Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/31/20230160

Analysis of concrete architecture of Bitcoin Core

Yiwen Zheng

School of Computing, Queen's University, Kingston, Ontario, Canada, K7L 3N6

19yz62@queensu.ca

Abstract. Bitcoin Core serves as the foundational software responsible for verifying and validating all transactions and blocks within the Bitcoin blockchain, hence upholding the security and integrity of the whole Bitcoin network. This study extensively examines the internal mechanisms, structure, and conceptual framework of Bitcoin Core. This study aims to conduct a comprehensive analysis of the Bitcoin Core architecture, with a focus on evaluating its capacity to fulfill the rigorous demands of a decentralized Bitcoin network. This study not only provides a comprehensive understanding of the essential elements that drive the Bitcoin network, but it also explores the numerous factors that have a substantial impact on the operational availability of Bitcoin nodes. All of these factors are essential for ensuring the efficient operation of the Bitcoin network, encompassing the physical environment, architectural designs of nodes, and maintenance requirements. This study provides a comprehensive analysis of the underlying mechanisms of Bitcoin, shedding light on its inherent robustness and the factors that contribute to its consistent performance within the ever-changing landscape of digital currencies. The objective of this endeavor is to furnish a detailed exposition of the mechanics of Bitcoin, so enhancing its prevalence and familiarity among individuals.

Keywords: bitcoin, bitcoin network, architectural design, blockchain.

1. Introduction

In January 2009, Bitcoin sprang out of nowhere, shaking up the financial world with a radical new concept: decentralized, secure money transfers outside the FIAT currency ecosystem. Nevertheless, the blockchain technologies underlying Bitcoin can be applied to a lot more than just digital currency. Public-key encryption, the proof-of-work principle, and peer-to-peer (P2P) technologies all come together in a blockchain, making it a distributed, verified database [1]. The goals of the Bitcoin ecosystem are what make it so complicated: they are that anyone should be able to add transactions to the Bitcoin blockchain and that there should be no one authority over the network [2]. Bitcoin's future success depends on a solid and trustworthy architecture that can accommodate its rapidly expanding user base and ever-increasing volume of transactions. The usage of concrete in the creation of the network's nodes is an essential part of Bitcoin's design. In this paper, the author will examine the Bitcoin core's implementation details and assess how well they serve the requirements of the network as a whole. The climatic conditions, structural design, and maintenance needs that influence the longevity and reliability of concrete in Bitcoin nodes will also be investigated in this paper. It intends to increase the resilience and lifespan of this crucial infrastructure by conducting a thorough investigation of the Bitcoin network's concrete design.

DOI: 10.54254/2755-2721/31/20230160

2. Top-level subsystems

Based on our prior studies, we know that Bitcoin Core consists of 8 distinct subsystems and that its primary architectural style is a hybrid of the Peer-to-Peer and Layer styles. However, after reviewing the code, we realized that the Layer style needed to be replaced with the client-server style, and we identified two new subsystems: API & RPC and crypto. We also reworked or renamed several of the older subsystems. For example, Node was renamed to Node networking to make clear that its purpose is to facilitate decentralized and peer-to-peer communication between nodes in the Bitcoin network. To better reflect its expanded role in storing Bitcoin-related data, the term "blockchain" was rebranded as "storage," as depicted in Figure 1.

First, API & RPC. These Bitcoin Core subsystems allow programmatic access to the underlying infrastructure. Because of this, programmers can create their own applications and services on top of Bitcoin Core without having to code every feature from scratch. Second, crypto. Encoding and decoding, as well as other cryptographic primitives, can be found in the Bitcoin network's Crypto subsystem. These algorithms cover things like digital signatures, encryption, and hashing. Despite its former position as a subsystem under Validation, this part of Bitcoin Core is now regarded vital because of the work it does to reduce Validation's load by making use of library functions and making the network more efficient.

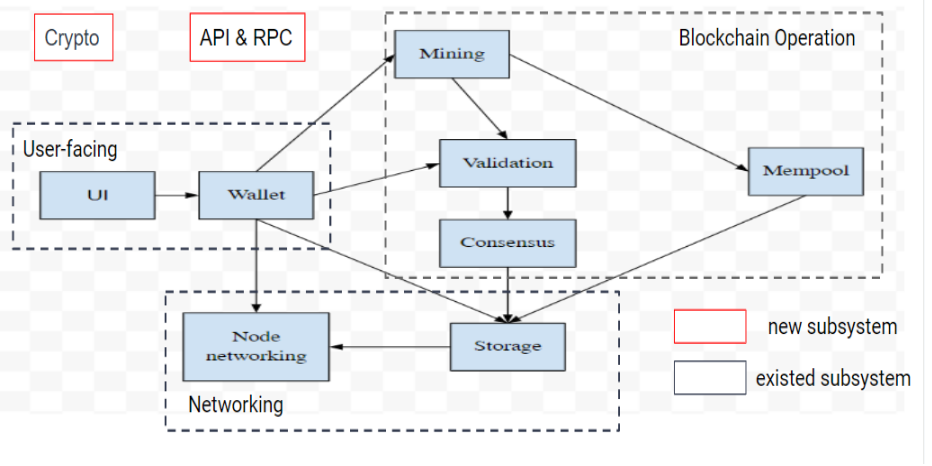

Figure 1. Top-level conceptual architecture [3].

