CISC 322/326 Assignment 1

CONCEPTUAL ARCHITECTURE OF BITCOIN CORE

Group 8: EVANGELION-01

Jinlun Zhang: 16jz61@queensu.ca

Jing Tao: 21jt35@queensu.ca

Jihong Zhang: 19jz138@queensu.ca

Xueyi Jia: 18xj8@queensu.ca

Hejin Wang: 15hw45@queensu.ca

Yunhan Zhou: 17yz101@queensu.ca

February 19th, 2023

Table of Contents

1.0 Abstract

In our technical report, we will provide a high-level overview of the conceptual architecture of the Bitcoin Core system by reading the Bitcoin Core developer documentation, searching the Bitcoin Core resource collection on Google, YouTube and so on. In summary, the Peer-to-Peer Network architecture is primarily used to help the Bitcoin Core system to implement their business requirements such as decentralized consensus, and the Client-Server architecture for developing each node in the Peer-to-Peer Network (an instance of Bitcoin Core) is explored to ensure the software's reliability such as guaranteed synchronous responses from the modules of the software (which differs from the Publish-Subscribe architecture style), non-repudiation in transactions, quick requests and responses transmission quality and convenience, etc. Since the Bitcoin System is likely to become a world of potential for new users who are eager to earn money through cryptocurrency trading, we write this report to cover all the important system components to help the readers to understand the software and its architecture.

2.0 Introduction and Overview

Bitcoin core is a peer-to-peer electronic payment system which uses a cryptocurrency to transfer value over the internet or act as a store of value like banknote in your wallet. Before 2008 Global Financial Crisis, Physical currency has always been the predominant way to buy things, where banks and financial institutions control it all. With technology evolving seemingly with every second and enhance of awareness of people's financial management, a new payment system had to be established, that's the reason why Bitcoin was created, it appeared as computer file and stores in your online digital account, essentially a public listing of all the transactions. To some extent, it can be the dawn of a new age.

One of the merits in Bitcoin is that it's seen as one of the most secure methods of currently purchasing goods or services. This comes from the incorporation of Blockchain system.

Each transaction involving Bitcoin is recorded on a public server called the Blockchain, this made available on a variety of computers globally. So, the Blockchain system is designed to make it incredibly difficult to make fraudulent transactions. Also, Bitcoin System software can be regarded as an automated trading bot that monitors the crypto market for data, gathers that data, and uses it to implement calculated, informed trading strategies on your behalf. That means, when you finish using your demo account and prepare to live trade, the algorithm will begin to trade on your behalf.

In the past ten years, Bitcoin Core has gone through 6 essential iterations, with each version adding new features, security improvements, and bug fixes, it was reported on February 2, 2023, that Bitcoin broke through 24,000 US dollars per coin, continuing to hit a new high in the previous period. However, in order to avoid consuming a lot of power and computing power and taking up too much hard disk space of blockchain, it still needs to keep pioneering and developing in the future.

This report will touch upon the system's conceptual architecture analysis, including subsystems, concurrency, control, data flow, use cases, sequence diagrams and so on. After intensive discussions and extensive researches on documentations of Bitcoin Core, we conclude that the architecture style adopted by Bitcoin Core is Client-Server style integrated with Peer-to-Peer network architecture style, and we will show our proof of this summary. Finally, we will also talk about what we learn through the analysis and give an overview of our findings.

