Bitcoin Core Enhancement Proposal Report

CISC322

Monday, April 10th, 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

ABSTRACT

This report aims to propose an Architectural enhancement for Bitcoin Core. The enhancement is cancellable transactions. For this feature, the proposal describes its motivation and benefits, along with two methods of implementation. In addition, the SAAM analysis will identify key stakeholders, and NFRs and compare and contrast the two implementations concerning said stakeholders and NFRS. Then, the impacts of the enhancement on the existing subsystems and components of Bitcoin Core will be discussed. To demonstrate the benefits and functionality of the new feature, two use cases will be described. Finally, the report concludes with the possible risks and testing mechanisms to ensure proper integration.

INTRODUCTION & OVERVIEW

Over the last couple of months, our team has achieved a thorough understanding of the functionalities and architectures of Bitcoin Core. Through the past two reports, we have studied the components of Bitcoin Core to develop a conceptual architecture, followed by the formulation of Bitcoin Core's concrete architecture and delved into its details. This report's purpose is to propose a new feature we feel Bitcoin Core could implement. The newly proposed enhancement allows users to cancel unmined transactions. We will demonstrate two implementations of our proposed enhancement, where each will contain a conceptual architecture and explanation of any new interactions with existing components. The report will use an SEI SAAM analysis to determine the stakeholders that the new feature affects and choose the best implementation to proceed with based on their individual advantages and disadvantages with respect to stakeholder NFRs and through logical reasoning. The next section will take a look at how the new feature impacts both high and low-level conceptual architectures of Bitcoin Core, along with the directories and source codes that are impacted by the newly implemented feature. Finally, this report will discuss how we plan on going forward with testing the newly proposed feature, along with the potential risks that come with it.

PROPOSED ENHANCEMENT

Description

Our proposed enhancement feature aims to increase the ease of use and expand the functionality of Bitcoin Core by allowing users to cancel unmined Txs. With this feature, a user will be able to revoke their Tx that has not been mined or proved by miners and have it returned to the unconfirmed transaction pool (UTXO).

In a traditional centralized banking system, even if the user realizes they made a mistake right after transferring money to someone, there is no way to cancel or revoke it. They must contact the bank and wait for intervention which might be impossible. For example, RBC Royal Bank says payments are "instantaneous and final" and they will not reverse such payments[1]. However, decentralized finance can help this problem by leveraging the fact that Txs made in

Bitcoin Core take some time to be authorized by miners, providing an opportunity for error correction.

At this point, in Bitcoin Core, Txs cannot be reversed or cancelled once they are initiated. Some wallet applications offer a Replace-by-fee (RBF) feature that enables users to replace their unconfirmed Txs with a higher transaction fee to speed confirmation[2]. However, it is important to note that this feature does not reverse or cancel the Txs.

Implementation 1

In the first implementation, a new component called “Quarantine” will be introduced, which is expected to depend on Txs & Mempool, RPC, and Wallet components, as illustrated in Figure 1. When a user requests to cancel their Tx through the Wallet, the Cancel Tx process will be executed by invoking the RPC. Initially, Txs & Mempool will verify the locktime, “the earliest time a transaction can be added to the blockchain[3],” and confirmation status. If the locktime has not been reached and the Tx remains unconfirmed, it will be redirected to Quarantine. To prevent payment fraud or double spending, Quarantine will request RPC to confirm the cancellation with the recipient within a specific timeframe. If the recipient confirms, the Tx will be reversed to UTXO status; otherwise, it will be returned to Txs & Mempool to resume confirmation.

The architectural style will remain unchanged from the original Publish-Subscribe style. The new Quarantine component will also broadcast messages that subscriber components can receive. Event notifications, including the execution and confirmation, will be disseminated through broadcast messages and received by all subscribers of the Quarantine component distributed through the system. The effect on the concurrency is also assumed to be trivial due to the nature of asynchronous communication of the Publish-Subscribe style.

