning methods in the future. Security is also a concern in the future, since hacker capabilities evolve over time. For example, the advent of quantum computing could revolutionize the cybersecurity field and affect Bitcoin Core in unprecedented ways. Already, the system has had to replace old security systems and add support for existing privacy aids. Security is of the utmost importance when one network contains thousands of user funds.

Control and Data Flow

The network of nodes begins the secure transaction process by timestamping the transaction. The timestamp server works by taking a hash of a block of items to be time stamped and publishing the hash. Every timestamp contains the last timestamp which creates a chain, since each added timestamp reinforces the ones before it. All new transactions are broadcast to all nodes in the network, each node then collects new coming transactions into a block and works to find a difficult proof-of-work for its block.

Proof-of-work is implemented by incrementing a nonce in the block until a value is found that gives the block's hash the zero bits it requires. Once a node finds a proof-of-work, it will then broadcast the block to all nodes. The nodes will only accept the block if all transactions in the block are valid and have not already been spent. Once the nodes have checked the block and accept it, they work on making the next block in the chain using the hash of the newly accepted block as the previous one, as mentioned earlier. The longest blockchain is what nodes will consider to be the correct one and they will keep working on extending that the correct one.

In the situation that two nodes broadcast two separate versions of the next block at the same time, other nodes may receive either first. To not risk messing up the branch, the nodes will work on whichever they receive first and save the other branch as well just in case it becomes longer. The tie between the two branches in question will be broken once the next proof-of-work is found and one branch becomes longer, proving that it is the correct branch. Block broadcasts are tolerant of dropped messages, so if it does not get a block it will request it when it gets the next block and realizes there is one missing. Nodes are able to leave and rejoin the network whenever and will accept the longest proof-of-work chain as their own proof of what happened while they were gone.

Concurrency

As transactions are broadcast, they are added to the memory pool of pending transactions on the P2P network. These are then verified and grouped into blocks with proof of work. The new blocks are once again broadcast to the P2P network. Light and Full nodes then fetch blocks from the network asynchronously and attempt to add them to their blockchain. Nodes can leave and rejoin the network without issue -- accepting the longest blockchain as the correct one once it joins the network. Furthermore, if there is a conflict or discrepancy when many transactions are occurring concurrently -- the longest blockchain with the most proof of work is accepted by nodes as their blockchain.

The utilization of concurrency in Bitcoin Core is achieved through the simultaneous execution of multiple tasks. These tasks encompass downloading the decentralized public ledger (i.e., the blockchain), processing new transactions, and validating new blocks, among others. To perform these tasks, Bitcoin Core implements a multi-threaded approach.

The user can specify the number of threads and the work queue size when making an RPC request. Furthermore, the system utilizes threads to actively handle HTTP requests from users who communicate with Bitcoin Core through the JSON-RPC interface.

The validation of blocks, which involves the application of consensus rules, is accomplished through a dedicated set of threads. Bitcoin Core automatically determines and utilizes the available threads on the user's

computer for this task. Network communication, including the sending and receiving data from peers, is managed by another set of network threads.

A dedicated set of threads actively manages the user's wallet. These threads are responsible for signing transactions, encrypting and decrypting the wallet, updating the wallet to ensure compatibility with the latest features, and sending and receiving notifications to inform the user of relevant events.

Lastly, a set of threads actively controls the scheduler. This group comprises a series of tasks that the system executes at predetermined times, such as at a specific time of day or certain time intervals, such as hourly. These tasks include checking the best chain that a node can be on, switching to a different chain if necessary, flushing the memory, wallet, and chain state to conserve resources, and scanning the blockchain for transactions associated with the wallet.

In conclusion, Bitcoin Core utilizes concurrency to enhance its functionality and efficiency. Utilizing multiple threads to perform various tasks simultaneously reduces the time and computational power required to perform these tasks one -by-one, and on their own. The utilization of concurrency in Bitcoin Core demonstrates the importance of this concept in the realm of digital currency and highlights the sophisticated technology behind it.

