Video URL: https://youtu.be/O6N4n-aJ8Sg

Concrete Architecture of Bitcoin Core

By: The Foobar Fighters

Roles and Responsibilities

LEADER: Makayla Mcmullin → Overall architecture analysis, abstract, intro/conclusion, reflexion

PRESENTER: Daniel Dickson → Overall architecture analysis, slides, Lessons & Limitations

PRESENTER: Aniket Mukherjee → Subsystem architecture analysis, slides

Maia Domingues → Subsystem architecture analysis, slides

Devon Gough → Subsystem architecture analysis, slides

Lucas Patoine → Overall architecture analysis, Diagrams & Use Cases, Understand analysis

EVERYONE: Data Dictionary, General Research, Naming Conventions

Derivation Process for Concrete

- Examined both the GitHub repositories associated with Bitcoin Core, source code provided with assignment, as well as its Understand model.
- After constructing a general idea of what our concrete architecture looked like, we cross-referenced with our conceptual architecture to see the differences and similarities and further evaluate our model.
- We determined that our model does fit into a peer to peer architectural model, as per our first report while the Bitcoin Core client itself follows a style akin to the Pub Sub style.

Concrete Analysis of Subsystems

Overview

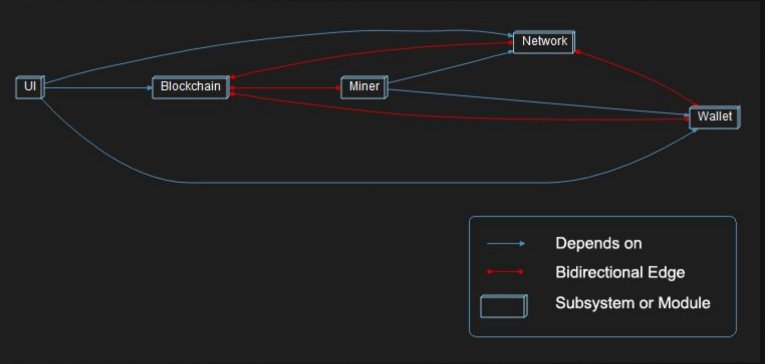
- P2P and PubSub Styles
- Validation Manager is the “Connector”
- Consists of various components that constantly interact

Blockchain

- Mempool and Validation
- Depended on by UI
- Bidirectional dependency with Miner, Network, Wallet

Wallet

- Handles information about keys and transactions
- Depended on by Miner
- Bidirectional Dependence with Blockchain, Network



Concrete Analysis of Subsystems

Mining

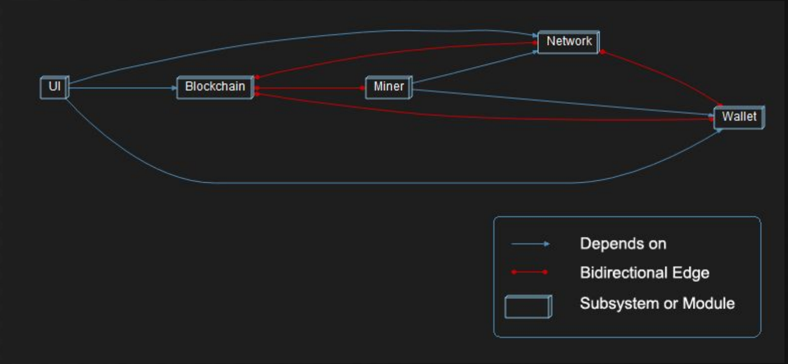
- Generates “blocks” of new bitcoin for users to mine
- Depends on Network and Wallet
- Bidirectional Dependence with Blockchain

Network

- Primary P2P component of the system
- Depended on by UI
- Bidirectional dependency with Miner, Blockchain

Validation

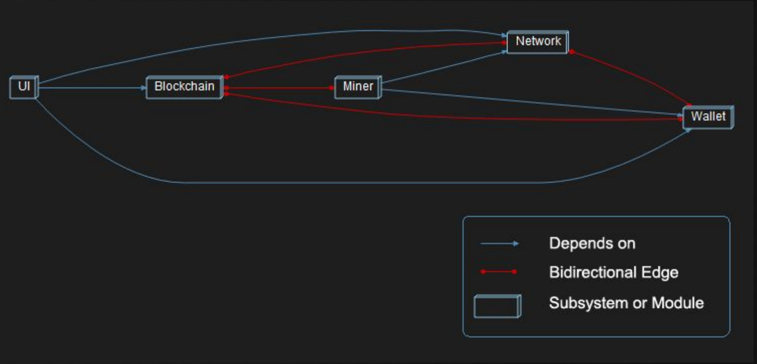
- Validates transactions and activity within Blockchain
- Bidirectional dependency on Miner, Mempool
- Will go more in depth later



