Concrete Architecture of Bitcoin Core & Wallets

CRYPTONITE – Group #16

Amanda Misek – 19aam18@queensu.ca

Benjamin Tiong – 19bkyt@queensu.ca

Bilal Imran – 20mbi@queensu.ca

Christian D'Souza – 20ceds@queensu.ca

Decheng Zhu (Eric) – 18dz6@queensu.ca

Vanshita Uthra – 19vu1@queensu.ca

Table of Contents

1.0 Abstract

This report describes the development of the Bitcoin Core architecture from concept to concrete implementation. With the addition of the Bitcoin Protocol, SPV, and Consensus Rule subsystems, we identified key dependencies and highlighted differences between the initial and final architectures. We also dive deep into our chosen second-level architecture, Bitcoin Wallets. We perform reflexion analysis on it and investigate the gaps between the conceptual architecture and the concrete architecture. Furthermore, our team provides two sequence diagrams displaying the decentralized payments and transaction verification of those payments in the high level subsystems using only concrete function calls.

2.0 Introduction & Overview

The purpose of this report is to dive deeper into the concrete architecture of Bitcoin Core and the it's wallet subsystem. Our report is sectioned into four parts including, architecture, use cases, external interfaces, and conclusion and lessons learned.

In the first section, Architecture, we analyze the software dependencies within the conceptual and concrete architectures, explain new and unexpected dependencies and summarize our findings in a reflexion analysis. The architecture part is divided into two. Our report analyzes a high-level conceptual and concrete architecture of Bitcoin Core and then we dive further into Bitcoin Core Wallets as a second level subsystem. Here we complete another reflexion analysis comparing both the conceptual and concrete diagrams and conclude with a description of Bitcoin Core concurrency.

The second section in our report covers the systems use cases and showcases UML diagrams to further explain. The two cases our report touches on include decentralized payments highlighting the Bitcoin Core's P2P network, and the process of verifying transactions. This portion not only illustrates the structure of our system, but also showcases the interactions and dependencies between the systems referenced throughout the use cases.

External Interfaces is the third portion analyzed in our report. This section elaborates on the information content transmitted to and from the system, otherwise known as the public properties and methods available.

Lastly, our report ends with our conclusion and lessons learned portions. This section reflects on the architecture and our findings as a whole as well as discusses some road blocks our group encountered along the way.

3.0 Derivation Process

The derivation process consisted of a combination of our ideas to form a concrete architecture that could be agreed on. Firstly, we made use of the tool 'Understand', allowing us to view the dependency graphs of Bitcoin Core given its source code. The folders and files in the source code were grouped into the matching subsystems found in our conceptual architecture to produce a more understandable diagram, while files that had little to no importance in the architecture were left behind. This way we were each able to view the concrete architecture of Bitcoin Core with the new dependencies. When discussing our versions of the concrete architecture of Bitcoin Core, we came to an agreement that components such as Bloom Filters and Block Explorers did not flow well with the rest of the components in the architecture.

