Concrete Architecture of Bitcoin Core

Presented by group 14, “Kryptic”
https://youtu.be/ljLX8Adxd6Q

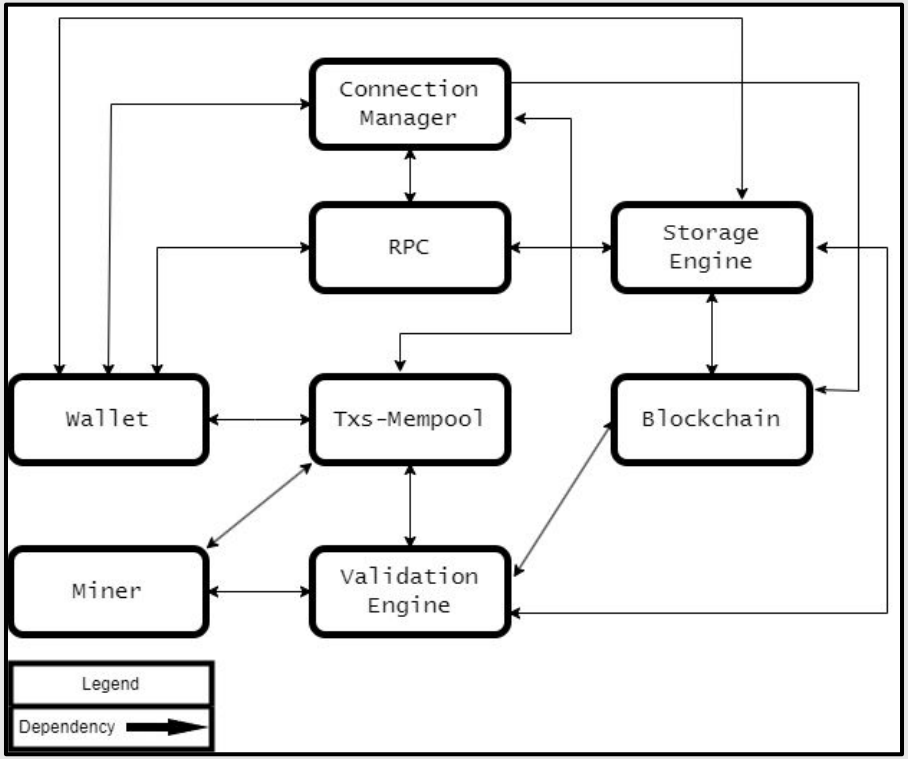
Team Kryptic

- Eric Lam – 20229013 (Leader)
- Andrew Zhang – 20210066 (Presenter)
- Dylan Chipun – 20224970 (Presenter)
- Amy Hong – 20219853
- Sueyeon Han – 20217002
- Asher Song – 20112257

