CISC 322/326 - Software Architecture
Assignment #1: Report
Sunday, February 19, 2023

Conceptual Architecture of Bitcoin Core

Team Bingus II
Connor Decan - 19ctd3@queensu.ca
Allen Geng - 20lg5@queensu.ca
Kevin Jiang - 19shj1@queensu.ca
Yuanqi Liang - 18yl204@queensu.ca
Jayden Ting - 19tkt1@queensu.ca
Kevin Zhang - 19jyz5@queensu.ca

Abstract

Bitcoin has emerged as a means of wealth accumulation since its inception, and it is reliant on the Bitcoin Core system. Bitcoin Core is the original and most widely used full-node software implementation of the Bitcoin protocol, and it is an open source program that allows users to transact and confirm at will within the network. This report specifically centers around the peer-to-peer network of Bitcoin Core and provides a comprehensive analysis of the architecture and inner workings of the Bitcoin Core, starting with its conceptual architecture such as architectural style, design patterns, subsystems and components, as well as use cases, external interfaces, a data dictionary, and naming conventions.

Introduction and Overview

On September 15, 2008, starting with the collapse of Lehman Brothers, the financial crisis broke out in the United States and spread rapidly around the world. In response to the suddenness of the situation, and to take back control of the economy from the financial elite and provide an opportunity for ordinary people to participate in the financial center, Bitcoin was proposed in 2009 as a response to the crisis (Frankenfield, 2022). It is a decentralized cryptocurrency and operates as a virtual currency, bypassing the control of individuals, organizations, or financial institutions (*Who is the mysterious Bitcoin creator Satoshi Nakamoto?* 2022). In 2023, Bitcoin experienced substantial growth, with a 28% increase in January alone (Browne, 2023). The success of Bitcoin is based on the technology behind it, specifically the Bitcoin Core. Bitcoin Core is a client software designed for the Bitcoin network by Bitcoin. Its purpose is to enable users to engage with the Bitcoin network. The software is constantly updated and audited by a global community of developers. Bitcoin Core provides users with wallets that can be used to authenticate and perform transactions with nodes through the application itself or external links and also enables secure transactions for users and the validity of every transaction within the network. Therefore, the software architecture is significant.

The initial report provides an overview of Bitcoin Core's structure and interconnections by referencing the official website's open-source code and developer guide on Github. We agree that Bitcoin Core follows a peer-to-peer architecture style with benefits like extensibility, the ability to respond to network issues by establishing new links to others, and protecting user's privacy. We also categorize the components of Bitcoin Core into eight distinct parts including Peer Discovery, Messages, Initial Blocks Download, Blocks-First, Headers-First, Block Broadcasting, Orphan Blocks, Transaction Broadcasting.

Simultaneously, we examined the components and communication methods of the eight modules and integrated them into the subsystem to make it more coherent within the overall structure. We utilized the developer guides to divide the subsystem into several parts, which include Block Chain, Transactions, Contracts, Wallets, Block Verification, Network and Mining. Within these parts, we identified more specific components such as proof of work and fork detection, wallet programs and wallet files, and peer discovery, communication, and broadcasting. Additionally, we created a global control flow to demonstrate the workflow of Bitcoin Core. The global control flow illustrates that the primary function of Bitcoin Core is to synchronize parallel processes, which is in line with its concurrent nature. We suggest two use cases, Standard P2PKH Transaction and P2PKH Script Validation, to effectively demonstrate how our proposed architecture can be implemented.

Finally, we have provided a data dictionary for the report with explanations of proper nouns and share our thoughts about what we learned in Bitcoin Core.

Architectural Style

Peer-to-Peer

Bitcoin operates using a peer-to-peer network enabling block and transaction exchange. Full nodes download and verify every block and transaction before relaying them to other nodes.

Peer Discovery

When nodes start for the first time, they need a reliable way to connect to the network. They start by querying DNS seeds to get DNS A records with the IP addresses of full nodes accepting new incoming connections. DNS seeds are maintained by Bitcoin community members, and may either provide manually updated IP addresses or an automated process by scanning the network. Once a program is connected to the network, peers can send it *addr* messages with the IP addresses and port numbers of other nodes.

