CISC 322/326 Assignment 3

ENHANCEMENT PROPOSAL OF BITCOIN CORE

Group 8: EVANGELION-01

Jinlun Zhang: 16jz61@queensu.ca

Jing Tao: 21jt35@queensu.ca

Jihong Zhang: 19jz138@queensu.ca

Xueyi Jia: 18xj8@queensu.ca

Hejin Wang: 15hw45@queensu.ca

Yunhan Zhou: 17yz101@queensu.ca

APRIL 12th, 2023

1.0 Abstract

Previously, we discussed the concrete architecture in Bitcoin Core, and we showed how components and dependencies operate to help clients in reality, especially for “Miner”, which is the key part of mining. However, it still lacks flexibility so that people only have a single way for mining. In order to increase the development space and trading methods of Bitcoin transactions, we decided to develop a new enhancement, we call it “Miner Rental Functionality”. This new technology allows each user to rent their idle miners to others, or to let people who need more miners borrow other people's miners. The lease rules of rental functionality can be decided by the customers themselves, for example, the rent can be determined according to the time interval the miner is rented, or the person who borrows the miner pays half of the gold that mined by the miner as the rent. In this report, we will talk about the rationale and practicality of this approach and provide use cases to illustrate the implementations of “Miner Rental Functionality” in detail, then we will summarize their effect and limits.

2.0 Introduction and Overview

Referring to Concrete Architecture, we know that the "Miner" component within Bitcoin Core is in charge of generating candidate blocks and incorporating them into the blockchain, then clients can prepare to mine. Based on this, clients can be provided another way to get Bitcoins by “Miner Rental Functionality”, experienced miners can borrow multiple miners to accumulate wealth, and those who without technology can choose to rent out their own machines in exchange for money, or they can decide the dividend according to the amount of Bitcoin of miners get when miners are lent out. Through the RPC-API, miner borrowers and creditor can issue commands related to mining, such as the time start or stop for the mining process, adjusting mining parameters, and obtaining mining-related information, such as the current mining difficulty or the status of the mining operation. It's a win-win solution! Now we can talk about the Functional and Non-Functional requirements, a SAAM analysis for alternatives, how the enhancement influences on the high- and low-level conceptual architecture and two use cases and sequence diagrams.

3.0 Proposed Enhancement

The enhancement we proposed is called the “Miner Rental Functionality”. The main idea of the enhancement is that, since the miner component in the Bitcoin Core architecture can be located on a different machine (such as the dedicated ASIC miners) and can be utilized by the users remotely through RPC, it is practicable and desirable to enable users to either rent out or rent in the miners. From the technical perspective, since each Bitcoin Core instance follows a client-server architectural style as we have justified in previous assignments, it is realizable to allow other users on the Bitcoin Core P2P network to remotely control someone’s miner through their own RPC-API component and Connection Manager component, provided that the someone authorizes the miner to be publicly accessible (renting out the miner). This means that the “Miner Rental Functionality” enables users to either rent out their own miners for a certain period in exchange of bitcoin or pay bitcoin in exchange of using powerful mining infrastructure for a certain period if they currently do not have one. As a result, the number of idle miners in the network should be minimized, which maximizes the security of the blockchain as more miners actively validate the transactions in the network (Antonopoulos, 2014).

Furthermore, this enhancement is also beneficial for the users from the risk management perspective. Firstly, as for the rentees, this enhancement provides them with an option to evade the substantial upfront costs of setting up their mining infrastructures, which should directly increases the entry confidence of potential Bitcoin Core users who do not have miners but desires to mine, because their risks of investing in the “mining business” are considerably reduced due to this “Miner Rental Functionality” as the only cost the mining can incur is the rental fee. On the other hand, as for the renters, this enhancement realizes a lower bound on the returns of their miners. Since mining is a competitive process where users compete

against each other to solve the Proof-of-Work algorithm and aim to be the first one of validating a new block of transactions (cypherpunks), the probability of a user successfully mining a block is relatively low, resulting in high risks of receiving zero returns after spending considerable utility expenses and effort. However, as the “Miner Rental Functionality” provides an option for the users with functional miners to rent out their miners, a lower bound on the returns of utilizing their miners is guaranteed, leading to a lower risk of being a mining or a full node in the Bitcoin Core P2P network, which should also indirectly motivate more users in the network to equip a miner for gaining certain extra earnings. Additionally, this enhancement also increases the earning ceilings for the superior players in the mining competition. For example, the people who have mastered a set of proficient “mining skills” by practicing and updating their mining hardware can acquire more miners to accumulate wealth as they can borrow extra miners from others to increase their success rate of appending a block to the blockchain.

In addition, from deployment perspective, this “Miner Rental Functionality” enhancement should also enable the rentees to configure the mining settings and monitor the mining performance synchronously, and some information returned from the miner should be accessible by both the renter and rentee. For example, the report of an error or illegal action can be received on both renter’s and rentee’s App, so that the renter can fix the problem of the miner or modify the rentee’s authorization (access right) promptly to ensure reliability and security.

Since Bitcoin mining has driven innovation in the field of computer hardware and software, miners are constantly looking for ways to increase their hash rate and efficiency, "Miner Rental Functionality” will lead to the development of new and more powerful mining equipment, which can inspirate clients to cooperation and mining.

The additional incentives of implementing the enhancement include:

- Avoid Bitcoin Core users to resort to third-party services and platforms that provide mining rental services, which not only saves the brokerage fees for the users, but also eliminates the intrinsic risks involved in the unregulated mining rental services such as the potential for scams.

- Providing one more reasonable option of spending bitcoin as the current usage of bitcoin is limited.

- Attract more people to use Bitcoin Core as the "Miner Rental Functionality” can be the gospel for people with low risk tolerance and people who need to earn a sum of money in a short time with the idle mining equipment to rent.

4.0 SAAM Analysis

4.1 Implementation 1

The first implementation of “Miner Rental Functionality” is enabling users to rent out or rent in miners by relying on the existing Bitcoin Core P2P network and the enhanced client-server architecture of the Bitcoin Core instance. As shown on Figure 1, according to the design of implementation 1, any user with a functional miner should be able to initiate a broadcast of “starting or terminating renting out the miner” to the P2P network, and this broadcast should be received by all the nodes on the P2P network iteratively just like how a normal transaction broadcast is propagated and received on the network. This implies that every node will maintain a separate list of “renter peers” on the network in addition to the “local peers” to keep track of the currently available renters, and because both of the “renter peers” list and the “local peers” list are required to be updated continuously, it is plausible to integrate the “renter peers” storage into the existing “Local Peers Storage” top-level component, which is reflected on Figure 1 as well.

