HYPERLEDGER

EB 20/21	Enterprise Blockchain Technologies	Number:	3
Module I - Introduction		Issue Date:	-
Background: A Primer on Blockchain Technology		Due Date:	-

Instructors Guide

This document provides a proposal for a solution for Lab3. This laboratory is based on previous work elaborated by the contributors [1–3].

1 Hands on Blockchain

1.1 Theory & Business

Exercise 1: What is the advantage of blockchain being a decentralized, Peer-to-Peer system instead of a centralized one?

Being a decentralized, peer-to-peer system, the chain becomes more resilient to integrity attacks and malicious action. This is due to blocks and their contents being verified. If we had a centralized system, it could become a single point of failure and attack since changing that "main" chain could alter the whole system.

Exercise 2: Consider a private blockchain running three nodes. Given that it is managed from a single organization, how do we assure decentralization?

Although one organization controls the blockchain, decentralization can be achieved if there are stakeholders with different incentives, i.e., nodes representing various company departments. However, executive management may enforce decisions on all departments, which can compromise decentralization. To deal with this, it is advisable that at least one other party is impartial, desirably.

Exercise 3: What stops a malicious entity from being able to alter blocks in a permissionless blockchain?

For permissionless blockchains, for a block to be accepted by other blocks, the entity who proposes it must participate in the consensus. For example, in Proof-Of-Work, the participant needs to solve a hashing puzzle that includes the hash of the previous block. If an attacker were to alter a block, it would have to solve the Proof-Of-Work for that block and recalculate all Proof-Of-Works from subsequent blocks the hash of the altered block has changed. Although not impossible, the deeper you go in the chain, the more probabilistic unlikely it becomes, depending on the computational power the attacker has.

Exercise 4: What about permissioned blockchains? Can a malicious entity change in these as well?

Since permissioned blockchains require credentials for participation and are usually more private, the presence of malicious entities is much less of a problem. Nevertheless, if we were to consider a malicious entity existing in the system, it would need a majority of malicious entities to control the chain.

Exercise 5: How does a client insert a transaction in the system?

Firstly, a client inserts the transaction into a transaction pool, which each node has. This transaction pool is different for every node since each client inserts transactions at nodes of his preference, and these transactions may not be sent to every node in the system.

Exercise 6: How do we know if a transaction was created by a client and not by a malicious node?

A: The transaction must be signed with the sender's private key to prove the authenticity and non-repudiation of the transaction. Hence, a malicious node cannot create transactions of its own accord.

Exercise 7: Why can't permissionless blockchains use the classical Consensus algorithms and must instead rely on other forms of Consensus like Proof-Of-Work?

A: One of the main advantages of the permissionless blockchain is its openness to the internet, meaning anyone can enter and leave the system at will and participate as they so desire. Classical Consensus algorithms require that the number of participants N is known, which is unfeasible to obtain in an open Peer-to-Peer system like a Blockchain. Furthermore, since it is open to the internet, several byzantine clients are present in the chain, which would be impossible to predict. Hence, Byzantine Consensus algorithms would not work since we do not know either the number of participants (N) or the maximum number of possible faults/byzantine entities (F).

Exercise 8: Why can't a node fake a result from Proof-of-Work?

A: Proof-of-Work relies on hashing functions. These functions have the following properties:

- Even Distribution, meaning that there is an equal probability of obtaining any hash value, so we cannot predict which value we can obtain from a certain input;
- One-way, meaning that it is hard to find the input value from the function result;

- Collision resistance, meaning that it is hard to find a pair where inputs x and y, with x != y that hash (x) == hash (y).

Exercise 9: Why do we want to add difficulty to the Proof-of-Work?

A: By increasing the difficulty level, one can control how fast the blockchain grows and how quickly new blocks are added. This is key to avoid malicious users to overwrite and change older blocks and the chain as a whole. If we add a difficulty of 0 (meaning a normal hash), it would be trivial for an attacker to simply remove a transaction from a block and recalculate all the hashes. By adding a difficulty equal to, for example, calculating a hash for a block every 10 minutes, it disincentivizes attackers to attack the chain. Since they have to recalculate all the block hashes after the altered one and at that time, new blocks would appear. This means that the chances of its new chain getting longer than the others are smaller.