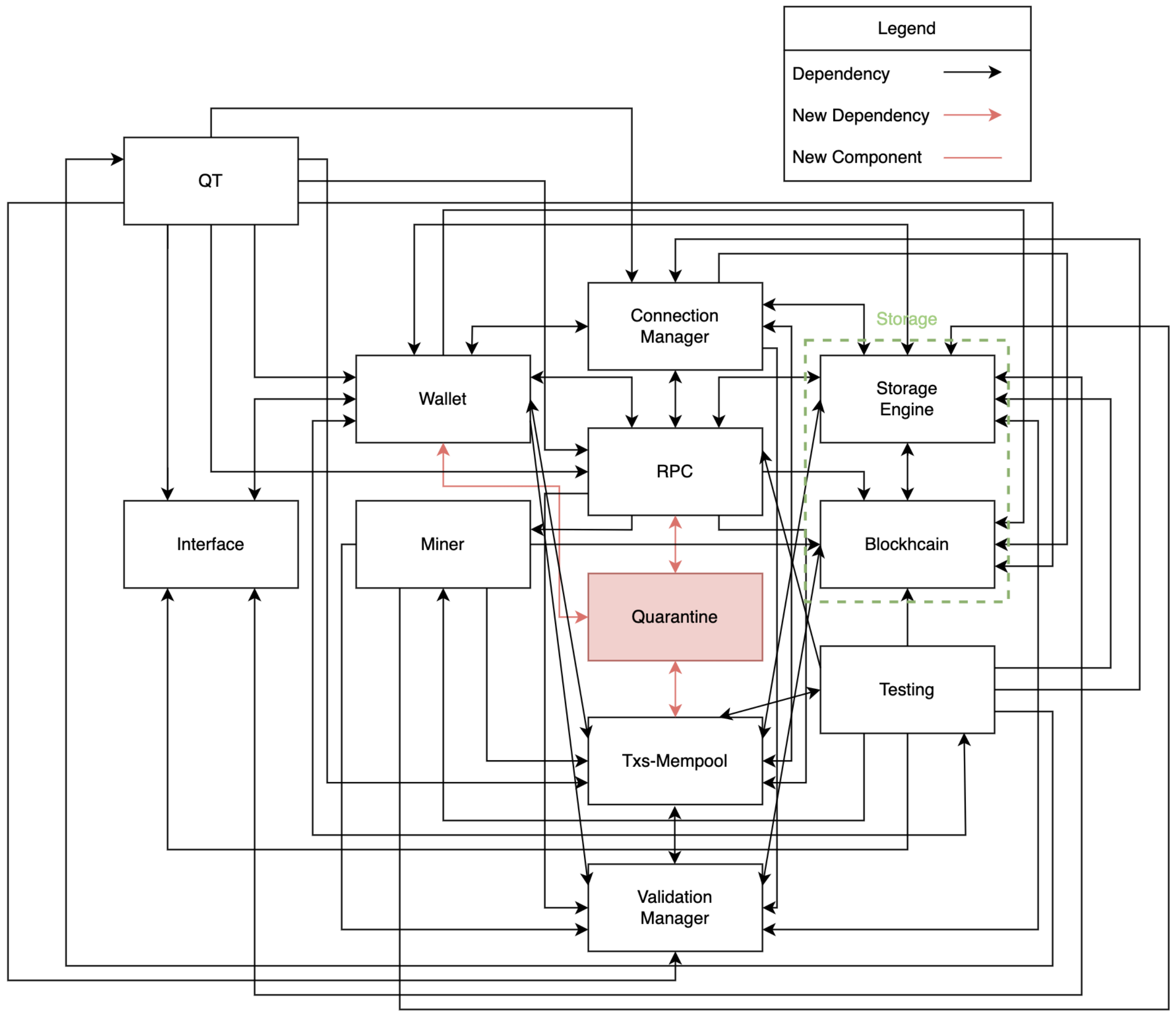


Figure 1. Conceptual Architecture of Implementation 1

Implementation 2

Our second implementation of cancelling transactions can be added without adding a new module or component into the system. We will need to alter the inner components of the components implementation 1 depended on, which are Txs & Mempool, RPC, and Wallet. The process will virtually start the same way with a user prompting the Mempool for the Tx to be cancelled through their wallet, causing the Cancel Tx process to be executed by invoking the RPC. Then, the Txs & Mempool will verify the locktime in the same way and provide a confirmation status. However, the management of a failed locktime and unconfirmed Tx will need to be managed by the Mempool alone. As the Mempool is doing the process of cancelling and figuring out the logistics of the locktime and confirmation, we can simply have Mempool also resolve the Cancel Tx by requesting the RPC to confirm the cancellation with the user in a specified timeframe. If the user confirms, then the Mempool reverses the Tx's UTXO status. If the user rejects, then the Mempool will resume confirmation.

SEI SAAM ANALYSIS

Major Stakeholders

Bitcoin Users: The most important stakeholders are the individual users who use Bitcoin. They are the most essential part of the operation of the Bitcoin network and thus are the most common and significant stakeholders. This can be any individual user who uses Bitcoin Core to manage their Bitcoin as an everyday currency, for investing, etc. The most important NFRs to these stakeholders are:

- **Usability**: Users want the new feature to be easy to use and learn (i.e. minimized difficulty and complexity)
- **Availability**: Users want the new feature to be available to use whenever they need it
- **Security**: Users want to be ensured that the new feature is safe to use and will protect user information.

Bitcoin Core Developers: Bitcoin Core is open-source software. As a result of this, there is an open community of individuals who can view Bitcoin Core's source code and contribute to the source code by proposing changes, updates, suggestions, etc. This community of individuals can be classified as Bitcoin Core Developers, and thus are another group of stakeholders. The most important NFRs to these stakeholders are:

- **Performance**: Developers want the software to still perform as optimally as possible
- **Testability**: Developers want to be able to test the new feature and all modules that interact with this new feature
- **Maintainability**: Developers want to be able to implement the new feature and have the system sustain itself without the intervention of the developers

Businesses: Businesses that accept Bitcoin as a form of currency would also be stakeholders since they will also be affected by the newly proposed enhancement. Businesses will need to ensure that if there are any changes to Bitcoin Core, they are able to update the tools and software used that implement Bitcoin Core to maintain the functionality of their tools and properly integrate Bitcoin Core. Some businesses that accept Bitcoin as a form of currency include Wikipedia, AT&T, Microsoft, Overstock, and more [4]. The most important NFRs to these stakeholders are

