Bitcoin Core Conceptual Architecture Report

CISC322

Friday, February 17th, 2023

Eric Lam – 19ewhl1@queensu.ca – 20229013

Andrew Zhang– 19az32@queensu.ca – 20210066

Dylan Chipun – 19dmc10@queensu.ca – 20224970

Amy Hong – 19yh94@queensu.ca – 20219853

Sueyeon Han – 19sh77@queensu.ca – 20217002

Asher Song – 17acds@queensu.ca – 20112257

TABLE OF CONTENTS

ABSTRACT

This report outlines and examines the conceptual architecture of Bitcoin Core, an open-source reference implementation of a client software for the decentralized bitcoin network. Bitcoin Core allows users to represent a node in Bitcoin's peer-to-peer network. Bitcoin Core's conceptual architecture was sourced from documentation of system functionality from the official Bitcoin Core website, the initial Bitcoin protocol written by Satoshi Nakamoto and various existing bodies of research on Bitcoin Core's architecture. The study and usage of blockchain technology for financial purposes are novel, yet increasing in widespread popularity. As such, it is beneficial to study the architecture of the earliest publically available implementation of this new form of currency management. This paper seeks to outline Bitcoin Core's conceptual architecture which may be further utilized to discover the concrete architecture of this system.

INTRODUCTION & OVERVIEW

In the past couple of years, the rise of bitcoin has been outstanding. It is slowly becoming a form of currency accepted in countries such as Canada, the U.S., and the U.K. Bitcoin is an open-source project created by Satoshi Nakamoto. Open source not only means that it is free to use, but also that bitcoin is a project developed by an open community. This means that anyone can view Bitcoin's source code and participate in the development with their contributions. Because Bitcoin is open source, it makes it easy to use and also creates a development team that values collaboration and innovation with just about anyone. What started from a community of just the developer, Satoshi Nakamoto, has now grown to have more than 400 contributors by 2016. The first implementation was simply known as "Bitcoin", however, it has grown and evolved into what is now "Bitcoin Core".

Before looking into the architecture behind Bitcoin Core, we must first get an understanding of how Bitcoins operate. Bitcoins currency works on a peer-to-peer network where transactions are verified and recorded on the blockchain. New bitcoins are created through a process known as mining, which involves using hardware such as the CPU or the GPU to complete calculations and problems first to add a new block to the blockchain. As a reward, miners receive a new bitcoin. A huge benefit of using Bitcoin is to create transactions between users without dealing with the middlemen such as banks. This creates a fast and secure low-cost payment. Bitcoin Core is the reference implementation of the Bitcoin system. This means that Bitcoin Core is the authoritative reference on how each part of the technology should be implemented in the system. It is programmed to decide and verify transactions and blocks on the Bitcoin network.

The purpose of this report is to take a deeper look at the conceptual architecture of the Bitcoin Core system. We will be discussing details of its implementation, such as the different components that make up the system. One of the components is the wallet which stores the user's private key and enables transactions using bitcoins. The transactions and blocks handle the exchange of bitcoins and validation engines verify and authenticate transactions to ensure security and accuracy. The miner solves complex problems to create new blocks while peer discovery ensures each node is connected allowing efficient communication. While looking at the Bitcoin Core architecture, we will also analyze their functionalities and how components interact with each other, the control, data flow, and concurrency between components become significant to allow the operation of the system to run smoothly. The diagrams that model the system's flow such as a sequence diagram and use cases provide an easy understanding of the pathing and decisions the system goes through. The use cases - making a transaction using Bitcoin Core and receiving payment in the form of the Bitcoin currency, both demonstrate how the system architecture utilizes its subsystem components and how these elements interact. Moreover, the external interfaces involved provide a thorough analysis of the system as a whole.