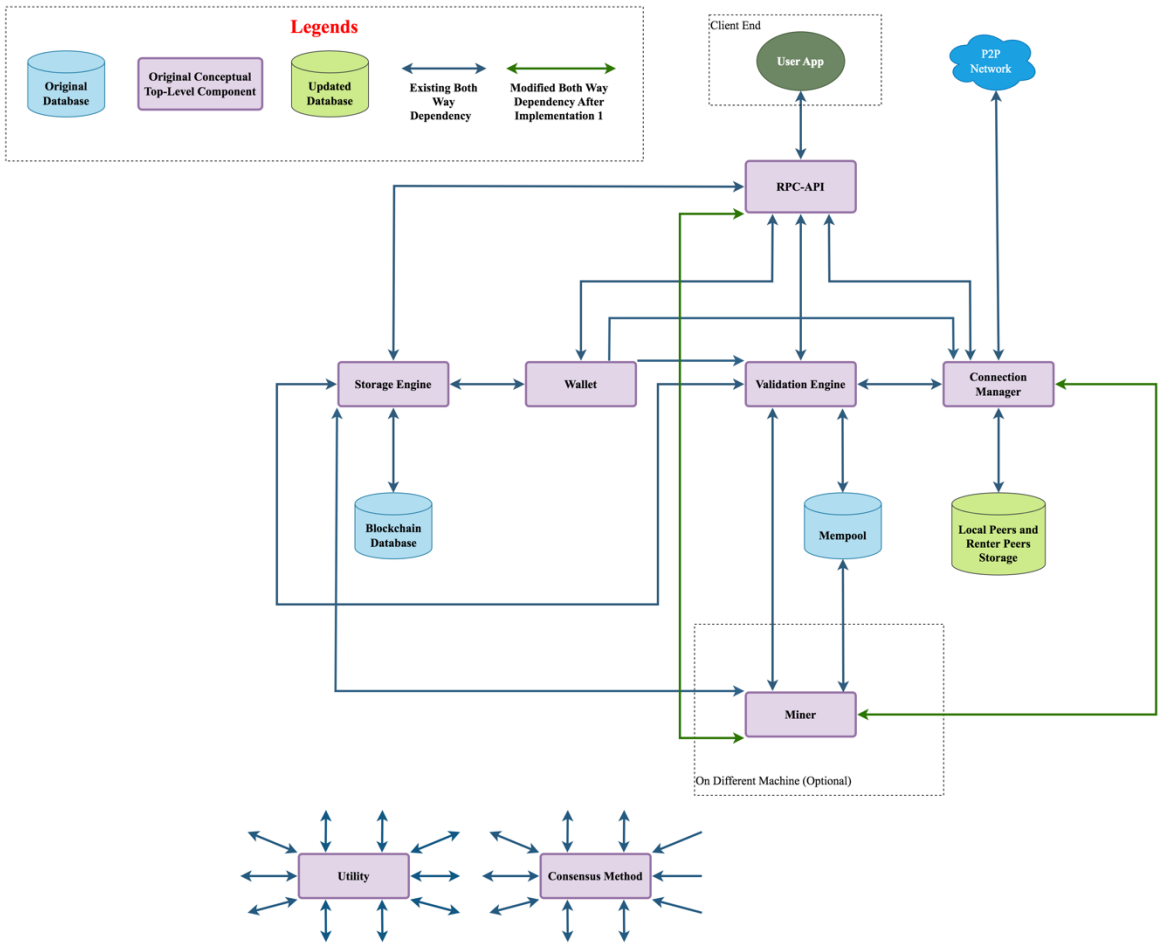

Figure 1: Bitcoin Core Updated Conceptual Architecture after Implementation 1

Then, if a user (rentee) wants to rent in a miner from a renter, he will firstly search his "renter peers" storage to find out the renters whose miners either have the lowest response time or the desirable specifications, meaning that when a renter broadcasts a miner renting out service, the miner's specifications, which are automatically identified by Bitcoin Core, will be attached on the broadcast, and the latency of the miner to the rentee is identified by the Connection Manager component on demand. Moreover, the miner rental fees per minute in the unit of bitcoin, which are set by the individual renters, are also included in the broadcast, and so when a rentee decides to rent in a specific miner, he will send a transaction of equivalent amount of bitcoin for his rental period to the renter to obtain the authorization to access the renter's miner. This transaction to the renter is handled as an ordinary valid transaction in the Bitcoin Core P2P network, but the only difference is that the rental transaction will trigger an update on the "renter peers" storage in each node of the network to reflect the unavailability of the miner.

Once the rental transaction is received and validated by the renter, a direct connection between the renter's miner and the RPC-API component of the rentee is established ad hoc through the Connection Manager components of both parties as shown on Figure 2. In such case, the renter's control to his miner is restricted for the rental period such as disallowing him to shut down the miner, configure the miner, or stop mining, because those authorizations are "transferred" to the rentee, which explains why the previous one-way dependency of Miner on Connection Manager becomes two-way, as the instructions from the rentee is transmitted to the Miner through the P2P network and the Connection Manger of the renter. However, the renter can still receive some information returns from his miner such as the miner status

report, or the report of an error or a failure, to ensure that the renter can stay informed about the miner's general operations and can resolve any technical issue encountered on time without causing any major disruption to the rentee's usage, which also explains why the previous one-way dependency of RPC-API on Miner becomes two-way.

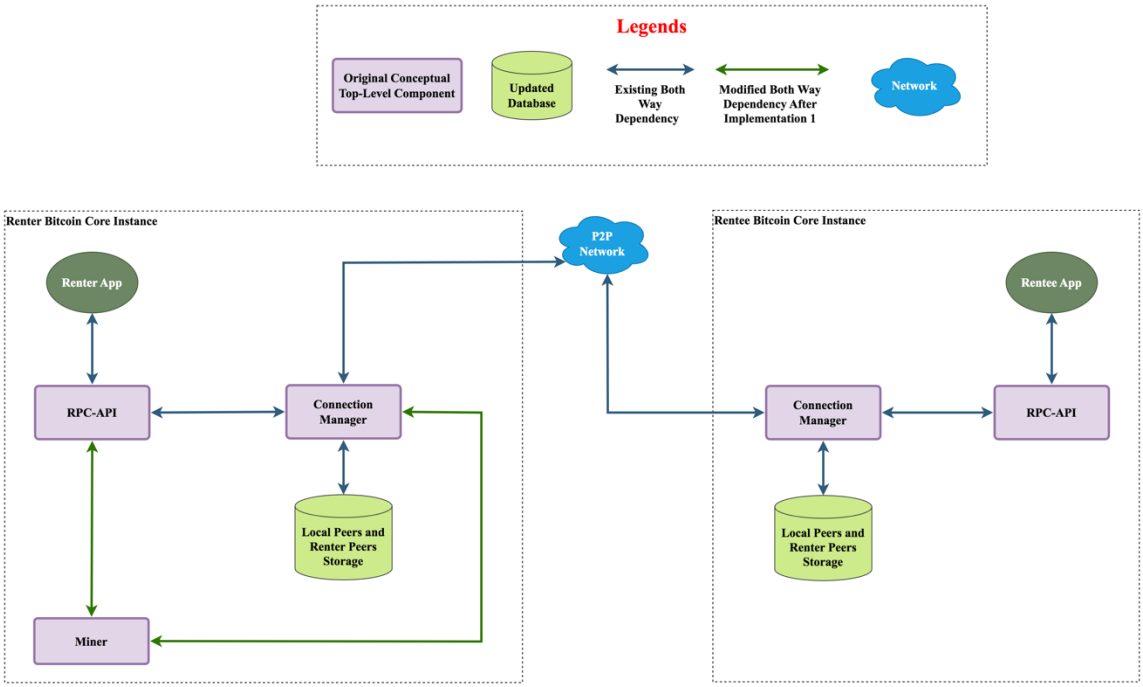

Figure 2: Simplified Conceptual Illustration of Implementation 1 of Miner Rental Functionality

After the rental period ends, the connection between the renter's miner and the RPC-API component of the rentee is terminated, and the full access of the renter to his miner is restored, meaning that the miners for rent must have a built-in timer to initiate a connection break once the rental period concludes. Finally, if the renter decides to continue to rent out the miner, he will resend the broadcast of "renting out the miner" to the Bitcoin Core P2P network like before to update his miner's availability in the network.