- **Performance**: Businesses want this new feature to be high in performance (response time, throughput, etc.) so they can continue normal operations without difficulty and satisfy consumer needs
- **Security/privacy**: Businesses want this new feature to ensure the safety of both company and consumer information
- **Accessibility**: Businesses want this new feature to be accessible to all of their consumers

- **Maintainability:** Businesses want the software to be stable itself so they do not have to invest resources into maintenance.

Comparison of Enhancements

	Pros	Cons
Implementation 1	**Usability**: The approach is straightforward. The new interactions between modules are not complex and users will find this new feature to be easy to use since triggering the cancel event requires a single broadcast through the Quarantine module. **Security:** The new feature requires the confirmation of the event within a given timeframe through the recipient by making use of the RPC. This makes verification of cancellation easier to integrate and deals with the issue of payment fraud. **Accessibility:** The new feature is easy to access and can be accessed by all stakeholders. As long as users can access their Wallet, they will be able to access the new cancel feature.	**Availability:** Due to this feature's use of a timeframe, it would not be possible to cancel a transaction if the user fails to cancel before the transaction becomes verified. If the user is unaware of this time frame, it can cause problems and would require them to speak with the recipient of the Bitcoin transaction. This feature is only available to use for as long as the timeframe allows. **Testability:** Testing will be expensive and complicated since we are adding a new module to the existing system. New test cases will need to be formed, which will involve modifying the existing Testing module. **Maintainability:** Since this implementation proposes the integration of a new module, this increases the complexity of Bitcoin Core's source codes. This increases the difficulty in future changes, maintainability, and evolution.
Implementation 2	**Maintainability:** Implementation only requires changes to existing	**Maintainability:** Maintenance costs will be higher since we are adding

	components to the system **Testability:** This implementation is easier to test as there already exists testing for these components **Security:** The security of the system will be improved as our original security system now has additional checks related to user actions built into the existing components, such as verification of transactions to prevent double-spending or fraud.	new features to components which need to update with new facts constantly **Performance:** Since we are adding more functionalities to the components, our whole system's potential suffers from a lower performance **Evolvability**: Since there is not a new component, the system will have less potential plasticity