Bitcoin core records known peers for direct connections after startup without using the DNS seeds. Peers may leave the network or change IP addresses, so multiple connection attempts may be made. If these Bitcoin Core cannot create a connection using its records in 11 seconds, it falls back to DNS seeds. If DNS seed servers do not provide a response in 60 seconds, it falls back to attempting a connection using hard-coded IP addresses that were active around the time the version of the software was released.

Messages

Aside from *addr* messages, peers may send and receive different messages for requesting and receiving information. The *getblocks* messages request inventories contains a header hash, requesting the best block chain for a block with that header hash.
The *inv* message passes up to 500 inventories to a node.

The *getdata* message requests blocks using header hashes. It either requests 128 blocks with the blocks-first method or 16 blocks with the headers-first method.
The *block* message contains a serialized block of data.
The *getheaders* message contains multiple header hashes from the block to request the local best block chain from a sync node.
The *headers* message contains up to 2000 headers as a response to a *getheaders* message.
The *tx* message contains a transaction.

Initial Blocks Download (IBD)
Used both during the initialization of the node on its first startup as well as when there are many blocks that need to be downloaded. Bitcoin Core uses the IBD method whenever the last block on its local best block chain has a header time over 24 hours old.

Blocks-First (BF)
The goal is to download the blocks from the best block chain in sequence. The node sends a *getblock* message to a sync node. The sync node responds with an *inv* message. The node then sends a *getdata* message to request full blocks, and the blocks are received through a *block* message.
The primary advantage of BF IBD is its simplicity. However, it fully relies on a single sync node to download data. The result is the node being reliant on the upload speed of the sync node. The node also is subject to download restarts as it won't be able to identify the block chain as non-best until IBD is near complete. It can also cause the node to be subject to disk fill attacks and high memory use, relating to download restarts, where the sync node may be sending blocks out of order, resulting in wasted disk space and high memory use.

Headers-First (HF)
The goal is to download the headers for the best header chain, partially validate them, then request blocks and headers in parallel. The node sends a *getheaders* message to a sync node. The sync node responds with an *inv* message. After partial validation, it can request more headers and request blocks using *getdata* messages, sending 8 requests to full node peers for 16 blocks each, removing the reliance on the upload speeds of the sync node. The HF node uses a moving download window to avoid the effects of stalling nodes on the download and validation of data.

Block Broadcasting
When a miner discovers a new block, it uses one of several methods. The unsolicited block push method sends a block message to each of its full node peers. The standard block relay method is bypassed as the miner knows that its peers do not have the block. The standard block relay method has the node send an *inv* message to each of its peers, referring to the new block. BF peers that want the block reply with a *getdata* message. HF peers reply with a *getheaders* message with the highest-height header on its best header chain and some further back for fork detection. This message is followed by a *getdata* message. This results in a HF peer being able to refuse orphan blocks. A relay node may skip the round trip overhead of an *inv* message followed by *getheaders* by instead having the miner send a *headers* message containing the full header of the new block. HF peers that receive this message will partially validate the block then request the full contents with *getdata* if the header is valid. This option can be enabled by the HF node by sending a special *sendheaders* message during the connection handshake

Orphan Blocks
BF nodes may download orphan blocks.The node will request inventories of any blocks containing blocks the node is missing using a *getblocks* message. The downloading node will validate the blocks, and will validate the orphan block if a parent of the block is validated.

Transaction Broadcasting
In order to send a transaction, an *inv* message is sent. The node will respond to a received *getdata* response using *tx*, which the receiving peer will forward in the same manner given that the transaction is valid.

Subsystems

Block Chain
The block chain component keeps the time stamped recorded of transactions that have occured. They are responsible for maintaining the integrity of the transactions. Which are preventing double spending and modification of previous transactions. Each block stores the Merkel Root which contains the transactions, a block header, and a hash to the previous block head that allows the blocks to be chained together.

> **Proof of work:** To protect the block chain, a proof of work system is used. A number will be generated by the system and acts as the boundary for hashing value. In order for a block to be added, it needs to show it did work by hashing a number below the boundary. Take in consideration of the proof of work, and the

chained structure, an attacker would need to redo all the hashing of the previous block in order to make a modification, which is extremely difficult.

Fork detection: The fork detection process will examine the block chain and their proof of work to detect hard fork in the chain. The fork will occur if an upgrade to the consensus rule has been made, which causes blocks using different rules to be forked. The fork detection process aims to prevent forked chains with large height being present in the block chain, as they may cause financial loss for parties involved. If a hard fork is detected, it will send an alert to notify the operator about the situation.