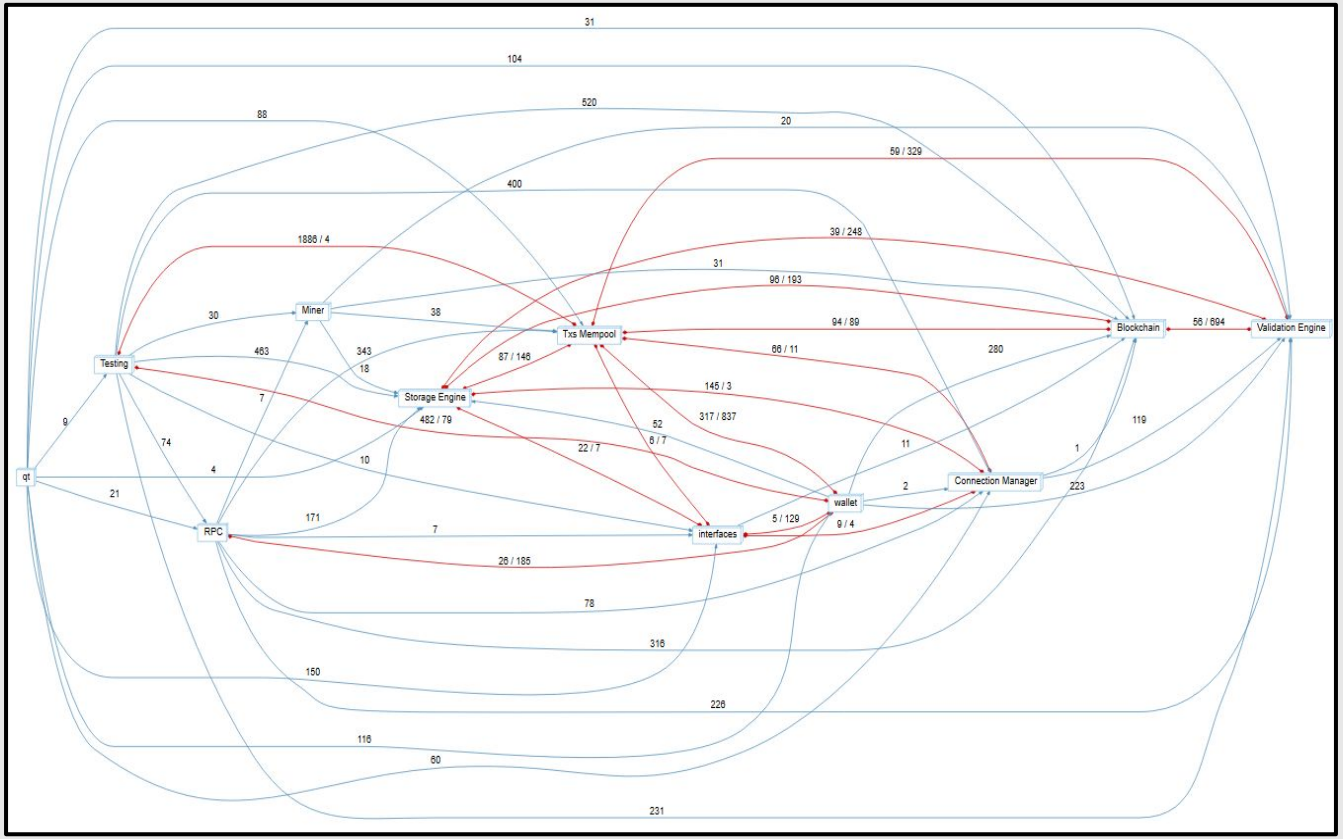
Derivation Process

- Updating the conceptual Architecture
- Using Understand to analyse our options for concrete architecture