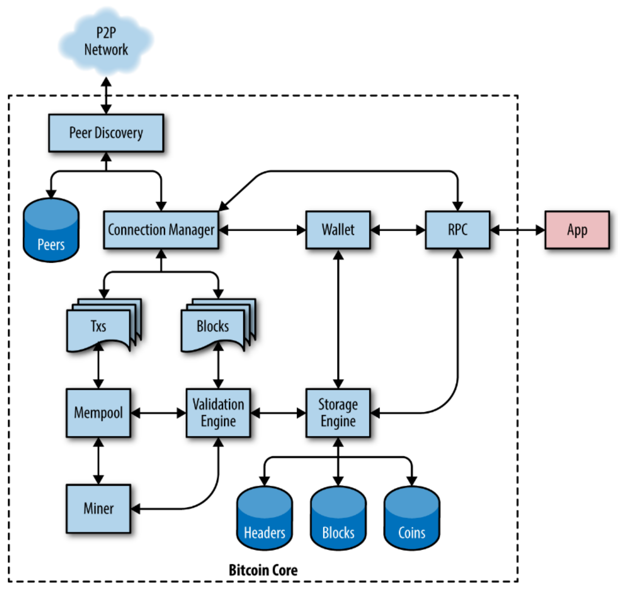
ARCHITECTURE OF BITCOIN CORE

Overview

The architectural style of Bitcoin core follows the layered style. Each layer provides us with a specific set of functionalities when working together, enabling the Bitcoin protocol. The architecture is broken down into these layers from top to bottom:

- Application Layer: This layer includes components that contribute to the user interface, such as the Wallet component and the RPC.
- Network Layer: This layer handles communications between nodes in the network. The components that belong to this layer include Peer Discovery and Connection Manager.
- Blockchain Layer: This layer is responsible for the validation of transactions and blocks, as well as ensuring an updated copy of the current Bitcoin blockchain. The components that belong to this layer are Txs, Blocks, Mempool, and Validation Engine
- Database Layer: This layer is responsible for the storage and retrieval of data from the blockchain. The databases that are designed in this layer are Headers, Blocks, and Coins. The components that belong to this layer are the Storage Engine
  Bitcoin Core is part of another architecture style, and that is the architecture style that

Bitcoin is designed on, peer-to-peer. Bitcoin is decentralized, meaning that no individual entity has full control over the network, rather it is controlled by all participants of the network. This is how Bitcoin can achieve individuals conducting direct transactions with each other without the need for a centralized banking system.



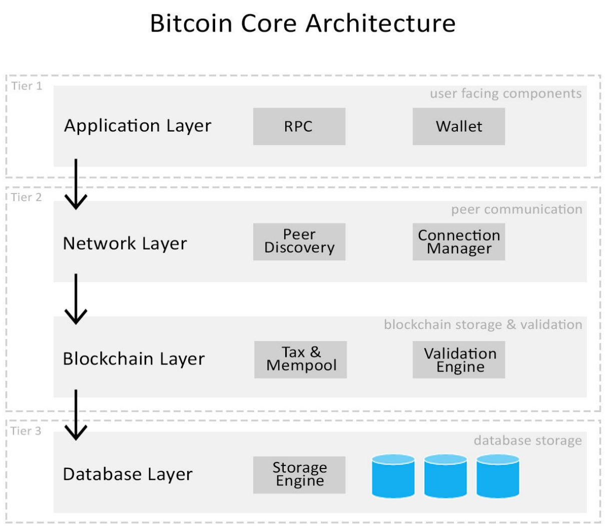


Fig. 1 Bitcoin Core Architecture Fig. 2 Bitcoin Core Layered Architecture

Peer & Peer Discovery

Bitcoin Core implements Bitcoin's Peer-to-Peer Networking system by representing the current user as a singular node. Bitcoin core implements a full node, capable of discovering and being discovered by outbound nodes. The discovered nodes are added to the connection manager. Bitcoin Core allows the current node to discover peers through their IP address ports. Peer discovery includes searching Bitcoin Core's addresses database for existing known peers on the network. This is accomplished by the Address Manager, which keeps track of the IP addresses of both known and unknown peer nodes. Addresses are cryptographically hashed into sections including pre-validated known nodes, and stranger nodes. The peer discovery system also allows users to enter custom peer addresses and discover peers using DNS seeds from servers that randomly resolve to peer nodes existing in the Bitcoin network. As a last resort, the Bitcoin core architecture keeps a database of hard-coded peer IP addresses.