Responsibility Division Between Developers

The Bitcoin Core system was initially developed by Satoshi Nakamoto in 2009. In the very first block on the Bitcoin Blockchain, Nakamoto gave some reasoning for the development of Bitcoin core, stating: "Yes, [we will not find a solution to political problems in cryptography,] but we can win a major battle in the arms race and gain a new territory of freedom for several years." This attitude and desire for greater freedom has driven the development of Bitcoin Core around the world, as well as other P2P systems.

Furthermore, because the project was conceived by a single person, Bitcoin Core has a well planned and relatively consistent foundation for other developers to build upon. After Nakamoto left the project in 2010, there was a large community of programmers willing to aid in the development of Bitcoin in the hopes of affording greater freedoms to themselves and their peers. The project has been open source ever since, however there has been a consistent group of committers as well as a designated maintainer, charged with ensuring the project's stability. The maintainers of the project achieve this through a regimented development process as well as rigorous testing.

Thus, like most open source projects, the responsibility division between programmers working on Bitcoin Core is loose and flexible. This allowed for desired features to be added quickly -- if developers felt passionate about said feature. However, this also means that larger, more tedious improvements or additions are not undertaken as often as in a company -- as they often had no financial or social incentive to do so.

Use Cases

The sequence diagrams for each use case are presented in the Diagrams section as Figures 3 and 4 respectively.

Use Case #1 - Creating a Transaction

One of the main uses of Bitcoin is private transactions. This first use case involves two users, A and B, in the scenario of A transferring funds to B through Bitcoin Core. User A will interact with their wallet through the user interface of their downloaded app. Once user B generates a public key and shares it with A, the transaction can be created. After using the app to add in necessary transaction details such as recipient and amount, A's Bitcoin node will broadcast the transaction to the other nodes in the network. The transaction is then validated by other nodes and added to the blockchain. These funds now appear as Unspent Transaction Output (UTXO) and B's wallet displays it as spendable funds.

Use Case #2 - Mining

Another use case involves the process known as mining, which is the process of earning and creating bitcoin through solving complex equations that verify transactions in a given block. While the creation of new bitcoin is an incentive in mining, it is not the purpose of it, instead mining is how bitcoin's security becomes decentralized. A user receives new transactions from the P2P network using an RPC. After receiving this list of transactions, a block template is created and sent back to the mining software which iterates through every possible value for the block header and generates a corresponding hash. Once this is completed, the mining software combines the header with the block and broadcasts the completed block to the network to be added to the blockchain.

External Interfaces

Bitcoin Core takes advantage of a Graphical UI that easily displays actions that users can take on the system. In addition to the GUI, there is also a command line interface (CLI) for more tech-savvy users. On the GUI, users can view their current balance and recent transactions. They can choose between the types of fees they would like to pay, and configure their privacy settings if they have installed Tor or a similar platform. Users can track their received payments, the bandwidth that they use, and their watch-only Bitcoins.

There is also the CLI and lightweight wallet interfaces. On the CLI, users can also create new addresses for receiving payments. They can view their Bitcoin balance, and send payments to multiple different addresses. Users will also be notified of new blocks and transactions. With lightweight wallets, users can use them similarly to the built-in wallet on the normal interfaces, with similar levels of security and privacy. However, these require users to configure their lightweight wallets and ensure the lightweight wallets are securely and privately connected to Bitcoin Core on every usage. Otherwise, lightweight wallets can leak data and be more vulnerable to attacks. To prevent this, users must connect their lightweight wallets to a trusted peer, which is a full Bitcoin Core node.