Exercise 10: How does the difficulty value work in Proof-of-Work?

This question might mean different things: what is the purpose of the difficulty in proof-of-work, or how is it updated. The goal of the difficulty value is to define how long it should take for a block to be created. The higher the difficulty, the longer it takes to create a block. In Bitcoin, proof of work is configured to yield a block every 10 minutes. Hence, the difficulty is variable, depending on the number of miners on the network.

Exercise 11: How does a node share its new block?

When a block is created, it can be shared with the other peers in many different ways. A node can either broadcast its chain once it finishes a new block to inform the other nodes. A node can periodically check the other nodes' chains to know if new blocks/-longer chains exist.

Exercise 12: What are the advantages and disadvantages of sharing the chain by broadcasting the block or by verifying other chains from peer nodes?

Regarding the broadcasting method:

- advantage
  - The node only needs to verify the block itself, since the chain behind it is already verified locally;
- Disadvantage
  - A node may not have the previous block from the one obtained, due to delays in the sharing of blocks;

  - All correctly verified block must be kept for the lifetime of the chain (or at least to a certain predicted depth) since we never know when a chain will become the new longest one;

Regarding the verification of chains method:

- advantage
  - Avoid problems of broadcasting, only relies on checking each neighboring known peer;
- Disadvantage
  - The node needs to verify each chain's entirety, which includes verifying each signature of each block. Given that these signatures are with asymmetrical encryption, it will take a long time to verify the whole chain, especially the longest it becomes.

Exercise 13: What happens when multiple nodes mine a new block at the same time?

When two nodes create a new block, a temporary fork happens in the system. Both blocks and blockchains are legitimate, so nodes can choose to mine in either one of them. However the chain where a new block is mined first will be chosen by the miners for future work. Hence the fork will resolve by itself with the mining of new blocks.

Exercise 14: Why do the miners choose the longest chain to work and not a forked, smaller chain?

There are multiple reasons for this. From the blockchain perspective, the longer the chain is, the harder it is to alter past blocks, upholding the chain's immutability. From the miner perspective, its goal is to obtain the reward for mining a block. Suppose a miner is working on a block, and a longer chain is discovered. In that case, the chances of his own chain being accepted by the other nodes get smaller, and the chances of a new block appearing on the longest chain get higher. Working in the longest chain gives the highest chance of a block being accepted and upheld in the system, possibly giving the miner the reward.

Exercise 15: What happens to the other block and transactions that were created in the forked chain that was discontinued?

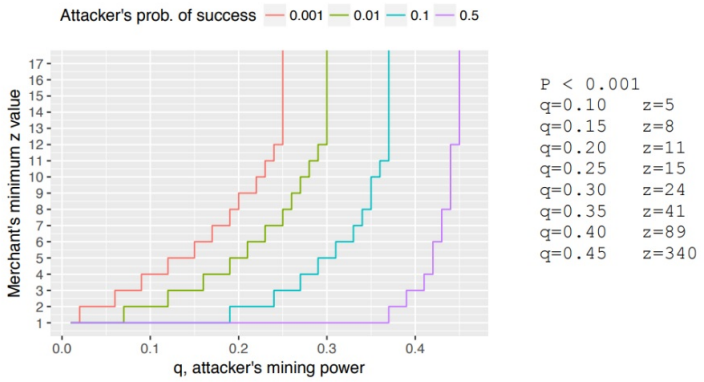
Since the nodes and the system have started working on the longest chain and the chain discontinued, the block is considered invalid. The transactions are not accepted by the system.

Exercise 16: What if the service was already made and the payment (in the form of a transaction) disappears?

This is one of the challenges the blockchain must face. Since there is a chance the block may disappear from the chain due to forked, longer chain appearing, a transaction must only be considered "final" once the block reaches a certain depth. This depth must be chosen depending on the predicted computational power from the system's biggest entities and the probability we want to uphold.