Concrete Analysis of Subsystems

Mempool

- Storage space for transactions not yet in Blockchain
- Bidirectional dependency on Validation, Miner



Validation of Transaction Subsystem

- When we examine some files responsible for the validation logic, we can see that there is a lot of data being fed into the src/validation.cpp file, which is responsible for the validation logic of Bitcoin Core, used to validate blocks and transactions coming in and outgoing to other sources
  - Trace.h likely manages to trace back and store wallet data so that transactions between bitcoin wallets can occur correctly and securely, with a level of anonymity, which is why it feeds data into src/validation.cpp to begin with
  - Another thing that makes a transaction valid is the transaction limit on each bitcoin transaction, there's a maximum amount of bitcoin that a transaction can be for, and if it exceeds that limit, then it's an automatic tipoff to the fact that the transaction isn't authentic and should be marked as invalid

Validation of Blocks

- Once a block is good and validated, it can be put into the mempool along with other blocks
  - Users may perform proof of work algorithms to mine these blocks and increase their balance of bitcoin
- Chainstate and chainstate manager validate that the block being analyzed is part of the larger blockchain saved in the network, and compare network version to local version to detect discrepancies
- Also various functions that src/validation.cpp uses to push its own data onto other files essential to bitcoin cores functionality,
  - blockvalidationstate function
    - which would serve as a way to invalidate a transaction or block and ensure that it won't be added to the mempool
- On each block, several tests are run in accordance with all the parameters that blocks on the blockchain should be able to pass, if they are compliant with bitcoin core protocols
  - Adds another layer of security on top of the ones previously established, with the anonymity that Bitcoin Core affords the user

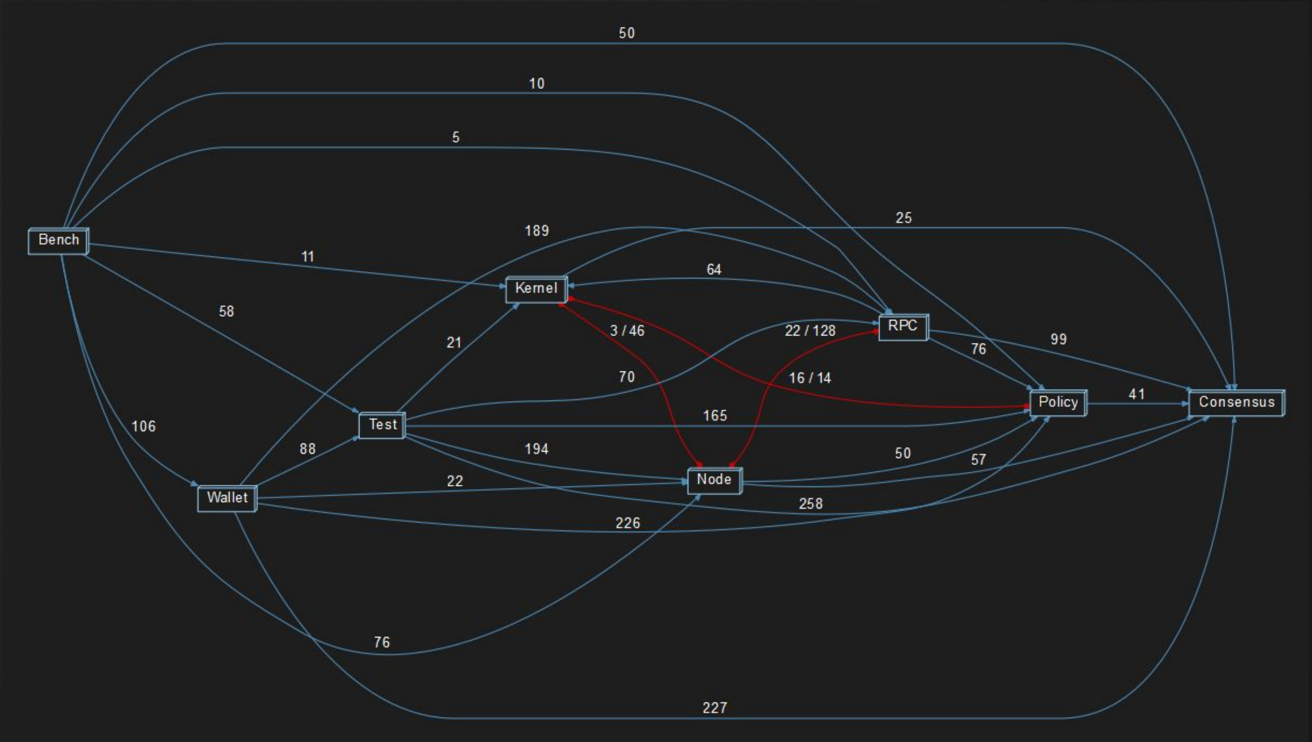
Reflexion Analysis (Overall)

