Architectural Enhancement of Bitcoin Core

CISC 322
Wednesday, April 12 2023

Group 39: Yash
Mercy Doan (mercy.doan@queensu.ca)
Yash Patel (19yp29@queensu.ca)
Cain Susko (20cjbs@queensu.ca)
Alice Slabosz (19aas15@queensu.ca)
Shauna Tuinstra (19st58@queensu.ca)
Samuel Lownie (19syl@queensu.ca)

Table of Contents

Abstract

In this report, we put forth an enhancement to Bitcoin Core. Our enhancement entails the incorporation of a Simplified Payment Verification (SPV) wallet into the Bitcoin Core software system. We discuss the rationale behind this decision, the implementation process for such an enhancement, the components affected by this change, and the associated benefits and risks. Utilizing the SAAM analysis, we create two distinct implementations of this proposed enhancement and determine the superior option based on their impacts on stakeholders and their respective Non-Functional Requirements (NFRs). Finally, we present a use case diagram that illustrates how this enhancement will interact with and enhance the given architecture.

Introduction and Overview

The swift expansion and adoption of cryptocurrencies, led by Bitcoin, have required continuous advancements regarding technology and software to accommodate the growing number of users and subsequently, their transactions. While the Bitcoin Core Software system has been instrumental in supporting the Bitcoin ecosystem, there is a need for improvements to enhance user experience, accessibility, and maintain incredibly high levels of security. One such improvement is the integration of a lightweight client, specifically a Simplified Payment Verification (SPV) wallet into the Bitcoin Core software system. This report presents a comprehensive proposal outlining the benefits, required architectural changes, potential risks, and architectural analysis of this proposed enhancement.

The primary objective of this report is to explore the feasibility and benefits of incorporating an SPV wallet into the Bitcoin Core system. An SPV wallet is a lightweight client that offers a more convenient and accessible solution for users who don't have the resources or desire to run a full node. Unlike full nodes, which require the user to download and maintain the entire blockchain, SPV wallets rely on other full nodes to provide the necessary information for conducting transactions. This streamlined approach results in reduced resource usage, faster synchronization times, and the ability to operate on mobile devices, which is appealing to a wider user base.

This report is structured into the following sections to provide a comprehensive understanding of the proposal: 1) Proposed Enhancement, 2) Benefits, 3) High-Level Component/Interface Changes, 4) Impact and Risks of our Enhancement, 5) SEI SAAM Analysis, 6) Use Cases and Diagrams, 7) Data dictionary, 8) Naming conventions, 9) Conclusions and Lessons Learned. Each section delves into specific aspects of the proposed enhancement, offering a well-rounded assessment of its feasibility, benefits, and potential risks.

This report begins with a detailed overview of the proposed enhancement, which highlights the rationale behind introducing the SPV wallet into the bitcoin core software system. The benefits of implementing an SPV wallet are discussed, emphasizing its ability to offer a balance between convenience and security for its users. Subsequently, this report delves into the architectural changes required to support the proposed enhancement, outlining the necessary modifications to components and interfaces within the Bitcoin Core software system. Potential impacts and risks associated with the proposed architectural changes will also be explored and examined, which will offer a well-rounded understanding of the enhancement.

To further solidify this proposal, an architectural analysis using the SEI SAAM method is conducted to evaluate two different approaches for realizing the proposed enhancement. This analysis involves identifying major stakeholders and their non-functional requirements (NFRs) which are concerning the enhancement. This report evaluates the impact of each proposed solution on the identified NFRs and stakeholders, ultimately determining the most suitable approach for implementing the SPV wallet.

Additionally, the report includes sequence diagrams to illustrate the flow of at least one use case through the proposed architectural modifications, providing a visual representation of the changes. To aid in understanding, a data dictionary and naming conventions section are included within the report, defining key terms and concepts related to this proposal.

To conclude, this report will present an extensive proposal for the integration of a lightweight SPV wallet into the bitcoin core software system. By examining the benefits, requirements within the architectural changes, potential risks, and conducting a detailed SEI SAAM analysis, this report aims to provide a comprehensive basis for further development and implementation of the proposed SPV wallet enhancement. This will ultimately contribute to a more accessible, efficient, and secure Bitcoin Ecosystem.

Proposed Enhancement

Bitcoin Core is a peer-to-peer electronic payment system, where each node is a computer running the Bitcoin software. Bitcoin Core is a full node client, meaning it stores a full copy of the accepted blockchain and is able to precisely verify transactions according to Bitcoin protocols and rules. After validating proposed transactions, the node will broadcast it to the other nodes it is connected to. Once a sufficient number of nodes are in agreement on the transaction's validity, it is added to a pool of valid transactions.