3.0 Architecture Analysis

3.1 Components Breakdown and Interaction

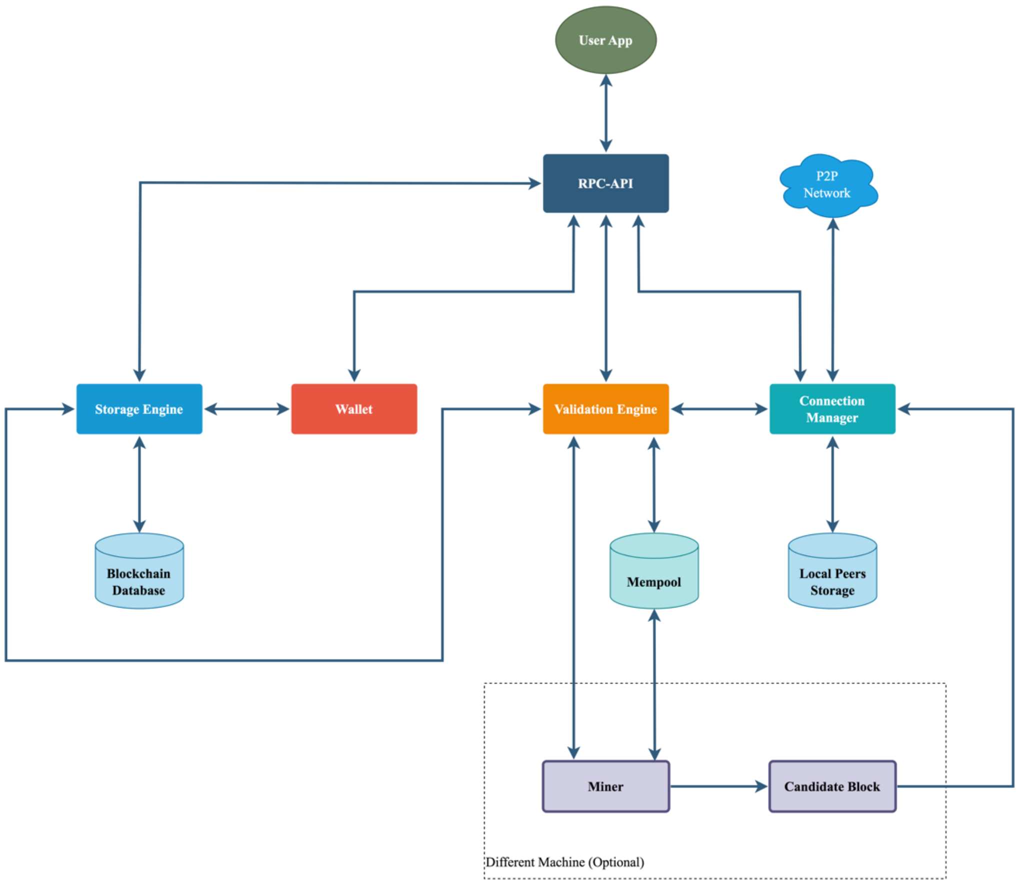

Figure 1: Bitcoin Core Conceptual Architecture (Client-Server Style)

The Bitcoin Core Software can be divided into separate functional components as shown on Figure 1. The details and interactions of those components are explained below.

To begin with, the user can acquire a control access to the Bitcoin Core through a user application, which can be a third-party application like Bitwasp. This application is communicating with the main components of Bitcoin Core through the RPC (Remote Procedure Call) API component, which grants the user the ability to spend funds from his/her wallets, affect consensus verification, read private data, and even perform operations that can cause loss of money, data, or privacy (So Bitcoin Core must be utilized and configured to reduce the risks that its RPC interface will be abused). To complete user operations like receiving and spending funds to and from the user wallet, the RPC-API component is interacted with the user wallet component, the storage engine component, the validation engine component, and the connection manager component.

To be more specific about those components, the wallet component has the control access to check the user's remaining funds, manage keys and addresses, track the account balance, and create and sign transactions. To complete those operations, the wallet component must communicate with the storage engine component (while the user can also communicate with the storage engine directly through the RPC-API component), where the database of bitcoin transaction blocks (the public ledger or the local copy of the blockchain) can be accessed by the storage engine component to fulfill requests like appending a received validated block of transactions or assisting in initiating an unsigned user transaction.

The storage engine component is also connected with the validation engine component that validates the individual transactions and the blocks of transactions received from the other nodes in the Bitcoin P2P Network. This is the key component of decentralized consensus in the Bitcoin P2P Network as it ensures that only valid transactions will be propagated on the network. After being validated by the validation engine for the first round (basic validity checking of transactions such as proper formatting and signature verification), the validated but unconfirmed individual transactions will be accumulated in the "mempool" storage component. As one can expect, the "mempool" component, which is also called as the memory pool or the transaction pool component, is served as a temporary buffer to keep track of the transactions that are verified (known to the network) but are not yet included in the blockchain (unconfirmed), meaning that any invalid transaction is prohibited from being added to the "mempool".

Additionally, the "mempool" component also communicates with the miner component to complete the mining functionality of Bitcoin Core. The miner is mainly responsible for generating the block that aims to be appended to the blockchain (the public ledger). To do so, it will retrieve the validated transactions from the "mempool" component, and send them to the validation engine component for the second round of validation (a more thorough validation to ensure that the transactions meet various criteria, such as that the inputs are valid and unspent, the output values are greater than or equal to zero, and the total transaction size is within the limits of the network, etc). Then, the transactions that past the second round of validation will be aggregated in the candidate block created by the miner component. Meanwhile, the miner component will also attempt to determine a solution to the Proof-of-Work algorithm (technically, it is trying to compute the hash of this candidate block's header and sees if the hash value is smaller than the current target) that is required to validate the candidate block. Moreover, due to the significant computing power the miner requires, the miner component can be placed in a different machine that runs a specialized computer-hardware system designed to mine bitcoin like "mining rig", and this specialized mining hardware is directly connected to a server that runs a full Bitcoin Core node.

As for the last key component in Bitcoin Core, the connection manager is indispensable in the Bitcoin P2P Network as it is responsible for discovering and connecting to the other nodes in the network to receive and propagate the transactions and blocks. Due to the nature of the Peer-to-Peer architecture style, the connection manager component is related to a local storage space to preserve the information (like the IP addresses) of a subset of active peer nodes in the network. As a result, when a validated transaction or a mined block needs to be broadcasted to the network, the connection manager will retrieve the information of known peer nodes from the storage and establish the connections with those peer nodes to complete the first iteration of broadcasting. Then, the connection managers of those peer nodes receiving the broadcast will connect to the peer nodes by using the information they stored in their local spaces to complete the broadcasting iteratively until the transaction or the block is rippled through the whole Bitcoin P2P Network.