Connection Manager

The connection manager is Bitcoin Core's implementation to manage interactions between peer nodes. The Net subcomponent sits at the core of the Bitcoin core stack and handles network communication with peer-to-peer networks. The connection manager utilizes Semaphore socket threads to form outbound connections with up to 8 peers and accept inbound connections from 125 peers. This component is essential to the functionality of all components which require interactions with peer nodes, as the primary message thread in the connection managers is responsible for managing the queue of inbound messages from peer nodes, and for sending outbound messages toward various sockets. In addition, the connection manager processes inbound messages by selecting appropriate actions depending on the query or request. Typically functions are called to send these requests to other components deeper in Bitcoin Core's architecture for further processing. As such, the connection manager aids in accepting the current block state of the global Blockchain for the validation engine and connecting to peer nodes to perform Bitcoin transactions and accept Blocks. This component ensures the security of the system by implementing Denial-of-Service Prevention. Nodes that send malignant information are assigned a score, increasing their chances of being disconnected from the current peer and banned from future connections.

Wallet

The Wallet component of Bitcoin Core is responsible for controlling the user’s money, managing keys and addresses, tracking the balance, and creating and signing transactions. The wallet does not contain bitcoin but instead contains pairs of private/public keys. The keys are used to sign transactions, proving that a user owns the transaction outputs, which are the coins. There are two types of wallets, nondeterministic and deterministic. Bitcoin wallets were originally nondeterministic, meaning each key is independently generated from a random number through a Bitcoin Core client. However, these wallets are being replaced by deterministic wallets since it was cumbersome to manage, back up, and import. Deterministic wallets contain keys that are derived from the seed through the use of a one-way hash function. These keys can be generated again if one has the original seed, and allows for sufficient wallet export or import. The HD Wallet is the most advanced form of deterministic wallets, where the wallet contains keys derived from a tree structure.

The Wallet component interacts with the RPC API component, where the RPC API provides the wallet with numerous commands that can get and manipulate data in the wallet. Some commands are getwalletinfo, getbalance, createwallet, gettransaction, importwallet, sendmany, walletlock, and many more.

RPC

An RPC is short for Remote Procedure Call. This means that a program is calling for a procedure to execute and run in a different address space. This RPC allows for an easier runtime for the programmer as they do not need to include details of the remote occurrence. It is as simple as a client calling the server for information through the entire network, the RPC is the actual call.

There are different names for each call depending on what you want to call from the server. The server is also where our databases will be stored. Bitcoin Core is built with wallet support by default, therefore all kinds of RPCs can be called, such as createwallet, getwalletinfo, walletlock, and loadwallet.

There are other RPCs that the frontend API can call that are not related to the data of the wallet. It can also reach the Storage Engine and the Connection Manager. Such as we were trying to find information on known peers in an address manager. Of course, these RPCs can be attached to things and run on frontend APIs to allow users to access their data.

In application, these will be the connectors between the main components. In the Garlan and Shaw models, they will represent the communication lines. A filter system or event system seems unideal as these connectors will not be changing information. Depending on style ___ running time may be slow due to massive inputs of RPCS. Security in terms of authorization and integrity is important.

Txs & Mempool

Txs, short for "transactions", are the most important part of the bitcoin system. They are, in essence, data structures that encode the transfer of value between participants in the bitcoin system. Everything else in bitcoin is designed to ensure that transactions can be created, propagated on the network, validated, and finally added to the blockchain. Each transaction is a public entry in bitcoin's blockchain.

Bitcoin transactions create outputs. These outputs are recorded on the ledge, and most of these outputs create spendable chunks of bitcoin called UTXO. UTXO is tracked by every bitcoin client in the UTXO set. New transactions spend one or more of these outputs from the UTXO set. Transaction outputs consist of two parts; an amount of bitcoin denominated in *satoshis*, and a cryptographic puzzle that determines the conditions required to spend the output.

Transaction inputs identify which UTXO will be spent and provide proof of ownership. To build a transaction, the Wallet chooses a UTXO it controls and finds one with enough value to make the required payment. For every UTXO that's consumed for the payment, the wallet makes one transaction input pointing to the UTXO and unlocks it with an unlocking script.