This implementation of the "Miner Rental Functionality" has various advantages. Firstly, as it is based on the existing Bitcoin Core P2P network to broadcast the miner rental announcement and request, no additional top-level component is required to be developed and added into the architecture of Bitcoin Core, meaning that the level of modifications to the existing conceptual architecture of Bitcoin Core is minimized in such implementation and so the client-server Bitcoin Core node architectural style with P2P network remain unchanged, leading the testability of the new enhancement to be high as many existing testing cases can be reused and the number of possible bugs associated with the new enhancement should be minimal due to the limited changes made to the original architecture. Secondly, since each node in Bitcoin Core P2P network maintains its own copy of the "renter peers" list in its own storage space, and can broadcast a miner rental announcement or request without load concerns on the number of miner rental announcements or requests currently on the network once qualified (such as having a functional miner to rent out or enough "bitcoin balance" to rent in), a high scalability can be achieved easily in this implementation due to the ease of scale-out. Thirdly, this "Miner Rental Functionality" is presupposed to achieve a100% availability (24*7*52) as there is no central server regulating the rental service as a whole and so should perform without periods of loss of availability.

Nevertheless, this implementation of "Miner Rental Functionality" also has downsides. For example, as aforementioned, since each miner for rent must have a built-in timer to initiate a connection break, and

there is no central server to enforce a universal type of timer for each legitimate miner for rent, it is highly likely that the timers between miners are not consistent, leading to potential compromises on the security and the reliability of this rental service.

Regarding the effects of the enhancement under implementation 1 on Bitcoin Core, the evolvability of the software should be significantly improved, because many additional features or enhancements can be researched and implemented for the “Miner Rental Functionality”. For example, a feature that can automatically determine the current market price of renting out a specific miner can serve as a reference for the renters when they are setting their miners’ rental fees. Moreover, a feature that allows a rentee to extend his existing rental period during or at the end of his rental period should be feasible to be included in the update patch of the “Miner Rental Functionality”. On top of that, the existing testability and maintainability of Bitcoin Core should not be deteriorated by the enhancement as it incurs minimal modifications to the existing conceptual architecture of Bitcoin Core.

4.2 Implementation 2

The second implementation of the "Miner Rental Functionality" involves a new server called the "Central Miner Rental Server" or CMRS, which manages users to rent in or rent out miners. All the users on the Bitcoin Core P2P network can access the services of CMRS after being authenticated. This implementation does not require any changes to the conceptual architecture of a Bitcoin Core instance. Instead, a new node, CMRS is needed to be developed and integrated into the P2P network. This CMRS consists of 2 components. One is called the “Rental Service Connection Manager”, which manages the matches and connections between the renters and the rentees, and the other component, which is called the “Renter Peers Storage”, supports the operations of “Rental Service Connection Manager” as the private database residing in CMRS that contains an active list of renters, as shown in Figure 3. Under such implementation, each peer in the network is not required to utilize its own storage space and computing resources to maintain a copy of the “renter peers” list.

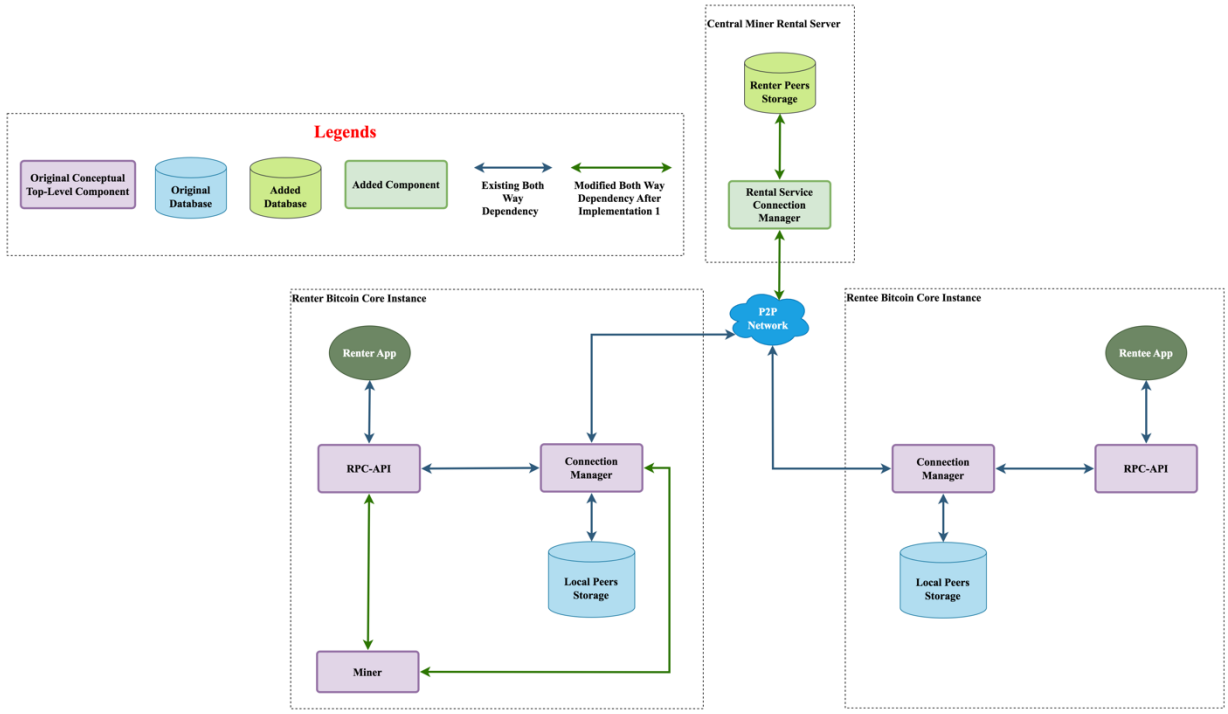

Figure 3: Simplified Conceptual Illustration of Implementation 2 of Miner Rental Functionality

To access the miner rental service provided by the CMRS, where the server is maintained by the most stable nodes in the Bitcoin Core P2P network, a rentee must first establish a connection with the server and initiate a rental request of a specific miner found in the “rental peers” list the server provides in the form of bitcoin transaction, as the server stores the renters’ mining machine information in the "renter peers storage" database, which enables the CMRS to track available miners and provide the miner information with prospective rentees. Likewise, when a renter seeks to rent out his miner, he must also connect to the CMRS through the P2P network and broadcast the miner rental announcement to the server. Once the CMRS receives the rental transaction from a rentee for a specific miner, it will retain a small proportion of the transaction as the “agency fee” and send out the remaining balance in the transaction to the renter to indicate a successful “rental match”, while the “agency fee” will be sent out to the owners of the CMRS (the most stable nodes on the network) immediately, leading to the unnecessity of having a wallet-like component in the CMRS. After the successful matching through the CMRS, the server will establish an indirect connection between the renter and rentee to manage the communications between the 2 parties such as passing the instructions from the rentee to the renter’s miner as shown on Figure 3. Then, same as implementation 1, the renter’s control of his miner is restricted, and the rentee has the capability of configuring and running the miner according to his specific needs.