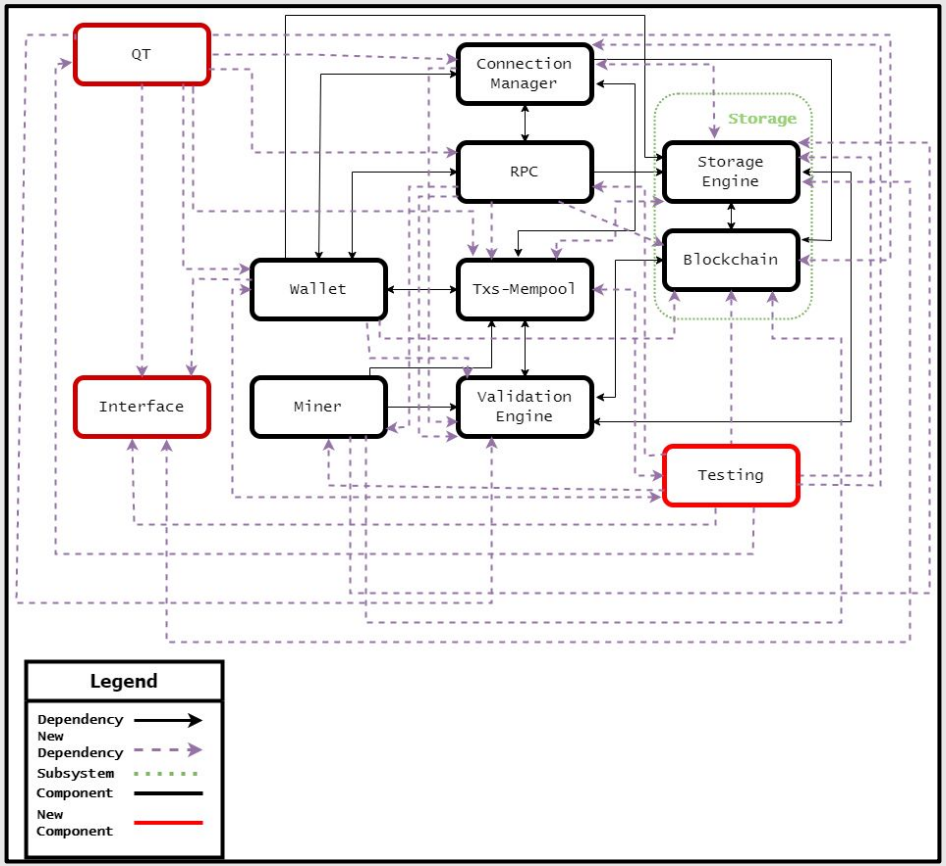
Conceptual Architecture Updated



Understand Analysis Diagram



Concrete Architecture



Divergences at the High Level

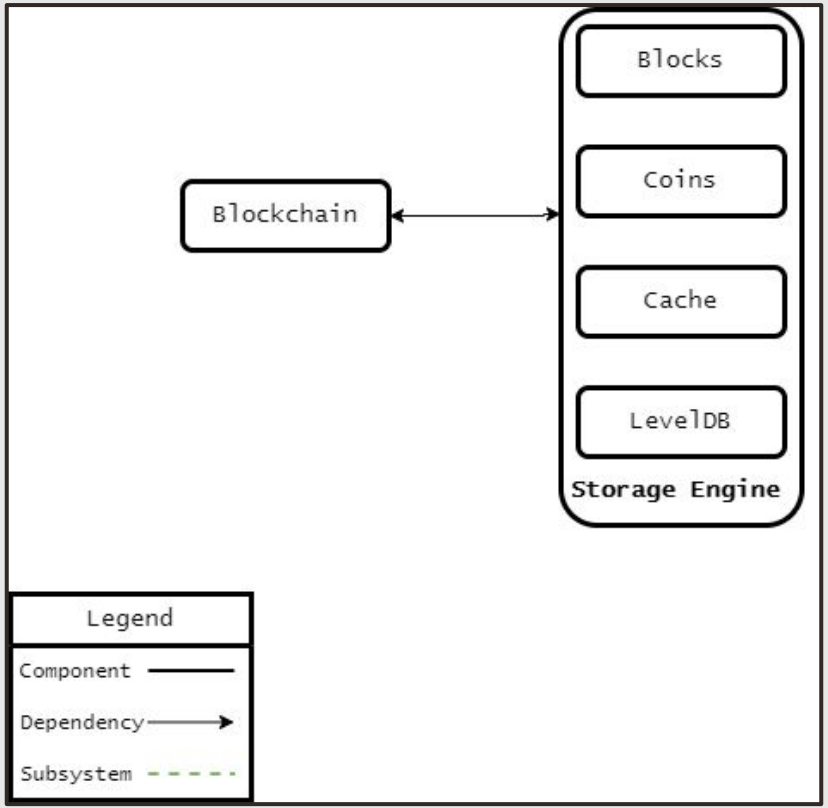
Dependency Changes:

- Testing (added)
- Storage Engine ↔ Txs & Mempool
- Wallet → Validation Engine
- Storage Engine ↔ Connection Manager
- Connection Manager → Validation Engine
- Miner, Wallet → Storage Engine
- RPC → Miner, Txs & Mempool, Blockchain, Validation Engine, Storage Engine
- Miner → Txs & Mempool (not ↔)
- Miner → Validation Engine (not ↔)
- Wallet, Miner → Blockchain

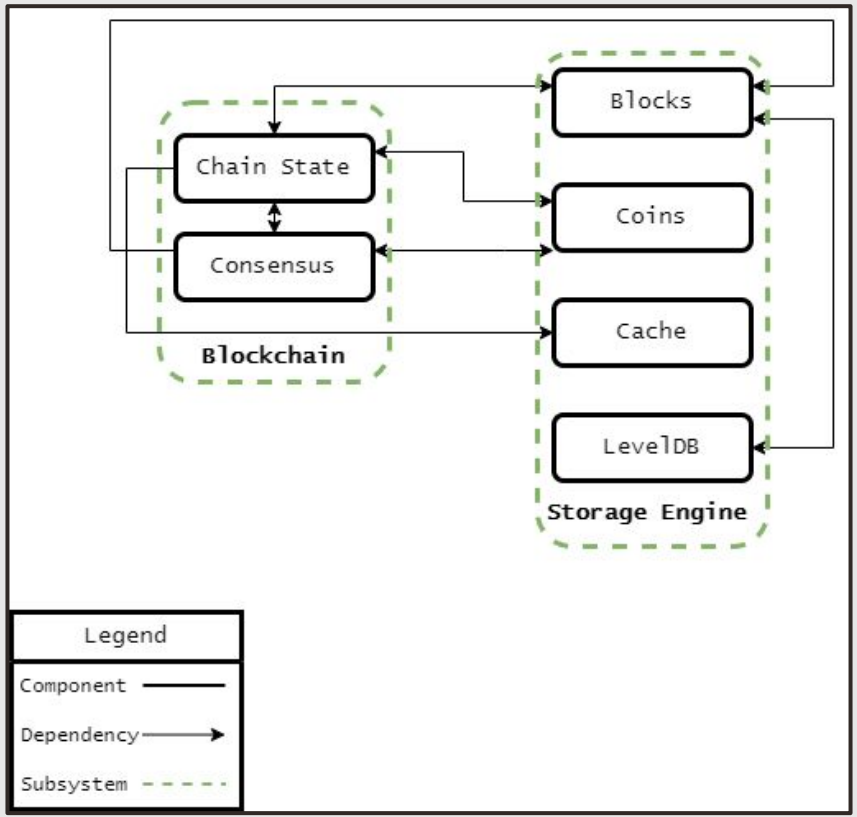
The Storage of the Blockchain

- Peer to Peer style
- no central database
- Components (blocks, chainstates, coins, etc...)