In the Bitcoin Core architecture, when a peer node receives transactions and blocks, they will send the transactions toward the txs component. The transactions are then sent to the mempool. The mempool is a component that acts as a "virtual waiting room". It is a temporary storage unit that collects unconfirmed transactions waiting to be processed by the miner component and then included in the blockchain. Once a transaction is confirmed and included in a block, it is removed from the mempool. The transaction is only added to the mempool after it has been verified by the validation engine.

Blockchain

Bitcoin Core will require the blockchain to be validated when a new node is created. The current iteration of the initial blockchain validation utilizes a headers-first approach to downloading the blockchain. The new node first downloads the header of each block in the current longest blockchain, then downloads the remaining body of each block through concurrent connections with several peers. This blockchain is saved on the map "chainActive", which contains the longest fully verified blockchain.

The Bitcoin Core architecture manages blocks through a block index database. Through this database, the current block tree state is saved on the node. Bitcoin core stores the validation status, the number of transactions saved on the block, and the number of blocks in its chain. To extend the blockchain, Bitcoin core stores headers which have completed more total work than the current tip in “setBlockIndexCandidates”. The best head among these is stored within, “pindexBestHeader”, it may replace the current tip once the entire block is validated.

Miner

Mining is the process in which blocks are added to the blockchain. By adding blocks to the blockchain, a block reward and transaction fees are paid out.

There are two ways to mine: solo mining and pooled mining. A single miner can try to create new blocks on its own and is given a large payment upon completion. Otherwise, multiple miners can try to find more blocks together and split the smaller payments. The literal process of mining uses the Bitcoin Core wallet capabilities and utilizes the machine’s CPU to mine.

Typically, a solo miner will use the “getblocktemplate” RPC and go through the process of generating the hash and header for the block. Any completed block is broadcast to a network through bitcoin. There is a program “bitcoind” that implements the bitcoin protocol. Bitcoind is used here for getting new transactions from the network. Bitcoind pays out for the finished block, allows the mining software to inspect transitions, finds info for constructing block header, and checks for the network target.

In terms of architecture, the miner is a component that acts on its own and does things by itself. It is important that messages and information sent by this component are not altered in any way as it is the most crucial in terms of what creates a monetary value for this whole system. It’ll be required that this is consistent 99.999999 percent of the time. There is no scalability for this mining system. Security in terms of integrity, encryption, and non-repudiation are incredibly important.

Validation Engine & Storage Engine

The validation engine, as it is named, checks the validity of incoming transactions and blocks to prevent direct theft, bait and switch, fabricated transactions, chain hijacking, and transaction withholding. A validated transaction is sent back to the mempool so that the miner can use them to mine for a new block. As Bitcoin Core is a full node, it contains a complete blockchain transaction history which is used to independently check if blocks are valid. Once a block has been validated, the validated block is passed on to the storage engine to be stored in a database and relayed onto the network. The storage engine is the component that manages the blockchain for the system in a database. The validation engine also tells the miner to stop mining for the current block if the current block has been received from other nodes and validated.

EXTERNAL INTERFACES

Bitcoin Core uses many different components and external interfaces to carry out the transactions involved with Bitcoin along the network. To begin with, the Graphical User Interface (GUI) is used to send and receive transactions. It displays information about the user such as the user's balance and allows the user to move along the system. Different files that are held within the architecture such as the wallet are used to store the user's private key. The database is used to store the blockchain which records every transaction made on the network locally. The application programming interface (API) allows external applications to work along with Bitcoin for their own purpose. This creates more opportunities for developers to use bitcoin for things such as trading cryptocurrencies or allowing merchants to use Bitcoin for transactions. Messages are used for the communication between the nodes within the bitcoin architecture. They help verify transactions and add security to the blockchain. These components allow the flow and security of the Bitcoin Core architecture

USE CASES

Scenario #1 - New user receiving a bitcoin

When a new user downloads and runs a mobile wallet application, it will automatically create a wallet, which is a collection of addresses and keys. These addresses will become known addresses in the network only after being referenced by another sender. If someone wants to send a bitcoin to this user, a new transaction will be constructed and signed using this specific sender's private keys. Once the transaction is included in a block and added to the blockchain, it gets published and broadcasted through the well-connected nodes in the network and eventually listened to by the recipient's wallet that contains the matching address.