2.1. Concrete architecture

Figure 2 depicts the categorization of our ten subsystems into the five categories of user interface, developer interface, blockchain operations, infrastructure, and security. The parts that the user interacts with are called "user-facing," whereas the parts that the developer interacts with are called "developer-facing." All actions performed on a blockchain are referred to as "Blockchain Operations," while "Networking" refers to the sharing of information between computers. To conclude, Bitcoin's network is protected by Security.

DOI: 10.54254/2755-2721/31/20230160

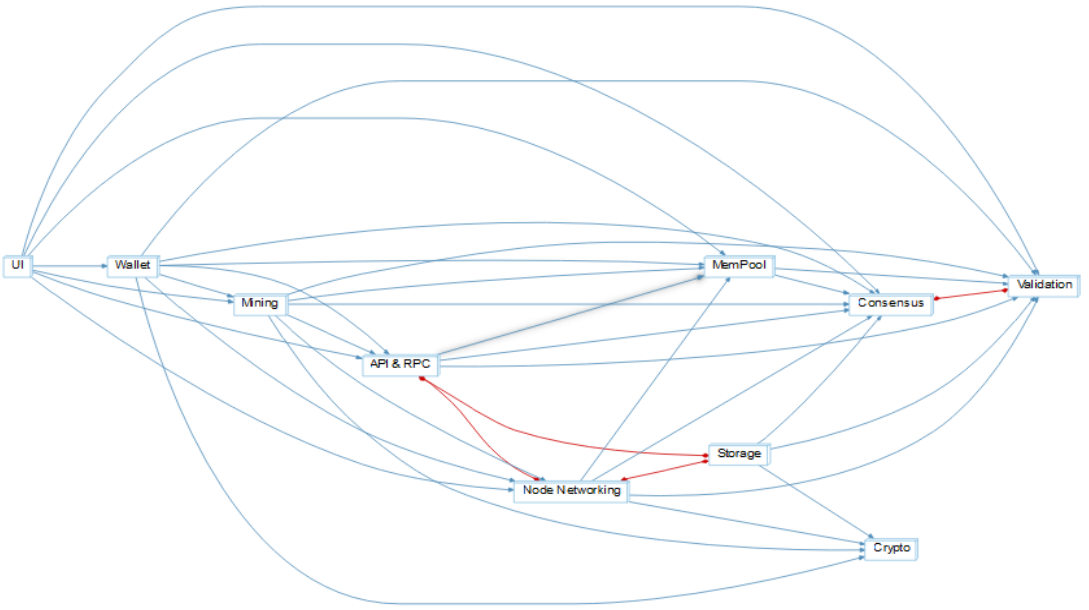

Figure 2. Top-level concrete architecture (by Understand) double arrow in red (Original).

2.2. Subsystem introduction

User-facing: The User Interface (UI) and Wallet subsystems are an integral part of Bitcoin Core's User Orientation group, which is responsible for facilitating user access to Bitcoin network resources. User interface features include the presentation of account balances, transaction histories, and other relevant data. It also allows users to make Bitcoin purchases, transmit Bitcoin to other users, and execute other Bitcoin-related tasks. Users' private keys and transaction activity are handled by the Wallet subsystem, which communicates with the UI. The Wallet subsystem generates and signs the required Bitcoin transaction whenever a user begins a transaction via the UI. The UI queries the database for the user's balance and transaction history, and requests transactions and signatures from the user.

Developer-facing: The Bitcoin Core API and RPC subsystems are part of this group because they allow for programmable interaction between Bitcoin Core and developers. Developers can leverage the API & RPC subsystem to build on top of Bitcoin Core by making use of the exposed functions and data structures. By processing requests and providing responses to third-party users, this component facilitates communication between other modules.