Transaction

The transaction system allows the user to spend bitcoin. Each transaction consists of a version number that specifies the rules this transaction will follow, an input which spends the output from the previous transaction, an output that waits for the future input of the next transaction to spend it, and a locktime for specifying the earliest time this transaction can be added to the block chain. The transactions are also chained together, the output of one transaction will be linked to the input of the next transaction. A transaction can have multiple outputs, sending to different transactions, but to prevent double spending, an output can only be used by one transaction as input. The input will have a transaction identifier and an output index number to identify the output it should be linked to. It will also contain a signature script which provides information that satisfies the public script created from the output of the last transaction. For a transaction to occur, the receiver will send their public key to the sender, while the sender creates a pubkey script that only allows the holder of the public key to spend the bitcoin. In order to spend the bitcoin, the receiver will need to create the signature script that agrees on the condition of the pubkey script.

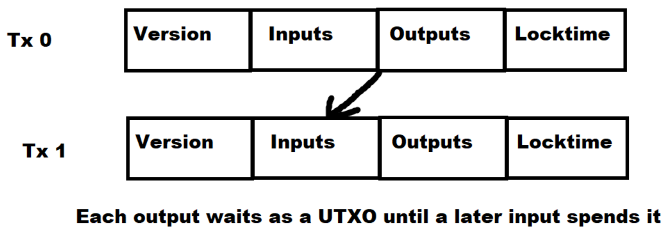

Figure 1: Interaction between transactions

Wallet

Wallet Program: The wallet system allows users to make transactions. It can be considered as the primary interface for the user. The goal of the system is to allow the user to spend or receive bitcoin. The three main features of the system is to distribute public keys, signing, and network. A wallet program does not need to consist of all three features, they can have specific purposes such as signing only, network only, or distribute only. How the wallet should be structured will depend on the desired requirements of the problem. For example, distributing only wallets would be a good fit for a difficult-to-secure system. Where hardware wallets provide much better security than Full service wallets.

Wallet File

The wallet file stores and manages a user's keys. The wallet could be non deterministic where keys are generated randomly, or deterministic where keys are generated from a seed.

Block Verification

The block verification module verifies the integrity of a block chain. This verification is meant to help the user to verify payments, and make sure the blockchain they are adding to is a valid one. Its goal is to enforce security of the software. The verification could be either done with a Full Node, or simplified payment verification method (SPV). Where Full Node will examine the entire block chain from the base block providing better security, SPV will only look at the headers,merkel root, and request transactions from full nodes when needed, providing efficiency.

Network

The network system allows the peers to connect to each other, maintaining a peer to peer network. The system may be responsible for peer discovery,and broadcasting. It is a system that allows communication between the peers in the network.

Peer Discovery, Communication: Allows new programs to join the network by going through DNS name, and finding a full node that is accepting new connections. For performance issues, the system uses dynamic DNS seeds to get the IP address of active nodes. This can avoid the latency when trying to connect to a node that is down.

Broadcasting: When a new block is discovered by a node, it will be broadcast to the entire network. This is to share information with all peers in the network.

Memory Pool

Mempool stores unconfirmed transactions in non-persistence memory. These records of unconfirmed transactions are used by miners to identify transactions they intend to mine. They can also be used for SPV validation.

Mining

The system allows adding blocks to the block chain, which increases the security by increasing the height of the block chain. The newly created block will be broadcasted using the network system, and be added to the block chain for all peers.

Global control flow

The *Bitcoin Journal* included many diagrams to illustrate the process of how Bitcoin works. Examples of transactions, server timestamps, disk space, etc. The figure below shows the transactions diagram.

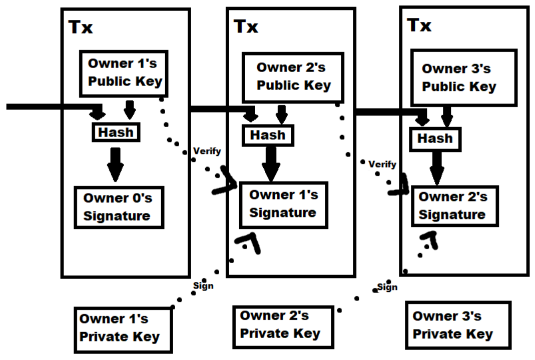

Figure 2. Transaction system in Bitcoin

To explain the transaction diagram, we must first understand how the coin structures, the coin is defined as a chain of digital signatures according to the *Bitcoin Journal*, it includes all of the previous owner's public key and signature, as well as the current owner's public key. As for the global data flow and control flow, the figure below shows the architecture of the Bitcoin system.

