École Polytechnique Fédérale de Lausanne

Optional Semester Project

Professor: Rachid Guerraoui

Improving the Efficiency of a Blockchain Protocol

Author:
Iva Najdenova

Supervisor:
Matej Pavlovic

Contents

List of Figures

List of Tables

List of Charts

1 Introduction

Blockchain technology is nowadays considered to be applicable in every system that needs to preserve data in form of a registry, which is at the same time resistant to data manipulation. It is the preferred choice when it comes to implementing a digital ledger service, as all its transactions are cryptographically secured and provide integrity.

Details about the classic blockchain protocol used in the Bitcoin system can be found in the article [3]. There are several inconveniences regarding this approach like high latency of transactions' confirmation and waste of resources, caused by the unsuccessful miners when trying to append a new block. A solution for these two problems is proposed in the article [1] and this new modified blockchain protocol is named Collaborative Block Reinforcement Protocol.

The goal of this project is to improve the first implementation, [2], of the abovementioned Reinforcement protocol, and consequently obtain results which show the benefits of this approach when compared to the standard Bitcoin Blockchain protocol. To be more specific, we show that the Collaborative Block Reinforcement Protocol is better resistant to malicious behavior, i.e. the blockchain is harder to fork by malicious miners than the Bitcoin blockchain. When it comes to the implementation, we were using Python as programming language, together with the asynchronous framework Twisted, because it is suitable for distributed, event driven programming.

The structure of this report, apart from introduction and conclusion, includes four main sections. In the first part, we cover the main characteristics of the Collaborative Block Reinforcement Protocol and how it compares to the Bitcoin Blockchain Protocol. Furthermore, in the second and third sections, we describe the malicious strategies for canceling blocks and the results from our experiments respectively. Finally, in the fourth division, we include some technical details about our implementation of the Collaborative Block Reinforcement Protocol.

2 Overview of the Collaborative Block Reinforcement Blockchain Protocol and Comparison with Bitcoin

The article [1] defines the principles of the Collaborative Block Reinforcement Blockchain Protocol:

1. Miners mine on top of what they believe is the last block of the blockchain, independently of what the next block will be.
2. During this mining process, some POW (proof of work) is generated at each miner locally, even if that miner does not succeed in appending a new block.
3. When a miner *m* succeeds in computing a POW that is sufficient to append a new block, it *proposes* a new block by broadcasting it, together with its POW.
4. The proposal of a new block is interpreted as a request for support by the other miners, which may decide to reinforce that block by sending their POWs to *m*.
5. m, after collecting the reinforcements from other miners, commits the block by broadcasting the reinforcements.
6. Miners start mining on top of the new block (it is only possible to mine on top of committed blocks).

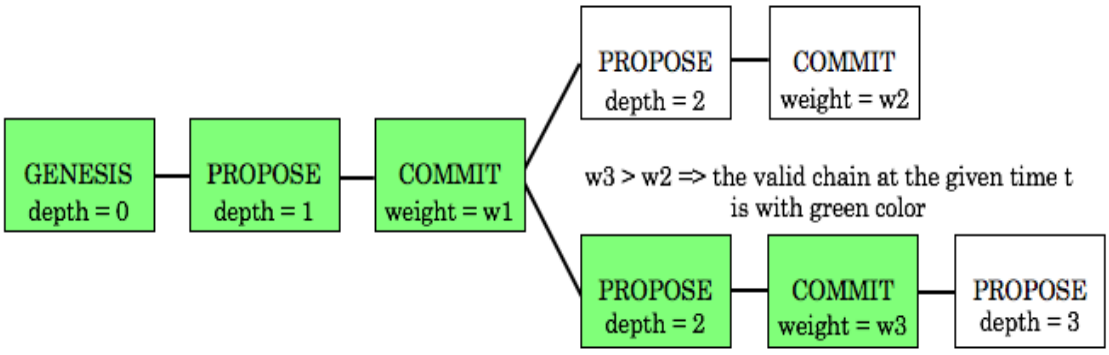

Figure 1: Collaborative Block Reinforcement Blockchain Protocol

It is based on the classic Bitcoin protocol, which on high level is: [1]

1. Miners mine on top of what they believe is the last block of the blockchain by trying to compute a POW involving both the last block and the new block to be appended.
2. When a miner m succeeds, it broadcasts the new block, together with its POW.
3. Miners start mining on top of the new block.

Unlike Bitcoin where the longest chain is considered to be the valid one, the Reinforcement protocol defines the valid blockchain to be the heaviest one (Figure 1), i.e., the linear chain with the largest cumulative amount of POW associated with its blocks (a block can be considered valid if it has the heaviest sub-tree). The POW associated with a block consists of the POW required to append that block, and the POWs that are used to reinforce it. [1]

The weight of a block is computed as the sum of the weights of all reinforcements backing that block, and the weight of each reinforcement is: $w = \frac{d}{h}$, where d is the mining difficulty and h is the hash produced by the nonce.[1] In order to avoid having enormous weights produced by one-time luck, that could unfairly cancel other blocks, in our implementation we decided to bound the weight of each reinforcement and compute it as: $w = min(1, \frac{d}{h})$.