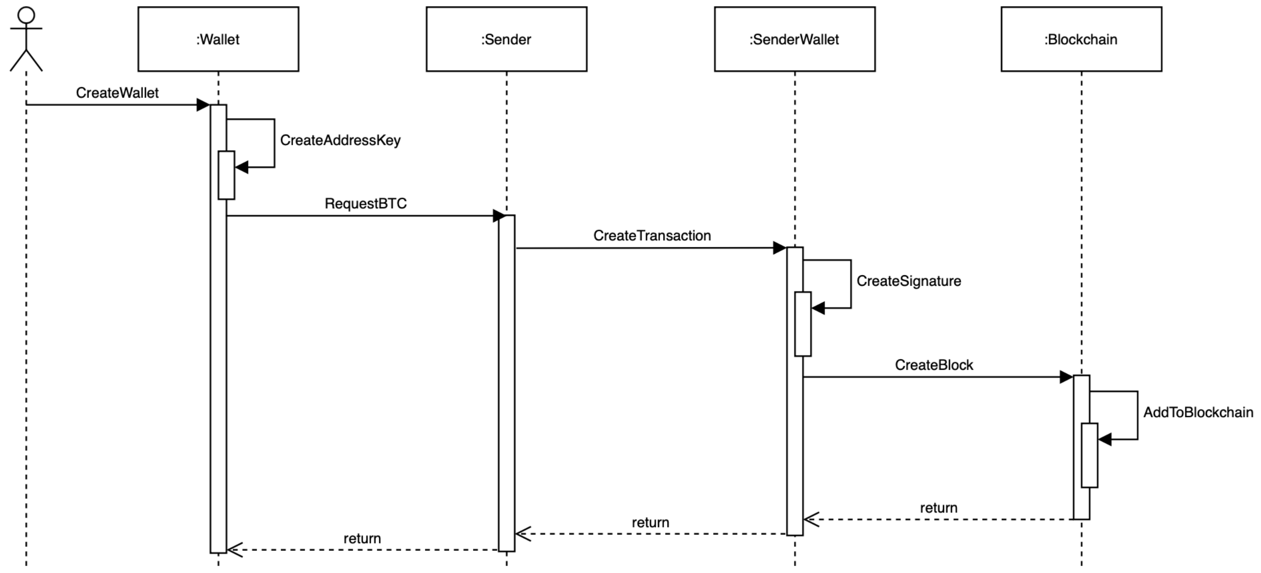


Fig. 3 Use case scenario #1

Scenario #2 - Making a payment at a retail store

Suppose the receiver from use case scenario #1 wants to make a payment at a retail store. The store will create a unique QR containing a destination address and an amount to pay. When the user scans the code and authorizes the payment, the designated amount of bitcoin value gets transferred from the original owner to the new owner. This transaction will be created using a previous transaction's output and the original owner's private key to prove that the user indeed owns the bitcoins, and eventually attaches to the new owner's address. These processes encumber that the transferred amount must be signed using the new owner's key in order to be spent.

The transaction has to be transmitted to the bitcoin network to become part of the blockchain. This process includes the propagation of transactions and blocks to all users of the bitcoin core and verification by a mining process. When new transactions are seen by the network nodes, miners will construct a new block called a candidate block and add the unverified transactions into it. Then, they attempt to prove the validity through computing Proof-of-Work. Eventually, one miner finds the solution, announces the winning block to the network, and now the payment becomes part of the blockchain.