Determined Better Implementation

Implementation 2 was determined to be the better addition to the architecture in terms of maintainability and security. Developers in the open community will benefit from the maintainability and the general security of the new feature will be beneficial to users and businesses. While implementation 1 gave the benefit of usability, accessibility, and security, these were mainly benefits for businesses that accept Bitcoin Core, which is our lowest priority stakeholder. Bitcoin Core developers suffered from the increased complexity of testing and more complicated maintainability. In addition, Bitcoin users suffer from a lack of availability, and businesses that accept Bitcoin suffer from more complex maintainability. Overall, the advantages of implementation 2 outweigh what implementation 1 can offer to stakeholders.

ENHANCEMENT IMPACTS ON BITCOIN CORE

Impact on high-level conceptual architecture

Both proposed implementations do not affect the Publish-Subscribe architecture style. The only impact the changes have on the architecture is on the Txs & Mempool, RPC, and the Wallet subsystems with the addition of a new subsystem, Quarantine, in the first implementation.

Impact on low-level conceptual architecture

Quarantine: This is a new subsystem that would be added for the first implementation. The addition of this component would introduce three new dependencies all between itself and the Txs & Mempool, RPC, and Wallet components. The Quarantine component is where a Tx is redirected if the Tx's locktime has not been reached and remains unconfirmed when a Cancel process is called within the Wallet and broadcasted by the RPC. If the Tx in Quarantine has its cancellation confirmed by the RPC, then it is marked as a UTXO. Otherwise, it is returned to the Mempool.

Txs & Mempool: The Txs & Mempool subsystem receives a Cancel call from the RPC and verifies if the Tx's locktime has been reached. If locktime has not been reached, in the first implementation, the Tx in question is sent to Quarantine, but in the second implementation, Txs & Mempool acts as the Quarantine component and the Tx remains in the Mempool awaiting a cancel verification from the RPC.

RPC: In both implementations, the RPC gains a Cancel process that tells the Txs & Mempool component if the locktime of the Tx has been reached and sends out a request to verify the cancellation with the recipient.

Wallet: The Wallet gains a new feature, to start a cancel for a Tx. When a user incites a cancel of a Tx, the Wallet relays that to the RPC for it to broadcast the cancel feature.

Impact on directories and files in subsystems

The following files would need to be modified in order to implement the new proposed enhancement:

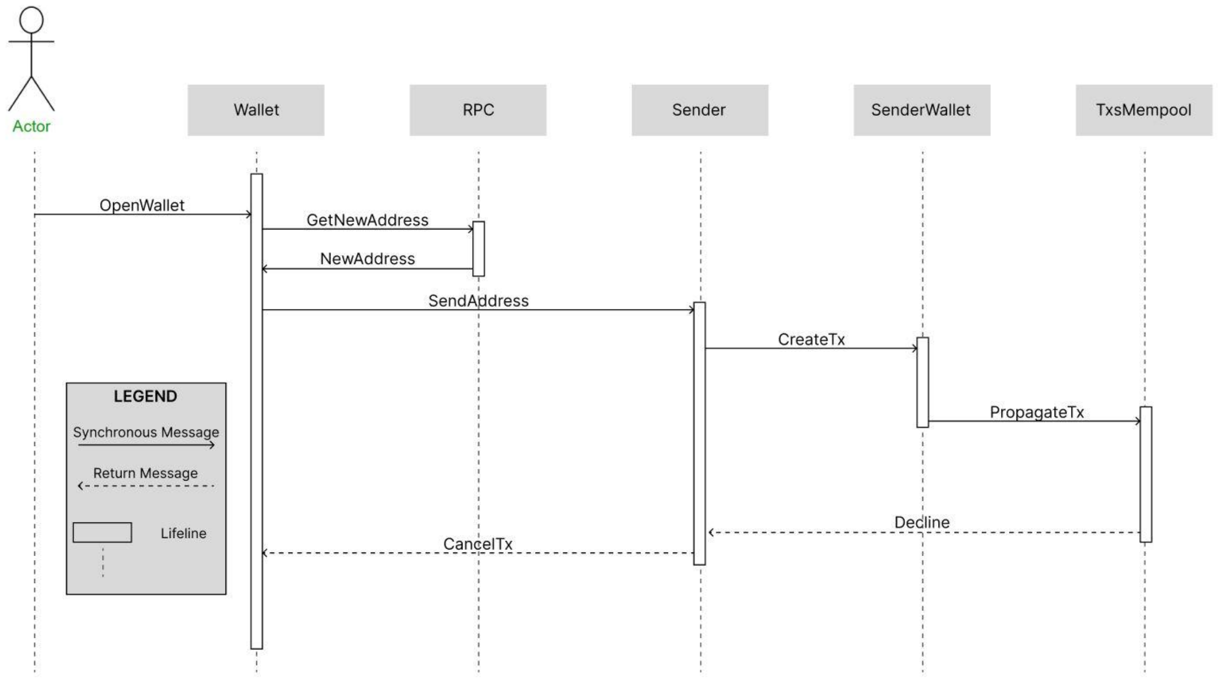
- bitcoin\src\txmempool\txrequest.cpp
- bitcoin\src\txmempool\txrequest.h
- bitcoin\src\txmempool\txmempool.cpp
- Bitcoin\src\txmempool\txmempool.h
- Bitcoin\src\txmempool\utxo_snapshot.cpp
- Bitcoin\src\txmempool\utxo_snapshot.h
- Bitcoin\src\rpc\mempool.cpp
- Bitcoin\src\rpc\mempool.h
- Bitcoin\src\rpc\mempool.cpp
- Bitcoin\src\txmempool\lockedpool.cpp
- Bitcoin\src\wallet\pubkey.cpp
- Bitcoin\src\wallet\pubkey.h
- Bitcoin\src\wallet\wallet.cpp
- Bitcoin\src\wallet\wallet.h

We theorize that the following files would need to be added in order to implement the new proposed enhancement:

- Bitcoin\src\txmempool\lock_status.cpp
- Bitcoin\src\txmempool\lock_status.h
- Bitcoin\src\txmempool\txcancel.cpp
- Bitcoin\src\txmempool\txcancel.h
- Bitcoin\src\rpc\cancel_confirmation.cpp
- Bitcoin\src\rpc\cancel_confirmation.h