We expect the benefits from having the Reinforcement protocol to be: shorter delay between the time when a transaction is proposed and the time when the transaction can be considered confirmed with high probability, as well as minimization of the amount of work that is "wasted" by unsuccessful miners in systems using the Bitcoin Blockchain Protocol. [1] We assume that it takes less time for a transaction to be considered confirmed with high probability if the chain is harder to fork, and therefore better resistant to malicious behavior.

3 Block-Canceling Strategies

In order to evaluate whether the Collaborative Block Reinforcement protocol is truly better resistant to malicious behavior than the Bitcoin one, we had to conduct several tests. In those experiments, the malicious miners were "told" to follow one of the following block-canceling strategies:

- Cancel Every Block

Malicious miners performing this strategy, wait until an honest node adds a block to the chain, and then they try to cancel it by appending either a heavier block for the Reinforcement version (Figure 2) or two blocks in a row for the Bitcoin version (Figure 3) before the honest ones do.

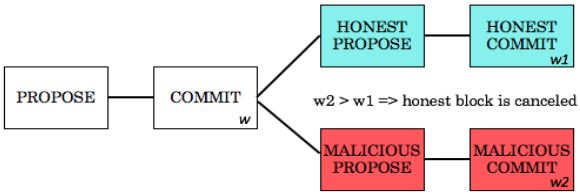

Figure 2: Cancel Every Block - Reinforcement Version

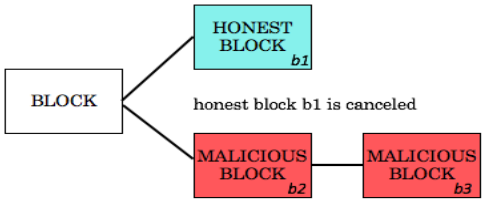

Figure 3: Cancel Every Block - Bitcoin Version

As mentioned in [2], regarding the Reinforcement version, if a malicious miner is the first to find a new block, it advertises it to all other malicious nodes via a separate channel so the others know that they should wait for a propose from that node and then reinforce it. But if some of the honest nodes proposes a new block first, the malicious miners give up on this depth and start mining on top of the new honest block so that they can cancel the future honest block from the next depth.

In the Bitcoin version, similarly, when a malicious miner is the first to find a new block, it advertises it to all other malicious nodes via a separate channel so the others know that they shouldn't append theirs if found in the future, but they give up on this depth only if the honest miners have appended 2 blocks before them.

In order to improve the implementation of this malicious mining for the Bitcoin version in [2], we included selfish mining, which requires, when receiving the secret malicious block, all malicious nodes to immediately start mining on top of it instead of waiting for the honest block to show up. This way it will be comparable to the Reinforcement version because there the malicious nodes never stop mining and discovering hashes, later used as reinforcements. Another issue with the Bitcoin malicious mining in [2], was a coding error that was causing only one particular malicious miner to propose every time, which was obviously hindering the full usage of the malicious computing power. Once fixed, the results drastically changed in the direction of our expectations.

○ Cancel a Particular Block

Malicious miners performing this strategy need to pre-agree on the depth of the block they want to cancel, and later they fork the chain on that particular depth by appending malicious blocks until they succeed in creating heavier/longer branch than the honest one (Figure 4 and 5). But if the honest branch becomes much longer (Bitcoin version) or heavier (Reinforcement version) than the malicious one, the probability of canceling the chosen block becomes very low (tending towards zero) so the malicious nodes give up. If the majority of miners is honest, as time goes by it will be less and less likely malicious nodes to cancel the block they have agreed to, because they will need to be constantly luckier for a long time in finding hashes than the honest nodes.