The drawback of using full nodes is the computational cost of verifying the entire blockchain. The blockchain is a database of transactions that have occurred, and is perpetually increasing in size. Full nodes must download the entire blockchain to check thousands of individual blocks and transactions. This can be a strain on user resources, where each check means spending computing power, computational space, and computation time. While transaction security is of the utmost importance, it is possible to verify transactions without downloading the full blockchain.

We thus propose a lightweight client, such as the simplified payment verification (SPV) wallet. SPV wallets were described in the Bitcoin whitepaper, allowing transaction verification with Merkle proofs instead of checking the full blockchain. SPV wallets rely on other full nodes to provide them with the necessary information to prove a transaction's validity and prevent double spending. We discuss the benefits, risks, and impacts of SPV wallets below.

Benefits

SPV wallets have many benefits. The Bitcoin wiki [1] lists the following advantages:

- a wallet can store all necessary block headers in around **50 MB** - this covers the entire block chain (as of January 2020, with 80 bytes per block and around 620,000 blocks in the chain). The total **grows linearly** at around **4 MB** per year (i.e. it increases by 80 bytes with each block mined, regardless of the size of that block).
- contrast this with the **hundreds of gigabytes** which would be required to store the entire chain, if SPV were not being used.
- The size of the data required for the Merkle paths is of maximum $64log_2n$ bytes, where n is the total number of transactions in one block.

The reduced cost of running Bitcoin Core is very convenient for users who do not have the necessary resources. Clearly, SPV wallets mitigate the problem that it is meant to solve, and increases the accessibility of Bitcoin Core to a much larger audience of casual users. Some other benefits include security and portability. While SPV wallets do not manually check the entire blockchain, the SPV verification method can discover fraudulent transactions very quickly. In addition, SPV wallets can be used on mobile devices so that users can make transactions without being at their computer. Overall, SPV wallets provide a good balance between convenience and security for many users.

High Level Component/Interface Changes

Adding Simplified Payment Verification (SPV) to Bitcoin Core would include four high level component/interface changes. The first change would be to change the block validation. Currently Bitcoin Core (full node implementation) downloads the entire blockchain and validates every block, but with SPV implementation the node would only download block headers and use them to validate transactions.

The next change would be to the peer-to-peer network component. Essentially the only difference here is that SPV nodes would rely on full nodes to provide block headers and Merkle branches in order to validate transactions.

The third change involves wallet implementation. SPV wallets and current Bitcoin Core nodes have different requirements for wallet implementation, since SPV wallets need to be able to quickly retrieve the UTXO set for transaction verification. This means there should be changes to the wallet code in order to support efficient indexing and retrieval of UTXOs. The final change would be to the user interface, since the SPV implementation requires UI to only display transaction data and basic wallet functionality, while the current Bitcoin Core implementation can display more detailed information about the blockchain state.

Impact and Risks of Enhancement

There are several impacts and risks of enhancing Bitcoin Core with SPV. The impacts include increased accessibility, reduced bandwidth and storage requirements, and faster transaction times. Increased accessibility is due to the fact that SPV doesn't run a full node and therefore attracts those who may not want to. Reduced bandwidth and storage requirements happen because SPV allows users to verify transactions with only a portion of blockchain. This reduces the bandwidth and storage requirements for running a node. Faster transaction times happen when simplifying payment verification, transactions will be much quicker.

On the other hand, risks include reduced decentralization, increased security risks, and difficulty in validating consensus rules. Reduced decentralization occurs because SPV clients rely on full nodes to feed them transaction information. When there aren't enough full nodes, which could happen, this could lead to a more centralized network. Increased security risks happen since SPV clients don't verify the entire blockchain. This leads to security risks when the full node is compromised. Difficulty in validating consensus rules occurs because SPV clients might be unable to validate all consensus rules of the network, which could lead to problems with fork detection or other consensus-related problems.

Adding SPV wallet support would initially require developing the additional feature. However, due to the simplicity of its tasks, the maintainability would rarely need to be changed once implemented. Regarding evolvability, the SPV would have to be updated if the transaction validation method or actual block implementation is changed. Similarly, the SPV would have to be tested if its dependent components are modified. However, because transaction checking is so vital to Bitcoin Core, it is unlikely that the overarching method of verification will be drastically changed. Since the SPV is meant to help users use less resources when running a node, the performance of the system becomes more efficient at the cost of less thorough transaction validation. Overall, implementing the SPV wallet is likely a one-time commitment, with few updates going forward.

SEI SAAM Analysis

The Carnegie Mellon Software Engineering Institute's Software Architecture Analysis Method is a set of procedures and policies outlining best practices for assessing the feasibility of non-functional requirements within a system architecture. Using the concrete architecture we derived in our previous report, we can begin to create a new architecture based on the "five basic functions of user interface software".