Overall, to enable users to complete various requests remotely by using the RPC to call the corresponding components of the system synchronously with guaranteed responses from those components (unlike the Publish-Subscribe style where the call for services can be asynchronous and not be replied), the Bitcoin Core can be considered as a software developed using the Client-Server architecture style, in which a Peer-to-Peer architecture style network is integrated to achieve the decentralized trust and consensus mechanism that underpins bitcoin's security and efficiency feature. Moreover, due to the ease of adding new components or upgrading the existing components the Client-Server software architecture possesses, a high modifiability of the system is obtained to support possible future changes in the system such as adding one more miner component or updating the current validation engine.

3.2 Use case

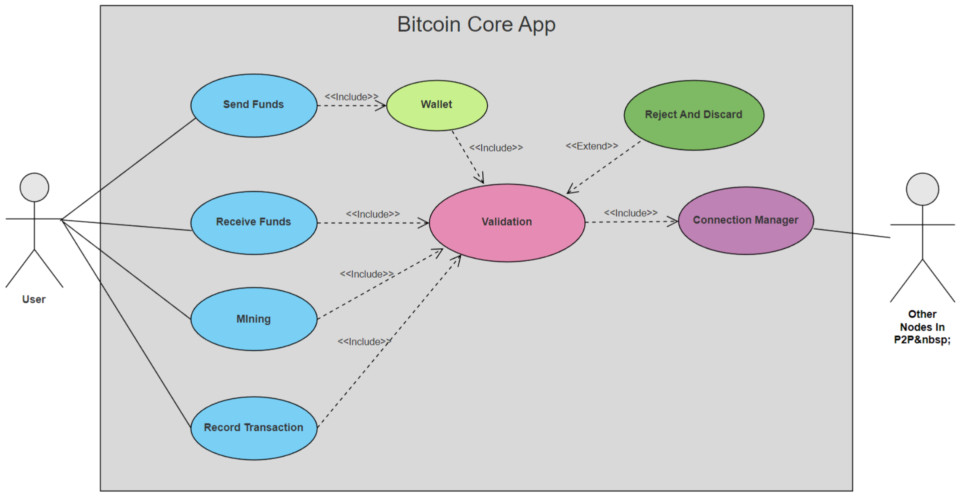

Figure 2: Bitcoin Core Use Case Diagram

The Bitcoin core use case diagram is shown in Figure 2. This use case diagram simply explains what the bitcoin core system can accomplish. As can be seen in Figure 2, we have four base cases, which are send funds, receive funds, mining and record transaction. Then the send funds case has an included case called wallet, which is used to store and manage user's balance. The four cases of wallet, receive funds, mining and record transaction have a common included case – validation. Validation is performed on both individual transactions and blocks of transactions that are received from other nodes in the Bitcoin P2P network, as well as on blocks of transactions created by the "mining" component. Validation has an extended case reject and discard transaction and included Connection manager. And the Secondary actor is other nodes in P2P network.