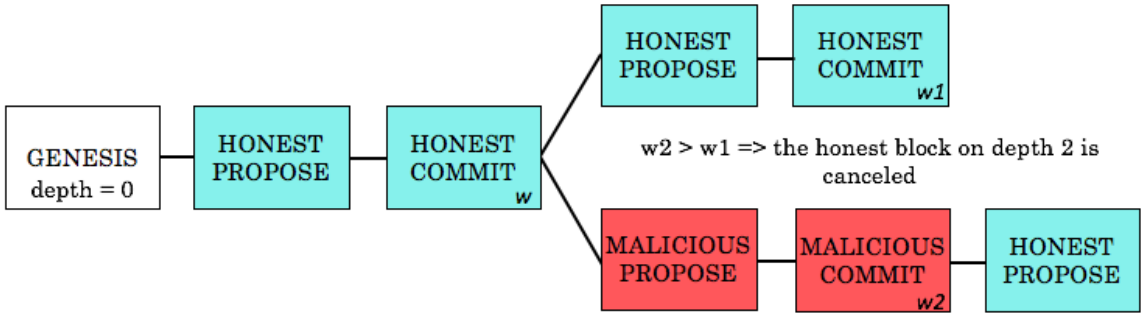

Figure 4: Cancel a Particular Block - Reinforcement Version

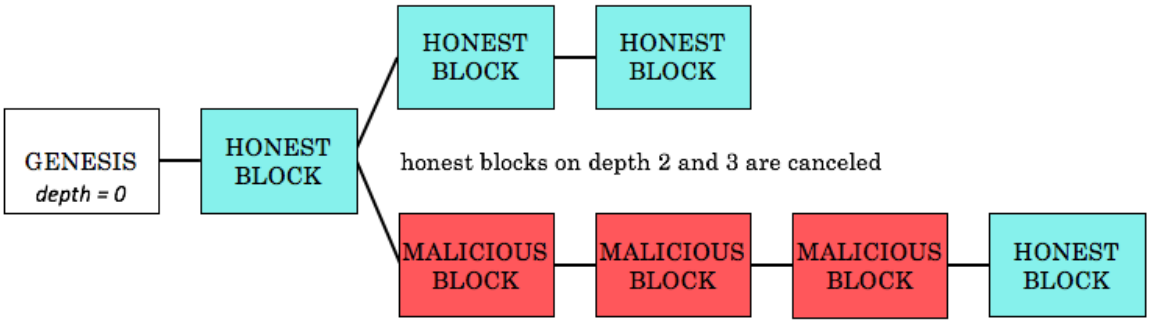

Figure 5: Cancel a Particular Block - Bitcoin Version

The results obtained in [2] for this strategy were also contrary to our beliefs as for the before mentioned Cancel Every Block strategy, so we had to come up with ways of improving them.

We started by following the analysis notes from [2] for the Reinforcement version:

"Firstly, the malicious miners have slightly more time to find a hash than the honest miners. When a propose block is committed by an honest node all the other honest nodes send their reinforcements and stop mining, waiting for the commit block. This makes it impossible to find a reinforcement with weight greater than 1. Also, this gives the malicious nodes some extra time to find a nonce large enough if they didn't find it yet."

Taking into consideration this statement, we made both honest and malicious nodes never stop mining and continue sending reinforcements one by one immediately upon being found, when they are in either Reinforcement Sent[2] or Reinforcement Collecting[2] state.

Furthermore, we realized that the way of computing the weight of every proof of work in [1] and [2] ($w = \frac{d}{h}$; d->mining difficulty, h->hash) is causing very unstable results, because a one time luck might produce a very heavy block that can cancel blocks from other branches in a somewhat unfair manner. For this reason, we decided to set an upper limit - the value 1, for every proof of work that is backing up the block.

Another improvement that made the Reinforcement Version even better resistant to malicious behavior, was setting the SWITCH_TH[2] to 1 instead of 0. This variable denotes that miners should switch a branch only if another one is heavier than their branch plus the currently found reinforcements, by this value. This approach avoids the case where malicious miners are lucky at the beginning by a little bit and succeed in canceling the desired block. The SWITCH_TH variable should remain 0 for the Bitcoin version because there the notion of weight difference lower than 1 doesn't exist (for example malicious nodes cannot have advantage of 0.5).

We should also note that the coding error for the Bitcoin version, mentioned in the previous strategy (Cancel Every Block), where only one malicious miner was always appending blocks, had impact on this strategy as well.

Another issue that we ran into, was that after some period of mining, few of the nodes stopped appending new blocks. After suspecting that this is because of problems with the CPU, we included the "time.sleep(0.015)" code in the mine() method of the Hash class[2] to slow the hashing down. It improved the CPU load, but didn't fix our issue. After spending some time investigating what the potential cause could be, we found out that we should manually increase the size of the internal threadpool by adding reactor.suggestThreadPoolSize(40) before the callFromThread method.

After making all corresponding changes, we were pleased with the outcome of having a blockchain which is harder to fork than the one from the classic Bitcoin blockchain protocol (shown with the experiments in section 4). This implies that with the Collaborative Block Reinforcement Protocol it takes less time for a transaction to be considered committed with high probability, and showing this was exactly one of the main goals of the project.

4 Experiments

In this part, we are going to observe and compare the behavior of both Collaborative Block Reinforcement Protocol and the regular Bitcoin Blockchain Protocol, under several different configurations. Parameters that vary in these experiments are: the number of honest and malicious miners, the block-canceling strategy that the malicious nodes follow (cancel every or cancel a particular block) and sometimes the variable SWITCH_TH (explained in section 3).

For the experiments that make use of the Cancel Every Block Strategy, nodes have the possibility to cancel 25 blocks and therefore they are mining till the depth 25 (RF Version) or 50 (Bitcoin Version). At the end we count how many honest blocks were canceled. For more consistent result, we perform 5 runs for both versions and take the average number of canceled blocks.

For the experiments that make use of the Cancel Particular Block Strategy, we perform 20 runs for both Reinforcement and Bitcoin Version and then we count how many times have the malicious miners succeeded in canceling the chosen block.

The testing conditions are same as the ones explained in [2], with one difference that now a 1.6 GHz Intel Core i5 processor was used. The exact values of the configuration constants are given in the appendix.

The aim of the experiments that follow is to see with which blockchain protocol, it is more difficult malicious miners to fork the chain.

○ Configuration I: 4 Honest 3 Malicious Miners

We start with the same configuration as in [2], so that we can compare them and visually see the improvements coming from our new implementation. Now we set the SWITCH_TH to 1 for the Reinforcement Version and 0 for the Bitcoin version.

▪ Cancel Every Block Strategy

As indicated before, in this strategy, malicious miners try to cancel as many blocks as possible out of 25.

The tables (Table 1 and 2) and chart (Chart 1) below show that on average malicious miners in the Bitcoin version are more successful in canceling blocks when compared with the Collaborative Block Reinforcement Protocol. This was not the case in the implementation [2], where the Bitcoin Blockchain Protocol demonstrated better performance against malicious behavior (Chart 2).

