Blockchain 4

This set of slides was developed based on the reference books as listed in the course document.

7 Major Tasks

There are seven major tasks that need to be addressed when designing and developing a software system that manages ownership by using a purely distributed peer-to-peer system of ledgers in an open and untrustworthy environment:

1. Describing ownership
2. Protecting ownership
3. Storing transaction data
4. Preparing ledgers to be distributed in an untrustworthy environment
5. Distributing the ledgers
6. Adding new transaction to the ledgers
7. Deciding which ledgers represents the truth

Task 5: Forming a System of Distributing the Ledgers

Distributing the Data Store Among Peers

The above turned the blockchain-data-structure into an *immutable append-only data store*, which can be used as a *manipulation-resistant ledger* for transaction data.

Having one immutable append-only history of transaction data in insolation may be of limited value for the goal of clarifying ownership based on a group of computers that acts as witnesses of ownership-related events. Hence, we need establishing a purely distributed peer-to-peer system that allows sharing of information about transactions.

We will not go further on how to achieve that.

7 Major Tasks

There are seven major tasks that need to be addressed when designing and developing a software system that manages ownership by using a purely distributed peer-to-peer system of ledgers in an open and untrustworthy environment:

1. Describing ownership
2. Protecting ownership
3. Storing transaction data
4. Preparing ledgers to be distributed in an untrustworthy environment
5. Distributing the ledgers
6. Adding new transaction to the ledgers
7. Deciding which ledgers represents the truth

Task 6: Adding and Verifying New Transactions to the Ledgers

We will discuss this task when we discuss Bitcoin.

Task 7: Deciding Which Ledgers Represent the Truth

7 Major Tasks

There are seven major tasks that need to be addressed when designing and developing a software system that manages ownership by using a purely distributed peer-to-peer system of ledgers in an open and untrustworthy environment:

1. Describing ownership
2. Protecting ownership
3. Storing transaction data
4. Preparing ledgers to be distributed in an untrustworthy environment
5. Distributing the ledgers
6. Adding new transaction to the ledgers
7. Deciding which ledgers represents the truth

Choosing a Transaction History

The Challenge of Choosing a Transaction History

The arrival of new blocks at the in boxes of the individual nodes is highly influenced by the message delivery capabilities of the network. Messages:

- may get lost,
- may be delivered with time delay, or
- may arrive in any order.

As a result, the nodes of the network do not have the identical information at their disposal at the same time.

The challenge: how to find a way to identify one unambiguous history of transaction data in the face of all message delivery adversities and **without falling back to a centralized solution?**

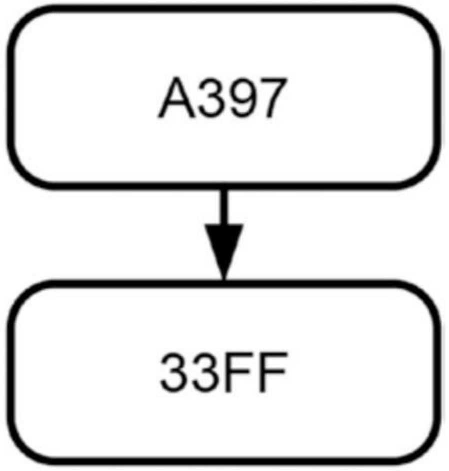
How?

The idea of selecting a transaction history based on the computational effort that was spent for creating it has led to the following two criteria:

- **The longest-chain-criterion**
- The heaviest-chain-criterion

The longest-chain-criterion

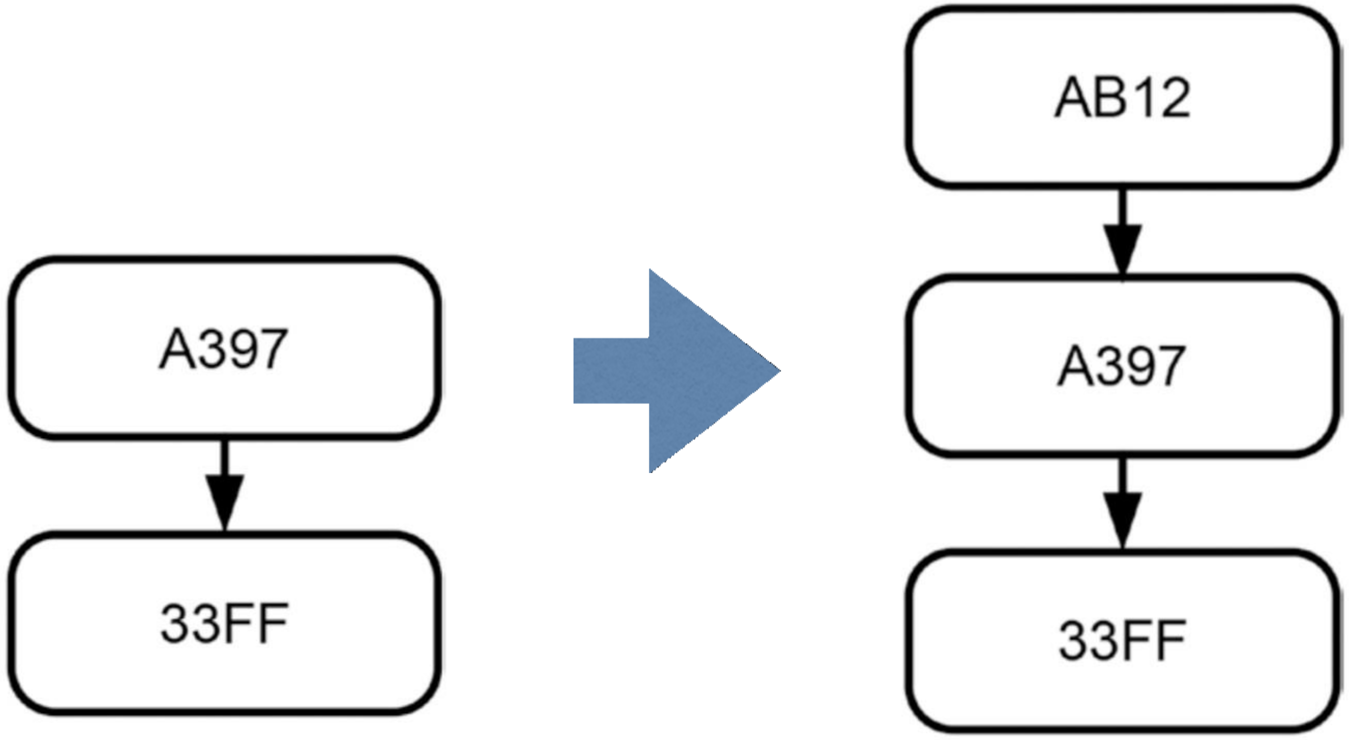
The longest-chain-criterion is based on the idea that the blockchain-data-structure that comprises the most blocks represents the most aggregated computational effort. Consider this initial situation:



where all the nodes of a distributed system maintain and agree on the identical version of the blockchain-data-structure.