As for blockchain operations, Mining, Validation, Consensus, and Mempool are all parts of this larger system. New blocks are created and added to the blockchain by the mining subsystem, while the validating subsystem ensures that all transactions and blocks are legitimate. All network nodes must be in agreement about the blockchain's current state, and that's what the consensus subsystem is for. Transactions that have not yet been confirmed are stored in a pool called the mempool. All of these components collaborate to guarantee that new transactions are added to the blockchain without compromising its security. In terms of network components, the Node Storage and Networking components are a part of this larger category. Communication between Bitcoin nodes is handled by the node networking subsystem, which relays data about pending transactions and blocks to other parts of the network for processing. Data is written to disk and read back in by the storage subsystem as needed. To guarantee that the Bitcoin network is always running smoothly, these auxiliary systems collaborate.

Lastly, as for security, among these is the Crypto subsystem, which performs the essential cryptographic duties for protecting the Bitcoin network. Digital signatures, encryption, and hashing are all examples of such operations. The cryptographic libraries subsystem ensures that the other subsystems may carry out their duties in a safe and secure manner by providing them with cryptographic functions.

DOI: 10.54254/2755-2721/31/20230160

The validation module, for instance, checks digital signatures of transactions and blocks with the use of cryptographic methods.

2.3. To develop a high-level architecture

The purpose of Reflexion Analysis is to compare the conceptual architecture to the concrete architecture and look for missing or unexpected dependencies. We can learn more about the system and tailor our changes to fit its design by focusing on these differences.

UI ⇒ Validation: The UI subsystem depends on the Validation Subsystem for checking the validity of user input before submitting it to the Bitcoin network. UI ⇒ Consensus: The UI subsystem might rely on the Consensus subsystem for displaying information about the current state of the blockchain or enforcing consensus rules. UI ⇒ Mempool: The UI subsystem might depend on the Mempool subsystem for displaying information about unconfirmed transactions or submitting new transactions to the mempool. UI ⇒ Mining: The UI subsystem might rely on the Mining subsystem for displaying mining-related information, controlling mining operations, or monitoring the mining process. UI ⇒ API & RPC: Aiming to interact with or controlling the graphical user interface programmatically, enabling developers to create custom user interfaces or scripts that interact with the Bitcoin Core software. UI ⇒ Node Networking: The UI subsystem might rely on the Node Networking subsystem for displaying network-related information, controlling node connections, or monitoring the network state. UI ⇒ Storage: The UI subsystem might depend on the Storage subsystem for accessing transaction history, block data, or other stored information to display it to the user. UI ⇒ Crypto: The UI subsystem might rely on the Crypto subsystem for displaying information about the cryptographic primitives used in the Bitcoin Core software or for allowing users to perform cryptographic operations.

Wallet ⇒ Consensus: The Wallet subsystem has a direct dependency on the Consensus subsystem in the concrete architecture, which was not shown in the conceptual architecture. This dependency is due to the fact that the Wallet subsystem needs to ensure that transactions are executed in accordance with the consensus rules before they are executed. Wallet ⇒ Mining: The Mining subsystem depends on the Wallet subsystem for updating balances and transaction history when new blocks are mined and added to the blockchain. Wallet ⇒ API & RPC: The conceptual architecture may not have anticipated a direct dependency between the API & RPC and Wallet subsystems. However, the API & RPC subsystem might directly interact with the Wallet subsystem to provide developers with access to wallet-related operations, such as creating new addresses, signing transactions, or managing keys. Wallet ⇒ Consensus: The conceptual architecture might not have anticipated a direct dependency between the Consensus and Wallet subsystems. But the Consensus subsystem might interact with the Wallet subsystem to update the user's balance or transaction history since new blocks are added to the blockchain.

Mining ⇒ Consensus: The Mining subsystem has a direct dependency on the Consensus subsystem in the concrete architecture, but in the conceptual architecture, this function does not show. This dependency is because the Mining subsystem needs to ensure that the transactions it includes in the block are executed in accordance with the consensus rules. Mining ⇒ Storage: The Mining subsystem might be required by the Storagesubsystem for retrieving transaction data, block data, or other information necessary for generating new blocks and adding them to the blockchain. Mining ⇒ API & RPC: In the conceptual architecture, there might not be an expected dependency between the Mining and API & RPC subsystems. But we find that the Mining subsystem could interact with the API & RPC subsystem to provide developers with access to mining-related operations, such as starting or stopping mining or adjusting mining parameters. Mining ⇒ Node Networking: The Mining subsystem depends on the Node Networkingsubsystem as it is responsible for sharing newly mined blocks with other nodes in the network. When the Mining subsystem successfully mines a new block, the Node Networking subsystem broadcasts the new block to other nodes to update their local copies of the blockchain. Mining ⇒ Crypto: The Mining subsystem depends on the Mining subsystem for verifying proof-of-work, hashing block headers, or generating block hashes.