Bitcoin	# Honest wins	# Malicious wins	% Malicious wins
1°	19	6	24%
2°	20	5	20%
3°	21	4	16%
4°	18	7	28%
5°	17	8	32%
AVG	19	6	24%

Table 1: Cancel Every Block with 4 Honest and 3 Malicious - Bitcoin Version

Reinforcement	# Honest wins	# Malicious wins	% Malicious wins
1°	21	4	16%
2°	24	1	4%
3°	21	4	16%
4°	25	0	0%
5°	22	3	12%
AVG	22.6	2.4	9.6%

Table 2: Table 1: Cancel Every Block with 4 Honest and 3 Malicious - Reinforcement Version

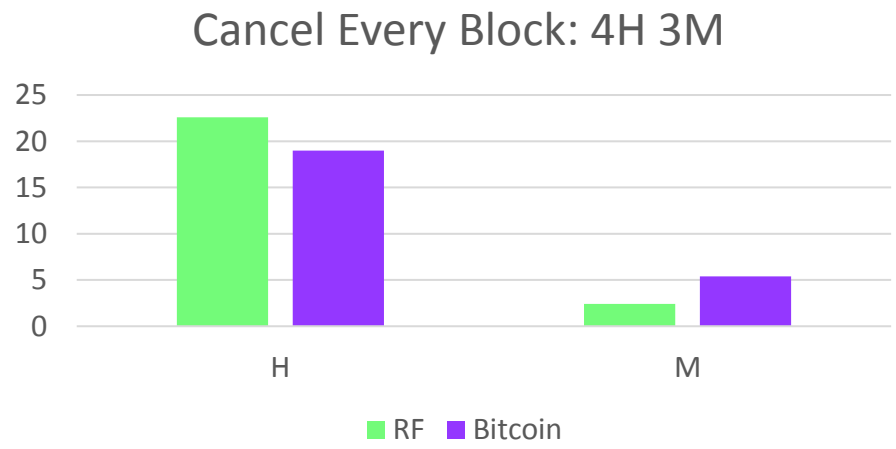

Chart 1: Cancel Every Block with 4 Honest and 3 Malicious Miners

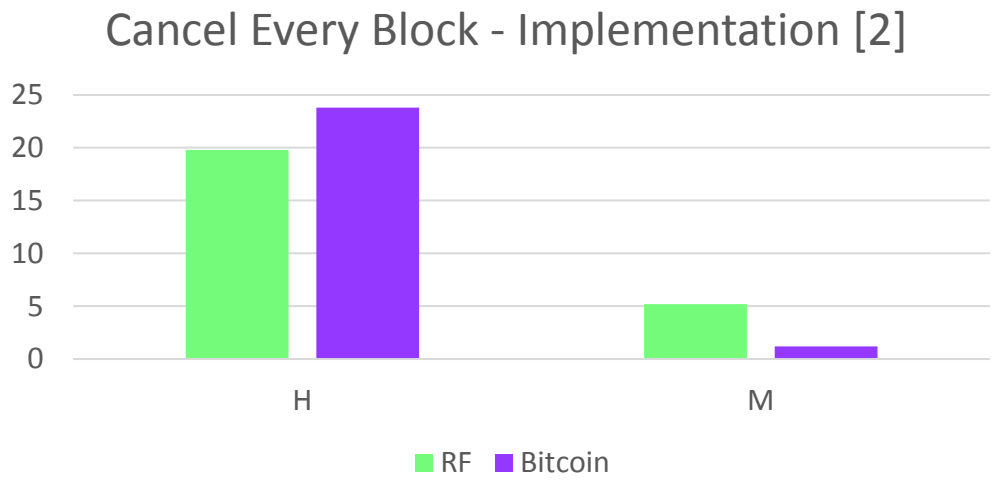

Chart 2: Cancel Every Block with 4 Honest and 3 Malicious Miners - Implementation [2]

- Cancel a Particular Block Strategy

As explained in the previous section, in this strategy, malicious miners pre-agree on the depth of the block they will try to cancel. For testing reasons we count how many times out of 20 they succeed in canceling the desired block.

The tables (Table 3 and 4) and chart (Chart 3) below show that malicious miners in the Bitcoin version are much more successful in canceling the pre-agreed block when compared to the Reinforcement version. This was again not the case in the implementation [2], where the Bitcoin Blockchain Protocol demonstrated better (but unrealistic) performance against malicious behavior (Chart 4).

Bitcoin	# Honest wins	# Malicious wins	% Malicious wins
20 Runs	4	16	80%

Table 3: Cancel a Particular Block with 4 Honest and 3 Malicious - Bitcoin Version

Reinforcement	# Honest wins	# Malicious wins	% Malicious wins
20 Runs	14	6	70%

Table 4: Cancel a Particular Block with 4 Honest and 3 Malicious - Reinforcement Version