Diagrams

Sequence Diagrams

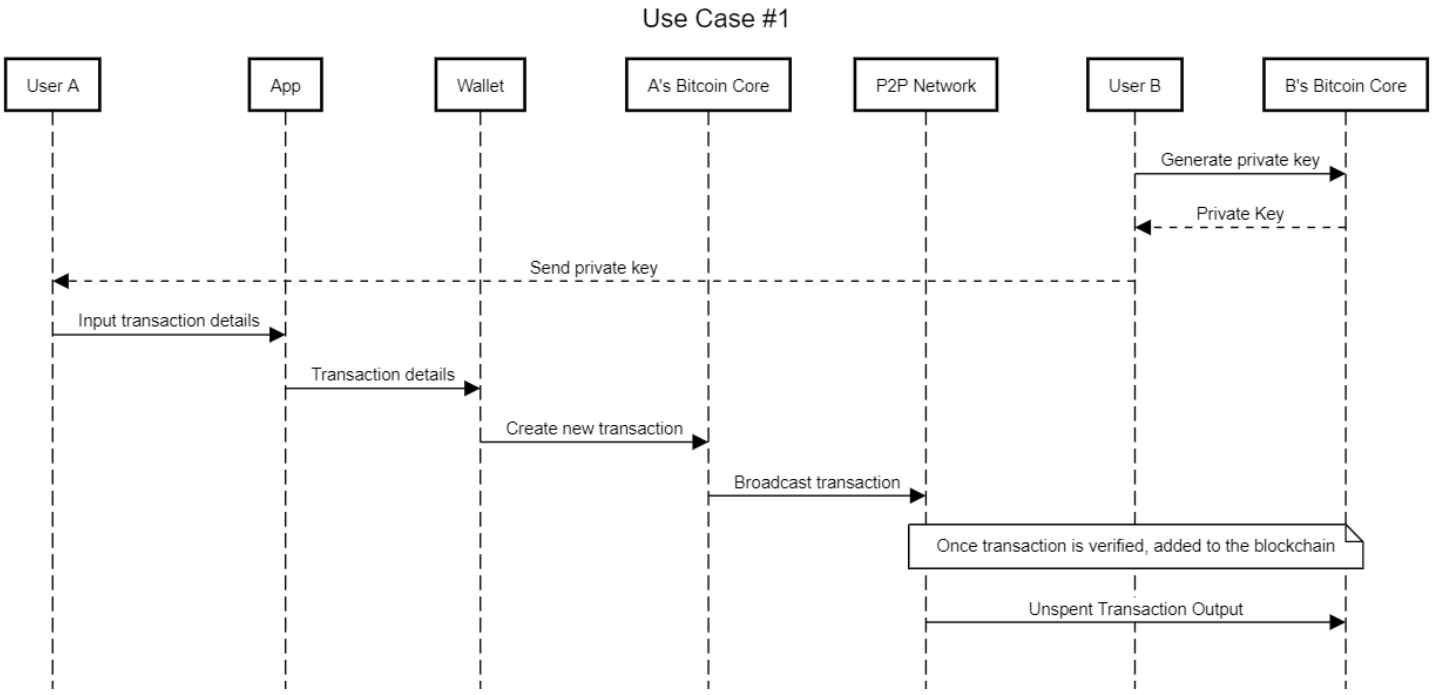

Figure 3: Sequence diagram for the use case of user A sending currency to user B

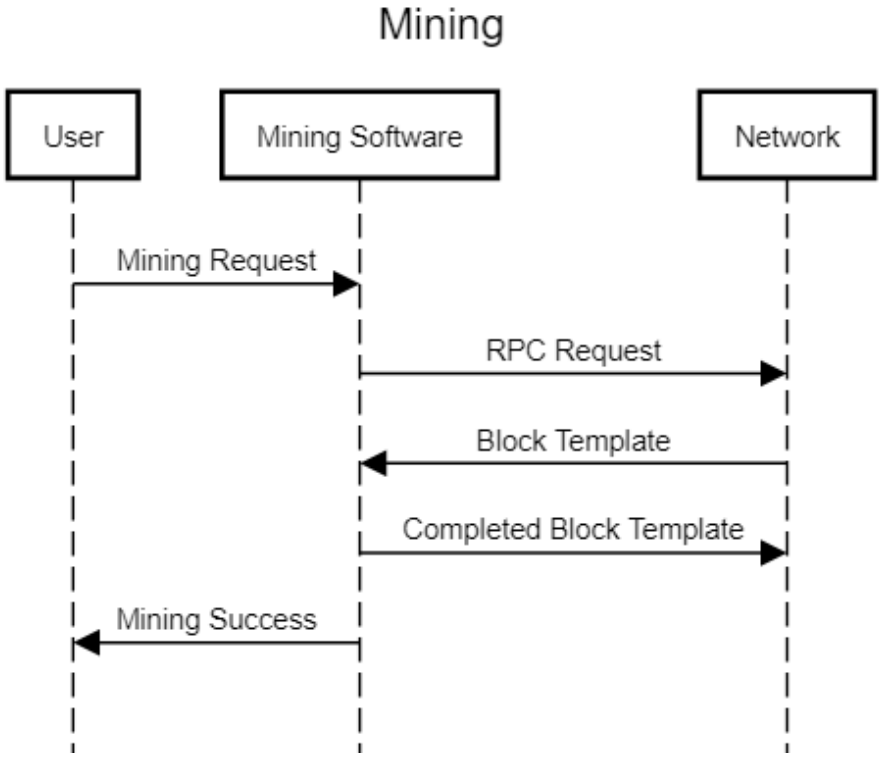

Figure 4: Overview of mining architecture showcasing the relationship between systems.

Data Dictionary

Block	A container data structure that schedules transactions for inclusion in the public ledger, the blockchain; a block is made up of two components: a header and a body
Fee	The amount remaining when the value of all outputs in a transaction are subtracted from all inputs in a transaction; the fee is paid to the miner who includes that transaction in a block.
Node	A computer running the Bitcoin software
Wallet	An application that serves as the primary user interface, controls access to the user's money, manages keys and addresses, tracking their balance, and creating and signing transactions

Naming Conventions

GUI	Graphical User Interface
P2P	Peer-to-Peer
RPC	Remote Procedure Call
SPV	Simplified Payment Verification
UTXO	Unspent Transaction Output

Lessons Learned

For Conceptual Architecture, we searched through plentiful amounts of documentation in order to understand the main concepts of the system. At times it was difficult to determine what was relevant information in the context of conceptual architecture, as the documents were lengthy in terms of both size and depth. At the same time, these detailed readings will prove useful when it comes to the next two assessments and was a valuable experience for understanding how the system worked at its core.

For System Evolution, we learned there was a blog that summarized the major developments of every version of Bitcoin Core. This was much better than the alternative, which was looking through the detailed release notes on Bitcoin Core's GitHub.

Control and Data Flow is an especially important component of Bitcoin Core, since it highlights what makes this software such a necessary system for those who wish to complete bitcoin transactions. Here, we looked at the details of what happens when a transaction occurs and what is done to make the process secure.