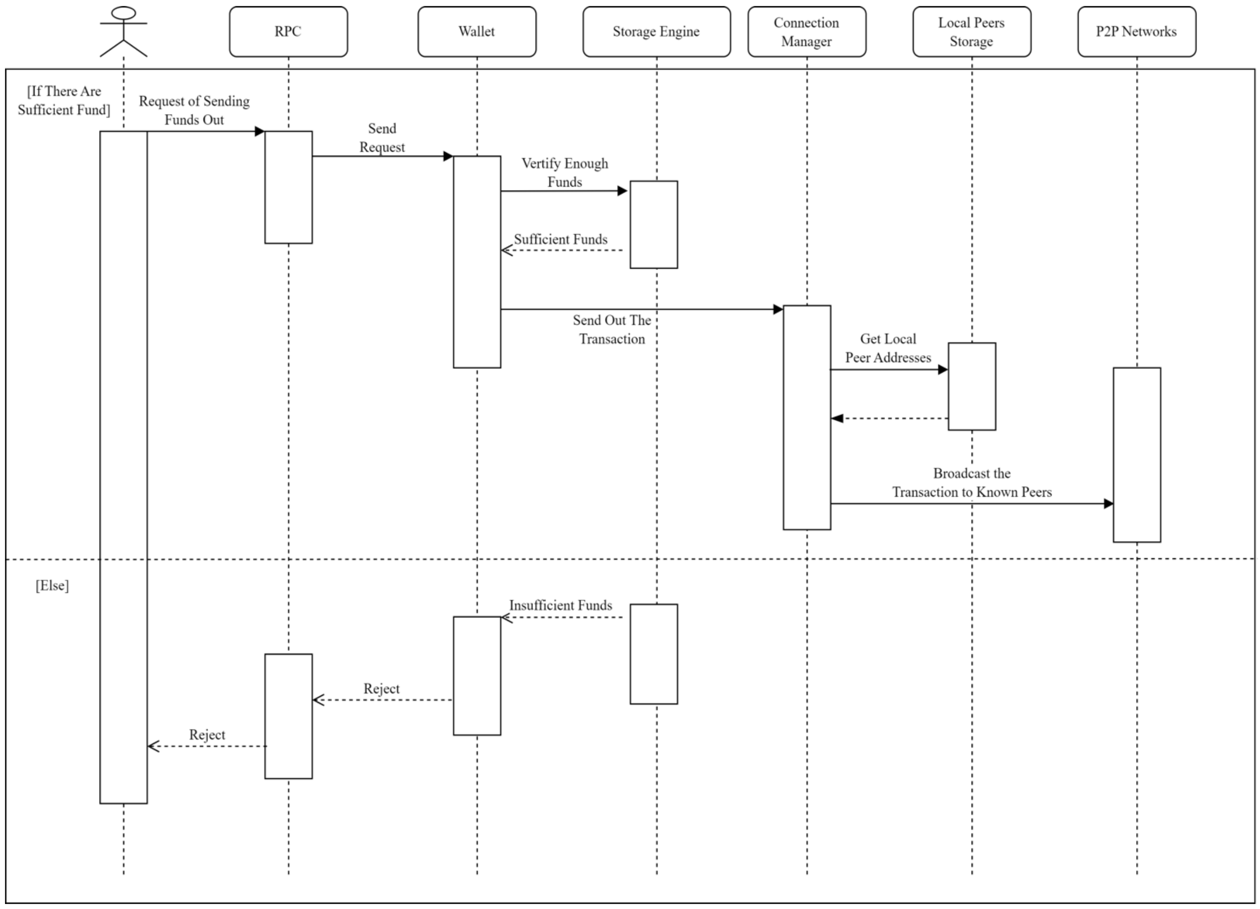

Figure 3: Sequence Diagram of sending funds

This Sequence Diagram simply explains the process of sending funds. As can be seen in Figure 3, User access RPC to request send funds. And RPC access Wallet, then Wallet accesses the storage manager to check if there are sufficient funds for the user, if it sufficient, then proceed, otherwise reject. Then, if sufficient fund, wallet access the connection manager to send out the transaction. The connection manager will broadcast the transaction to the local peers that are stored in the local list of peers.

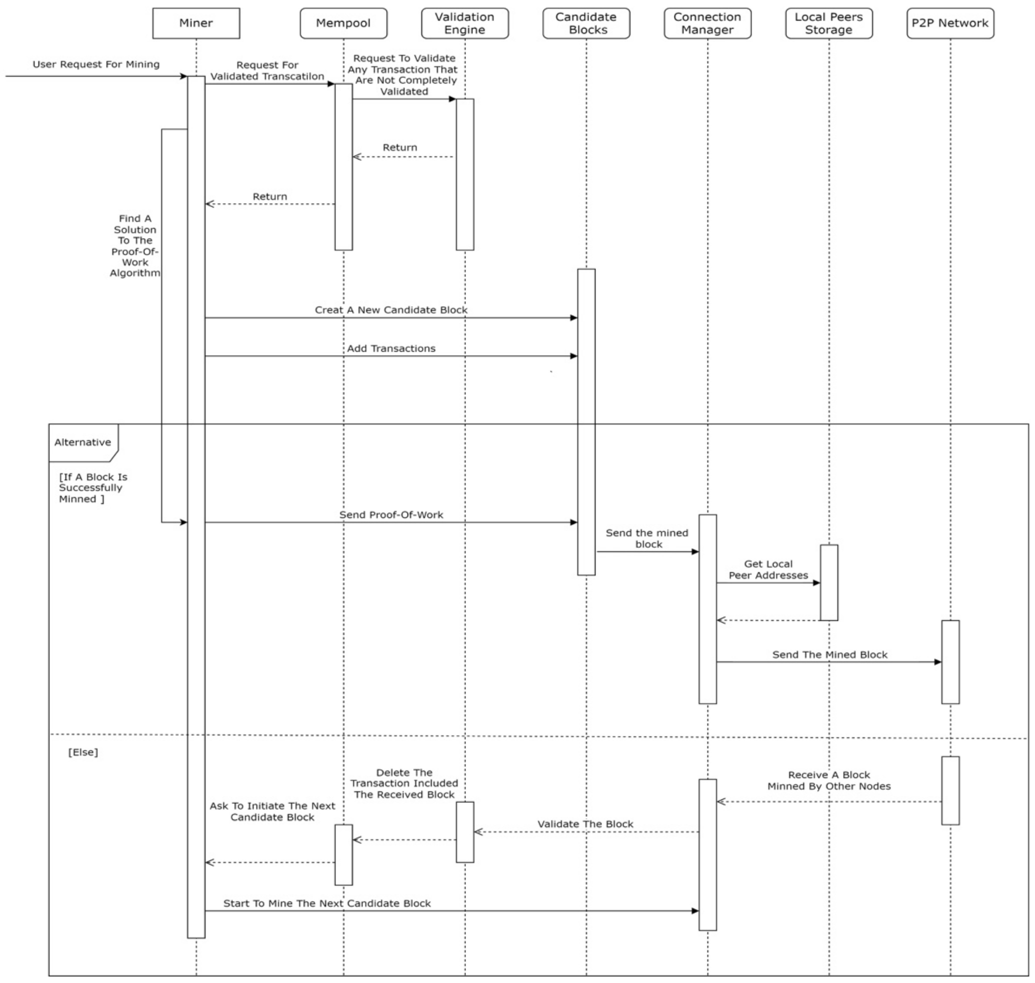
Miner
Mempool
Validation Engine
Candidate Blocks
Connection Manager
Local Peers Storage
P2P Network
User Request For Mining
Request For Validated Transcatiion
Request To Validate Any Transaction That Are Not Completely Validated
Return
Return
Find A Solution To The Proof-Of-Work Algorithm
Creat A New Candidate Block
Add Transactions
Alternative
[If A Block Is Successfully Minned]
Send Proof-Of-Work
Send the mined block
Get Local Peer Addresses
Send The Mined Block
[Else]
Delete The Transaction Included The Received Block
Receive A Block Minned By Other Nodes
Ask To Initiate The Next Candidate Block
Validate The Block
Start To Mine The Next Candidate Block