Table 1.

The five basic functions of user interface software.

FC	**Functional Core**	Performs primary domain oriented functions and data manipulation
FCA	**Functional Core Adapter**	Aggregates domain data to be used by the Functional Core
D	**Dialogue**	Mediates the domain specific and presentation specific components
LI	**Logical Interaction**	Provides toolkit of independent objects (data structures) to the Dialogue
PI	**Physical Interaction**	Provides toolkit for input & output

Note - adapted from SAAM: A Method for Analyzing the Properties of Software Architectures, *Section 3.1*

Using the functions in Table 1 as guideposts, we followed the structure laid out by SAAM, which is made up of nodes: Processes, Sub-processes, Active data-repository, and Passive data-repository. These nodes are linked with thin edges, indicating data-flow -- or thick edges, indicating control-flow (Kazman, 2007. Section 2.2). Using these guidelines, We derived the architecture:

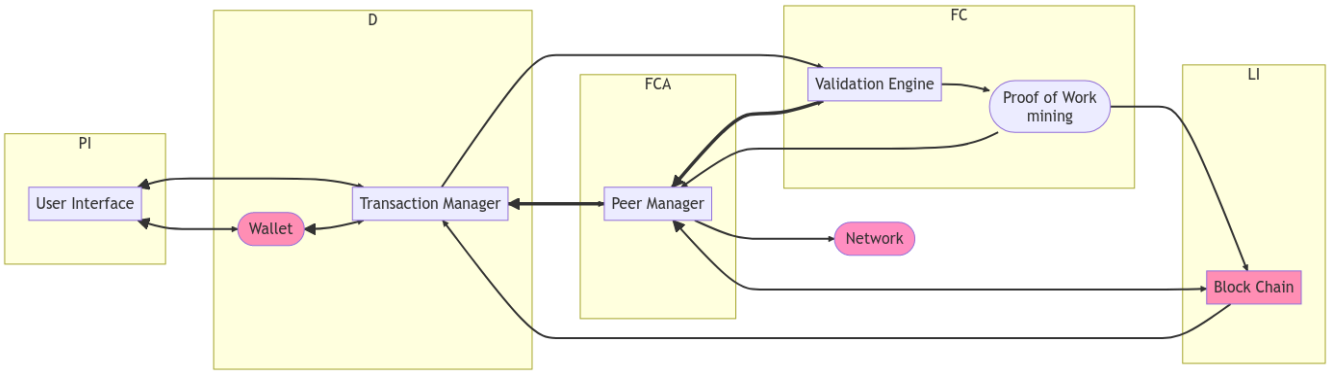

Figure 1.

Bitcoin Core Architecture (annotated)

Note: Uses structure outlined in SAAM: A Method for Analyzing the Properties of Software Architectures, *Section 2.2*

After determining this architecture, we may now identify the non-functional requirements which must be maintained through the enhancement process. Once these are properly defined, we can analyze whether our proposed enhancements are feasible within the architecture -- as well as how it affects stakeholders and their non-functional requirements. On initial examination of the architecture, it appears that the Bitcoin Core architecture is sufficient to support the enhancement of an SPV wallet as much of its logic and structure is internal. To further provide justification for this enhancement and its architectural changes, we complete the SAAM analysis.

Stakeholders and their NFRs

Users refers to the current user of Bitcoin Core and naturally, they are concerned with the system that belongs to them. From this, we derived the main NFRs for this enhancement were: Optimization, Security, and Scalability. As we discussed in our previous report, in order to even use the software, the user must perform an initial block download of the blockchain, which is roughly over 340 gigabytes as of the time of this writing, and most likely will only continue to grow. Not only that, but this process takes a considerable amount of time and causes a high usage for the network and CPU.

There are issues with using one's machine as a full node, as not only does it require users to run their machines for at least 6 hours a day, but it also comes with its own set of risks. Several people have placed parts of known computer viruses in the Bitcoin blockchain, and while it can't infect the user's computer, the computer's antivirus will most likely quarantine the data anyway, making the program less efficient. There are also people who want to disrupt the peer-to-peer network that is essential for running Bitcoin Core, such as limiting a user's available download bandwidth.

The nodes of the Bitcoin Core peer-to-peer network are also stakeholders in this enhancement. The full nodes' have a critical role maintaining the consensus that provides Bitcoin with its secure transaction capabilities. These nodes also require that there are enough full nodes in the network so that the system can perform all necessary network tasks. So, one of the full nodes' NFRs is concerned with capacity. The exact number of full nodes supporting the network that is necessary for all network requests to be handled accurately and quickly is outside the scope of this report. But given the rapid rate of growth and expansion in the world of cryptocurrency, this NFR can be phrased as "the system must be able to handle 400,000 requests across the network while maintaining speed of responses". The other major NFR for the full nodes in the network has to do with security, specifically encryption and integrity. In order for consensus to be maintained and the blockchain to continue being the absolute authority of transaction history, data passing between a full node and an SPV wallet must be completely accurate, secure, and unchanged.