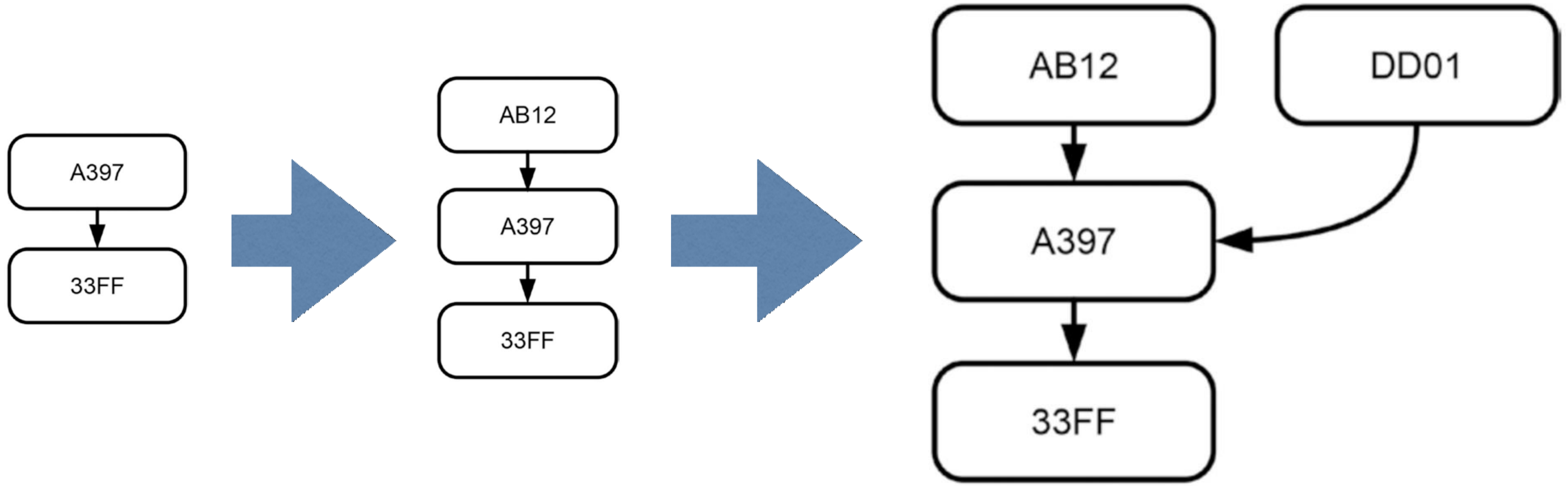
The longest-chain-criterion

Later, one node solved the hash puzzle of a new block and sent it to its peers. The blockchain-data-structure that the *majority* of nodes maintain is:



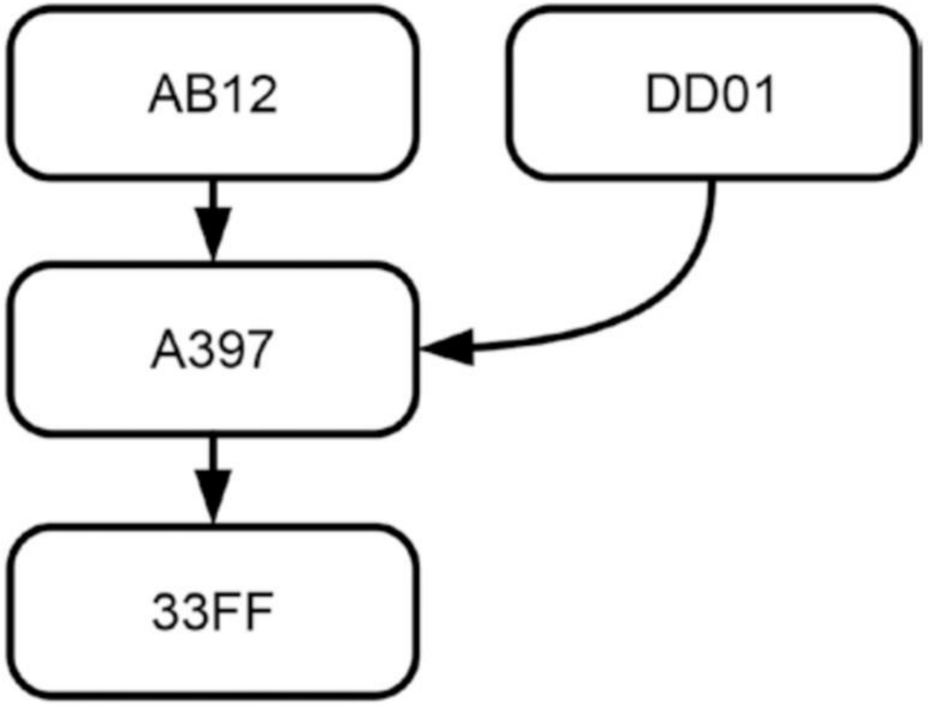
The longest-chain-criterion

Due to a delay in the message passing, a *minority* of nodes have not received block AB12 yet. Hence, they still try to extend the initial chain. Eventually, one of them successfully solves the hash puzzle for a new block with the hash value DD01 and passes it to its peers:



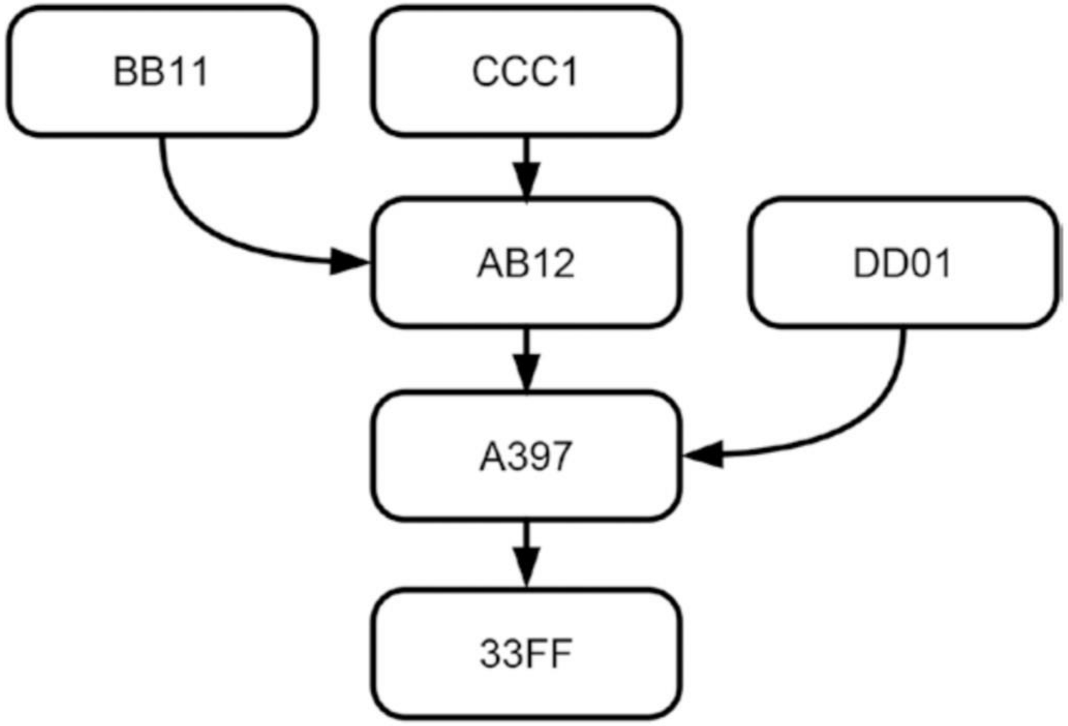
The longest-chain-criterion

In this situation, the nodes are free to decide which branch to extend. Some nodes may strive to find a new block that refers to block AB12 as its predecessor, while other nodes strive for finding a new block that refers to block DD01 as its predecessor.