Figure 4: Sequence Diagram of Mining

This Sequence Diagram simply explains the process of mining. As can be seen in Figure 4. First, the user sends a mining request to the miner, and then the miner sends request for validated transaction to mempool. Mempool will send request to validation engine to validate any transaction that are not completely validated. Then the validation engine will return the validated transaction to mempool, also the validated transaction will be sent by mempool to miner. Miner will create a new candidate block and add transactions to candidate blocks. Meanwhile, the miner is finding a solution to the proof-of-work algorithm. If a block is mined, the miner will send the proof-of-work to candidate blocks, and candidate blocks will send the mined block to connection manager. Then the connection manager will try to get the local peer addresses in local peer storage. If the local peer is found, its addresses will be returned to the connection manager, and connection manager will send the mined block to P2P network. Instead, if there is no block be mined, the P2P network will receive a block mined by other nodes, then send it to connection manager. Which the connection manager will ask the validation engine to validate the block, then the validation engine will ask mempool to delete the transaction including the received block, and the mempool will ask miner to initiate the next candidate block. Finally, the miner will start to mine the next candidate block to connection manager.

3.3 Evolution

Bitcoin Core is a free, open-source software project that has been evolving since it was first released in 2009. It has undergone numerous iterations, with each version adding new features, improvements, and bug fixes. As of February 2023, the current version of Bitcoin Core is 0.22.1. It's worth noting that sometimes iterations don't result in an increase in version number, which means that Bitcoin Core has gone through an uncountable number of evolutions.

The following describes some of the previous versions with significant improvements, which helps to understand Bitcoin Core as a whole:

Version 0.8: This 2013 release contains several performance improvements, including using the LevelDB database to store transactions and block indexes and implementing optimizations for transaction validation by using Pieter Wuille, which increases the speed and capacity of the network.

Version 0.10: This 2015 release includes several security improvements: Changing signatures from OpenSSL to libsecp256k1 keep signatures constant and deterministic. OpenSSL signatures are more secure against anomaly attacks, and subnet matching for access control purposes now matches by checking binary network addresses instead of using string wildcards.

Version 0.13: This 2016 release introduces Segregated Witness (SegWit). It increases the Bitcoin network's capacity and enables low-level P2P changes by separating transaction signatures and transaction data. Specifically, when a waiting party receives a Feefilter message, the node will not send Invs for any transaction that does not match the filter rate.

Version 0.18: This 2019 release includes several improvements to the user interface and user experience, as well as the introduction of the Partially Signed Bitcoin Transaction (PSBT) format, which makes it easier to create and sign transactions offline, and the unique addition of a new bitcoin-wallet component tool.

Version 0.21: This 2021 release includes the introduction of the Taproot upgrade, which improves privacy and reduces transaction fees by allowing more complex smart contracts to be executed on the Bitcoin network.

Version 0.22: This version is the most current release, which has been updated to enable the RPC server to handle a limited number of concurrent RPC requests.

Overall, the development of Bitcoin Core has focused on improving performance, security, and functionality while maintaining the decentralized, peer-to-peer nature of the Bitcoin network, which mirrors the peer-to-peer software architecture model of the software.