4.0 Conceptual Architecture

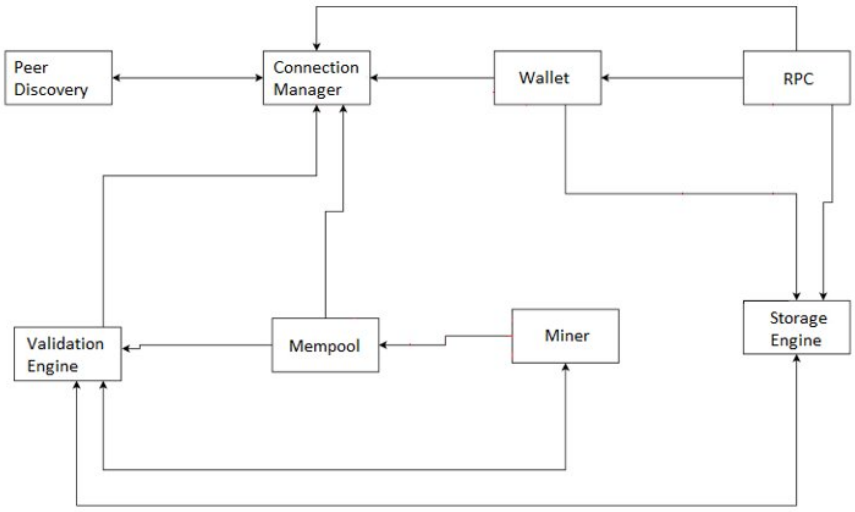

Figure 1. Conceptual Architecture

Based on the feedback we received for our assignment 1 report, we created a conceptual architecture that had been overlooked initially. Using arrows as visual aids, we created an architecture that clearly illustrates the dependencies between subsystems, drawing inspiration from group 17, Build Generation Wealth.

5.0 Concrete Architecture

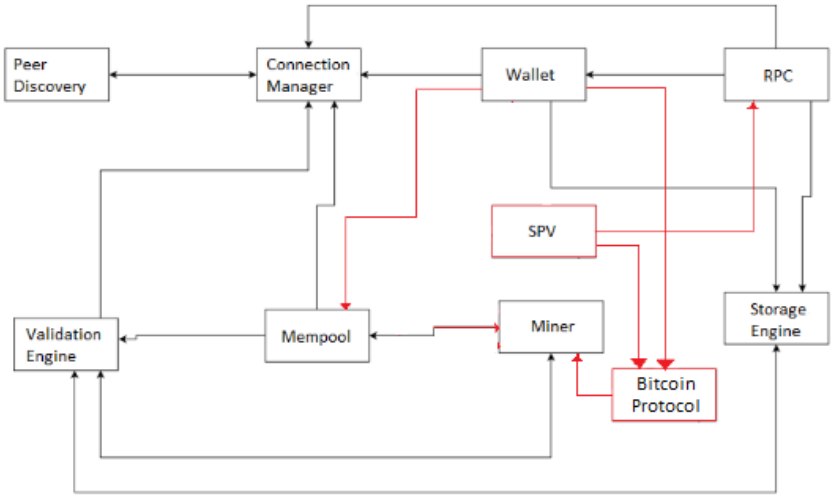

Figure 2. Concrete Architecture

5.0.1 Concrete Subsystems and Interactions

In the derived concrete architecture, three new subsystems are introduced:

Bitcoin Protocol → defines the rules and mechanisms for the network's operation

SPV (Payment verification) → a method for verifying transactions that does not require downloading the entire blockchain

Consensus Rule → a set of rules that ensure that all nodes on the Bitcoin network agree on the state of the blockchain and that incoming transactions and blocks are valid.

5.0.1.1 New Subsystems

- **RPC**: Protocol used by Bitcoin nodes to communicate with each other
- **Wallets**: Software program used to store and manage private keys, which are used to sign transactions.
- **Connection Manager**: Responsible for managing the network connections between Bitcoin nodes.
- **Peer Discovery**: Process of finding and connecting to other Bitcoin nodes on the network.
- **Storage Engine**: Responsible for managing the Bitcoin blockchain database.
- **Miner**: Node that participates in the Bitcoin network by solving mathematical problems to add new blocks to the blockchain.
- **Mempool**: Data structure that stores unconfirmed transactions on a node.
- **Validation Engine**: Responsible for validating transactions and blocks.

5.0.1.2 Subsystem Review

All the subsystems in Bitcoin Core concrete architecture interact with each other seamlessly to help the system function. When a user creates a bitcoin transaction, their wallet broadcasts the transactions to other nodes in Bitcoin Core's P2P (Peer-to-Peer) Network.The Peer Discovery subsystem helps the wallet find nodes to broadcast the transaction to. The Connection Manager subsystem ensures that the wallet stays connected to other nodes on the network, so that it can broadcast transactions.The Storage Engine subsystem stores the blockchain, which contains information about all Bitcoin transactions that have ever occurred.The Validation Engine subsystem verifies that new transactions and blocks follow the consensus rules before they are added to the blockchain. The Mempool subsystem stores unconfirmed transactions, allowing nodes to prioritize their own transactions and ensure that they are confirmed in a timely manner. Wallets create new transactions, broadcast them to other nodes on the network, and interact with the Mempool to monitor the state of the network.and add them to the blockchain, while also verifying the validity of transactions and blocks that are broadcast to them by other nodes. All of these subsystems work together to create a secure, decentralized, and efficient network that allows users to send and receive Bitcoin