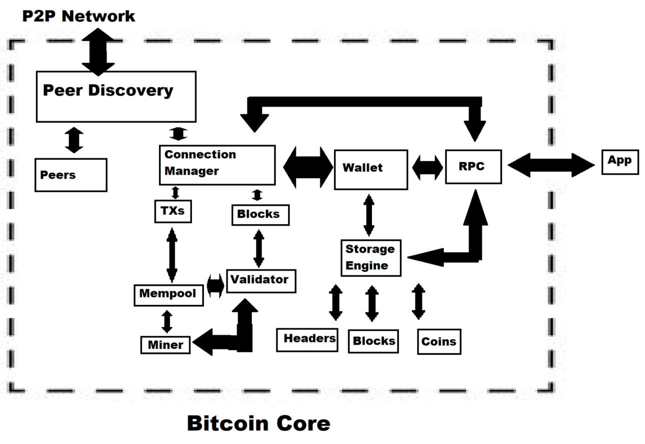

Figure 3. Bitcoin core architecture

The application that is used by the user, communicates with the RPC, then the RPC can access the wallet, storage engine, and connection manager, through this the user can initiate trading with others, check their storage, or put money in from their wallet. The connection manager is the center of the structure here, it allows trading, access to the user's wallet, and connection to other users.

Concurrency

While most other programs want to support concurrency and let users run multiple threads at the same time, in the case of Bitcoin, it is actually something that needed to be avoided as it may cause someone to be able to sell their Bitcoin twice, as known as double spending. Bitcoin developers used minting as the primary way to ensure that double spending does not happen, minting is something that happens when the cryptocurrency, in this case, Bitcoin, is first put on the blockchain. Minting is validating that the currency is now legitimate, and it has different implementations according to which organization developed it. In the case of Bitcoin, their minting would decide which transaction comes first if there is double spending that happens, and would only allow the first transaction on record. To let the user get a clear sight, and not worry about being the victim of someone else double spending, the development team also enforced that the transactions must be publicly announced.

Use Cases

Use Case 1: Standard P2PKH Transaction

Let’s say users A and B want to make a standard P2PKH transaction. After A has generated a private/public key pair using their wallet and sends the hashed public key to B’s address, B’s wallet “unhashes” the key which allows it to initialize the P2PKH transaction output containing information on how someone can use that output if they can prove the ownership of the corresponding private key. B’s wallet then broadcasts this transaction output to the network which categorizes it as an UTXO, this transaction is also added to the blockchain. When A decides to spend the UTXO, they must generate an input that references the transaction output using their wallet, this is done by creating a Transaction Identifier and output index. A’s wallet must also create a signature script that contains data on A’s unhashed public key and a signature used to confirm the ownership of the corresponding private key, the Transaction Identifier is broadcasted to the network along with the output index and signature script to be validated before further broadcasting them or including it in another block of transactions.