Storage ⇒ Consensus: The Mining subsystem has a direct dependency on the Consensussubsystem in the concrete architecture, but not in the conceptual architecture. This is because the Storage subsystem

DOI: 10.54254/2755-2721/31/20230160

needs to ensure that the information it stores about transactions and blocks is consistent with the consensus rules. Validation ⇒ MemPool: The Validation subsystem has a dependency on the MemPool subsystem in the concrete architecture. This dependency is because the Validation subsystem needs to check the validity of transactions before they are added to the Mempool.

API & RPC ⇒ Mempool: The API & RPC subsystem rely on the Mempool subsystem for accessing unconfirmed transactions, submitting new transactions to the mempool, or querying the mempool for specific transactions. API & RPC ⇒ Consensus: The API & RPCsubsystem depend on the Consensus subsystem for accessing block data, querying the state of the blockchain, or enforcing consensus rules programmatically. API & RPC ⇒ Validation: For checking the validity of transactions or blocks, or retrieving information about the validation status of transactions. API & RPC ⇔ Node Networking: For monitoring and controlling network-related operations, like connecting or disconnecting from peers, fetching network statistics, or broadcasting transactions. API & RPC ⇔ Storage: For accessing and managing data stored on disk, such as transaction history, block data, or other stored information.

Node Networking ⇒ Mempool: The Node Networking subsystem depend on the Mempool Subsystem for sharing unconfirmed transactions with other nodes, receiving new transactions from other nodes, or synchronizing the mempool state between nodes. Node Networking ⇒ Consensus: We can refer that the conceptual architecture might not have anticipated a direct dependency between the Consensus and Node Networking subsystems but the Consensus subsystem might be different. The Consensus subsystem could interact with the Node Networking subsystem to disseminate new blocks and receive updates about the blockchain state from other nodes in the network. Node Networking ⇒ Validation: The Node Networking subsystem have an unexpected dependency on the Validation subsystem. This might be because the Validation subsystem needs to communicate with other nodes to request missing data or verify the validity of certain transactions or blocks by comparing them with information held by other nodes in the network. Node Networking ⇒ Crypto: For encrypting and decrypting data transmitted between nodes, or for securing node identity using cryptographic keys. Node Networking ⇔ Storage: The Storage subsystem depend on Node Networking for synchronizing stored data with other nodes in the network, such as updating the blockchain with new blocks or fetching missing transaction data.For encrypting sensitive data stored on disk, such as private keys, or for hashing data when indexing stored information.

Storage ⇒ Consensus: The Storage subsystem might rely on the Crypto subsystem for encrypting sensitive data stored on disk (e.g., private keys) or for hashing data when indexing stored information. Storage ⇒ Validation: In the conceptual architecture, there might not be an expected dependency between the Storage and Validation subsystems. Actually, the Storage subsystem could depend on the Validation subsystem to ensure that only valid transaction and block data are stored on disk. Meanwhile, the Validation subsystem might need to access the Storage subsystem to retrieve historical transaction data for validation purposes. Storage ⇒ Crypto: The Storage subsystem might rely on the Crypto subsystem for encrypting sensitive data stored on disk (e.g., private keys) or for hashing data when indexing stored information.

Consensus ⇔ Validation: the Consensus subsystem is responsible for determining the rules that govern the addition of new blocks and transactions to the blockchain. It relies on the Validation subsystem to verify whether transactions and blocks conform to those rules.

DOI: 10.54254/2755-2721/31/20230160

3. Wallet — secondary subsystem conceptual and concrete views

3.1. Conceptual view of wallet module

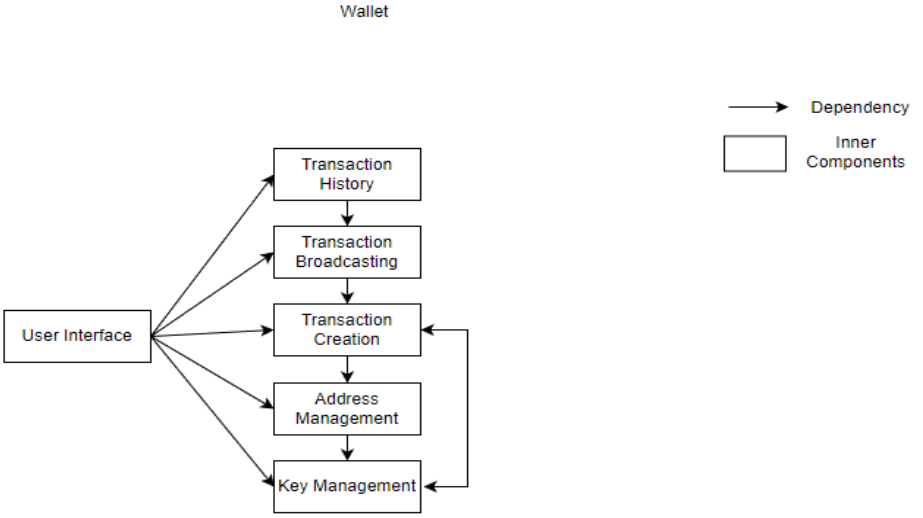

Figure 3. Conceptual view of wallet [5].