Bitcoin Protocol - The Bitcoin Protocol is a set of rules that determine the behaviour of Bitcoin Network nodes. The Bitcoin protocol is made up of the following parts: transaction format, block format, mining rules, and consensus rules. Transactions are organized as inputs and outputs so the protocol specifies how they should be formatted and used to transfer funds between parties. The Bitcoin Protocol specifies the format of Bitcoin blocks . The protocol also defines the format of blocks and how they are added to the blockchain as well as specifies the mining process used to create new blocks in terms of mining. Finally, the Bitcoin Protocol includes consensus rules. These rules specify how transactions and blocks are validated and how conflicts are resolved. The protocol also defines the rules for checking which blocks and transactions are valid and invalid. Those rules are consensus rules, a set of rules that all nodes on the Bitcoin Network must follow in order to keep the blockchain consistent and secure. The validation engine enforces these rules, making sure that all transactions and blocks are valid and the Bitcoin network functions properly. For example, an important consensus rule is that in order for a transaction to be valid, it must be digitally signed by the owner of the funds. This helps in the prevention of fraudulent transactions and guarantees that funds can only be spent by the rightful owner.

SPV - Simplified Payment Verification is a method for verifying transactions on the Bitcoin network without downloading the entire blockchain. Instead, they rely on full nodes to provide block headers to prove the validity of transactions. SPV is used by lightweight clients, like

mobile wallets, to interact with the network without requiring the same level of resources as full nodes. Wallet and SPV clients are dependent on the RPC's component, as they use RPCs to communicate with other nodes on the network. SPV clients rely on consensus rules to verify transactions and ensure the integrity of the Bitcoin network. SPV clients rely on the fact that all nodes on the network are following the same consensus rules to ensure that the transactions they receive are valid. If a node was to violate the consensus rules, it would be rejected by other nodes on the network, and its transactions would be considered invalid.

5.0.2 Reflexion Analysis

There were a few discrepancies between the conceptual architecture and concrete architecture. By performing a reflexion analysis, we were able to determine the divergences between both architectures.

5.0.2.1 New dependencies

SPV → RPC

The SPV subsystem uses RPC commands to access important information from nodes regarding a wallet's state, while also using it to retrieve data involving any transactions made.

SPV → Bitcoin Protocol

The SPV subsystem also depends on the Bitcoin Protocol due to the latter's several features, including validating transactions for payments made by SPV, using addresses to verify payments, and network communications to receive transactions.

Bitcoin Protocol → Miner

The protocol creates and adds new blocks to the blockchain using the miners by following Bitcoin's protocol. This way new blocks can be validated and transactions can also be added to the blockchain.

Wallet → Mempool

In Bitcoin Core's concrete architecture, wallets and the mempool are closely connected. The mempool stores a list of unconfirmed transactions that have been broadcast to the network but have not yet been included in a block. When a wallet creates a new transaction, it adds it to the mempool so that the network can distribute it and include it in a block. Wallets depend on the mempool to determine the status of their transactions. When a wallet creates a new transaction, it checks the mempool to ensure that there are no conflicting transactions. Wallets also rely on the mempool to provide them with information about the state of the network.

Wallet → Bitcoin Protocol

The wallet depends on Bitcoin protocol's features such as validating transactions, network communications, and generating addresses. This allows the transactions to be managed properly by the wallet.