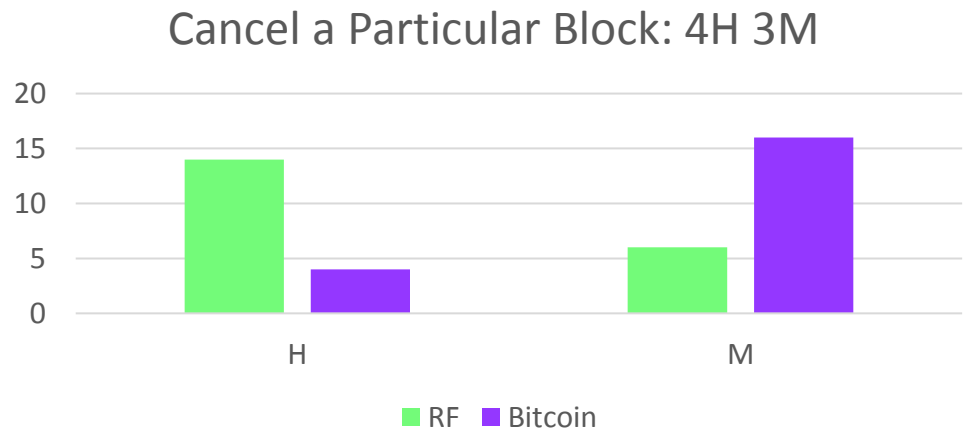

Chart 3: Cancel a Particular Block with 4 Honest and 3 Malicious Miners

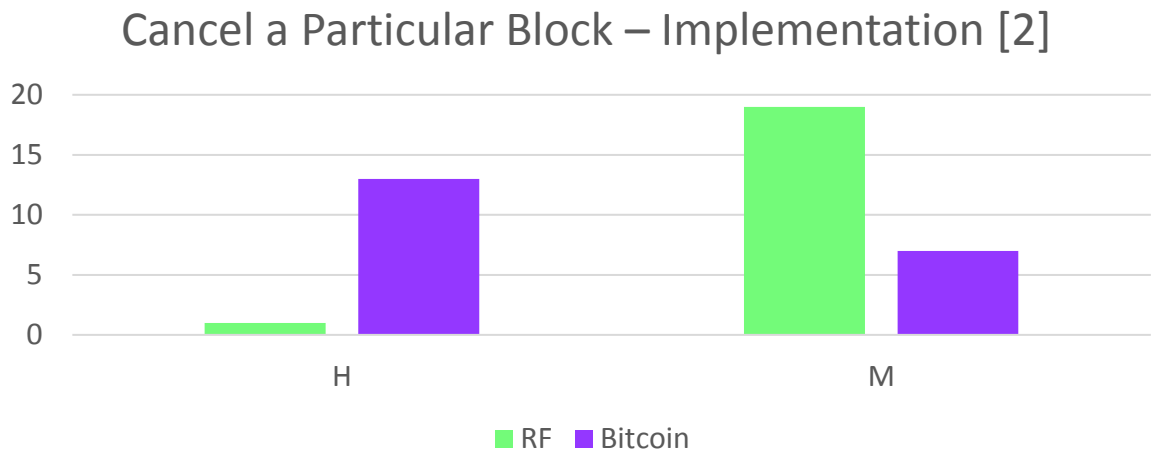

Chart 4: Cancel a Particular Block with 4 Honest and 3 Malicious Miners - Implementation [2]

In these experiments (Cancel a Particular Block strategy), as it was expected, malicious miners were successful only if they were luckier than the honest nodes at the beginning (Figure 6). Otherwise, it becomes less and less likely with the time for them to cancel the desired block (Figure 7).

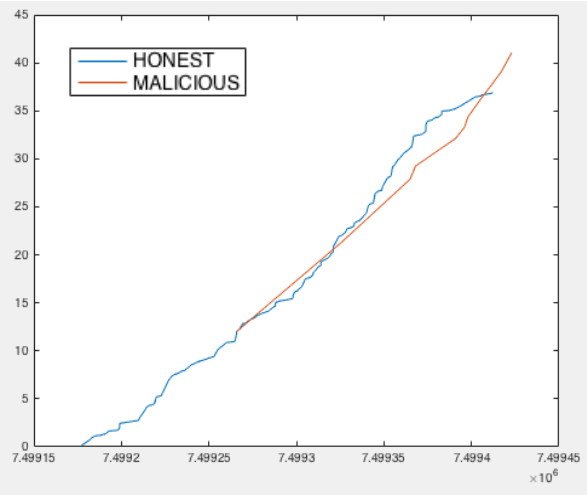

Figure 6: Cancel a Particular Block - RF version- 4 Honest and 3 Malicious - Malicious Winning Plot

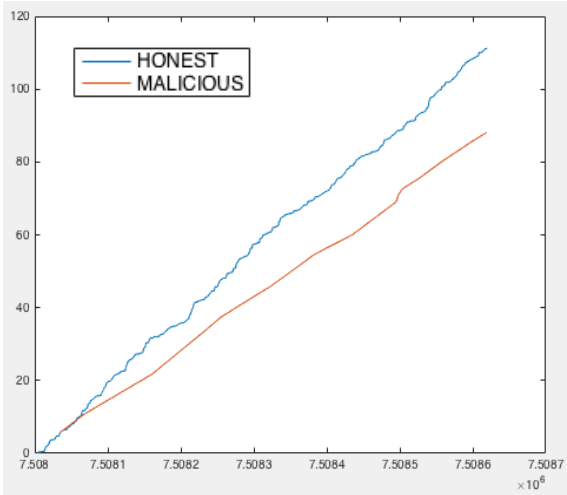

Figure 7: Cancel a Particular Block - RF version - 4 Honest and 3 Malicious - Honest Winning Plot

In order to see the comparison between the Reinforcement and Bitcoin version when the SWITCH_TH is set to 0 in both of them we conducted the same experiments and obtained the following results:

Cancel Every B.	# Honest wins	# Malicious wins	% Malicious wins
Bitcoin AVG	19	6	24%
Reinforcement AVG	20	5	20%

Table 5: Cancel Every Block with 4 Honest and 3 Malicious - SWITCH_TH=0

Cancel Particular	# Honest wins	# Malicious wins	% Malicious wins
Bitcoin (20 runs)	4	16	80%
Reinforcement (20 runs)	9	11	55%

Table 6: Cancel a Particular Block with 4 Honest and 3 Malicious - SWITCH_TH=0

As expected, we observe worse performance of the Reinforcement version than before when the SWITCH_TH was 1, but it is still slightly better than the Bitcoin version.