Exercise 17: Being a Peer-to-Peer service, how does a node discover which nodes to connect to join the system?

Depending on the Blockchain, there are many ways these can be done. It can be by either joining a known address (which is not very good for a P2P system), making a DNS request to known names, or accessing a known database where nodes insert their addresses.



Sources: Bitcoin: A Peer-to-Peer Electronic Cash System , Satoshi Nakamoto;
An Explanation of Nakamoto's Analysis of Double-spend Attacks, Ozisik and Levine

Figure 1: Relation between computation power, probability of overtaking the blockchain and block depth

1.2 Blockchain4Students

Consider the source code of Blockchain4Students. Start by exploring "blockchain-data-structure" and "'consensus", inside the blockchain folder.

Exercise 1: Consider the next code fragment. What type is the Blockchain4Students blockchain?

Public, permissionless blockchain, because everyone with the code (publicly available) and an internet connection can interact with it. However, the node discovery service can

be considered centralized.

```
app.route('/register/node', methods=['POST'])
f register_peer_node():
de_address = request.get_json()["node_address"]

sponse = {
essage': 'Node added',
otal_nodes': list(blockchain.peer_nodes),

ockchain.register_node(node_address)
turn json.dumps(response), 200
```

Exercise 2: We give rewards right after a block is mined. What is the problem with this approach?

As the blockchain has probabilistic finality, it should wait for a certain number of blocks to be confirmed (e.g., In Bitcoin is 6, which corresponds to about an hour).

Exercise 3: Each Transaction includes the ID from the node that created it. Why is this information necessary?

Transactions must be digitally signed by the nodes to guarantee its authenticity and integrity. Without the node ID, the nodes would not know which node signed what and what public key to use to verify the authenticity of the transaction.

Exercise 4: When a transaction request arrives at the node, it signs it with its own private key. What is the possible attack with this approach?

A malicious node could create fake transactions to benefit himself, like moving money from one wallet to another without the users' consent or even to the miner itself. To solve these problems, when users request a new transaction, they should send a signature of their request to guarantee the transaction's authenticity.

Exercise 5: When a node first connects, it contacts a discovery node to find other peers in the blockchain. Although this method is possible, why is it not realistic to use on broader scale blockchains?

Blockchains are, by definition, a decentralized entity. By making a single node responsible for node discovery, we are not only losing this decentralization as we are adding a single point of failure to the blockchain. If this node fails, no other nodes would be

able to connect. The more realistic way to deal with this problem is to add some form of name discovery, either by a DNS resolution (where failures could masquerade and could point to multiple nodes) or other more advanced systems of name discovery like UDDI [1]

Exercise 6: When the node obtains a valid chain bigger than its own, it discards its own chain for the bigger one. Why?

The miner (and the node) is to be the first to mine a block to obtain the reward from the system. If a miner is working on a block and it receives a chain with an already blocked mine further ahead, the chances of him finishing its current block and being accepted by the chain are very low. As such, a miner is better off giving up on their current work and starting a new block from the end of that chain.

Exercise 7: With our current implementation, when a node joins, he is not aware of any state of its peers. Add the functionality of when a node joins the system to ask and update its own blockchain.

After node discovery, when a node joins, ask each node for the /getChain command, parse and verify the chain and if any is bigger substitute.

References

[1] R. Belchior, A. Vasconcelos, and M. Correia, “Towards Secure, Decentralized, and Automatic Audits with Blockchain,” in *European Conference on Information Systems*, 2020.

[2] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, “A Survey on Blockchain Interoperability: Past, Present, and Future Trends,” *arXiv*, vol. 1, no. 1, p. 58, may 2020. [Online]. Available: http://arxiv.org/abs/2005.14282

[3] R. Belchior, “JusticeChain: Using Blockchain To Protect Justice Data,” Ph.D. dissertation, Instituto Superior Técnico, dec 2019. [Online]. Available: https://fenix.tecnico.ulisboa.pt/cursos/meic-a/dissertacao/846778572212223

[1] https://en.wikipedia.org/wiki/Web_Services_Discovery