To optimize rentee’s miner “browsing experience” on the CMRS, the server employs advanced algorithms that take various factors into account in the “rental miner recommendation system” such as the miner specifications, rental fees, and popularity (based on past availability). The algorithms should guarantee that the rentees can quickly and easily obtain the optimal processing powers that meet their demands. Moreover, the CMRS offers a high level of flexibility in terms of rental period options. For example, the rentees can specify the duration of their rental period either by a fixed amount of time or a specific number of blocks mined. Additionally, rentees can specify the amount of processing power they require to focus on the qualified miners only, which further improves their rental experience. On the other hand, the rentee can either choose to rent out their miners for a specific time period or until a specific number of blocks is mined by a rentee, providing them with more control over their mining operations.

This implementation offers multiple benefits. One of the primary benefits of this implementation is its high performance. As each renter and rentee can directly connect to the CMRS to access the miner rental service and find the optimal match, the maximum throughput and shortest response time of the “Miner Rental Functionality” can be achieved. Moreover, as a centralized server for managing the miner rental service, the server minimizes the time and resources required for rentees to find available miners. Additionally, each component of the central server is designed be able to be individually verified, and automated tests can be scheduled periodically to ensure the proper operation and prompt identification & resolution of any potential issues of the server. Moreover, the implementation's high maintainability is achieved by automated scheduled maintenance testing and repair on all individual server components. This guarantees that the server remains in optimal working condition and reduces the likelihood of downtime, providing users with a consistent and reliable rental service experience.

However, there are still potential drawbacks in CMRS that require careful considerations. Firstly, there is a high risk of compromised availability of CMRS due to the necessary upgrades or maintenance the server must undergo. In the meantime, unforeseen events such as network disruptions or power outages could cause single-point failure and interruptions in the miner rental service. Moreover, as all miner rental announcements and requests must pass through the central server, it introduces a level of centralization into the system, which violates the decentralized nature of Bitcoin Core and could potentially make the rental functionality more vulnerable to attacks by malicious actors. One additional drawback of this implementation is its associated cost on the renters and rentees. Since it is anticipated that the CMRS will levy “agency fees”, which are necessary to cover the costs of the server hosting and maintenance for the reliable and consistent operation of the rental service, this implementation can incur a higher expense than implementation 1 on the user side, which may be unsuitable for those looking to minimize the costs.

However, for those who prioritize the benefits of this implementation, this charge may be an acceptable price.

4.3 Implementations Comparison Regarding NFR

NFR	Implementation 1	Implementation 2
Performance	Medium Performance. Since this implementation relies on the existing P2P network of Bitcoin Core, the miner rental announcement and request may take long time to traverse the whole network to be received by every node in the network (low response time)	High Performance. Since every renter and rentee can directly connects to the central server to use the miner rental service, a maximal throughput and minimal response time can be realized
Scalability	High Scalability. Since each node in Bitcoin Core P2P network maintains its own copy of the "renter peers" list in its own storage space, and can broadcast a miner rental announcement or request without load concerns, a high scalability can be achieved easily due to the ease of scale-out.	Medium Scalability. Depending on the current request load and the number of connections to the central server, the server may exhibit a decrease in throughput and a subsequent increase in response time. Thus, to ensure performance, scalability must be controlled
Testability	High testability. Since the client-server Bitcoin Core node architectural style with P2P network remain unchanged in this implementation, existing testing cases can be revised and reused	High testability. Each component of the central server can be validated individually once the corresponding testing cases are produced, and the automated testing for this implementation can be scheduled periodically.
Security	High security. Due to the decentralized nature of the P2P network, there is no specific target for the malicious hackers, while the connection between the renter and the rentee is built ad hoc and is secured by both sides	Medium security. User private data disclosure and leaks may occur if the central server has vulnerabilities that can be exploited by hostile attacks
Availability	High Availability. Due to the decentralized nature of the P2P network, the rental service can be available for 24*7*52	Medium Availability. The central server can experience downtime for upgrade or maintenance and has the risks of single-point failure, leading the availability to be compromised
Maintainability	High Maintainability. Since this implementation does not alter the existing conceptual architecture of Bitcoin Core but is built on top of it, this "Miner Rental Functionality" can be maintained as a normal add-on feature of Bitcoin Core	High Maintainability. Automated regular maintenance tests and repairs can be run overnight for all the separate components of the central server (at the sacrifice of the availability)

Integration	High integration. Since this implementation is built on the existing architecture of Bitcoin Core, it can be integrated easily into Bitcoin Core. Also, this implementation can be smoothly incorporated into the broader Bitcoin Core application context, such as allowing a group of nodes on the network to form a mining pool and carve up the rewards of mining a block successfully.	Medium integration. Since an additional server node is required to be developed and added into the decentralized P2P network, numerous efforts are needed to integrate the “Miner Rental Functionality” into Bitcoin Core in this way. Also, this implementation may not be easily modified to adapt to the future evolution of Bitcoin Core

4.4 Stakeholder Analysis

This section will analyze our two new implementations from the perspective of the following three stakeholder roles: Users, Developers, and Investors:

The implementation of "Miner Rental Functionality" offers several benefits to users. It allows users with functional miners to monetize their resources by renting them out. Users needing mining power can rent miners for a specific period, which could lead to a more efficient distribution of mining resources and increased revenue for users with idle mining capacity. The "Central Miner Rental Server" also benefits users: rentees can connect directly to CMRS for the optimal rental matching as CMRS provides advanced recommendation algorithms and flexible rental options to allow users to find the best processing power with the best terms quickly and easily. Moreover, as the communications between the renters and rentees are handled by CMRS once the rental transaction is validated (rental relationship formed), the renters do not need to know the personal information of the rentees such as their IP addresses or identities on the network and vice versa, leading the users to preserve more privacy under the implementation 2 than implementation 1.

Since "The Miner Rental Functionality" (in implementation 1) is built upon the existing Bitcoin Core architecture, the number of modifications to the existing architecture is minimized. This also minimizes the impact on developers, as they can continue working on other aspects of Bitcoin Core without worrying about extensive changes. Besides, under implementation 2, as the "Central Miner Rental Server" does not require any modification to the existing conceptual architecture of Bitcoin Core instance, but only the development and integration of the CMRS node, it should be a relatively uncomplicated task for the developers.