Conclusion from the tests conducted with Configuration I:

The Collaborative Block Reinforcement Protocol outperformed the classic Bitcoin Blockchain Protocol, by being better resistant to malicious behavior in both Cancel Every Block and Cancel a Particular Block strategies.

Here, it is also important to highlight that our modifications to the implementation [2] were successful.

○ Configuration II: 3 Honest 7 Malicious Miners

In this case, the number of malicious miners is significantly greater than the number of honest nodes. The SWICH_TH constant is again set to 1 for the Reinforcement version and 0 for the Bitcoin one.

▪ Cancel Every Block Strategy

From the following tables (Table 7 and 8) and Chart 5, we can observe that both Bitcoin and Reinforcement version have approximately equal performance against malicious behavior. But this is a quite reasonable outcome in a system dominated by malicious miners, because honest nodes in the Reinforcement version win only if they are lucky enough to find the first propose block before the malicious do (Figure 2) and in this scenario malicious miners immediately give up. Otherwise, it is very unlikely to create a heavier block whilst having much less computing power. Similarly, for the Bitcoin version, in order honest nodes to be successful they also need a one-time luck to append the second block before the malicious do (Figure 3).

Bitcoin	# Honest wins	# Malicious wins	% Malicious wins
1°	9	16	64%
2°	10	15	60%
3°	8	17	68%
4°	11	14	56%
5°	7	18	72%
AVG	**9**	**16**	**64%**

Table 7: Cancel Every Block with 3 Honest and 7 Malicious - Bitcoin Version

Reinforcement	# Honest wins	# Malicious wins	% Malicious wins
1°	9	16	64%
2°	5	20	80%
3°	10	15	60%
4°	7	18	72%
5°	11	14	56%
AVG	**8.4**	**16.6**	**66.4%**

Table 8: Cancel Every Block with 3 Honest and 7 Malicious - Reinforcement Version

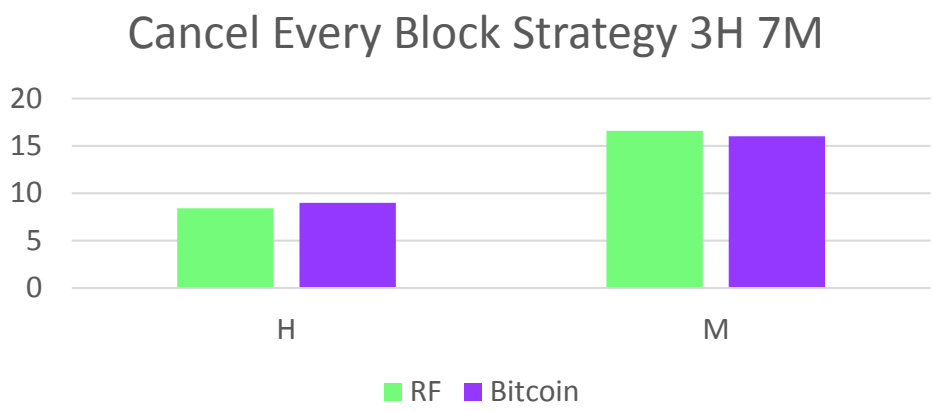

Chart 5: Cancel Every Block with 3 Honest and 7 Malicious Miners

- Cancel a Particular Block Strategy

When malicious miners were following this strategy, they were 100% successful in both Reinforcement and Bitcoin version. This outcome was expected because the malicious nodes don't give up if they don't cancel the block immediately, instead they carry on mining and they have the advantage of more computing power.

Bitcoin	# Honest wins	# Malicious wins	% Malicious wins
20 Runs	0	20	100%

Table 9: Cancel a Particular Block with 3 Honest and 7 Malicious - Bitcoin Version

Reinforcement	# Honest wins	# Malicious wins	% Malicious wins
20 Runs	0	20	100%

Table 10: Cancel a Particular Block with 3 Honest and 7 Malicious - Reinforcement Version

On the figure bellow (Figure 8), we can see the standard case of malicious miners canceling the pre-determined block, in their early stage of mining. Honest nodes never succeed in preventing the block cancelation, with this configuration.

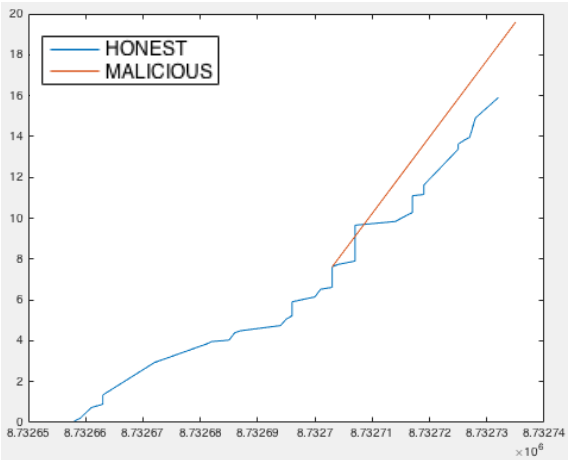

Figure 8: Cancel a Particular Block - 3 Honest and 7 Malicious - Malicious Winning Plot

○ Configuration III: 7 Honest 3 Malicious Miners

In this case, the number of malicious miners is significantly lower than the number of honest nodes. The SWICH_TH variable is, once more, set to 1 for the Reinforcement version and 0 for the Bitcoin one.