- Same peer-to-peer and Publish-and-Subscribe architecture style
- Discrepancies:
    - No Storage Manager in concrete architecture
        - Increased modularity
            - Simpler code that is easier to maintain
            - Freedom for independent subsystem upgrade and development
            - Error or crash in one memory system does not affect others
    - No internal Connection Manager in concrete architecture
        - Determined not to be needed
        - Increased modularity
        - Gets rid of potential choke point

Reflexion Analysis (Subsystem)

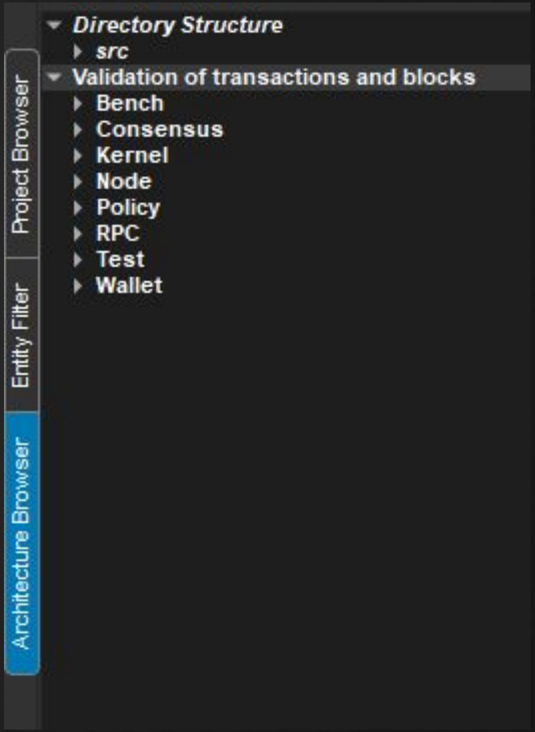
- Main discrepancies found:
  - Concrete architecture includes a clearly-defined dependency between the validation engine and the wallet, compared to the conceptual architecture which did not include a clear dependency between the two.
  - Inclusion of the mempool module more specifically within the validation engine rather than as a separate module in the architectural structure.
  - Enhanced validation engine architectural model in the concrete architecture when compared to the conceptual architecture
    - More robust breakdown of the validation engine and it's components with related dependencies is found, instead of including all validation components under one module

Dependency Graph



Concrete Architecture(Subsystem)

- Managed to break the validation subsystem into a couple of key components, the policy, rpc, test and wallets being the most important parts
- Since every node on the peer to peer network symbolizes a user, when transactions occur, all the data relevant to transactions is transmitted between all the nodes partaking in the transaction.
- Policy is set to make sure that transactions are regulated
- Blocks are also tested repeatedly to make sure that they are valid before being put into the mempool



- *Directory Structure*
  - *src*
- Validation of transactions and blocks
  - Bench
  - Consensus
  - Kernel
  - Node
  - Policy
  - RPC
  - Test
  - Wallet

Lessons & Limitations

- Difficulty accessing documentation of current version
  - Solved(?) by using older documentation, Understand, and manual code analysis
- Difficulty coordinating the group this time
  - Solved by using asynchronous work, though this presented some challenges of its own

- Positive takeaways: Understanding of real-world events and how to adapt to cryptocurrency in the modern world

Thank you for listening!