Mempool → Miner

Mempool's dependency on the Miner subsystem is one that was noticed after creating the conceptual architecture for Bitcoin Core, where Miner was shown to depend on the Mempool subsystem. Mempool is shown to depend on the Miner subsystem as well as it contains transactions that are unconfirmed, which are then cleared by miners and added to blocks.

5.0.3 Chosen 2nd-Level Conceptual Architecture

The Wallet Subsystem shown in the image below contains many components to form together a top-level view of the conceptual architecture. The architecture starts with the wallet user interface that serves as a GUI between the wallet and the user, with many controllers and managers of the wallet connected both ways as a double dependency. This way the components are able to send and receive information while connected directly in a process-control type of relationship.

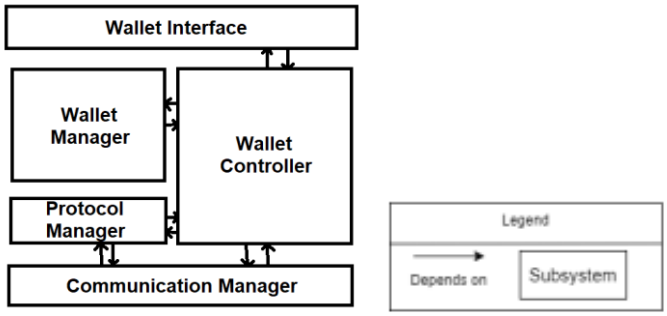

Figure 3. 2nd-Level Conceptual Architecture

5.0.4 Chosen 2nd-Level Concrete Architecture

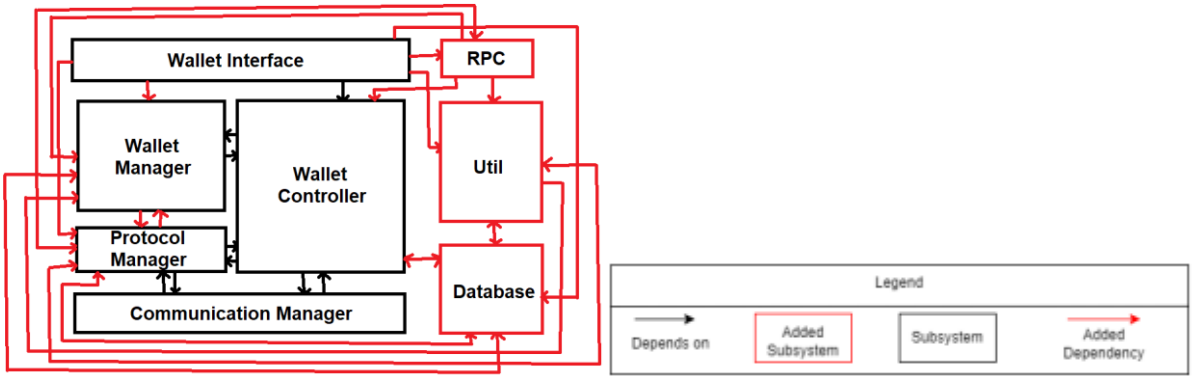

Figure 4. 2nd-Level Concrete Architecture

Wallet Interface **-** The initialization and interface of the bitcoin wallet, serving as a GUI between the user of the software and the digital wallet of Bitcoin Core.
Wallet Manager **-** Manages the commands the user makes to the coins in the bitcoin wallet, and decides the execution of such events, such as send, receive, dump, salvage, select(coin), spend, etc.
Wallet Controller **-** Controls the processes of the bitcoin wallet, containing the events that alter or control different aspects of the wallet, such as the fees, control(coin), feebumper, loading, transactions, etc.
Protocol Manager **-** Manages the different protocols and rules to be followed, ensuring correct relation of certain components and information, such as all the tests, signature checking, cryption, ismine (as in checking if some process belongs to you), etc.
Communication Manager **-** Manages the communication between the Wallet Controller and the Protocol Manager, to relay information in a secure way and ensure the integrity of the information passed between the components.
RPC (Remote Procedure Calls) **-** A subsystem within the bitcoin wallet that holds certain remote procedure calls such as addresses, coins, encrypt, signmessage, etc.
Util **-** Tools and utilities used by the bitcoin wallet.
Database **-** All the databases used by the bitcoin wallet to store information including the main database, wallet database, Berkeley database, sqlite, etc.