For Investors of the Bitcoin Core project, they could see potential growth in the Bitcoin ecosystem due to "the Miner Rental Functionality" because more users are able to access mining resources, the overall network security and mining competition could increase, contributing to the stability and growth of the ecosystem, consequently dragging more investors into the Bitcoin field. Moreover, under implementation 2, investors can even levy a portion of the "agency fee" collected by the CMRS. More importantly, the improved rental experience and high performance of CMRS may attract more users into Bitcoin Core P2P network, which is on the interests of the investors.

In addition, we analyze which NFR is the most important for each stakeholder from their point of view. In implementation 1 of "Miner Rental Functionality", the most significant NFR for users should be security: users should be confident that their transactions and rented miners are free from unauthorized access or manipulation. Integration is the essential NFR for developers: the enhancement should be integrated seamlessly with the existing Bitcoin Core architecture, minimizing disruptions and ensuring compatibility with other components, and should also be flexible to cater other future enhancements. The most

important NFR for investors should be scalability: The “Miner Rental Functionality” should be designed to easily handle increasing numbers of users and rental transactions without negatively impacting the overall performance of the Bitcoin network. In this way, investors could maximize their returns from investing in the “Miner Rental Functionality” due to its high coverage of users. For implementation 2 of "the Miner Rental Functionality", Security of data and communications should be paramount for the users. Since the system has a centralized server, the implementation should include strong security measures to protect users’ rental data and communications. As for developers, maintainability should be paramount for them, because it is in favor of developers to be able to easily maintain and update the various components of CMRS without compromising the system's overall functionality. For investors, the most important NFR must be performance, because a performant and well-designed CMRS will lead to a highly positive user experience, enabling and attracting more renters and rentees to use this “Miner Rental Functionality”, meaning that the investors of this enhancement can gain the maximum values from their investments.

According to the above comparative analysis of the 2 implementations of "Miner Rental Functionality", it should be clear that the Implementation 1 should outperform the Implementation 2, because that though the high performance demonstrated by the CMRS of Implementation 2 is desirable, Implementation 1 presents an overall better security (no single-point failure), availability (24*7*52), scalability (no load & connection concerns), and integration (ease to be integrated into the existing Bitcoin Core and flexible to be adapted to cater other future enhancements), leading Implementation 1 to be comprehensively better than Implementation 2 from the users’, developers’, and investors’ perspectives.

4.5 Testing

In Implementation 1, since the enhancement is built on the existing framework, and no additional top-level components are required, most tests can rely on the existing test cases since the number of modifications to the existing conceptual architecture is minimal. Therefore, the focus of testing should be on the interactions of the important components involved in the functional process of Implementation 1.

Firstly, to test whether the rentee can rent miners from the renter through the Connection Manager component, we should create test scenarios involving the renter broadcasting the availability of miners, the rentee receiving the renter’s broadcast (then updating his local “Renter Peers” storage accordingly), and establishing & terminating a direct connection between the renter's miner and the rentee's RPC-API component (ensuring the effective connection establishments and terminations at the beginning and end of the rental period).

Next, the RPC-API component will be tested on its ability to verify that the rentee has access to and control over the operations of the rented miners and the ability to send out the instructions to the remote miner after being authenticated by the renter’s RPC-API. In addition, the RPC-API ability to receive feedback information from the miner after being rented out will be tested to ensure that the renter retains access to basic miner information, such as status reports and error notifications, for the rental period.

Finally, to test the upgraded "Local Peers and Renter Peers Storage" component, we will evaluate the proper parallel functioning of the storage and management of the “local peer” list and the "renter peer" list, ensuring that each node on the Bitcoin Core P2P network will maintain an up-to-date list of available renters. Various test scenarios will be created and used to verify the robustness and performance of the “Local Peers and Renter Peers Storage” component.

Regarding Implementation 2, it introduced a new client-server architecture featuring a Central Miner Rental Server (CMRS) to manage users renting in or renting out miners. This implementation includes two main components that need to be thoroughly tested.

Firstly, as the "Rental Service Connection Manager" component is designed to manage the matches and connections between rentees and renters, we can test it by creating test cases covering various scenarios, such as multiple simultaneous connections and disconnections due to rental period initializations and expirations (load test) to verify that the component can handle many concurrent requests accurately, security tests to ensure the indirect communications between renters and rentees are least vulnerable, and tests to ensure that the recommendation algorithm can provide the rentees with the available miners that fit to the requirements specified by them.

Secondly, as the "Renter Peers Storage" component supports the operation of “Renter Service Connection Manager” as a private database, we can test it by focusing on its security, data integrity, and performance. Security tests, such as penetration tests and vulnerability scans, can be conducted to protect the sensitive rental information and data. In addition, it is important to evaluate the component's performance under various loads to ensure that it can handle many concurrent requests without compromising its responsiveness and efficiency. Furthermore, since all miner rental announcements and requests must pass through the central server, it is critical to prevent the single-point failure that could disrupt the miner rental service due to unforeseen events such as network outages or power failures. Hence, stress tests on the server will be performed to determine the server’s capacity limits and evaluate the server performance under extreme conditions. These tests should include high network traffic, resource exhaustion, and hardware failure to ensure that the server remains operational under adverse conditions.

4.6 Concurrency

Since the Miner Rental Functionality triggers major enhancements to top-level components like RPC-API, Connection Manager, and Local Peer Storage, the concurrencies between the components and within a component will be changed. For example, the rentee’s RPC-API and Connection Manager must be able to send out instructions to the rented miner and transactions concurrently to ensure that the normal transaction functionalities of Bitcoin Core are not compromised by the enhancement. Moreover, for implementation 1, the upgraded “Local Peers Storage” component must be able to concurrently maintain both “local peers” list and the “renter peers” list if there are simultaneous updates to both of them. As for implementation 2, the CMRS must be able to handle concurrent miner rental announcements and requests without conflicts, and the Connection Manager must be able to send and receive transactions to and from the network while maintaining stable and secure connections with the CMRS. Thus, though the conceptual architecture of Bitcoin Core does not have significant changes due to the enhancement, the effect of it on the concurrency of the existing architecture is not minimal.

5.0 Effects on Conceptual Architecture

5.1 Impact on High-Level Architecture

Although Implementation 1 does not have the "Central Miner Rental Server" (CMRS) in Implementation 2, they both share some same impacts on the high-level Bitcoin Core architecture.

Miner <=> Connection manager: The Miner is modified to be dependent on the Connection Manager for being able to receive instructions from the potential rentees that are transmitted through the Bitcoin Core P2P network to the renter’s Connection Manager, resulting in a two-way dependency between the 2 components (as the Miner also needs to return its current operational status to the rentees through the renter’s Connection Manager). In addition, the Connection Manager is enhanced to be able to maintain a stable and secure connection between the renter's miner and the rentee's RPC-API component or the CMRS during the rental period.

RPC-API <=> Miner: The RPC-API component is modified to be dependent on the Miner. Since the renter should be able to receive some feedback information sent from his Miner through RPC-API for

staying informed about the Miner's operation and resolve any technical issue, a dependency from RPC-API on Miner is formulated, resulting in a two-way dependency (as the Miner is completely dependent on the renter's RPC-API if the miner is no longer for rent).