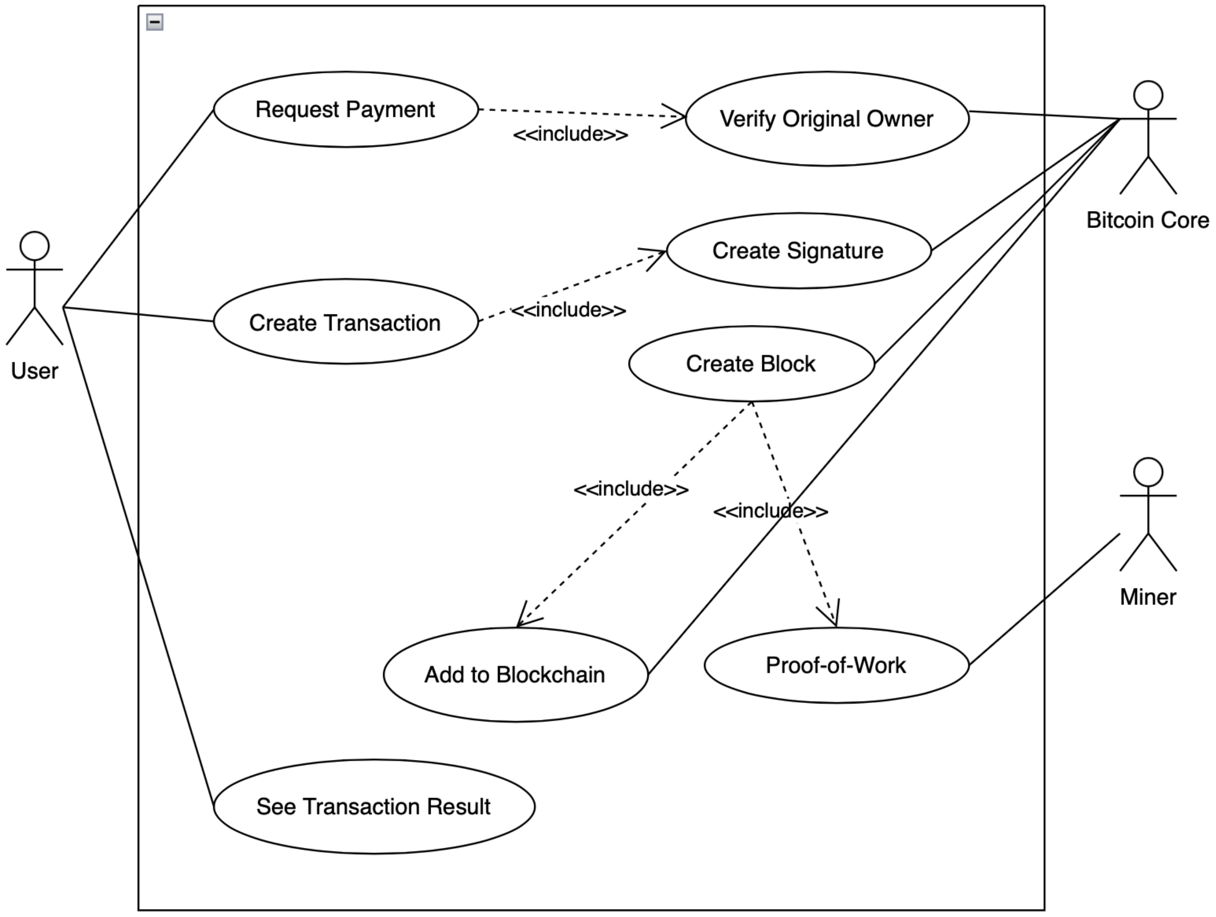


Fig. 4 Use case scenario #2

DATA DICTIONARY

Tip: The last block added to the blockchain stored in a Node instance.
Seed: The master key that keys in a deterministic wallet derive from
Satoshis: The smallest bitcoin unit
Header: The section of a block containing its metadata in an 80-byte format
Block: A block is a data structure in a blockchain which stores a record of transactions
Coin: A coin is a digital currency of a cryptocurrency
Miner: A node that repeatedly finds the solution to the SHA256 algorithm to valid new blocks
Proof-of-Work: A piece of data that needs a significant amount of computation to acquire
Wallet: An application that holds bitcoin addresses and keys which also can be used for bitcoin transactions

NAMING CONVENTIONS

UTXO: Unspent Transaction Output
RPC: Remote Procedure Call
API: Application Programming Interface

CONCLUSIONS

In conclusion, Bitcoin Core's layered architecture is a well-thought-out design that provides a strong foundation for the software's complex and critical functionality. The separation of concerns between different layers promotes modularity, scalability, and reusability, while also enhancing security. The use of this architecture style allows for individual components to be developed and maintained independently, making it easier to add new features and fix bugs. Additionally, the layered architecture allows for different layers to scale independently, meaning that the system can handle large amounts of data and transactions without sacrificing performance or reliability. Overall, the layered architecture style has proven to be highly effective for Bitcoin Core, providing a solid foundation for the software to function securely and efficiently while also allowing for future expansion and improvement.

