

The background features a large, semi-transparent Bitcoin coin centered behind the text. To the right, a faint line graph is visible, showing an upward trend. The overall color scheme is dark blue with teal and white accents.

Group 27 - Based Bitcoin Core Conceptual Architecture

Amy Cui, David Curtis, Jagrit Rai, John Alajaji, Logan Cantin, Matthew Vandergrift

Link to Video of this presentation

<https://youtu.be/tcBtMkDoNUQ>



Team

John Alajaji

(Member)

- Report: Introduction, Derivation Process, Conceptual Architecture Overview, Block Constructor, Mining, System Evolution
- Slides: Derivation Process, Conceptual Architecture Overview, Block Constructor, Mining, System Evolution


David Courtis

(Presenter)

- Report: Transaction structure and use case, control and flow, alternative architectures, merkle tree.
- Slides: Components, structuring, flow and concurrency

Amy Cui

(Team Leader)

- Report: Wallet programs and files, control and flow
 - Slides: Agenda, Introduction, Derivation Process, Wallet, re-formatting and styling for slides
- 

Logan Cantin

(Member)

- Report: P2P Network Interface, Cryptographic Component, Message Verification, Lessons Learned, Inter-node concurrency
- Slides: Message Verification, Cryptographic component, Lessons Learned


Jagrit Rai

(Presenter)

- Report: Blockchain, Blocks, System Evolution,
 - Slides: System Evolution,
- 

Matt Vandergrift

(Member)

- Report: Blocks, Contracts, Transaction Use Case, Division of Responsibilities, Alternative Architectures, Concurrency, overall Architectural style and Limitations
 - Slides: Division of Work, Transaction component/use case, blockchain, contents, alternative styles, concurrency, limitations.
- 

Contents of this Presentation

01 Introduction

02 Derivation

03 Conceptual Architecture Overview

04 Components

05 Use Case

-> Transactions

-> Mining

06 System Evolution

07 Control and Flow of Components

08 Concurrency

09 Architectural Styles

10 Alternative Architectures

11 Lessons and Limitations

12 Conclusion



01.

Introduction





What is Bitcoin Core?

- Conceptual architecture is a high level overview of the blueprint of a project.
- Bitcoin first introduced in 2008 in a paper by Satoshi Nakamoto titled “Bitcoin: A Peer to Peer Electronic Cash System”
- Bitcoin offers a safe way to transmit online transactions without relying on trust between third parties.
- This presentation will focus on the conceptual architecture Bitcoin Core which is the reference implementation of the Bitcoin protocol and which serves as the backbone of the Bitcoin network.



02.

Derivation





Derivation Process



Researching whitepaper to become more familiar with system.



Understanding the paper “Bitcoin: A Peer-to-Peer Electronic Cash System”, as well as the Bitcoin Core Developer's Guide

Brainstorming individual components and subsystems.

Gaining insight into the system's data dependencies and connections

Consider all possible main use cases

Adjusted our understanding by researching subsequent contributions.