Under implementation 1, one existing top-level component of Bitcoin Core is modified remarkably:

Local Peers and Renter Peers Storage: The existing "Local Peers Storage" top-level component in a Bitcoin Core instance is upgraded by incorporating the "renter peers" storage functionality, leading each individual node on the Bitcoin Core P2P network to be able to track the currently available renters on the network to support the decentralization nature of implementation 1.

Under implementation 2, a new node is introduced into the Bitcoin Core P2P network.

CMRS: The new type of node added into the P2P network, which serves as a centralized platform for the renters to connect with the prospective rentees.

Rental Service Connection Manager: One top-level component in CMRS, which is responsible for managing users to rent in or rent out miners, maintaining the communications between renters and rentees once a rental relationship is formed, and distributing the rental transactions to the corresponding parties.

Renter Peers Storage: A database component in CMRS, which maintains a private list of active renters to provide the necessary information for the Rental Service Connection Manager to display to the prospective rentees, which eliminates the need for individual peers to store and maintain their own list of renter peers.

5.2 Impact on Low-Level Architecture

Connection manager <=> Miner: Since the 2 implementations share some same impacts on the high-level Bitcoin Core architecture, the shared impacts should be seeable on the low-level Bitcoin Core architecture as well. For example, there will be a new file, named "feedback.cpp", in the Miner (Bitcoin-core/modules/Miner/feedback.cpp) that can return the miner's current operational status to the rentees through the renter's Connection Manager. However, there are also impacts on the low-level architecture that are unique to each implementation.

For Implementation 1, there will be an impacted file in the Connection Manager (Bitcoin-core/modules/ CMRS /Connection_Manager/net.cpp) that is enhanced to include two additional functions, where one is responsible for sending out the RPC-API instructions to the network for the rentees to be able to control the miners of the renters, and the other is responsible for handling the instructions received from the rentees. As for Implementation 2, the same file in the Connection Manager (Bitcoin-core/modules/ CMRS /Connection_Manager/net.cpp) will be enhanced to include two other additional functions, where one is responsible for sending out the RPC-API instructions through the network to the CMRS (which differs from Implementation 1 as the rentees are not directly connected with the renters) for the rentees to be able to control the miners of the renters, and the other is responsible for handling the instructions received from the CMRS (rather than from the rentees).

RPC-API <=> Miner: There will be an impacted file in the Miner (Bitcoin-core/modules/Miner/mining.cpp) that is enhanced to include one additional function, which is sending feedback information to the RPC-API for keeping the owner informed about the miner's current operational status and to resolve any technical issue. Correspondingly, there will be a new file, named "feedback_from_miner.cpp", in the RPC-API (Bitcoin-core/modules/RPC-API) that handles the feedback information received from the Miner, leading this two-way dependency between RPC-API and Miner to be shared by the 2 implementations as this dependency is fundamental to the "Miner Rental Functionality".

Local Peers and Renter Peers Storage (Implementation 1): There will be a new data structure file, named “renter_list.cpp” in the existing “Local Peers Storage” component (Bitcoin-core/modules/Local_Peers_Storage) that has two functions, where one is to store the information of currently available renter peers and to add new renters into the data structure, and the other is to delete the renter peers from the data structure once their miners are rented out or no longer available. Moreover, there will be a new file, named “synchronization.cpp” in the existing “Local Peers Storage” (Bitcoin-core/modules/Local_Peers_Storage) to support the parallel functioning of “local peers” list and “renter peers” list to ensure the data integrity in the component. These two new files lead to the extension of the existing “Local Peers Storage” to the “Local Peers and Renter Peers Storage”.

Renter Peers Storage (Implementation 2): This is a simplified version of “Local Peers and Renter Peers Storage” in Implementation 1 as it is a dedicated database component to maintain the list of active renters. This means that there will be a new data structure file, named “renter_list.cpp” in the Renter Peers Storage (Bitcoin-core/modules/ CMRS/Renter_Peers_Storage) that has two functions, where one is to store the information of currently available renter peers and to add new renters into the data structure if new rental announcements are received from the Rental Service Connection Manager, and the other is to delete the renter peers from the Renter Peers Storage once delete instructions are received from the Rental Service Connection Manager due to the specific miners being rented out or no longer available.

Rental Service Connection Manager: Under implementation 2, there will be a new file, named “receive_trasaction.cpp” in the CMRS (Bitcoin-core/modules/CMRS/ Rental_Service_Connection_Manager) that manages each rental transaction received from a rentee requesting for a specific miner (and will then partition this transaction balance to send transactions to the relevant parties). Moreover, there will be another new file, “receive_renter_from_node.cpp” in the CMRS (Bitcoin-core/modules/CMRS/ Rental_Service_Connection_Manager) that receives the miner rental announcements from the users who want to rent out their miners, and the corresponding rental information will be past and stored into the Renter Peers Storage (triggering the storing function in the “renter_list.cpp” of Renter Peers Storage). Finally, a new file called “miner_recommendation.cpp” will reside in the Rental Service Connection Manager to optimize the rentees’ miner “browsing experience”. This file retrieves the renter information from Renter Peers Storage and displays a list of available miners for rent in the increasing/decreasing order of the miner specifications, rental fees per minute, or popularities that can be specified by the rentees.

Rental Service Connection Manager <=> Renter Peers Storage: Since Rental Service Connection Manager will transmit the renter peers information updates to the Renter Peers Storage to keep track of the currently available miners on the network, and the Rental Service Connection Manager will retrieve the renter information from Renter Peers Storage for the recommendation algorithms to display the miner information in a customized way to the potential rentees, a two-way dependency between Rental Service Connection Manager and Renter Peers Storage is formulated for the smooth operations of CMRS.