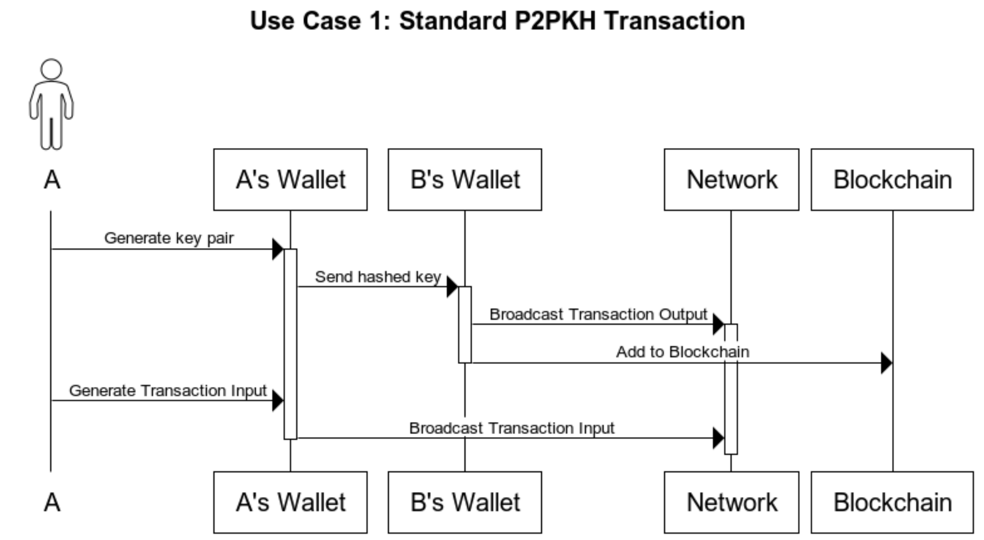
Use Case 1: Standard P2PKH Transaction
A
A's Wallet
B's Wallet
Network
Blockchain
Generate key pair
Send hashed key
Broadcast Transaction Output
Add to Blockchain
Generate Transaction Input
Broadcast Transaction Input

Figure 4. Sequence Diagram of Use Case 1: Standard P2PKH Transaction

Use Case 2: P2PKH Script Validation

In order to validate a P2PKH Script transaction, a user must evaluate the signature script and pubkey script of the transaction. First, the signature from the signature script is pushed onto the empty evaluation stack followed by the public key, this key is then copied using the “duplication operation” from the pubkey script. Next, the pubkey script uses the “hash operation” to transform the duplicated key into a hash of the public key. The pubkey script then pushes its own key hash onto the stack, if the pubkey script “matches” the signature script then the top two entries of the stack would be identical hashed keys. The pubkey script then performs the “equal verify operation” which checks if the hashed keys are identical and pops them from the stack, if the hashes are not identical then the transaction fails otherwise the pubkey script performs the “check sig operation” pops the signature and the authenticated public key and checks if they match. If the key matches with the signature and the signature was generated using all the data needed to be signed, “true” is pushed onto the stack which validates the transaction. If the key matches with the signature do not match, “false” is pushed onto the stack and the transaction is invalid.