3.4 Control and Flow of Data

In Bitcoin Core, the control flow has a C++ language implementation of the software source code management. The program's purpose is to ensure that new transactions are validated and added to the blockchain in a secure and decentralized manner. More importantly, Bitcoin Core is a decentralized system, which means that every component of the system interacts with each other in both directions.

Bitcoin core acts as a P2P network architecture where each application can interact with the other. As shown in the figure below, each app can be considered a node. When a node propagates a transaction to node l, node1 itself verifies it. When the verification is passed, node1 finds local peers to store the transactions. More importantly, Node1 also sends transactions to other nodes. Nodes sending transactions to each other also show that each node can interact with each other in the P2P architecture.

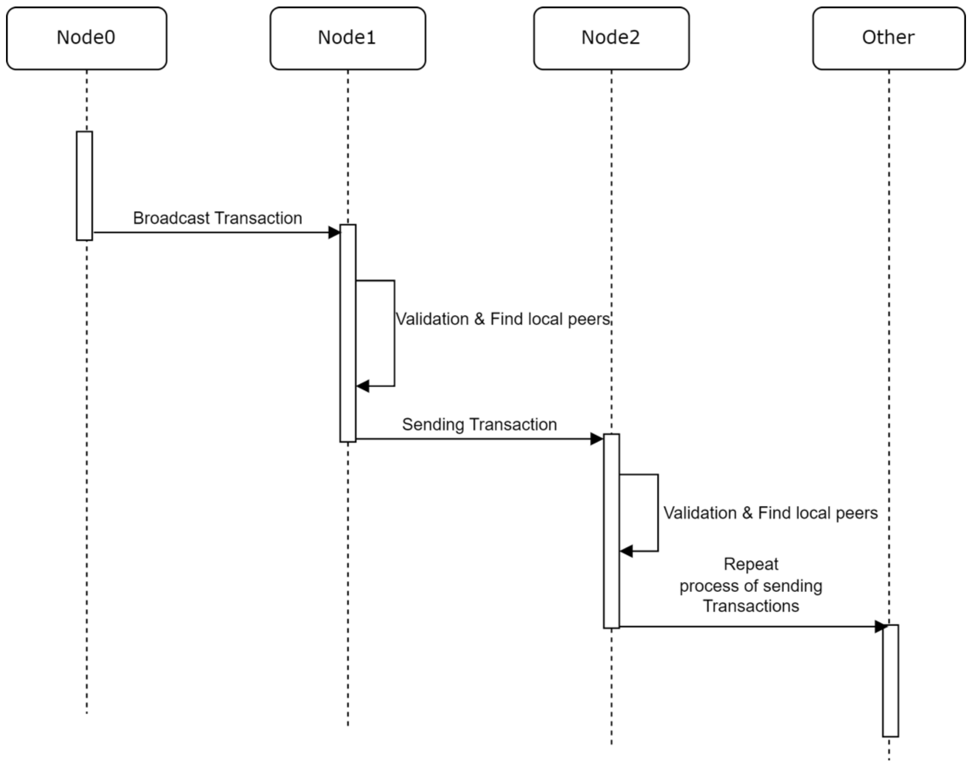
Node0
Node1
Node2
Other
Broadcast Transaction
Validation & Find local peers
Sending Transaction
Validation & Find local peers
Repeat process of sending Transactions

Figure 5: Data Flow Between Nodes in P2P Network

The following is a brief description of how the data flow works when a transaction is received with the followed diagram: When the Connection Manager of an individual receives a transaction request from other peers in the P2P Network, the Connection Manager will send a validation request to the Validation Engine. When the Validation passes, the Validation Engine sends both the request and the transaction to the Memory pool for storage. It propagates the authenticated transaction to the Connection Manager through the Memory pool. When the local peer component returns the address, the Connection Manager packages all the transaction information and returns it to the P2P network. However, when the authentication fails, Validation rejects the transaction and terminates the process directly.