6.0 Use Cases

6.1 Use Case 1:

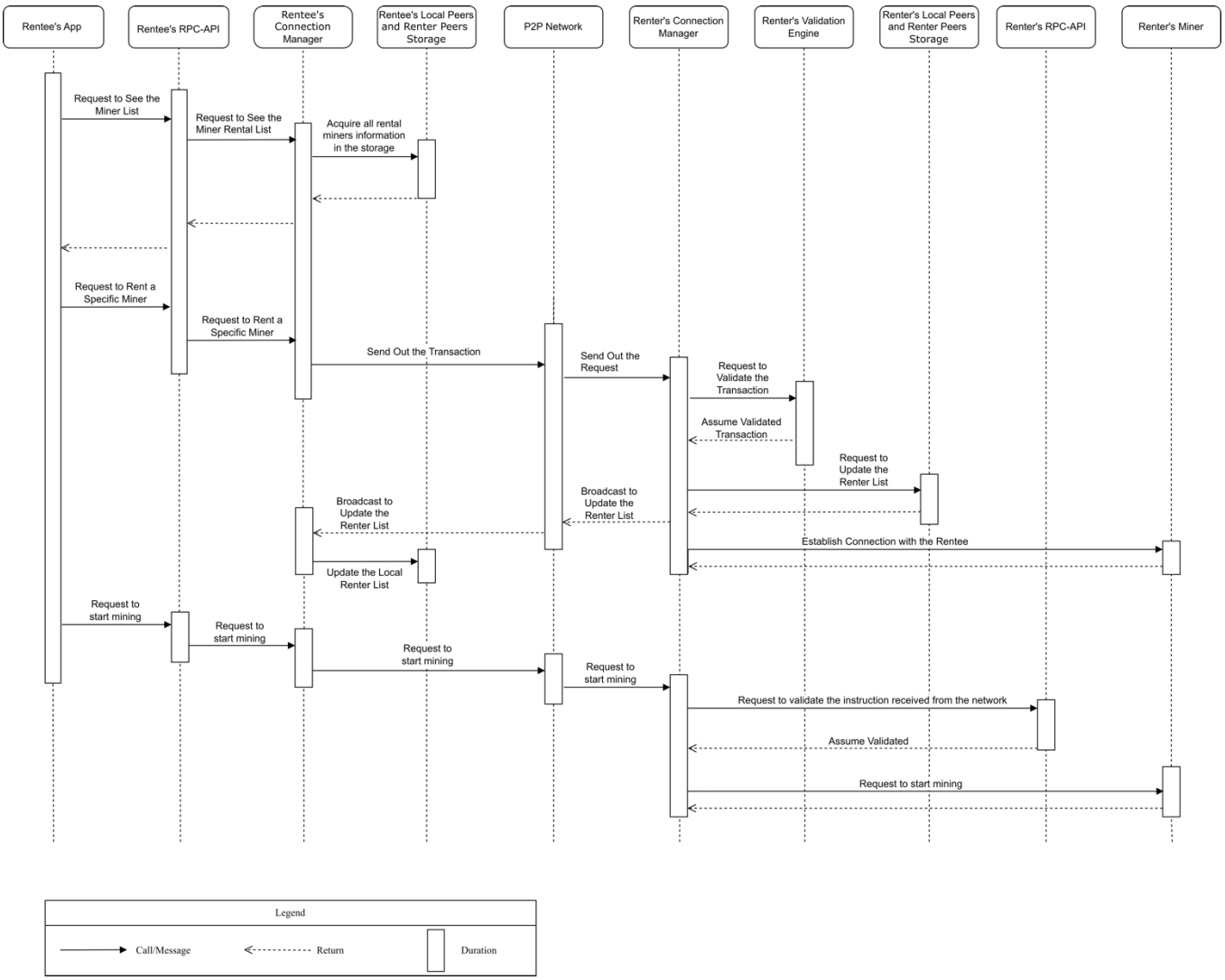

Figure 4: Sequence Diagram of renting in a miner under Implementation 1

This use case is the process of renting in a miner under implementation 1. Firstly, the Rentee sends the request to see the miner list using the RPC-API, the RPC-API will send request to rentee's Connection Manager to see the miner rental list. After rentee's Connection Manager acquires and returns all the rental miners information in the rentee's Local Peers and Renter Peers Storage to the Rentee's APP, the rentee can select the miner he wants to rent by sending a rental transaction to renter through the rentee's RPC-API, Connection Manager, and the P2P network. Once renter's Connection Manager received the rental transaction, it will request the renter's Validation Engine to validate the transaction. If it is validated, the Connection Manager will request to update the renter list in renter's Local Peers and Renter Peers Storage. At the same time, this renter list update will be broadcasted back to the network to update the renter list of all the nodes' Local Peers and Renter Peers Storage on the network. Then, renter's Connection Manager will establish the connection between the miner and rentee. If the rentee sends the request to start mining through the rentee's RPC-API, Connection Manager, and the Network, it will be received by the renter's Connection Manager, and it will request the renter's RPC-API to validate this starting mining instruction received from the network. Once it is validated, the renter's Connection Manager will request the Miner to start mining.