Bitcoin is the first currency system that is both decentralized and immune to the whims of the monetary authorities [4]. Figure 3 depicts the five fundamental components that make up the Bitcoin wallet's foundation. Create Transaction, Manage Keys, Manage Addresses, Send Transaction Broadcasts, and Track Transaction History are all Modules. Figure 2 shows how various parts of the architecture work together to efficiently and reliably manage Bitcoin network transactions.

When we first set out to learn about Bitcoin wallets, we thought that key and address management were the most important features. Therefore, we zeroed emphasis on these parts and their connections, going into depth about how they work together to keep a user's digital possessions safe and organized.

1)Transaction Generation: This component initiates and verifies new transactions. It takes information from the user, such as the address of the receiver and the amount to be sent, and uses the available UTXOs to create a legitimate transaction. Users then use their private keys to sign the transaction.

2)Key Management: This component is responsible for the creation, storage, and administration of the user's public and private keys. It protects users' private keys and makes public keys readily available for use in address generation and transaction signing.

3) Address Management: Users' Bitcoin addresses, which are obtained from their public keys, are managed via the Address Management module. Each created address is tracked together with its transaction history and current balance.

4)Transaction Broadcasting: This component sends a transaction to the Bitcoin network after it has been created and signed. It guarantees that the transaction will be broadcast to other network nodes and confirmed by miners when they are added to a block. In Bitcoin, a block of accepted transactions is added to the block chain and broadcast to all of the other nodes every 10 minutes, on average [6].

5) Transaction History: A wallet's transaction history is a record of all payments sent to and received from a given address. Dates, amounts, and addresses, as well as the ability to filter and search through one's transaction history, should be readily available to users.

These parts collaborate to give Bitcoin wallet users a safe and streamlined way to access the network. Each part of the wallet is crucial to its overall function, as it ensures the safety and security of the wallet and facilitates the processing of transactions.

DOI: 10.54254/2755-2721/31/20230160

3.2. Concrete view of wallet

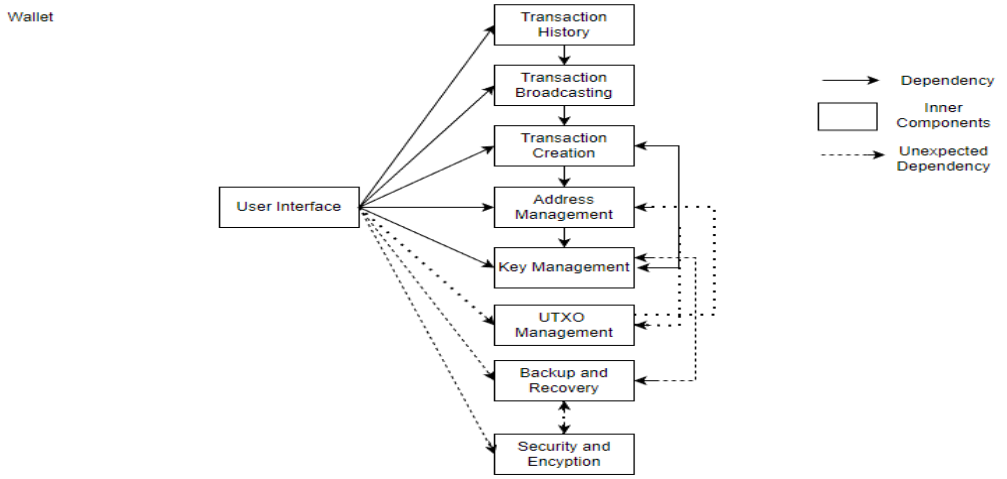

Figure 4. Concrete view of wallet [7].

For this purpose, we dove into the Wallet subsystem, finding the source code and analyzing it to better understand the dependencies inside the subsystem and obtain its concrete architecture. So, we get three new parts that weren't in the conceptual view; figure 4 shows in fine detail how the internal modules interact with one another and how data flows between them. There are three additional parts visible in Figure 4:

First, the UTXO management module: UTXOs are unsplittable Bitcoins that are tied to a certain owner and stored in the blockchain [7]. This feature ensures the integrity of the user's unspent transaction outputs (UTXOs) by keeping track of incoming transactions, updating the UTXO set, and making use of UTXOs when necessary. Relevant source files in Bitcoin Core include “wallet/coinselection.h”, “wallet/coinselection.cpp”, “wallet/wallet.h”, and “wallet/wallet.cpp”.