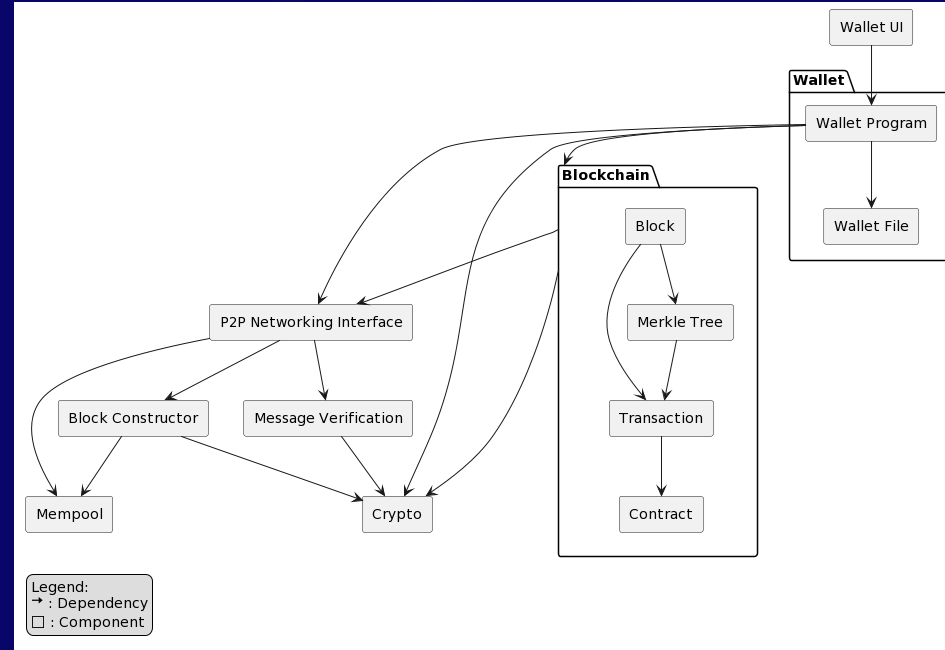


03.

Conceptual Architecture Overview

Conceptual Architecture Overview

- Peer-to-peer style
- System is broken into a few core parts
 - Wallet
 - Blockchain
- Other components tie back into the Wallet and Blockchain
 - E.g. P2P Network Interface fetches blocks to be added to the Blockchain





04.

Components



Components



Message
Verification



Cryptographic
Component



Blockchain Transactions



Contracts



Merkle
Tree



Block
Constructor



Mempool



Wallet



Message Verification



- Verifies a variety of messages that come in through the P2P network
- Performs the following types of verifications:
 - Block / transaction verification: ensures that a block / transaction is cryptographically valid
 - Broadcast storm prevention: prevents messages from propagating and multiplying infinitely in the network
 - Packet verification: ensures that received packets are well-formed and from trusted nodes to reduce attack surface





Cryptographic Component





- Contains the functions responsible for doing the cryptography tasks required by the Bitcoin protocol
- Two main types of functions:
 - Cryptographic Hashing: function that takes arbitrary input data and returns a fixed-size number. Used for proof of work in mining and uniquely identifying blocks
 - Digital signatures: Asymmetric encryption technique that allows a user to “sign” a piece of data to demonstrate that they generated it. Used for verifying transactions.





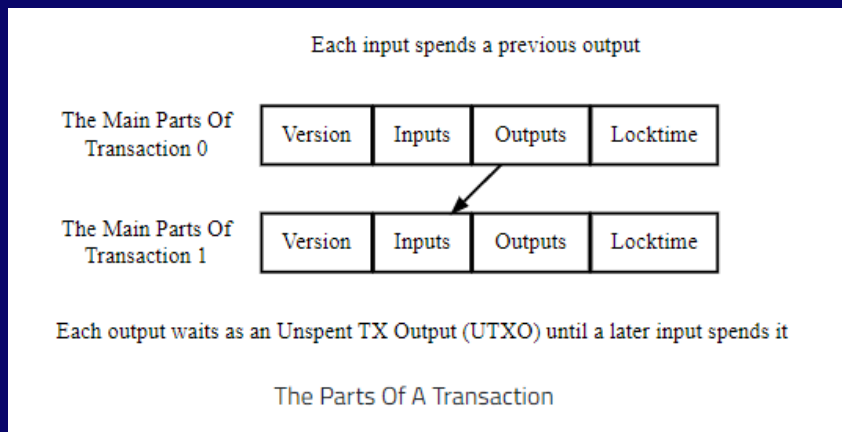
Blockchain



- The blockchain is a distributed, decentralized ledger system
 - Ledger means that it allows information to be tracked over time
 - Built of a connection of blocks which provides,
 - Transaction tracking
 - Immutability
 - Resilience against attacks
 - What are Blocks?
 - Collection of transactions (called transaction data) preceded by a block header
 - Tracks ownership of currency
 - Block header holds the Merkle root, the proof of work, along with the hash of the previous block's header
- 
- 

Transactions


- The transaction part of Bitcoin Core is what allows users to exchange satoshis.
- The data fields are inputs, outputs, a version number and a locktime.