SEQUENCE DIAGRAM AND USE CASES

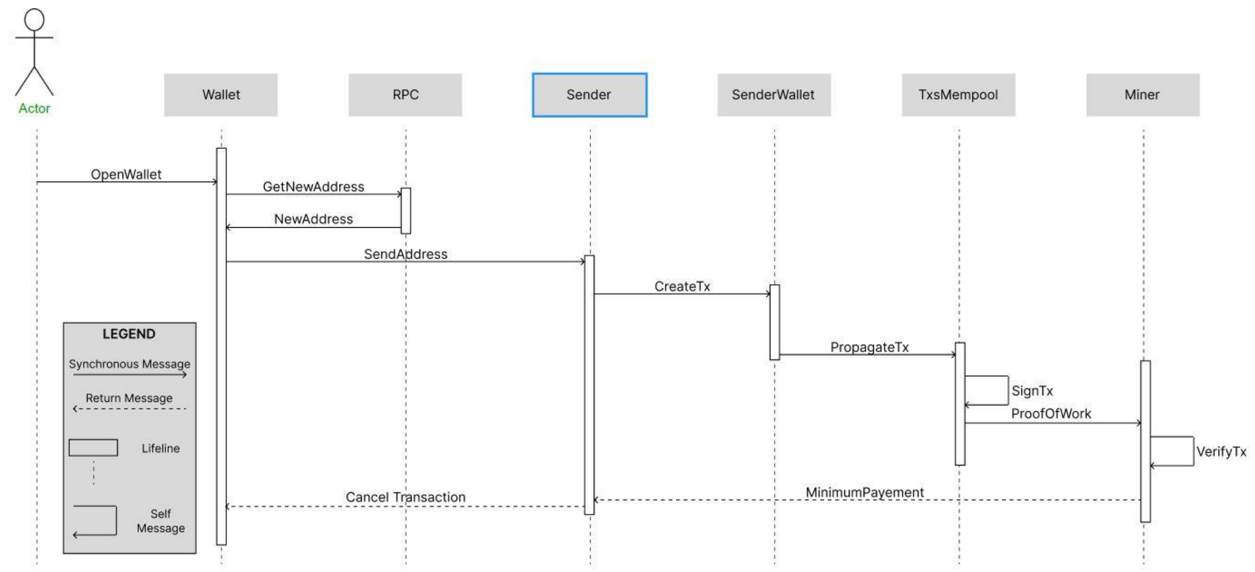
Use Case 1: User Wants to Cancel Unmined Transaction



A user wants to send another bitcoin to another user. He opens his wallet software and creates a new transaction to send 1 BTC to the user’s address. However, he realizes that the amount is not what he intended, which means that he would need to cancel the transaction. The user tries to cancel the transaction and resend it. The enhancement allows the user to cancel the unmined transaction in his wallet. This means that the transaction is returned to the UTXO where it can be picked up again and included in a future block.

To cancel the transaction, the user goes to the transactions page in her wallet software and selects the unconfirmed transaction. Once he confirms to cancel the transaction, it is removed from his wallet’s mempool and returned to the UTXO. The process begins once the transaction is sent. The network broadcasts the transactions and is verified and added to the mempool. Before the transaction begins mining, the user cancels the transaction. The miner removes the transaction from the mempool and is returned to the UTXO

Use Case 2: User Sends Below Minimum Payment Amount



a shop owner sells goods online and accepts Bitcoin as payment. He receives a payment from a user for an order worth 0.1 BTC. The shop owner notices that the transaction has a very low fee and is not being confirmed by miners. The shop owner is concerned that the payment may not be processed in time, which could cause delays in shipping the products to the user.

Because of the new enhancements, the shop owner can cancel the unmined transaction and ask the user to send a new one that would be confirmed by the miners. The shop owner opens his Bitcoin wallet and looks for the transaction sent by the user then cancels it. The transaction is returned to the UTXO and the user can send it again. The process begins when the unconfirmed transaction is broadcasted. It gets added to the mempool. Once the shop owner cancels the transaction, the miners remove the unmined transaction from the mempool as it does not get mined due to the minimum payment. The transaction is then sent back to the UTXO

TESTING

Testing must be utilized to ensure that both the functional requirements and the non-functional requirements, such as usability, security, and privacy.

Given the publisher-subscriber architecture, ensuring proper integration testing is crucial in maintaining the security of Bitcoin Core. In the first implementation, the Quarantine module broadcasts cancelled transactions to other components including the RPC, Wallet, and Transactions & Mempool subsystems. Integration testing should ensure that no information leaks about private user information occur throughout these exchanges by testing both individual interactions with a particular component, and large-scale testing of the whole system with the added feature.

In addition, unit testing of the specific functionalities of the new feature, such as in scenarios of valid transactions, already mined transactions, and invalid transactions. This ensures that the new feature does not result in a loss of usability of the existing features of Bitcoin Core. For instance, they were causing errors due to double-spending. That is, if a user decides to cancel a transaction, Bitcoin Core must ensure that the transaction does not go through and that all parties have the amount of currency they had before initiating the transaction. Using the example of the Quarantine implementation, testing must ensure that the RPC does not allow a translation to be cancelled if the time frame before the trade goes through is exceeded.