Conceptual Architecture of 2nd level subsystem



Concrete Architecture of 2nd level subsystem

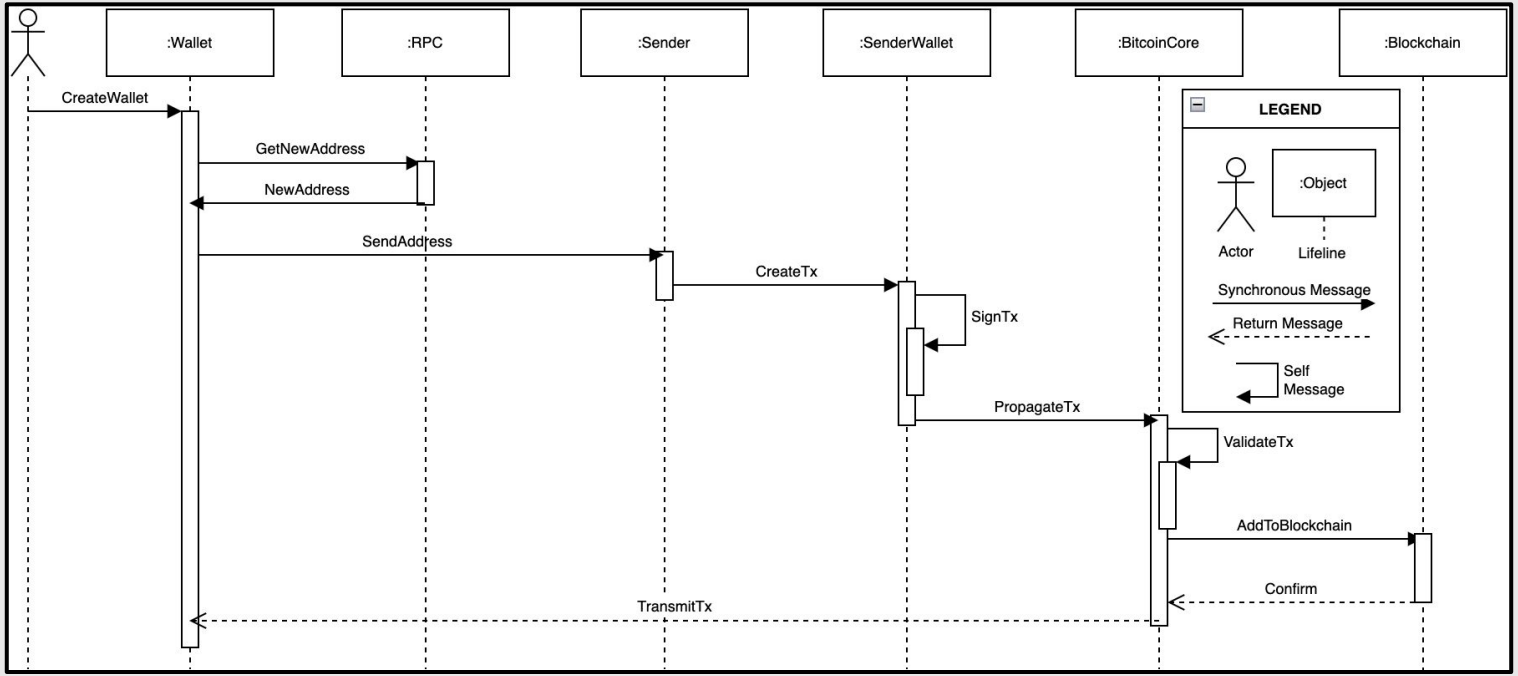


Divergences at the 2nd Level

Added Dependencies:

- Chainstate database
- LevelDB (block storage)
- UTXO (coins)
- Cache
- Consensus

Use Case Scenario



Concurrency and team issues

- The RPC is used in almost all cases
- Bitcoin core’s subsystems are very complex
- Assignment timing with the team was difficult

Lessons Learned

- We understood why conceptual architecture and concrete architecture do not match
- Translating the understand diagram to concrete architecture
- Importance of the report organization and structure

Conclusion