Source: https://developer.bitcoin.org/_images/en-tx-overview.svg



Block Constructor and Mempool



- Mempool queues incoming valid transactions for use by the Block Constructor
- Responsible for the construction blocks during the mining process
- Using a provided block template and a collection of transactions, the block constructor will generate a block together with a block header
- Generates a proof of work for the block utilizing the Crypto component
 - Gets a possible nonce values
 - Hashes it using the Crypto component
 - Verifies whether the number of zeros satisfy the network's target





Wallet

- A Bitcoin Wallet is composed of the Wallet Program component, the Wallet UI and Wallet File component
 - Wallet UI allows users to interact with their Bitcoin Wallet
 - Wallet Program is a software application that allows for transaction management and wallet key generation
 - Wallet File on the other hand is the backend component that stores the critical information needed to access and manage the funds.



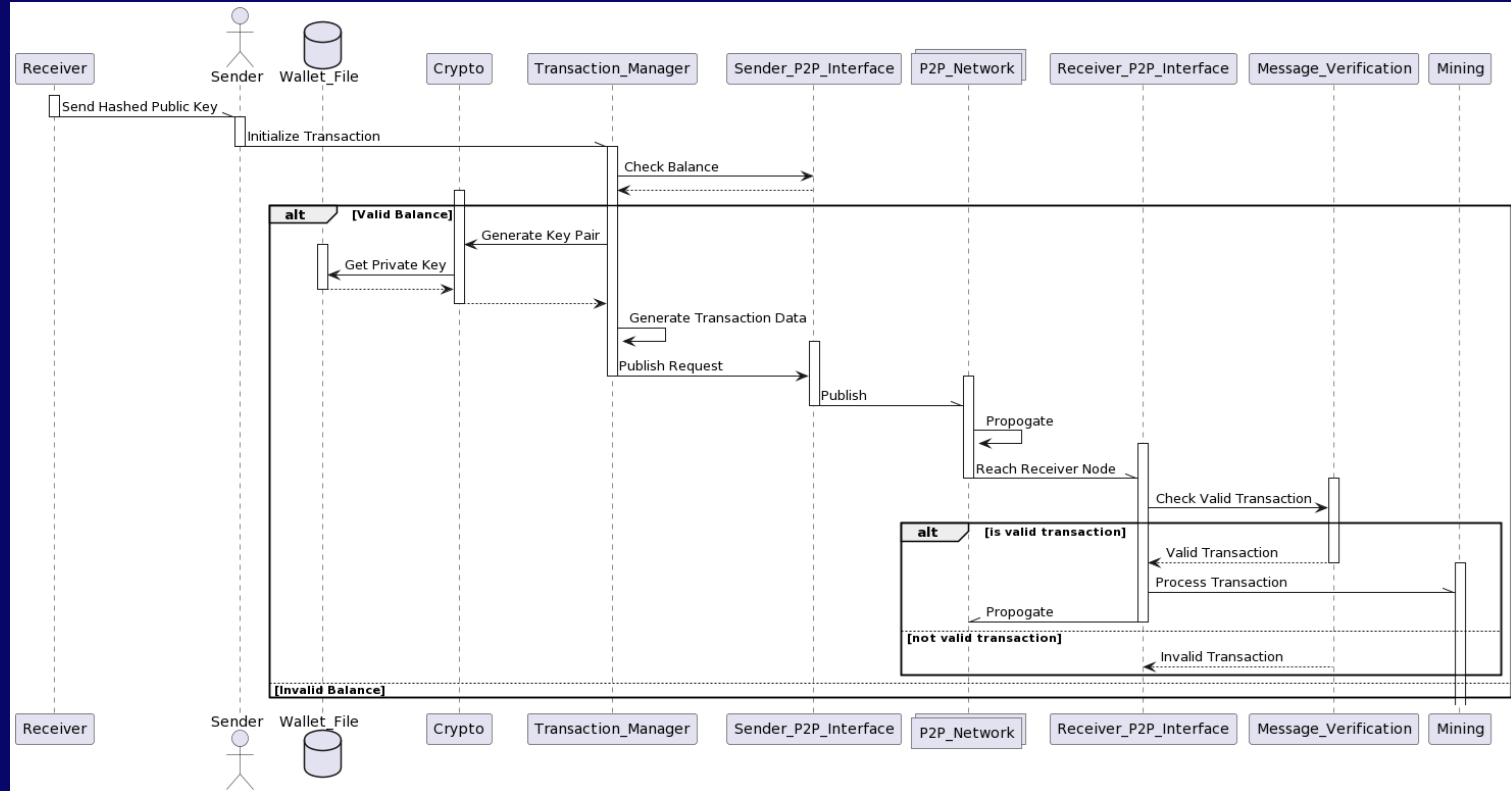
05.