This is also without taking developers into consideration. When attempting to improve upon something that is at the core of the software and essential to how it functions, it must either be more efficient or more feature-rich than what was previously there. Not only that, but developers are used to how the code works and attempting to alter something that is so essential, must be implemented with intent and room to grow in the future. The developers are concerned about the maintainability and privacy of the system. Any data leaks caused by a full node accessing address information that it is not authorized to would cause an immense backlash from users and lack of trust in the Bitcoin system.

With these issues in mind, the derived NFRs and the SPV Node address these in different ways. The SPV Node bypasses optimization by simply not having a stored blockchain, meaning there is no need for the user to download over 340 gigabytes worth of data at any point. Privacy was an initial issue, as an SPV node's requests for specific data can inadvertently reveal the addresses in their wallet, destroying the user's privacy. The solution to this was in bloom filters, which allow SPV nodes to receive a subset of transactions without revealing which addresses they are in. Finally, the implementation for SPV Nodes already exists within the repository for BitCoin Core, and while not every developer would have used this code, it would not take long to understand given both the documentation and the actual code.

Impact on NFRs

Since the SPV uses Merkle branches to verify transactions, NFRs about accuracy and security must be extended to these calculations. No matter the implementation, SPV wallets must always accurately calculate the Merkle proof to validate transaction data.

One possible way to implement this enhancement is to add an SPV component to the architecture that acts in parallel to the existing wallet component. This SPV component would have dependencies on components that the existing wallet also depends on and components that depend on the wallet would also depend on the SPV component. There are some exceptions to this due to the nature of the SPV. For example, the SPV component does not depend on the validation engine as it has its own internal methods for calculating Merkle proofs to validate transactions.

This implementation would require the user to download more than just the SPV client but would still save significant space as they would not download a copy of the blockchain. They also would still not need to use computational power to run a full node for long periods of time. Using this SPV client would

require the presence of the following additional components: Peer Manager, UI, Utilities, the Transaction primitive, and the Block primitive. The architecture of this implementation would be internally object-oriented, as per our group's findings in A2. However, the overarching architecture would be the Peer to Peer style, as every instance of Bitcoin Core is still part of the P2P network.

Some risks include having to integrate the SPV component with the other components, which are already very interconnected. It would need to be closely tested to ensure there are no unexpected dependencies and that the code is secure. This would involve unit testing for the SPV component to ensure the Merkle proofs work as expected. Its dependent components should then be tested to ensure that the system works properly with the addition of the SPV component. Once implemented, the SPV component should be very maintainable, since its core function (verifying transactions with Merkle proofs) will not need to be updated frequently, if ever. The performance of Bitcoin Core may be improved, because SPV wallets save computational time and space. This allows users with low resources to access Bitcoin Core, and for users with more resources, the SPV wallet may speed up their operations.