5.0.5 Reflexion Analysis

New Subsystems: Wallet → RPC, Database, and Util
Rationale:
A lot of database files were used to store information and could not quite fit into the components, so a new component was made to consist of all the databases in the wallet subsystem.
Util was added to store the tools and utilities used by the bitcoin wallet subsystem.
A smaller version of the RPC (Remote Procedure Calls) was stored inside the wallet directory with all the calls or components needed or related to the bitcoin wallet subsystem.

5.0.5.1 New and Removed Dependencies

<u>New Dependencies</u>

Wallet Interface → RPC
Wallet Interface → Util
Wallet Interface → Wallet Manager

Wallet Interface → Protocol Manager

Rationale: The dependencies between the wallet interface and other components of the bitcoin wallet system indicate how the nodes of Bitcoin communicate with each other, how the tools are used, how commands made by the user are managed and how protocols and rules are managed.

Wallet Interface ↔ Database

Rationale: The double dependency between the wallet interface and database shows how information can be accessed and displayed to the user. The information is stored in the database and later accessed by the wallet interface which acts as a GUI between the user and the bitcoin wallet.

RPC → Util

RPC → Wallet Manager

RPC → Wallet Controller

Rationale: The newly added RPC system that is a part of the bitcoin wallet system requires a few wallet utilities, and the information passed from the manager or controller of the wallet in order to function.

Database ↔ Wallet Controller

Rationale: This new double dependency was created as a connection between the database and the wallet controller was needed to allow the wallet access to information and data. The database stores user information thus the wallet controller can access this information and update the database.

Database ↔ Util

Rationale: The double dependency between the database and util systems was added to allow the database to access the tools and utilities required for the wallet to be stored. Likewise, the util system can access the database component to store and access data for tools and utilities required.

Util → Wallet Manager

Rationale: This one way dependency allows the wallet access to the tools and utilities used by the Bitcoin wallet. Furthermore, by connecting these two systems, the wallet can now manage data and transactions of bitcoin.

Wallet Manager ↔ Protocol Manager

Rationale: The new double dependency instance from the wallet manager system and the protocol manager system is a required connection, making information relay between them

a possibility, providing a direct connection between both of the manager component subsystems.

Removed Dependencies

Wallet Controller → Wallet Interface

Rationale: Compared to the conceptual architecture, the wallet controller does not need to have a two way dependency with the wallet interface for a functional bitcoin wallet. This demonstrates that the wallet interface only needs one dependency towards the controller to effectively manage and control the system, only passing information from the interface to the controller.

5.0.6 Concurrency

Bitcoin Core Wallet is a subsystem responsible for sending, storing and receiving bitcoin. The wallet provides many services and operations simultaneously creating its concurrent nature. The inner workings of the wallet system allow funds to transfer easily via a native wallet address; however, this system is not only restricted to Bitcoin Core but rather the whole Bitcoin network. This network is mapped out with a whole bunch of nodes which connect users on the transaction verification engine. As a result of this mapping, this means many transactions can be in operation at the same time, on the same network, via the Bitcoin Core Wallet. Overall, the Bitcoin Core Wallet provides users with a high-level cryptocurrency wallet with strong security, simplicity, and accessibility.