Use Cases

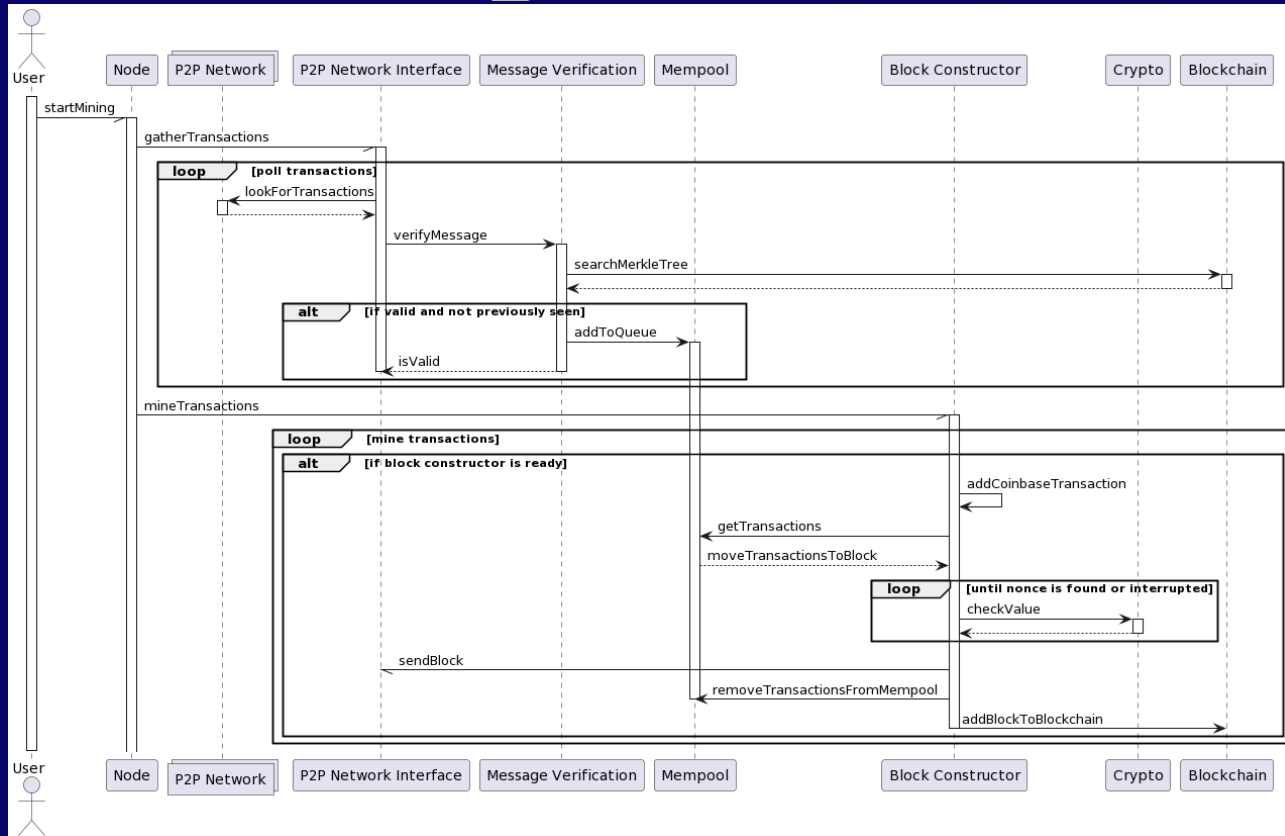


Note: First send of address happens +
outside of bitcoin core

User Transactions Use Case



Mining Use Case






06.