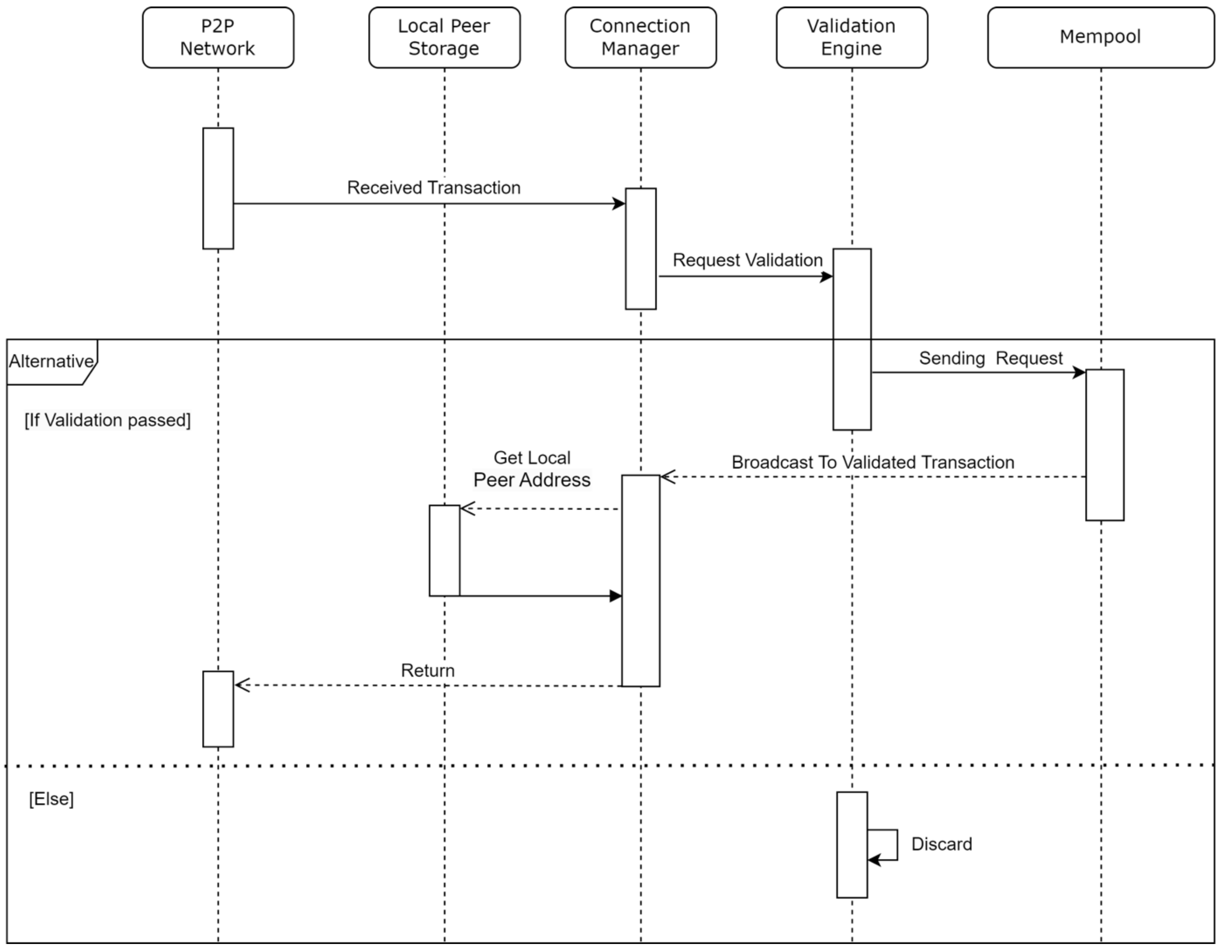

Figure 5: Data Flow When a Node Receives a Transaction Broadcast

3.5 Concurrency

Bitcoin Core has the duty of processing multiple operations at the same time, including the download of the blockchain, processing of new transactions and blocks, and responding to various events. Therefore, Bitcoin Core must consider a concurrency model that can not only enable multiple functions to run in parallel, but also ensure its operating efficiency. For this situation, multithreaded application is chosen by Bitcoin Core as the most suitable model.

A multithreaded application is a type of computer program that can perform multiple tasks concurrently, by dividing the workload among multiple threads. Therefore, multithreaded application is able to assign specific jobs to individual threads within the process and the threads communicate, through various means, to synchronize their actions. In multithreaded applications, concurrency issues which can be raised are race conditions and deadlocks. These two issues could happen when accessing shared resources without proper synchronization. To avoid these issues, the following techniques are used by Bitcoin Core:

Thread-local storage: Thread Local Storage (TLS) is a technique that enables individual threads within a multi-threaded process to reserve memory locations for storing data that is specific to each thread. The TLS API, which includes the TlsAlloc function, allows for the allocation of thread-specific data that can be bound to threads at run-time. The use of thread-local storage ensures Bitcoin Core to maintain that each thread has its own copy of certain data structures.

Locks: Locks are utilized to protect a shared data variable, such as the account balance illustrated in the example. If a lock object is used to guard all accesses to a data variable by surrounding it with a synchronized block, the accesses will be guaranteed to be atomic, meaning that they will not be interrupted by other threads. Bitcoin Core uses locks to ensure that only one thread can access a shared resource at a time.

Atomic operations: Atomic operations are a series of instructions that ensure that the access and updates to single word shared variables are done atomically. This implies that while atomic operations may not safeguard access to complex data structures in the same way that locks do, they offer a highly effective means of making sure that access to a single word is serialized. Bitcoin Core uses atomic operations to perform thread-safe updates to shared resources.

4.0 Developer contributes

Bitcoin Core is an open-source platform that determines whether blockchains have legitimate transactions. It contains a wallet, miner, full Blockchain database, and network routing node on the bitcoin P2P network. The platform provides highly protected transactions, exclusive privacy features, and high compatibility multifunctional wallet.