Second, backup and recovery module: This module provides backup and recovery mechanisms, such as seed phrases, encrypted backups, or hardware wallet integrations. Key source files in Bitcoin Core include “wallet/wallet.h”, “wallet/wallet.cpp”, “wallet/walletdb.h”, and “wallet/walletdb.cpp”.

Third, security and encryption module: This component ensures the security and privacy of the user's keys and data, with features such as wallet file encryption, secure storage mechanisms, password protection, and two-factor authentication. Key source files in Bitcoin Core include “wallet/crypter.h”, “wallet/crypter.cpp”, “wallet/wallet.h”, and “wallet/wallet.cpp”.

3.3. Reflexion analysis performed for the architecture of the chosen 2nd level subsystem

User Interface ⇒UTXO Management: For proper management of unutilized transaction outputs (UTXOs), the User Interface submodule depends on the UTXO Management submodule. Users need to see their accessible UTXOs on the wallet's UI so they may make transactions based on their actual balances.

User Interface ⇒Backup and Recovery: The Backup and Recovery submodule is essential to the User Interface submodule so that wallet data may be backed up and restored in an intuitive manner. In the event of device loss, damage, or theft, users must be able to quickly and easily restore their backups and retrieve their cash.

User Interface ⇒Security and Encryption: When it comes to keeping private information and financial transactions safe in a user's wallet, the User Interface submodule relies heavily on the Security and Encryption submodule. To guarantee the safety of the wallet as a whole, it is necessary to incorporate secure encryption technologies and extra security measures within the user interface.

DOI: 10.54254/2755-2721/31/20230160

UTXO Management ⇒Address Management: The Address Management submodule is necessary for the UTXO Management submodule to locate and record UTXOs that correspond to wallet addresses. Optimization of transaction fees and user access to their funds and privacy depend on accurate UTXO tracking.

Transaction Creation⇒UTXO Management: When making new transactions, the Transaction Creation submodule consults the UTXO Management submodule to determine which UTXOs to use. To create new transactions, UTXO Management first determines which unspent transaction outputs are accessible.

Backup and Recovery⇔Key Management: There is a two-way reliance between the Backup and Recovery and Key Management modules. The generation of mnemonic seed phrases or encrypted backups for secure wallet restoration is dependent on Key Management. To ensure that users can restore their wallets and access their cash in the event of a disaster, Key Management relies on Backup and Recovery to provide a user-friendly solution for backing up and restoring private keys.

4. Use cases

4.1. Use case 1: Peer to peer transaction

As shown in Figure 5, a Bitcoin transaction is initiated by the user using the UI by entering the recipient's address and the desired transfer amount. The wallet program then determines the appropriate inputs, outputs, and fees for the transaction. The wallet then communicates with the cryptographic libraries in order to sign the transaction with the user's private key. When the signature verification process is complete, the signed transaction is broadcast to the Bitcoin network using the API and RPC components.

It is the function of the Peer Discovery and Connection Management (PD & CM) component to forward the transaction to additional nodes as it moves through the network. The transaction is subsequently sent to the mempool, from which miners choose those that will be included in the following block. The mining process utilizes cryptographic libraries to carry out hashing operations and determine a legitimate Proof-of-Work (PoW) solution.

Following discovery of a PoW solution, mining software will submit the newly created block for validation. The validation subsystem uses cryptographic libraries to check the authenticity of transaction signatures and block data. If the block is mined and added to the blockchain successfully, the transaction is considered confirmed. The consensus algorithm then directs the PD & CM part to update the local blockchain and synchronize with other nodes, completing the Bitcoin transaction.