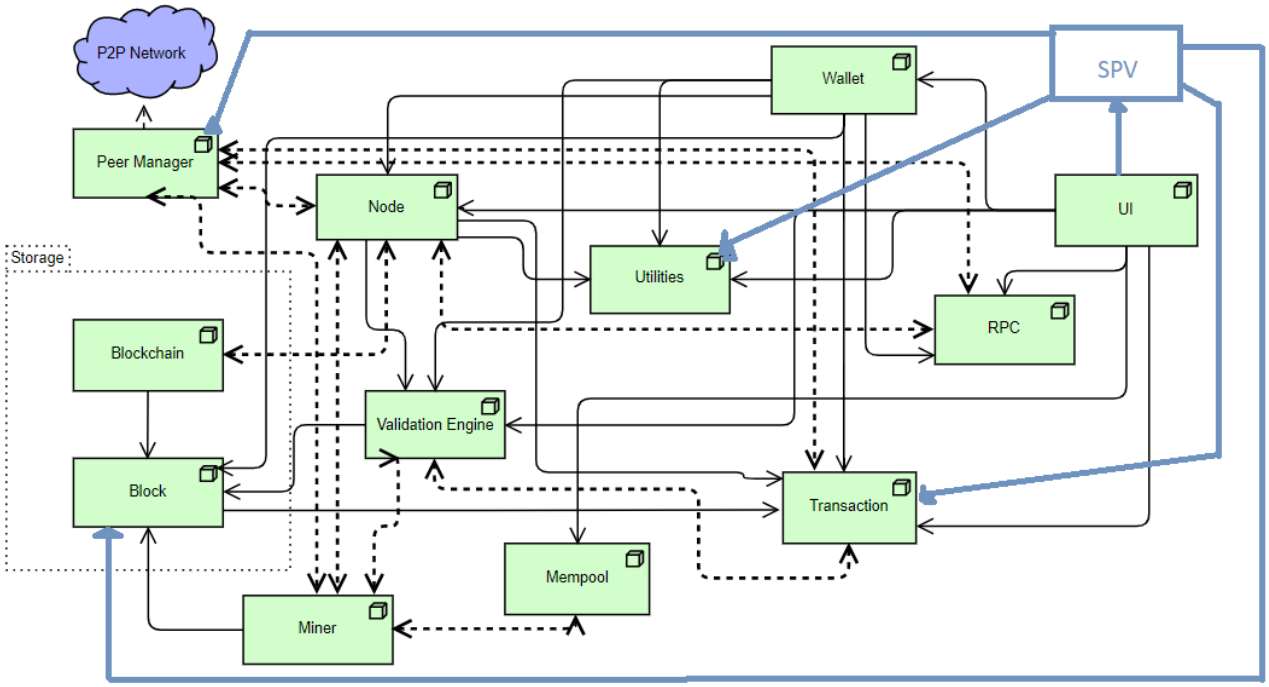

Figure 2

Architecture changes for including SPV using a parallel approach

A second possible way to achieve this enhancement could be to add an SPV component independent of the rest of the Bitcoin Core system. This would require that the SPV component also have its own UI subsystem to allow users to access their funds and create transactions. The only components that the user would have to have locally are the SPV client, the UI, and the peer manager. All other necessary components are accessed by connecting to full nodes through the peer-to-peer network. The architecture for this implementation is Peer to Peer, since the SPV becomes directly part of the P2P network. There would be no impacted directors or files other than the peer manager, since a node's peer manager will interact with the SPV's peer manager.

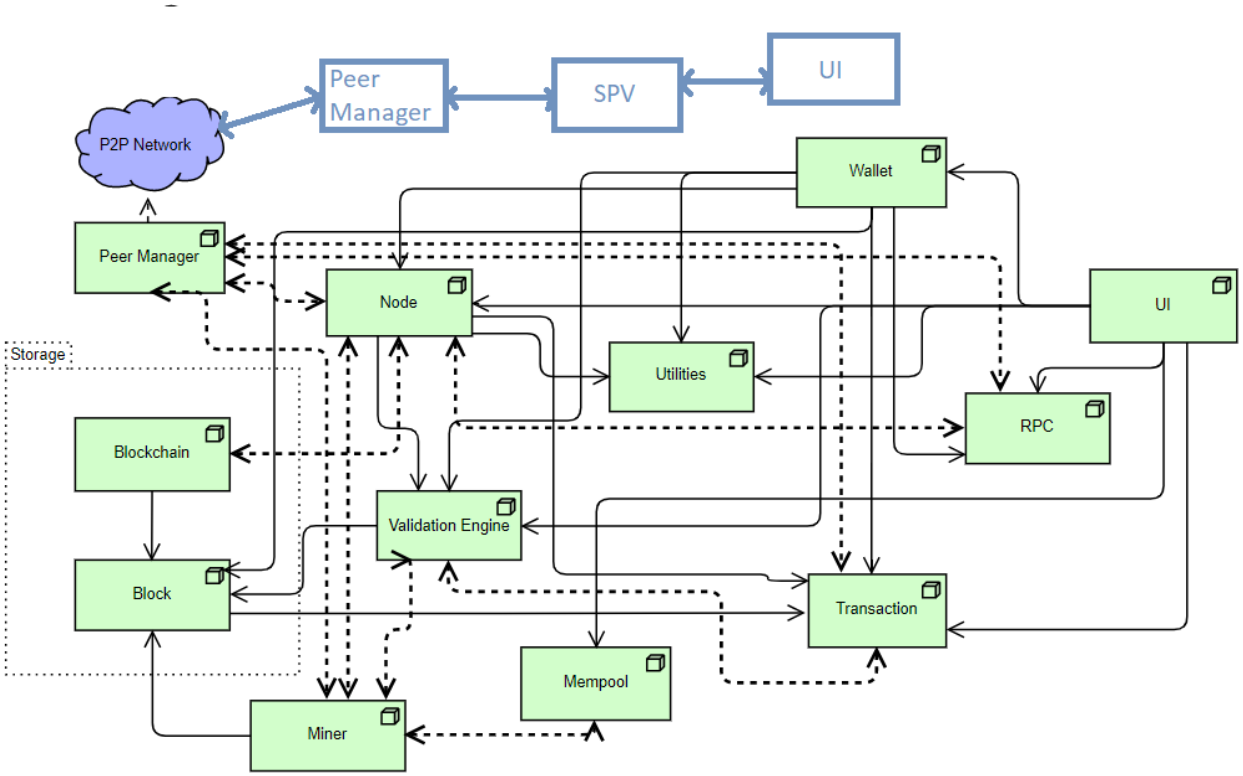

Figure 3

Architecture changes for SPV using an independent approach.