POTENTIAL RISKS

Under the current Bitcoin architectural system, multiple potential risks may arise in the Bitcoin Core system.

One major potential risk is that the new feature may compromise the integrity of Bitcoin transactions performed on Bitcoin Core. Double spending, where a user utilizes the same currency multiple times, increases in likelihood. For instance, if improperly implemented, users may be able to spend Bitcoins before a transaction is formalized, then cancel their transaction to have the Bitcoin reimbursed as a form of double-spending. In addition, the new feature adds complexity to the verification of transactions, this means that tracking transactions may become more difficult, leading to easier fraud. Additionally, new forms of DDOS attacks may be possible. If multiple peers attempt to cancel transactions at the same time, the overhead of the system may increase to the degree where the usability of the system is greatly diminished.

Another potential risk is difficulty in maintainability. Adding a substantial new feature, such as a whole new subsystem to cancel transactions increases the complexity of Bitcoin Core's codebase and requires significant resource and time investment. For a project that is open-source and volunteer-based, this may lead to difficulty in changes, maintenance, and enhancements to the system over time.

NAMING CONVENTIONS

UTXO:	Unspent Transaction Output
RPC:	Remote Procedure Call
Txs:	Transactions

LEARNED LESSONS

As we worked through the first two assignments from the ground up, there were many different processes of building potential parts and then breaking them down as we realize certain parts are not correct in terms of the architecture we were trying to build. With the feedback we have received from the first two assignments, we now have a much better overview of the Bitcoin architecture and when working on this assignment, we used our knowledge of what we already know to build out the proposal for our new enhancement. As we had a better grasp on the subject, we could build an enhancement that we saw was missing in the base architecture that a lot of users would benefit from in terms of useability and security. It was also much easier to consider all the aspects of what this new feature might bring, in terms of positives and negatives, security risks, maintainability, and other things that impacted our analysis.

All in all, our team learned to take our knowledge of architecture and stakeholder analysis to build something that brings better optimization and a better user experience of the application while keeping in mind what it would cost to implement the changes and what possible ways we could implement those things.

CONCLUSION

In conclusion, this report has proposed a new feature for Bitcoin Core, namely cancellable transactions, and presented two possible methods of implementation. Through the SAAM analysis, we have identified key stakeholders and their respective NFRs, and compared and contrasted the two implementations to determine the best course of action. The report has also discussed the impacts of the new feature on existing subsystems and components and presented two use cases to demonstrate its benefits and functionality. Finally, we have outlined the testing mechanisms to ensure proper integration and discussed the possible risks associated with the new feature. We believe that the proposed enhancement has the potential to significantly improve the user experience of Bitcoin Core and contribute to the continued growth and development of the system.

REFERENCES (EVERYONE - FORMATTED BY ERIC)

[1] *Cancel a payment or transfer*. Cancel a payment or transfer - RBC Royal Bank. (n.d.). Retrieved April 12, 2023, from https://www.rbcroyalbank.com/onlinebanking/bankingusertips/payments/cancel.html

[2] *Replace-by-fee (RBF) bitcoin in Trezor Suite application*. in Trezor Suite application. (n.d.). Retrieved April 12, 2023, from https://trezor.io/learn/a/replace-by-fee-rbf-bitcoin

[3] *Transactions*. Bitcoin. (n.d.). Retrieved April 12, 2023, from https://developer.bitcoin.org/devguide/transactions.htm l

[4] 2, E. A., 21, C. N. M., 12, W. M., 7, kral M., Kibet - 99Bitcoins support March 7, 10, J. F., 7, asad F., Kibet - 99Bitcoins support February 8, & 16, asri F. (2023, January 15). *Who accepts*

bitcoins in 2023? list of 20+ major companies. 99 Bitcoins. Retrieved April 12, 2023, from https://99bitcoins.com/bitcoin/who-accepts/