6.0 Use Cases

6.0.1 Use Case 1

Use case #1: Decentralized Payments (Peer-to-peer network)

To send out a payment in a distributed crypto payment system, the following steps are involved. Firstly, the user initiates a payment request through the Wallet Interface module, which serves as a graphical user interface between the user and the digital wallet. Secondly, the Wallet Manager module receives the payment request and processes it according to the user's instructions. This may involve selecting the appropriate coins to use for the transaction, calculating the fees, and generating the necessary transaction data. Thirdly, the Wallet Controller module controls the execution of the payment request, such as loading the necessary coins and signing the transaction. Fourthly, the Protocol Manager module ensures that the transaction adheres to the rules and protocols of the distributed crypto payment

system. This may involve checking the validity of the signature, ensuring that the user has sufficient funds, and verifying the recipient's address. Fifthly, the Communication Manager module securely relays the transaction information between the Wallet Controller and Protocol Manager, ensuring the integrity of the data. Sixthly, the RPC subsystem may be used to perform certain remote procedure calls, such as encrypting the transaction data or signing a message to be included with the transaction. Seventhly, the Util module provides various tools and utilities that may be used during the transaction process. Finally, the Database module stores all relevant transaction data, including the main database, wallet database, and other databases used by the distributed crypto payment system.

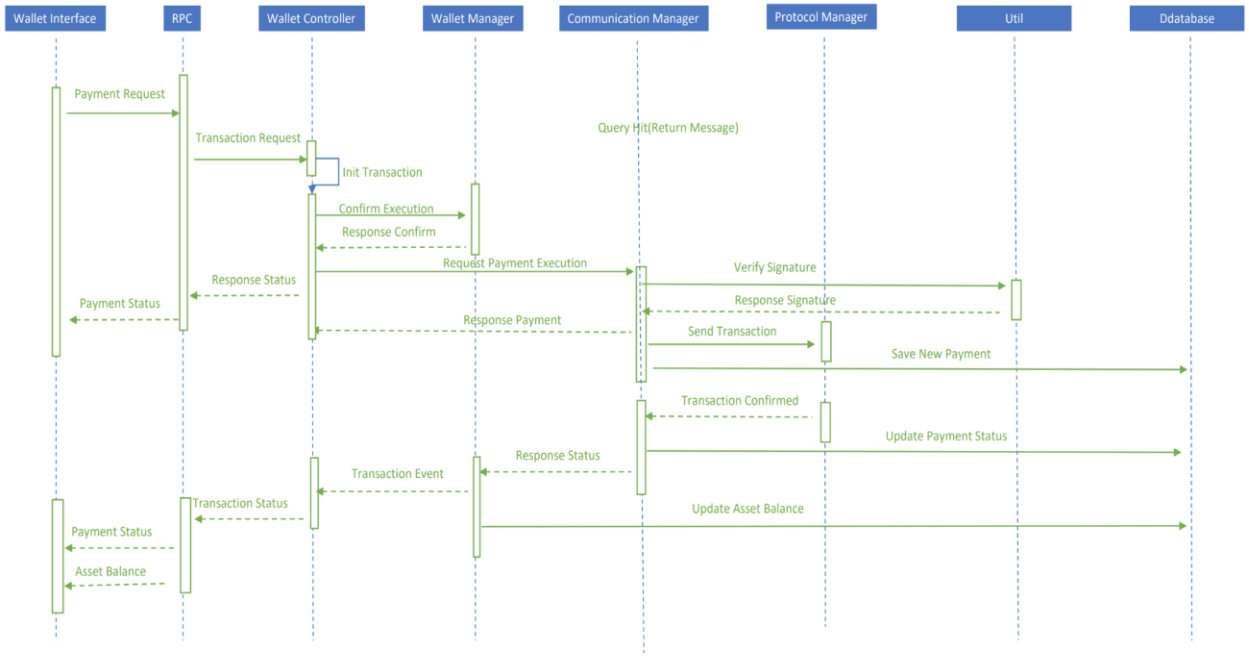

Figure 5. Use Case 1

6.0.2 Use Case 2

Use case #2: Verifying Transactions

To verify a payment in a distributed crypto payment system, the following steps are taken. Firstly, the user initiates a transaction by sending the required amount of cryptocurrency to the intended recipient's wallet address. Secondly, the Wallet Manager receives the transaction request and validates it, checking that the user has enough funds to cover the transaction and that the recipient's wallet address is valid. Thirdly, the Wallet Controller then takes over and controls the execution of the transaction, including setting the transaction fee, ensuring proper encryption and signing of the transaction, and updating the database with the transaction information. Fourthly, the Communication Manager facilitates communication between the Wallet Controller and Protocol Manager to ensure the integrity

of the information passed between them. Fifthly, the Protocol Manager ensures that all the necessary protocols and rules are followed, such as signature checking, encryption, and ensuring that the transaction inputs and outputs balance. Finally, once the transaction is complete, the user can verify it by checking the transaction details on the blockchain, which is a public ledger that records all cryptocurrency transactions. The transaction details will include the transaction ID, the amount sent, the recipient's wallet address, and the transaction fee.