System Evolution



System Evolution

- Bitcoin v0.4:
 - Encryption of wallet private keys (older system now crashed with encrypted wallets)
 - Bitcoin-Qt v0.5:
 - Changes to UI, focus on user experience (reach broader audience)
 - March 2014:
 - Rebranded as Bitcoin Core
 - Changes to bitcoind (the P2P Network Interface)
 - Formerly a server and remote procedure call client, now only a server
 - Bitcoin Core version 11.0
 - Block pruning (however incompatible with wallet)
 - Bitcoin Core 22.0
 - Current version
- 



Division of Work

- Mathematicians work on Cryptography
- Network engineers/programmers work on Peer-to-Peer Interface
- 1. Strong developers in data security work with the data storage system.
- UI designers should work on the UI.



07.

Control and Flow of Components



Data Control and Flow

- Transaction flow
 - Transaction is created and validated by the local client.
 - Transaction is published to the p2p network for mining and group validation
 - Transaction is pushed to the blockchain and minted.
- Blockchain flow
 - Nodes / miners connect to the network.
 - Nodes receive and propagate valid changes in the blockchain
 - Miners solve the proof of work problem in each block





08.






Concurrency





Concurrency



- Intra-Node Concurrency
 - Operation of peer to peer interface while other actions occur
 - Operation of UI while interacting with chain or mining
 - Block Constructor operates while full node continues running
 - 1. Inter-Node Concurrency
 - a. Networking: “best effort” strategy
 - b. Mining: Proof of work done individually then sync
- 
- 
- 
- 
- 



09.

Architectural Styles

Architectural Styles



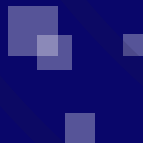
- Peer to Peer
 - Overarching net that connects the subcomponents together
 - Nodes communicating with each other in a decentralized manner.
- Object Oriented
 - Many of the components of a Bitcoin full node can be thought of as objects, such as the transaction object and the block object
 - Utilizes oop principles, such as inheritance and polymorphism
- Repository
 - Blockchain and the wallet file are a form of database
 - These interact dynamically with subsystems that access them





10.

Alternative Architectures



Architectural Alternatives

Likely would represent a significant departure from the principles of decentralization, security, and privacy that are at the core of the Bitcoin network.

- Publish Subscribe Style
 - Pros: Supports transient connections, and communication between nodes
 - Cons: The message bus becomes a single point of failure.
- Client Server Style
 - Pros: Efficient distribution of data, scalability, and no data duplication.
 - Cons: Issues of trust since server controls the currency.
- Sub Architectural Style: Object Oriented Style
 - The full node itself is made in C++ and employs inheritance and polymorphism, typical of this style





11.

Lessons Learned And Limitations






Lessons Learned



- We tried to independently research the components and write the sections of the report → ended up having inconsistent components
- Next time, we are going to work together to research and understand the components holistically, and then only work individually once everyone has a shared understanding of all of the pieces

Limitations

- No interviews were done or observed, only 2 main sources
 - Had to occasionally make educated guesses.
- 
- 
- 



12.

Conclusion





Thank you

Have Questions? Contact Us:

Amy Cui (19ayc1@queensu.ca)
David Courtis (20dhc@queensu.ca)
Jagrit Rai (19jr28@queensu.ca)
John Alajaji (18ja19@queensu.ca)
Logan Cantin (logan.cantin@queensu.ca)
Matthew Vandergrift (19mwv1@queensu.ca)