▪ Cancel Every Block Strategy

From the following tables (Table 11 and 12) and Chart 6, we can observe that our expectations of the Reinforcement version having better performance than the Bitcoin version are confirmed.

Bitcoin	# Honest wins	# Malicious wins	% Malicious wins
1°	20	5	20%
2°	21	4	16%
3°	17	8	32%
4°	22	3	12%
5°	22	3	12%
AVG	**20.4**	**4.6**	**18.4%**

Table 11: Cancel Every Block with 7 Honest and 3 Malicious - Bitcoin Version

Reinforcement	# Honest wins	# Malicious wins	% Malicious wins
1°	25	0	0%
2°	25	0	0%
3°	24	1	4%
4°	25	0	0%
5°	25	0	0%
AVG	**24.8**	**0.2**	**0.8%**

Table 12: Cancel Every Block with 7 Honest and 3 Malicious - Reinforcement Version

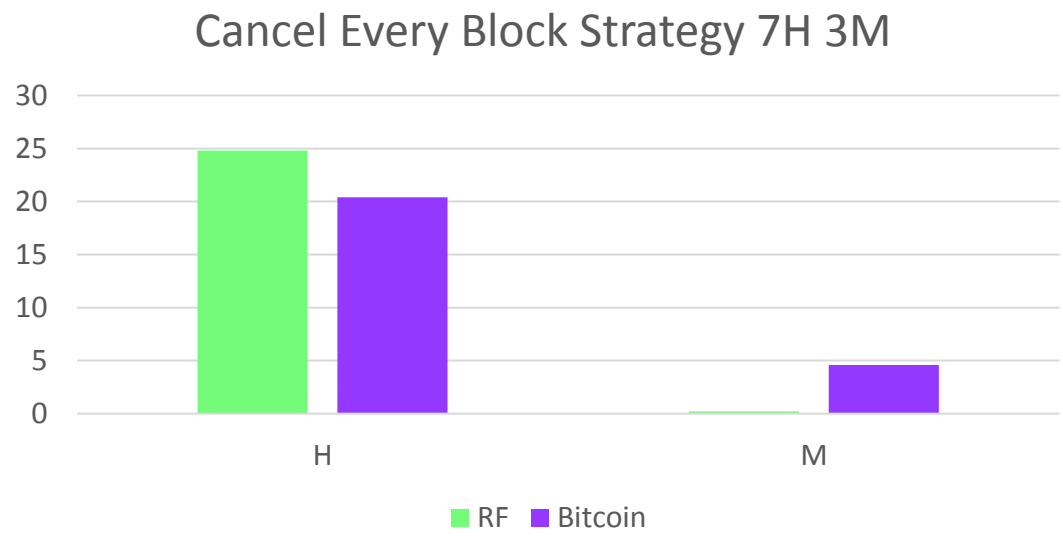

Chart 6: Cancel Every Block with 7 Honest and 3 Malicious Miners

- Cancel a Particular Block Strategy

When malicious miners were following this strategy, again the Reinforcement version proved itself to be better resistant to malicious behavior than the Bitcoin one (Tables 13 and 14).

Also, if we compare Figure 9 to Figure 7 where the number of honest nodes is lower and the number of malicious remains the same, we can observe that the area between the two curves is larger in Figure 9, as it is expected.

Bitcoin	# Honest wins	# Malicious wins	% Malicious wins
20 Runs	15	5	20%

Table 13: Cancel a Particular Block with 7 Honest and 3 Malicious - Bitcoin Version

Reinforcement	# Honest wins	# Malicious wins	% Malicious wins
20 Runs	20	0	0%

Table 14: Cancel a Particular Block with 7 Honest and 3 Malicious - Reinforcement Version

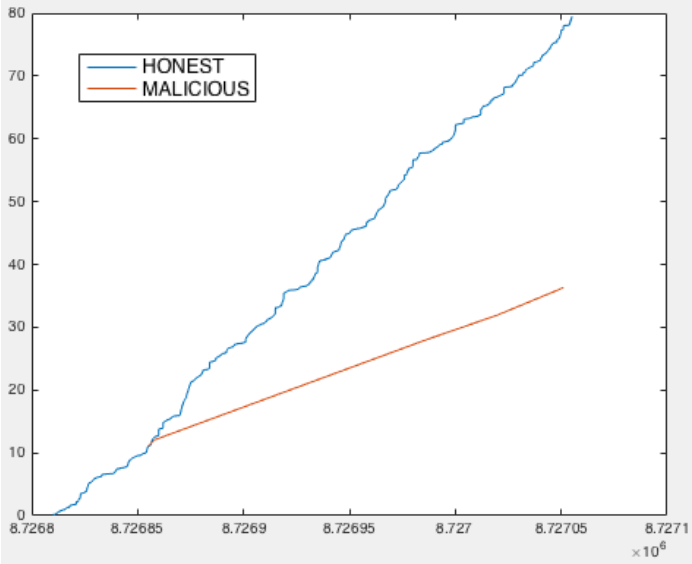

Figure 9: Cancel a Particular Block - RF version - 7 Honest and 3 Malicious - Honest Winning Plot

5 Some Technical Notes

In what follows, we give some important technical details about our implementation.