In our research on Concurrency, we gained insight into the advantages of incorporating concurrent operations in systems, particularly in the context of digital currency systems. Our findings indicate that the implementation of concurrent processes can significantly improve the efficiency and functionality of such systems, thereby conserving resources and reducing processing times.

Conclusions

Bitcoin Core is a software that allows bitcoin owners to safely make electronic transactions without relying on trust. The peer-to-peer network using proof-of-work to post a public history of transactions is what makes this a safe and practical method. All peers have access to the public ledger of transactions, called the blockchain, in order to verify payments and to prevent a massive problem at hand with bitcoin transactions called double spending. The three main divisions of submodules and components including the network and the nodes within it, transactions and the blockchain, and user wallets and interfaces, are what make this system functional.

Normally online transactions or e-payments rely on trusted third parties to work. However these payments between banks are fundamentally non-private and therefore always reversible by the bank. Bitcoin however leverages computation to build confidence in transactions. This is why Bitcoin Core is a necessary software for those who wish to complete bitcoin transactions without any risk.

References

[1] Bitcoin Wikipedia. (2021, March 1). *Gavin Andresen*. Bitcoin Wiki - Gavin Andresen. Retrieved February 19, 2023, from https://en.bitcoin.it/wiki/Gavin_Andresen

[2] Bitcoin Wikipedia. (2022, April 8). *Satoshi Nakamoto*. Bitcoin Wiki - Satoshi Nakamoto. Retrieved February 19, 2023, from https://en.bitcoin.it/wiki/Satoshi_Nakamoto

[3] bitcoin.org. (2023). *About: Bitcoin Core*. Bitcoin. Retrieved February 19, 2023, from https://bitcoin.org/en/bitcoin-core/

[4] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *SSRN Electronic Journal*. Retrieved February 19, 2023, from https://bitcoin.org/bitcoin.pdf.

[5] Namcios. (2022, November 28). *Bitcoin Core 24.0 released: What's new*. Bitcoin Magazine. Retrieved February 19, 2023, from https://bitcoinmagazine.com/technical/bitcoin-core-24-0-released-what-is-new