A full development system, according to Simon Sword's paper, should be divided into the technical lead, software developers, and software testing. Besides, as Bitcoin Core is an open-source project hosted on the GitHub platform, the administrators and cooperating contributors are both stakeholders. The technical lead is in charge of turning business needs into technological solutions for the development team leader, the technical lead is also responsible for creating and enforcing software development team standards and procedures. The software developer is subdivided into two parts: front-end and back-end, both of which are accountable; the software developers are responsible for creating cost and timing estimates based on the technical lead's technical requirements, as well as developing deliverables and conveying the software project status to the technical lead or project manager. Furthermore, the software test assures that the software solution meets business requirements and is free of bugs, errors, and defects. Finally, clear and effective communication is essential for project success. Identifying the stakeholders and their responsibilities is a vital component of this communication.

5.0 Lessons Learned

1) Learnings on Blockchain

a. Blockchain technology is a sophisticated database mechanism that enables the transparent exchange of information across a corporate network. A blockchain database holds information in blocks connected in a chain.

b. The data is temporally consistent since the chain cannot be deleted or altered without network consensus. Consequently, users may establish an unalterable or immutable ledger for recording orders, payments, accounts, and other transactions using blockchain technology.

c. The system is equipped with features to prohibit illegal transaction submissions and ensure consistency in the shared view of these transactions.

2) Learnings on Bitcoin P2P Network

a. For a stable and reliable P2P Bitcoin blockchain network, substantial users/nodes are required to be the full nodes to maintain the validated public ledgers.

b. Although all nodes in the bitcoin P2P network are equal, their responsibilities may vary based on the functionality they serve. A bitcoin node is a set of functions, including routing, blockchain database storage, mining, and wallet services.

3) Learnings on Teamwork

Although we are not familiar with each other before forming this team, the sense of responsibility still made us deploy tasks and search dataset to finish our project. Thanks to our leader, he encouraged us to communicate with each other face to face to bring more effort, we can be clear for direction and methods of individual work and solve the problems collectively. Everyone felt passionate about this project so that we completed it efficiently.

6.0 Conclusion

In conclusion, Bitcoin Core is a software that follows the rules of the Bitcoin protocol and operates as a full node in the Bitcoin P2P network. It is developed using the client-server architecture style, in which a Peer-to-Peer architecture style network is integrated and collaborated. The software comprises various modules, including the wallet, miner, full blockchain database, and network routing component, to complete various operations such as transaction creation, transaction broadcasting, transaction validation, and mining.

Concurrency is also a critical aspect of Bitcoin Core's operation. The software chooses a multithreaded application model that assigns specific jobs to individual threads within the process and enables multiple tasks to run concurrently to avoid concurrency issues such as race conditions and deadlocks. In addition, the development of Bitcoin Core involves various stakeholders, including technical leads, software developers, software testers, and administrators and cooperating contributors. These stakeholders have different responsibilities in transferring business needs into technological solutions, as well as ensuring the quality and effectiveness of the software solution.

Overall, Bitcoin Core's complex transaction handling mechanisms and concurrency management techniques are critical to its effectiveness as a decentralized platform for secure online transactions.

7.0 Reference

Chapter 3: "*Bitcoin Core: The Reference Implementation*" · GitBook. (n.d.). https://cypherpunks-core.github.io/bitcoinbook/ch03.html#bitcoin_core_architecture. Accessed February 17, 2023

https://github.com/bitcoin/bitcoin/blob/master/doc/JSON-RPC-interface.md

Crypto Understanding. (2021, August 13). *Understanding Bitcoin Core: The Reference Implementation* [Video]. YouTube. https://www.youtube.com/watch?v=wLYdcH37phE

What Are the Bitcoin Layers? (n.d.). Bitget Academy. https://www.bitget.com/en/academy/article-details/What-are-the-bitcoin-layers Accessed February 18, 2023

Block Chain — Bitcoin. (n.d.). https://developer.bitcoin.org/devguide/block_chain.html

What is Blockchain Technology? - Blockchaining Explained - AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/what-is/blockchain/?nc1=h_ls&aws-products-all.sort-by=item.additionalFields.productNameLowercase&aws-products-all.sort-order=asc Accessed February 18, 2023

Swords, S. (2020, January 19). *Software Development Project Roles and Responsibilities.* Atlas Computer Systems Ltd. https://www.atlascode.com/blog/software-development-project-roles-and-responsibilities/ Accessed February 18, 2023

Antonopoulos, A. M. (n.d.). *Mastering Bitcoin.* O'Reilly Online Learning. https://www.oreilly.com/library/view/mastering-bitcoin/9781491902639/ch06.html Accessed February 18, 2023

Version History - Bitcoin Core. (n.d.). https://bitcoin.org/en/version-history Accessed February 18, 2023