Using the parallel implementation would have a slight impact on users' NFRs relating to scalability as the size of the download necessary to run an SPV client would increase slightly in comparison with the outside implementation. However, this download size would still be much smaller than that of a full node, so the NFRs would still be satisfied. The NFRs regarding the full nodes as stakeholders would remain unchanged. The developers' NFRs would also remain unchanged. The independent implementation would maintain the users' NFRs.

Some risks of this implementation are the security of the new components. The Peer Manager, SPV component, and the UI will have to be thoroughly tested for exploitable vulnerabilities. Adding a new interface means adding more ways malicious users can hack into and manipulate the network. Traditional cybersecurity protocols should be taken, such as penetration testing. Once secured, the SPV will have to be maintained separately, meaning there must be developers available to update it if the P2P network is drastically modified. To use the SPV wallet, users would need to access a separate UI than their node's UI, which may be tedious. However, having the SPV subsystem be part of the P2P network and not a user's node may help save user resources.

Evaluation

Overall, the two methods for implementing this enhancement both provide features which adhere to most of the stakeholders' NFRs. However, from an architectural viewpoint, they are not equal due to their respective implementations.

The *parallel* approach involves having the new simplified verification process component be connected to numerous other system components. The new SPV module can be seen to be a part of the Functional Core, because verification -- whether simplified or not -- is the primary domain specific function Bitcoin Core performs. Thus, from Figure 1 and Figure 2, the *parallel* approach suggests that the SPV should be in the Dialogue or Functional Core Adapter in order to neatly interact with the Transaction Manager and Peer Manager. While this is a feasible approach to our enhancement, it doesn't take advantage of the existing Bitcoin Core architecture.

The *independent* approach entails connecting the new simplified verification to only the Peer Manager and User Interface. There is already an existing control-flow edge between these two components in Figure 1, so the addition of the SPV module requires the addition of only one data-flow edge, allowing it to be contained in the Functional Core.

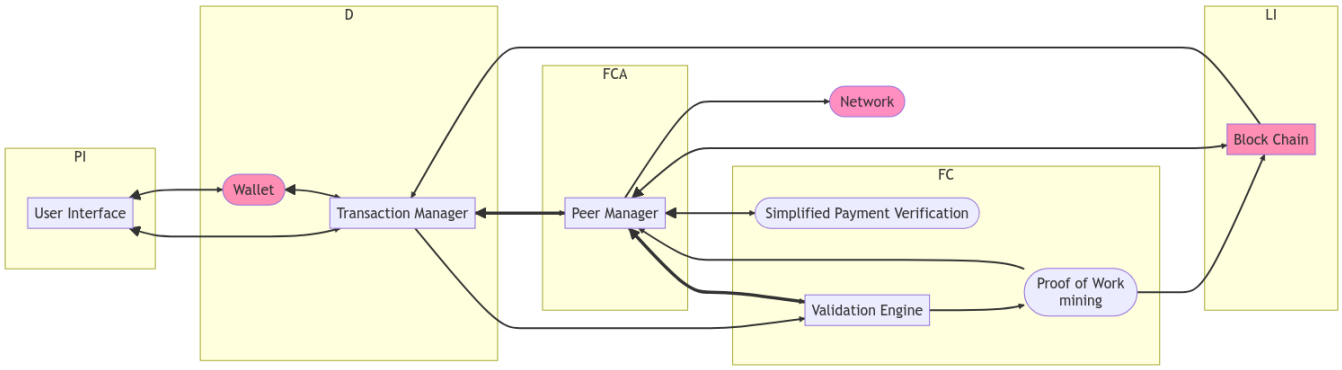

Figure 4.
Bitcoin Core Enhanced Architecture (annotated) based on Figure 1 and Figure 3.

Thus, using the Software Architecture Analysis Method laid out by the Carnegie Mellon University, Institute of Software Engineering, we have determined that our *independent* approach should be the ideal approach for our proposed architectural enhancement, based on our derived architecture of Bitcoin Core and the stakeholders' non-functional requirements. The independent approach would require additional directories to be added to the codebase (to provide functionality for the SPV and its Peer Manager and UI), as well as small alterations to existing files in order to provide the necessary connection between an SPV wallet and full nodes. Functions must be added to the Node component to access the blockchain and return only block headers to the requesting SPV wallet. Additionally, the peer manager of the full node clients must be able to handle these requests securely.