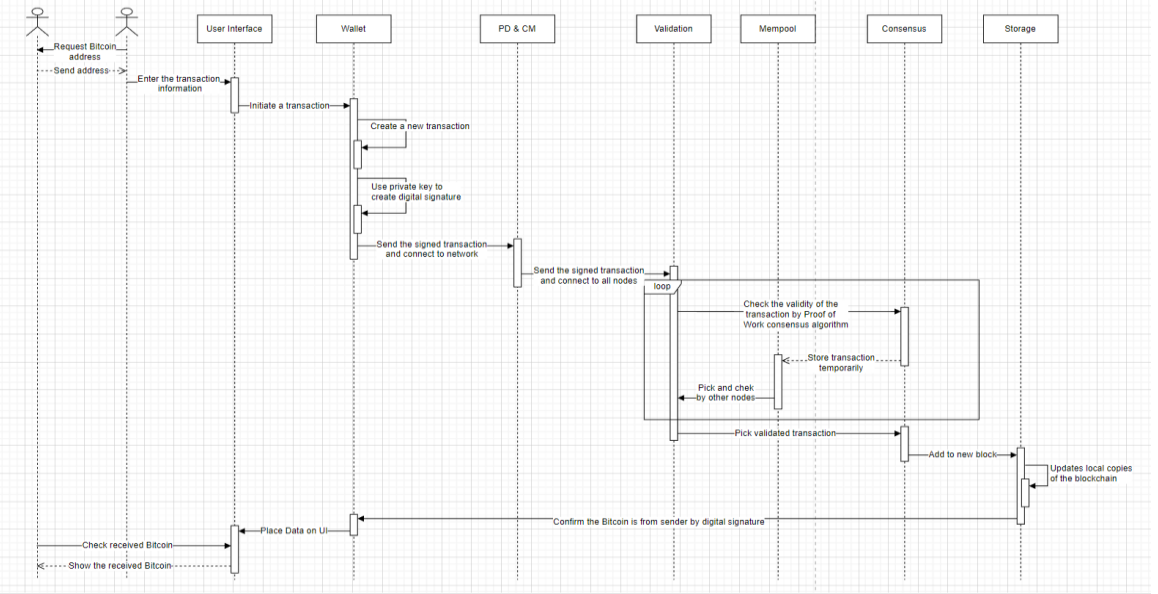

Figure 5. Use case 1: Peer to peer transaction.

DOI: 10.54254/2755-2721/31/20230160

4.2. Use case 2: Mining

The second use case explains how Bitcoin's core system handles mining, drawing attention to the interplay between the many Bitcoin subsystems and the miners who verify transactions and keep the blockchain secure (figure 6).

In order to begin interacting with the Bitcoin mining program, miners must first access the user interface (UI). The mining process begins with the Application Programming Interface and Remote Procedure Call (API and RPC) components finding Bitcoin network peers and connecting with them. Peer Discovery and Connection Management (PD & CM) is responsible for adding transactions to the mempool and retrieving the miner's wallet address, both of which are required for obtaining mining rewards.

Selecting transactions from the mempool, the mining software then sends a request to the consensus algorithm to validate the transaction threshold, yielding the corresponding results. Following their hashing with cryptographic libraries, these deals are sent back to the mining procedure. The subsequent step is for the transactions to be included in a new block and then transmitted for verification. The consensus algorithm is asked to review the block, and the results are then disseminated to other nodes via the PD & CM module. This procedure is repeated until the block is accepted. After the transaction has been verified, the block is sent to the consensus algorithm to be added to the blockchain, and local copies are then updated. At last, miners' wallets, which can be seen via the UI, receive their share of the mining earnings.