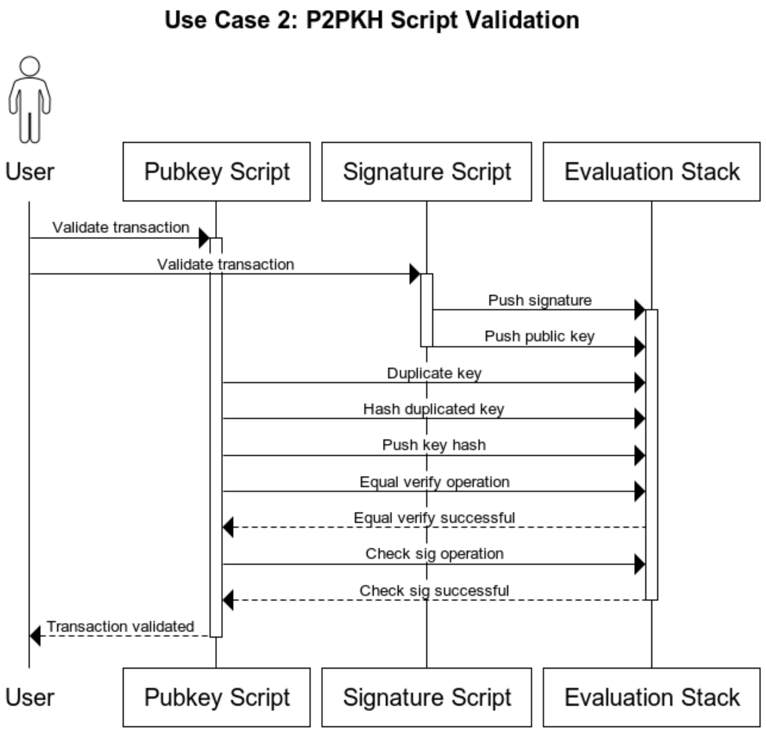

Figure 5. Sequence Diagram of Use Case 2: P2PKH Script Validation

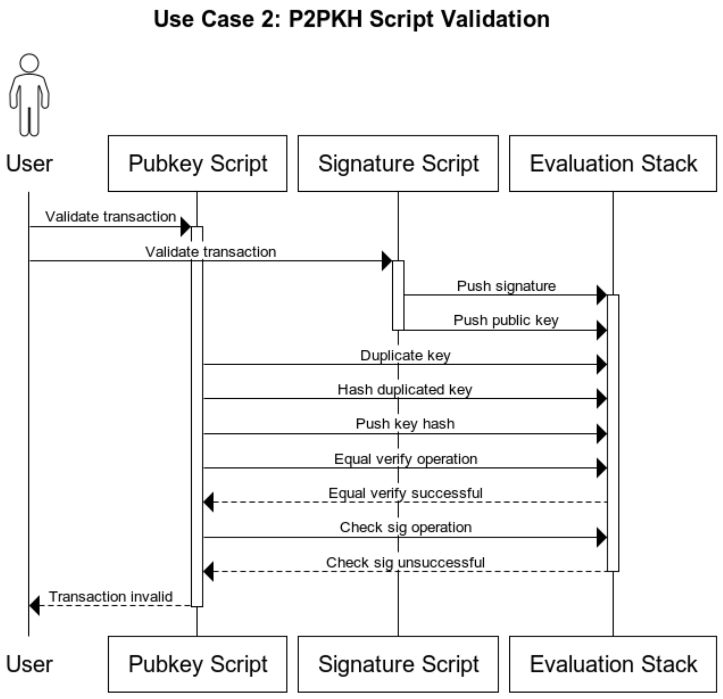

Figure 6. Sequence Diagram of Use Case 2: P2PKH Script Validation (Check sig unsuccessful)

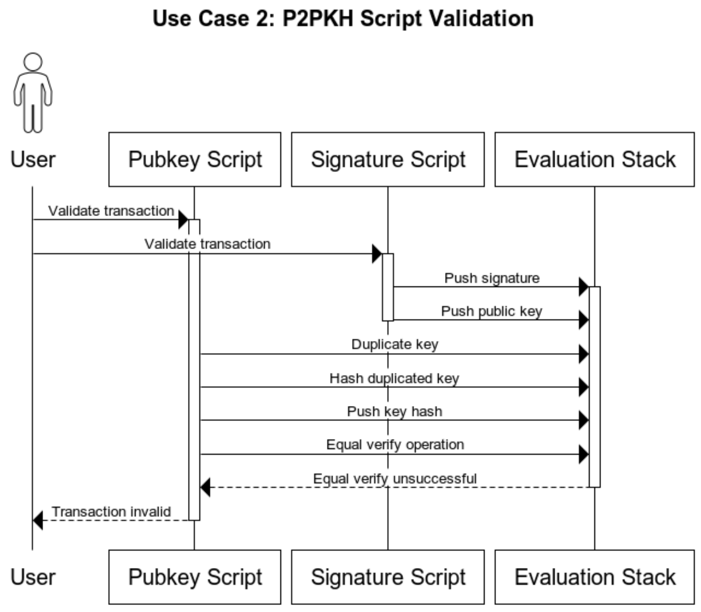

Figure 7. Sequence Diagram of Use Case 2: P2PKH Script Validation (Equal verify unsuccessful)

Data Dictionary

Initial block download (IBD): used anytime there is a large number of blocks that need to be downloaded

DNS seeds: DNS names hardcoded into Bitcoin Core that provide new nodes with the IP addresses of full nodes that may accept new incoming connections.

DNS A record: the IP address of a given domain.

Inventories: unique identifiers for information on the network. Blocks use the hash of its header for its unique identifier.

Blocks-first:: a method of initial block download that aims to download the blocks from the best block chain in sequence.

Header-first: a method of initial block download that aims to download the headers for the best header chain, validate, and then download corresponding blocks in parallel.

Sync node: a peer chosen for a node to synchronize its blocks with

Download window: a 1024-block moving window in which blocks must be finished downloading after being requested to before Bitcoin Core tries to swap connections with another node

Stale blocks. blocks whose previous block header hash field refers to block header(s) that are known, but are not part of the best block chain.

Orphan blocks: blocks whose previous block header hash field refer to a block header this node hasn't seen yet

RPC: remote procedure call, basically a server that was implemented to allow users to access the blockchain and process transactions.