Ensuring that this enhancement integrates into the existing Bitcoin Core landscape and improves user experience will involve two kinds of testing. The first is unit testing. Each added component (SPV, SPV Peer Manager, and SPV UI) will be tested individually to ensure its functionality. The SPV component will be tested on its correctness in calculating Merkle proofs and handling transaction creation. The SPV Peer Manager will be tested on maintaining a list of peers, sending and handling requests. The UI will be tested for ease of use and portability. The next form of testing will be wider testing of the entire system to make sure that the new components work with the existing ones and support each other. This will involve testing the retrieval and transfer of block headers, broadcasting transactions from an SPV wallet, and insurance of privacy when the SPV wallet requests information from a full node.

Concurrency is also an issue. Developing the SPV functionality for the parallel approach initially would be simple. It must simply be able to perform Merkle proofs. Concurrency comes into play when integrating the SPV component to the 3 other dependent components, some of which might be being updated at the same time Adding SPV features might delay some updates to the dependent components in order to ensure dependencies are properly tested. Implementing the SPV wallet with the independent approach would require dedicated developers to build most of the SPV wallet independent of the main Bitcoin Core system When the UI is ready to launch, the main Bitcoin Core architecture must implement functionality to accept the new subsystem in the P2P network This would require updating the main architecture's Peer Manager and testing it with the SPV wallet's peer manager in the P2P network

Data Dictionary

Block	A container data structure that schedules transactions for inclusion in the public ledger, the blockchain; a block is made up of two components: a header and a body
Fee	The amount remaining when the value of all outputs in a transaction are subtracted from all inputs in a transaction; the fee is paid to the miner who includes that transaction in a block.
Node	A computer running the Bitcoin software
Wallet	An application that serves as the primary user interface, controls access to the user's money, manages keys and addresses, tracking their balance, and creating and signing transactions
Full Node	A network node containing a wallet, miner, full blockchain database, and network routing. These can autonomously and authoritatively verify any transaction.
SPV Node	Also known as a lightweight node. Contains a wallet and networking routing but cannot mine or use the local blockchain to verify transactions.

Naming Conventions

GUI	Graphical User Interface
UI	User Interface
P2P	Peer-to-Peer
RPC	Remote Procedure Call
UTXO	Unspent Transaction Output
SPV	Simplified Payment Verification

Use Cases and Diagrams

Use Case 1: SPV Wallet Transaction

User makes a transaction which is verified using SPV within our determined architecture.

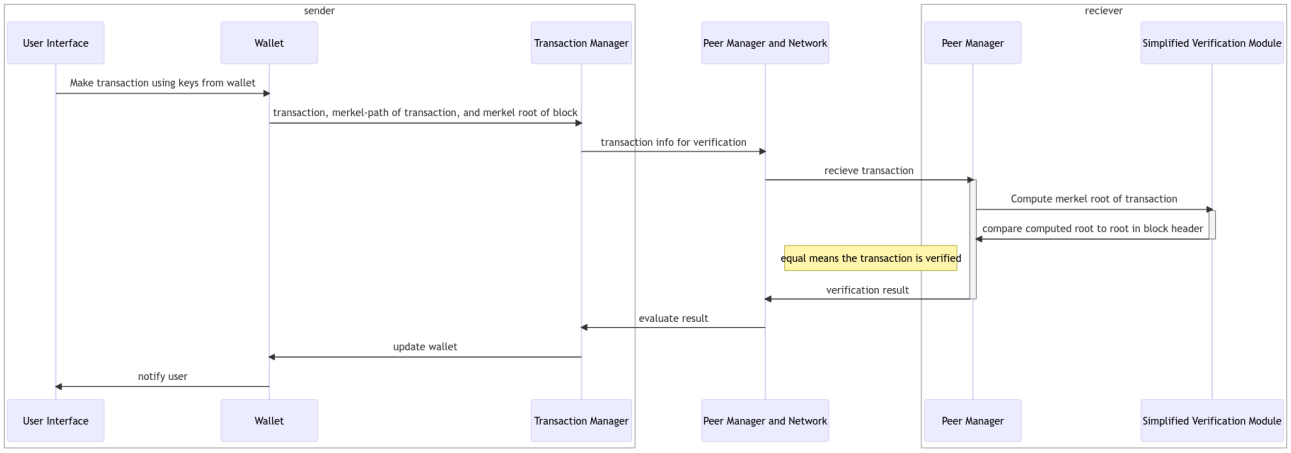

Figure 5.

SPV Transaction sequence diagram

Use Case 2: Non-SPV Wallet Transaction (for comparison)

User makes a transaction that is verified without the SPV wallet for comparison. The process requires much more thorough processing and validation. In particular, the validation engine in a full node can be very computationally expensive.