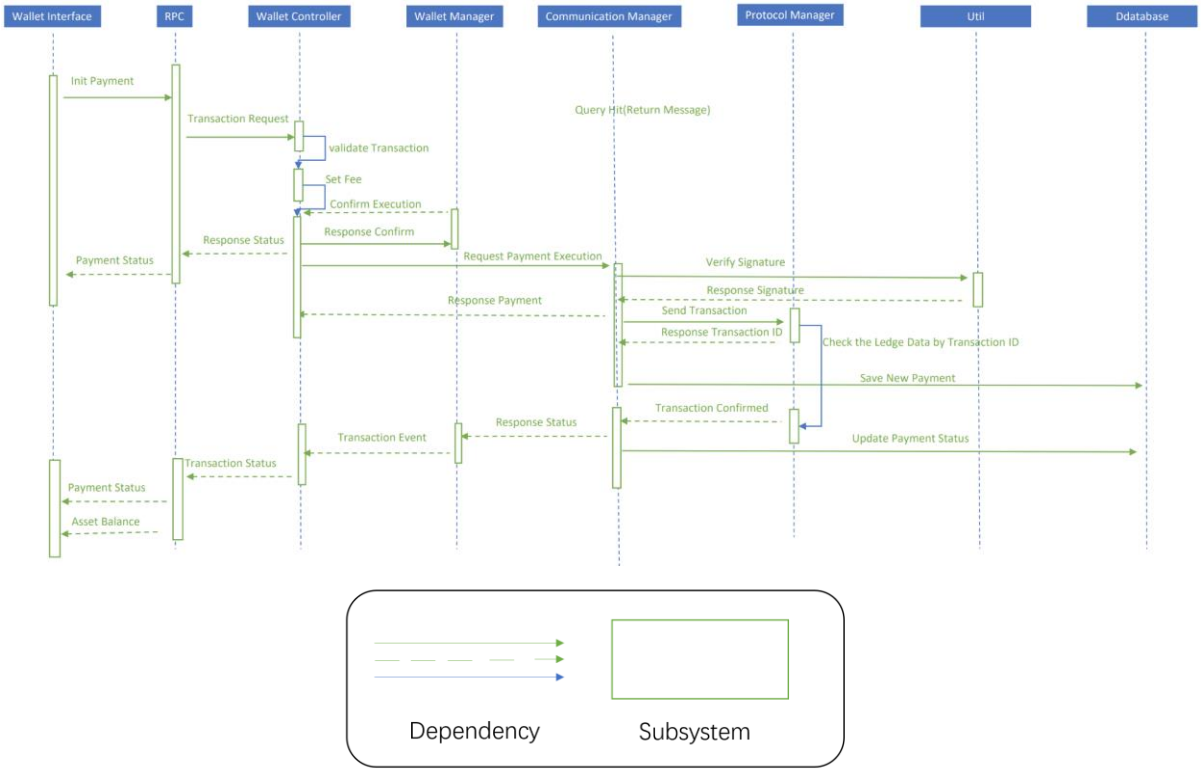

Figure 6. Use Case 2

When a transaction is submitted to the blockchain network, it needs to be confirmed by the network in order to be considered valid and added to the blockchain. This confirmation process is done through a consensus mechanism that ensures the integrity and security of the blockchain network. In general, there are two main types of consensus mechanisms used by blockchain networks: proof of work (PoW) and proof of stake (PoS). Bitcoin is a PoW-based blockchain network, miners compete to solve a cryptographic puzzle by using their computational power to validate transactions and create new blocks. Once a miner successfully solves the puzzle, they broadcast the new block to the network, and other nodes on the network validate the block and confirm that the transactions within it are valid. Once a block is confirmed by the network, it is added to the blockchain, and the miner who solved the puzzle is rewarded with newly created cryptocurrency.