6.2 Use Case 2:

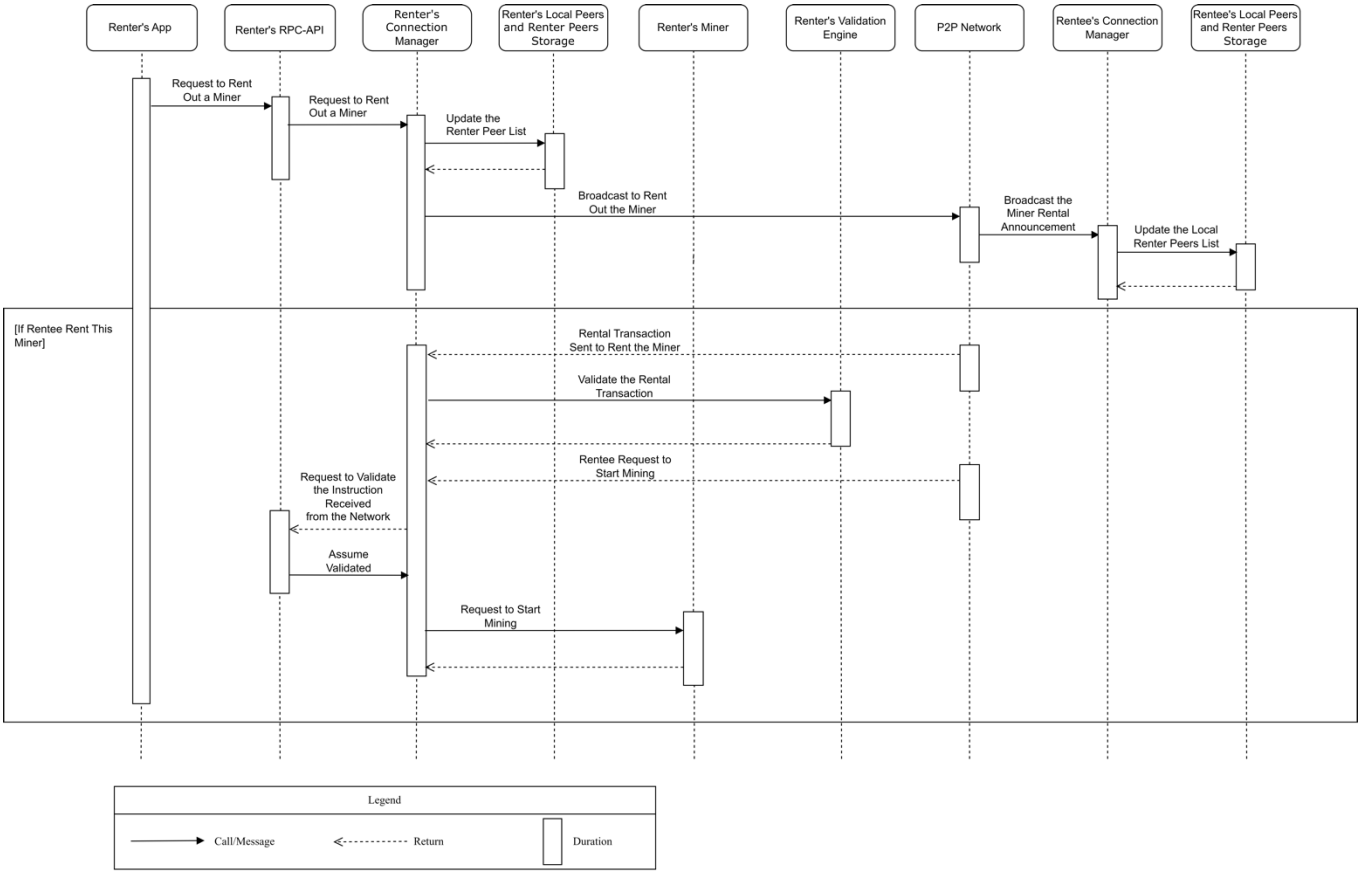

Figure 5: Sequence Diagram of renting out a miner under Implementation 1

This use case is the process of renting out a miner under implementation 1. Firstly, the renter sends the request to rent out a miner using the RPC-API, which will trigger the renter's Local Peers and Renter Peers Storage to be updated. After returning to the renter's Connection Manager, the Connection Manager will broadcast the rent out announcement to the P2P Network, leading this miner rental announcement to be received by every node on the network, which are the potential rentees. The rentee's Connection Manager will correspondingly send the request to update the local renter peers list in the rentee's Local Peers and Renter Peers Storage. If a rentee decides to rent this miner, the rental transaction will be sent through the P2P Network to be received by the renter's Connection Manager. Upon receiving the rental transaction, renter's Connection Manager will request the renter's Validation Engine to validate the transaction. If it is validated, the rentee can request to the miner to start mining through the P2P network. When receiving this start mining instruction, renter's Connection Manager will request the RPC-API to validate the instruction. If it is validated, the renter's Connection Manager will request the miner to start mining.

7.0 Potential Risks and Limitations

One potential risk can be that since this enhancement allows each user on the Bitcoin Core P2P network to be able to obtain the authorizations to control the miners of the other users on the network, there can be security vulnerabilities that could be exploited by malicious actors. For example, an attacker could attempt to bypass the security protocols or the rental server to obtain the controls of a large number of renters' miners on the network, enabling the attacker to gain absolute advantages on the mining

competition and to cause financial losses to the renters. One possible approach to mitigate this security risk is to implement monitoring and logging tools on the miners to detect and respond to any potential unauthorized access.

Another potential risk can be the legal and regulatory risk. The introduction of Miner Rental Functionality may draw increased scrutiny from regulatory bodies and lawmakers, who may view the functionality as a potential avenue for money laundering or other illegal activities. For example, the money launderers can buy many miners to rent out to the other users on the Bitcoin Core P2P network and use the proceeds from the rental fees to launder their illicit funds, which could lead to legal and reputational risks for Bitcoin Core and its community. To minimize this risk, proper know-your-customer (KYC) and anti-money laundering (AML) procedures may need to be implemented, and efforts should be made to ensure compliance with any applicable regulation.

The centralization of mining rights may also be a potential risk of the "Miner Rental Functionality", because allowing users to rent miners from other users can lead to a situation where a few users accumulate a large portion of the network's mining capacity. This centralization of mining capacity has the potential to undermine the decentralized nature of Bitcoin Core P2P network, which can become a hidden danger to the value of Bitcoin. Therefore, mechanisms may be needed to ensure that no single user or group gains excessive control over the mining resources on the network.

As for the limitations, since the "Miner Rental Functionality" relies on the voluntary participation of users in renting out their miners, it can lead to fluctuations in the number of miners available for rent at a given time, which is a limitation that may affect the rental market. For example, the rentees seeking to rent miners may not find a sufficient number of offering miners for rent due to the mismatch between supply and demand. Hence, it may be helpful to implement an incentive mechanism that encourages users to maintain a steady supply of available mining machines for rent, promoting a more consistent rental market to ensure an equilibrium between demand and supply.

8.0 Lessons Learned

After a semester's study and cooperation, the team members have gained a deep understanding of the software architecture of Bitcoin Core. Under the guidance of the team leader and the active cooperation of the team members, we quickly came up with a consensus idea about the enhancement to the current Bitcoin Core, the 'Miner Rental Functionality'. We studied and designed an implementation of the enhancement based on the proposed architecture of Bitcoin Core itself. Then, through brainstorming, we came up with another implementation with a dedicated server that goes against the Bitcoin Core decentralized nature. In the analysis of the implementations, we found that our ideas lacked deliberation, but after in-depth studies and discussions of how to realize the implementations in practice, we articulated more details to be dealt with. However, we believe that we can optimize the implementations of the enhancement if we can have an opportunity to consult with Bitcoin Core experts.

9.0 Data Dictionary & Naming Conventions

Renter: The individual user who owns a separate and dedicated mining machine and is willing to rent the machine out to another individual user on the Bitcoin Core P2P Network for a certain fee.

Rentee: The individual user who pays a certain rental fee to the renter in exchange for using the renter’s mining equipment for bitcoin mining during the agreed-upon rental period. Once the rental period expires, the rentee is prohibited from accessing the renter’s miner anymore unless a new rental period is initiated.

ASIC: Application-Specific Integrated Circuit. In the context of bitcoin mining, ASICs refer to the dedicated hardware devices that are designed specifically for the purpose of mining cryptocurrencies. For example, the SHA256 function is directly placed on the silicon chips of the devices.

SAAM: Software Architecture Analysis Method, this method focuses on architecture significant use cases. To compare and contrast candidate architectures, SAAM applies scenarios as benchmarks (Kommadi, 2021).

CMRS: Central Miner Rental Server under implementation 2. The new type of node that was added to the P2P network serves as a centralized platform for the renters to connect with the prospective rentees. It acts as a connection and communication agent between renters and rentees once the rental relationships are formed and it is maintained by the most stable nodes on the network.

Renter Peers Storage: A data structure that maintains an active list of currently available renters on the network, which supports the operations of "Rental Service Connection Manager" as the private database residing in CMRS.

10.0 Conclusion

As the Bitcoin mining market and its user base expand, there is a greater need for flexibility in bitcoin's mining capabilities, leading us to propose the "Miner Rental Functionality" as an enhancement to benefit Bitcoin Core users. The proposed enhancement's analysis and benefits have been performed in order to properly document and evaluate its values and performance. Functional requirements, non-functional requirements, and stakeholders were taken into consideration during the enhancement's SAAM analysis. Comparing the two implementations, efficiency and requirements were analyzed separately in detail. In addition, two use cases and sequence diagrams were brought up to further evaluate the execution of the enhancement. Finally, potential risks and major limitations that the enhancement will possibly cause in the production environment were considered.

11.0 Reference

Antonopoulos, A. M. (n.d.). *Mastering Bitcoin*. O'Reilly Online Learning. https://www.oreilly.com/library/view/mastering%20bitcoin/9781491902639/ch06.html#full_node_reference

Chapter 10: "Mining and Consensus" · GitBook. (n.d.). https://cypherpunks-core.github.io/bitcoinbook/ch10.html

Kommadi, B. (2021, December 13). SAAM : Software Architecture Analysis Method - Bhagvan Kommadi - Medium. *Medium*. https://medium.com/@bhagvankommadi/saam-software-architecture-analysis-method-36864cd8ea94

Johnson, G. (2023). How to Use Bitcoin Core to Mine. *Crypto Wisdom Australia*. https://cryptowisdom.com.au/how-to-use-bitcoin-core-to-mine/