LESSONS LEARNED

This assignment aims to research and outline the conceptual architecture of Bitcoin Core. In trying to accomplish this, we encountered many difficulties and setbacks. The first of these difficulties surrounds the decentralized nature of Bitcoin, and as an extension of Bitcoin Core. Higher-level concepts surrounding Bitcoin, such as its inception, functionality and documentation are spread in multiple locations across the Internet since no single party or stakeholder is responsible for Bitcoin. Additionally, in the future, we would find the broad architectural view of the Bitcoin Core System(layered) near the project's inception, instead of near its completion. This way, we would not need to edit and rewrite some components and concurrencies within the system to fit a layered architectural style after they have already been completed.

REFERENCES

Bitcoin Core 0.11 (CH 1): Overview. Bitcoin Core 0.11 (ch 1): Overview - Bitcoin Wiki. (n.d.). Retrieved February 18, 2023, from https://en.bitcoin.it/wiki/Bitcoin_Core_0.11_(ch_1):_Overview

Bitcoin Core 0.11 (CH 4): P2p network. Bitcoin Core 0.11 (ch 4): P2P Network - Bitcoin Wiki. (n.d.). Retrieved February 18, 2023, from https://en.bitcoin.it/wiki/Bitcoin_Core_0.11_(ch_4):_P2P_Network#Peer_discovery_.26_connectivity

Bitcoin Core 0.11 (CH 6): The Blockchain. Bitcoin Core 0.11 (ch 6): The Blockchain - Bitcoin Wiki. (n.d.). Retrieved February 18, 2023, from https://en.bitcoin.it/wiki/Bitcoin_Core_0.11_(ch_6):_The_Blockchain

Bitcoin Core Validation. Validation - Bitcoin Core Features. (n.d.). Retrieved February 18, 2023, from https://bitcoin.org/en/bitcoin-core/features/validation

Bitcoin Core. (n.d.). Retrieved February 18, 2023, from https://bitcoin.org/en/bitcoin-core/

Bybit Learn. (2021, October 26). *Bitcoin Mempool: What happens to unconfirmed transactions?* Bybit Learn. Retrieved February 18, 2023, from https://learn.bybit.com/blockchain/bitcoin-mempool-what-happens-to-the-unconfirmed-transactions/

Chapter 3: 'bitcoin core: The reference implementation'. Chapter 3: 'Bitcoin Core: The Reference Implementation' · GitBook. (n.d.). Retrieved February 18, 2023, from https://cypherpunks-core.github.io/bitcoinbook/ch03.html

Chapter 5: 'wallets'. Chapter 5: 'Wallets' · GitBook. (n.d.). Retrieved February 18, 2023, from https://cypherpunks-core.github.io/bitcoinbook/ch05.html

Chapter 6: 'transactions'. Chapter 6: 'Transactions' · GitBook. (n.d.). Retrieved February 18, 2023, from https://cypherpunks-core.github.io/bitcoinbook/ch06.html

Chapter 8: 'The Bitcoin Network'. Chapter 8: 'The Bitcoin Network' · GitBook. (n.d.). Retrieved February 18, 2023, from https://cypherpunks-core.github.io/bitcoinbook/ch08.html

Developer guides. Bitcoin. (n.d.). Retrieved February 18, 2023, from https://developer.bitcoin.org/devguide/index.html#

Introduction. Introduction · GitBook. (n.d.). Retrieved February 18, 2023, from https://cypherpunks-core.github.io/bitcoinbook/

Mining. Bitcoin. (n.d.). Retrieved February 18, 2023, from https://developer.bitcoin.org/devguide/mining.html

Open source P2P money. Bitcoin. (n.d.). Retrieved February 18, 2023, from https://en.bitcoinwiki.org/wiki/Bitcoind

A peer-to-peer electronic cash system. Bitcoin. (n.d.). Retrieved February 18, 2023, from https://bitcoin.org/bitcoin.pdf

RPC API reference. Bitcoin. (n.d.). Retrieved February 18, 2023, from https://developer.bitcoin.org/reference/rpc/index.html