7.0 External Interfaces

The external interfaces in the concrete architecture are the same as the conceptual architecture:

1) **RPC**
   Bitcoin Core's RPC (Remote Procedure Call) is a method that executes various commands on remote servers.

2) **LevelDB**
   Bitcoin Core's need for a database is covered by LevelDB, a Google provided library that stores an ordered mapping of key-value pairs.

3) **Bitcoin Network**
   The Bitcoin peer-to-peer protocol has tons of nodes running. These nodes need to interact with each other through networks. The network is used to transmit various transactions and blocks.

8.0 Data Dictionary

Bitcoin blocks - a collection of transactions that have been verified and added to the blockchain
Understand - a development tool used to visualize the provided source code
Subsystem - a group of components that create a system
UML - a visual design of the software
Blockchain - the public ledger of Bitcoin, responsible for recording the transactions that occur over a peer-to-peer network
Peer-to-Peer Network (P2P) - sharing of data between peers without centralized control

9.0 Naming Conventions

SPV - Simplified Payment Verification
RPC - Remote Procedure Call
P2P - Bitcoin Core's Peer-to-Peer Network
UML - Unified Modeling Language
PoW - Proof of Work
PoS - Proof of Stake

GUI - Graphical User Interface

10.0 Conclusion

Concrete architecture is the implementation of the system whereas conceptual architecture are the ideas that the developer comes up with for the system. The purpose of this report was to obtain a deeper understanding of the concrete architecture of Bitcoin Core and its subsystem, Bitcoin Core Wallet. Through our findings, we concluded that the P2P network is the most appropriate architectural style for the architecture of the system. Furthermore, the report analyzed the dependencies between the components and summarized the findings in the form of a reflexion analysis. A software called, Understand, was used to visualize the dependencies between the components of the Bitcoin source code and view the concrete architecture of the Bitcoin Core System. Using the feedback provided on assignment one, a new diagram for conceptual architecture was created and used to deduce the diagram for the concrete architecture with more subsystems and dependencies added. The report further elaborated on the use cases, giving a visual representation through the UML diagrams and the external interfaces. Thus, a deeper understanding of the concrete architecture of the Bitcoin Core System was obtained through the analysis of the dependencies and diagrams made on the software, Understand.

11.0 Lessons Learned

Like all group assignments, there were a few lessons learned. The first and main insight was discovering how significant the architecture section was. Initially we divided up our report into our usual sections similar to our previous assignments, clueless to how much work the architecture section truly was. Our group got caught up with poor communication and a perfect solution would have been to hop on a call and regroup instead of frantic conversation. The second lesson we learned as a group would be to ensure we are acquainted with the tools required for the report and then to understand and analyze the Bitcoin Core Wallet component. We found the learning curve on the software interface, Understand, to be quite intricate, thus trying to plan out our architecture became more difficult. To overcome this, we should have allotted more time to become acquainted with the new tool, Understand. This would allow our group time to comprehend the interface thus cutting back on the time spent while simultaneously constructing the architecture diagrams. Overall, our group hit some speed bumps along the way, but we pulled together in the end.

12.0 References

Daniel, Alan. “What Is Bitcoin Core Wallet – a Complete Review.” *Unblock.net*, 28 Sept. 2021, https://unblock.net/bitcoin-core-wallet-review/.

Müller, et al. “The Bitcoin Universe: An Architectural Overview of the Bitcoin Blockchain.” *Lecture Notes in Informatics*, 2018, https://dl.gi.de/bitstream/handle/20.500.12116/16570/DFN-Forum-Proceedings-001.pdf.