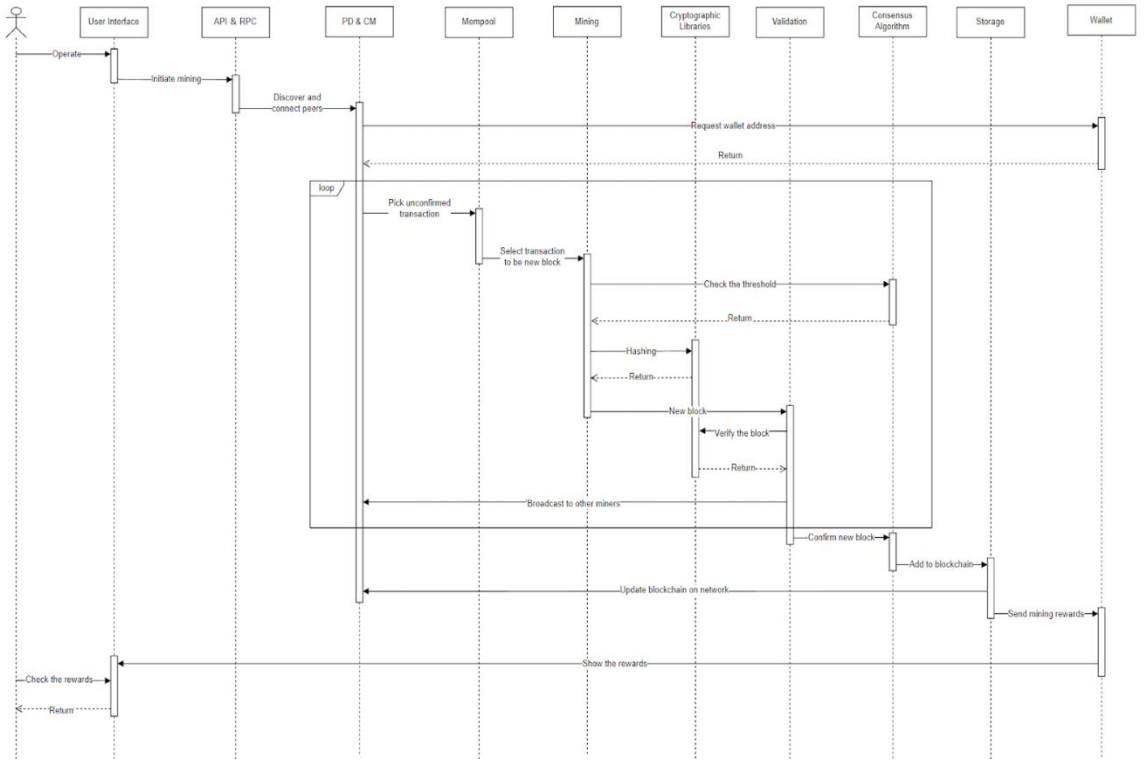

Figure 6. Use case 2: Mining.

5. Discussion

Our research into Bitcoin's design has taught us the significance of a solid foundation upon which to build a decentralized digital currency. We found that a client-server + Peer to Peer architecture is more suited for scalability, since it allows the network to be more adaptable to its surroundings and reduces the computing requirements for lightweight clients. Furthermore, we've witnessed the utility of Bitcoin

DOI: 10.54254/2755-2721/31/20230160

Core's subsystems for developers to interface with the system programmatically, allowing for the development of new applications and services without having to build all of the functionality themselves.

We also realized the importance of differentiating between the theoretical and practical design, as unforeseen relationships sometimes occur in practice. Source code documentation provides us with a more in-depth look at the architecture of the system, which is essential for decentralized and modular codebase analysis. For optimal transaction fees and user control over funds and privacy, accurate tracking of unspent transaction outputs (UTXOs) connected with wallet addresses is necessary.

Last but not least, we discovered how critical it is to communicate well and pay close attention to detail when conducting research and analysis. We included comments from earlier assignments, shifted to a client-server approach for our conceptual design, and strengthened our research by citing sources and numbering our figures. We think that by incorporating these learnings into our future work, we can help to ensure the long-term success of Bitcoin and similar decentralized digital currencies.

6. Conclusion

In conclusion, this research paper presents a thorough examination of the structural framework of Bitcoin Core's physical design. Through the implementation of the client-server + Peer to Peer software design, we have successfully enhanced the scalability and reduced the computational demands of lightweight clients. This achievement can be attributed to our unwavering commitment to this endeavor. Bitcoin's robustness is derived from a compilation of ten distinct subsystems, which can be classified into five overarching categories. To facilitate the ongoing development and study of the software, a reflection analysis was conducted to identify any dependencies that were either absent from the architecture or unanticipated. The paper additionally gives a case study on peer-to-peer transactions as a means to exemplify the practicality of the wallet subsystem. Furthermore, it conducts an analysis of both the abstract and tangible perspectives of the wallet subsystem.

References

[1] Bitstreams Endpoints, 2023. https://dl.gi.de/server/api/core/bitstreams/defdb872-e66c-4194-a45fe196e1c1b7a2/content

[2] Bitcoin Core Integration/staging tree, 2023. https://github.com/bitcoin/bitcoin

[3] M. T. Ghasr, M. J. Horst, M. R. Dvorsky and R. Zoughi, "Wideband Microwave Camera for Real-Time 3-D Imaging," in IEEE Transactions on Antennas and Propagation, vol. 65, no. 1, pp. 258-268, Jan. 2017, doi: 10.1109/TAP.2016.2630598.

[4] Bitcoin Core - P2P Digital Currency, 2022. https://doxygen.bitcoincore.org/dir_68267d1309a1af8e8297ef4c3efbcdba.html

[5] AKHIL GUPTA, AND RAKESH KUMAR JHAA. Survey of 5G Network: Architecture and Emerging Technologies, in IEEE Special Section on Recent Advances in Software Defined Networking for 5G Networks. 2015, Vol.24, 33-35.

[6] Febrero-Bande Manuel;González-Manteiga Wenceslao;Prallon Brenda;Saporito Yuri F. Functional classification of bitcoin addresses. Computational Statistics and Data Analysis. 2023, Vol. 181, 50-55.

[7] Duan Kun;Gao Yang;Mishra Tapas;Satchell Stephen. Efficiency dynamics across segmented Bitcoin Markets: Evidence from a decomposition strategy. Journal of International Financial Markets, Institutions & Money. 2023, Vol. 83, 101-103.