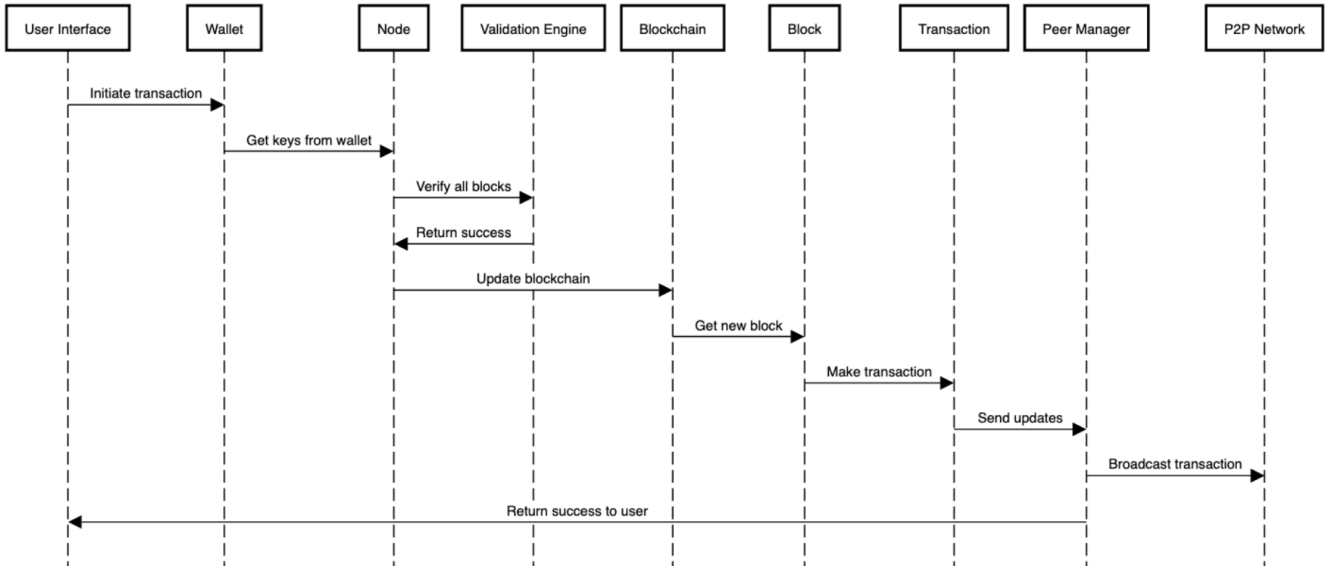

Figure 6.

Non-SPV Transaction sequence diagram

Lessons Learned

Selecting a feature of Bitcoin Core to enhance involved considerable thought, as the system was designed to serve as a foundation for future Bitcoin clients, and the potential ideas for enhancements and their implementations were extensive. We ultimately chose to incorporate an SPV wallet into Bitcoin Core due to its lower resource requirements compared to a full-sized node, particularly in storage space and portability.

Developing the proposal architecture necessitated further research on SPV nodes, their practical applications, implementation, and strengths and weaknesses. Through this research, we acquired valuable insights into the intricacies of Bitcoin Core. When implementing the enhancement, we also considered the high-level components affected by such a significant change, specifically the peer-to-peer network and wallet implementation.

In this report, we also gained experience in performing an SEI SAAM analysis, which required additional research since it was not covered in lectures. By examining reports from previous years and information found online, we acquired a solid understanding of how to conduct such an analysis in the future, if necessary.

As a group, we managed our time well, efficiently allocated tasks, and overall worked well together. It was a very busy semester, but we pulled through and contributed to each other's learning. We will definitely walk away from this project with an appreciation of the software development life cycle and the importance of teamwork.

Conclusion

The proposed enhanced feature of incorporating a SPV wallet into the BitCoin Core system comes with the benefits of reduced resource usage, increased portability by being able to be run on mobile devices, and quicker synchronization times. The risks of implementing this enhancement include security, reduced decentralization, and difficulty in validating consensus rules.

By performing the SEI SAAM Analysis, the team was able to devise two unique realizations of this enhancement by identifying the major stakeholders, the most important non-functional requirements for these stakeholders. Assessing further, the team evaluated how both implementations affected the previously identified stakeholders and NFRs and were able to determine that our independent approach was the best overall based on this analysis.

References

[1] Bitcoin Association. (2022, August 14). *Simplified payment verification*. Simplified Payment Verification - Bitcoin Wiki. Retrieved April 12, 2023, from https://wiki.bitcoinsv.io/index.php/Simplified_Payment_Verification

[2] Kazman, R., Bass, L., Abowd, G., & Webb, M. (2007). Saam: A method for analyzing the properties of software architectures. https://doi.org/10.21236/ada633431