- *Run*

In order to start the mining, it's needed to create a script that first configures the parameters with a command like this:

```
python3 setup.py -p -a -r 10 3 2
```

where -p is omitted for the Reinforcement version, and -a is omitted for the cancel a particular block strategy. The numbers after -r mean: total number of miners, number of malicious miners, and number of clients that broadcast transactions.

Then, we need to start all clients and all miners with the following commands where the number indicates the id of the proper client or miner (first numbers are malicious if any):

```
python3 client.py 1 &

python3 main.py 1 &
```

- *Reading the results*

Every miner produces a log that describes all the messages sent and received as well as all the blocks appended to the blockchain and the blocks on top of which he is mining.[2]

When using the Cancel Every Block strategy, the easiest way to find out how many blocks were canceled by the malicious miners in the Reinforcement version is to search for the word "WIN" in a log of a malicious miner, whereas in the Bitcoin version we can discover the number of blocks that weren't canceled if we search for the word "LOSE" in a log of an honest miner.

When using the Cancel a Particular Block strategy, in both Reinforcement and Bitcoin versions, the result is printed out in the terminal.

- *Constants*

The exact values of the constants used for our experiments are given in the Appendix.

The Bitcoin version is meant to be run with SWITCH_TH = 0, but if we want to change this value we need to make a modification in the code, because otherwise the miners are stuck at depth 1 when the SWITCH_TH is set to 1 for example. The exact condition causing this issue is in miner.py and is marked with FIXME comment.

- *Technical debt*

We did not fix an error in [2] that happens when blocks arrive out of order because of network issues. In that case a pool of blocks is used. This scenario happens only

if the number of miners is too big, but we made sure that was not the case for our experiments.

In the Bitcoin version, selfish mining (explained in section 3) is not implemented for the malicious node who finds the first block (not that simple because it needs to keep information about 2 different blocks).

However, this doesn't influence our conclusion that the Reinforcement version is better resistant to malicious behavior, as malicious miners in the Bitcoin version would be even more powerful when implemented.

6 Conclusion and Future Work

The aim of this project was to present an improved implementation of the Collaborative Block Reinforcement Protocol and show that it outperforms the classic Bitcoin Blockchain Protocol.

We started working on top of [2] and had to think of logical and coding interventions that will lead to a blockchain that is harder to fork by malicious miners than the Bitcoin Blockchain. This way, it is shown that with the Collaborative Block Reinforcement Protocol it takes less time for a transaction to be considered committed with high probability. Another benefit from this protocol is the minimization of the amount of work that is "wasted" by unsuccessful miners in systems which use the Bitcoin Blockchain Protocol.

In our experiments we were using different number of malicious and honest miners and 2 different malicious strategies for canceling blocks: Cancel Every Block and Cancel a Particular Block. Although we had been facing many challenges, at the end we managed to successfully demonstrate how the Reinforcement Protocol is better resistant to malicious behavior in both block canceling strategies when the system is dominated by honest nodes or honest and malicious miners have approximately equal computing power. In the case where the system is dominated by malicious miners, we observe almost the same performance with both blockchain protocols.

As a future task, we consider using a more realistic distributed environment, where every miner would run on a distinct machine and also building applications that will make use of the Collaborative Block Reinforcement Blockchain protocol.

Acknowledgements

I would like to thank Prof. Rachid Guerraoui and the LPD Laboratory for providing me with the opportunity to work on a project of my interest. Additionally, special thanks to my supervisor Matej Pavlovic for his guidance and help throughout the whole project.

References

[1]. PAVLOVIC M. Collaborative Block Reinforcement in Blockchain Systems

[2]. Shevchenko A. , Águas A. Semester Project Report

[3]. Zoo NAKAMOTO S. Bitcoin: A peer-to-peer electronic cash system

Appendix

- https://github.com/Iva1502/Collaborative-Blockchain-Reinforcement
- Content of src/Contants.py used for the experiments in part 4:

```
COMMIT_TH =
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8

REINF_TH =
0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF8

#should be set to 0 if the Bitcoin version is run
SWITCH_TH = 1

CANCEL_PARTICULAR_BLOCK_TH = 20

PORT = 5600

CANCEL_BLOCK_MIN_RANGE = 2

CANCEL_BLOCK_MAX_RANGE = 4

TRANSACTION_INTERVAL = 3

REINF_TIMEOUT = 1.5

COMMIT_TIMEOUT = 10

DELIVERY_DELAY = 0

CLEAN_PREVIOUS_BLOCKS = 100

PROPOSAL_TAG = "proposal"

COMMIT_TAG = "commit"

PROPOSAL_COMMIT_TAG = "proposal_commit"

REINFORCEMENT_TAG = "reinforcement"

REINFORCEMENT_INF_TAG = "reinforcement_information"

TRANSACTION_TAG = "transaction"

MALICIOUS_PROPOSAL_AGREEMENT_TAG =
"malicious_proposal_agreement"
```