Txs: transactions

Mempool: non-persistent memory storing unconfirmed transactions

Hard fork: A permanent divergence in the block chain commonly occurs when non-upgraded nodes can't validate blocks created by upgraded nodes that follow newer consensus rules.

Merkle Tree: A tree constructed by hashing paired data (the leaves), then pairing and hashing the results until a single hash remains, the merkle root. In Bitcoin, the leaves are almost always transactions from a single block.

Full Node: A computer that connects to the network

Pubkey script: A script included in outputs which sets the conditions that must be fulfilled for those bitcoins to be spent. Data for fulfilling the conditions can be provided in a signature script. Pubkey Scripts are called a scriptPubKey in code.

Signature script: Data generated by a spender which is almost always used as variables to satisfy a pubkey script. Signature Scripts are called scriptSig in code.

Signature: A value related to a public key that can only feasibly be created using the private key that created that public key.

P2PKH: Pay-To-Public-Key-Hash. A P2PKH payment address is made up of a hashed public key, the spender must generate a corresponding standard pubkey script to perform the transaction.

UTXO: Unspent Transaction Output

Conclusion:

To summarize, our conceptual architecture for Bitcoin Core demonstrates how a peer to peer architecture can be used to allow users to store a digital wallet, and connect with other users.

As mentioned in both the prior statement and at the beginning of the report, the conceptual software architecture used utilizes a peer to peer style. The reason this was chosen was because a core element of the system is allowing users to connect with one another. Rather than a client-server style, the peer to peer style works better for this type of software because it allows the users to communicate with each other directly, rather than having to go through a middleman.

The system uses many components to allow for complex processes to occur. One important one is the wallet, which allows the user to store their data and use it to share data with other users. There are also multiple ways in which the wallet could be implemented, as mentioned, such as fully virtual vs having a hardware component. Both methods have unique upsides and downsides. Another important component is the blockchain, and its related subsystems. It is what allows the whole system to function, as it uses block verification to verify features such as transactions, which ensures the safety of the users.

Lessons Learned:

Through this project, we learned about the different components of how a bitcoin architecture might be composed of. Specifically, we used information from locations such as the bitcoin core's github page to gain insight on how its architecture works, and what components it is made up of. We also used other bitcoin related sources, such as the official bitcoin website.

Throughout the assignment, there were also some physical issues that occurred in part due to the timing of when we worked on the assignment. During the week of the quiz, our group members were spending time studying for the quiz. The week after that was midterms, and the week after that was reading week. This made it difficult for the group to meet in person. From this, we decided that the easiest way to get work done despite having different schedules was to divide the work of the report in parts. As a result of this, however, the work may lack some consistency due to difficulties in communication, which will hopefully be improved in the future.

References

Antonopoulos, A. (2017, July 20). *Mastering Bitcoin*. GitBook. Retrieved February 14, 2023, from https://cypherpunks-core.github.io/bitcoinbook/

Binance Academy. (n.d.). *Bitcoin Core*. Binance Academy. Retrieved February 14, 2023, from https://academy.binance.com/en/glossary/bitcoin-core

Browne, R. (2023, January 17). *New Year, New Rally: Why bitcoin is up 28% This Month after a tumultuous 2022*. CNBC. Retrieved February 16, 2023, from https://www.cnbc.com/2023/01/16/why-is-bitcoin-btc-rallying-in-january.html#:~:text=Bitcoin%20has%20begun%202023%20on%20a%20positive%20note%2C%20with%20the,high%20bitcoin%20notched%20in%20Nov.

Cointelegraph. (2022, February 11). *Who is the mysterious Bitcoin creator satoshi nakamoto?* Cointelegraph. Retrieved February 16, 2023, from https://cointelegraph.com/bitcoin-for-beginners/who-is-satoshi-nakamoto-the-creator-of-bitcoin#:~:text=Why%20was%20Bitcoin%20created%3F,in%20a%20decentralized%20financial%20system.

Developer guides. Bitcoin. (n.d.). Retrieved February 13, 2023, from https://developer.bitcoin.org/devguide/index.html

Frankenfield, J. (2022, November 23). *What is bitcoin? how to mine, buy, and use it*. Investopedia. Retrieved February 16, 2023, from https://www.investopedia.com/terms/b/bitcoin.asp

Open source P2P money. Bitcoin. (n.d.). Retrieved February 13, 2023, from https://bitcoin.org/bitcoin.pdf