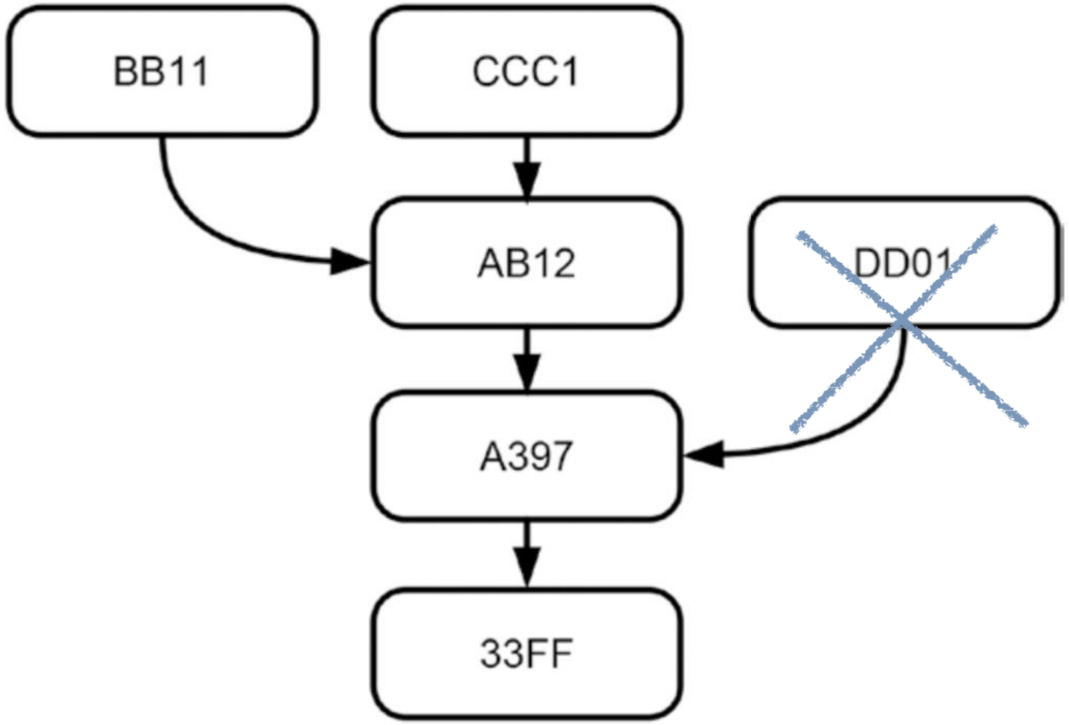
The longest-chain-criterion

Suddenly, the majority of nodes receive **two new blocks, BB11 and CCC1**, which both refer to block AB12 as its predecessor. This can happen due to two nodes finishing the proof of work for their blocks nearly at the same time. The result of incorporating these two new blocks into the blockchain-data-structure is a data structure that contains **three chains**.



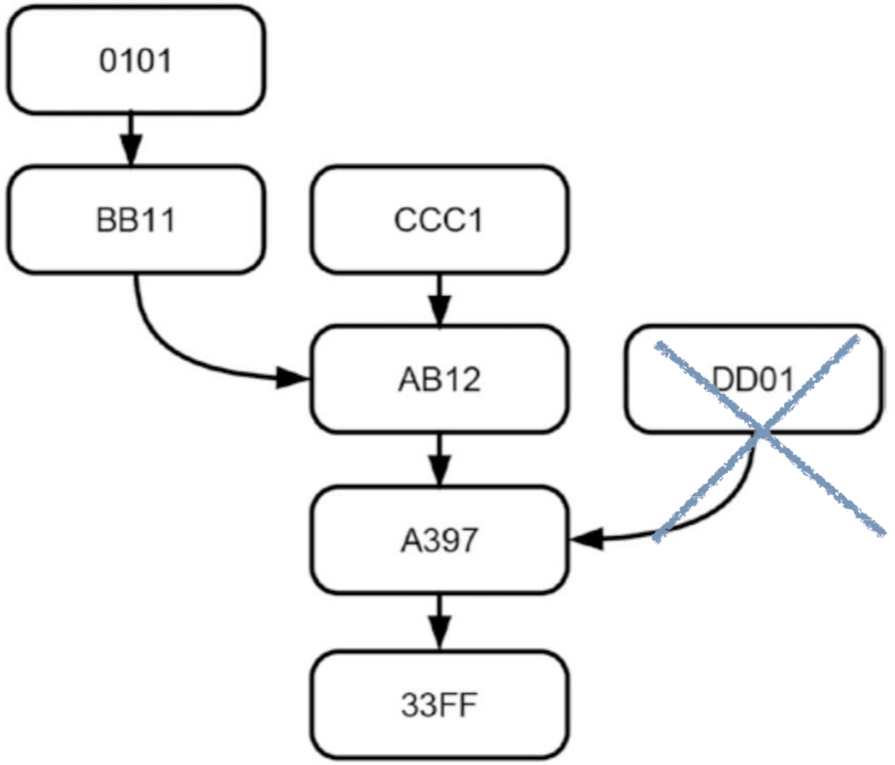
The longest-chain-criterion

The **longest-chain-criterion** clearly **rules out the shortest chain**, which is the chain **DD01 —>A397–>33FF**. However, the **longest-chain-criterion does not yield an unambiguous result** because there are **two chains of the same length.** As a result, some nodes may strive to find a new block that refers to block BB11 as its predecessor, while other nodes may strive to find a new block that refers to block CCC1 as its predecessor.



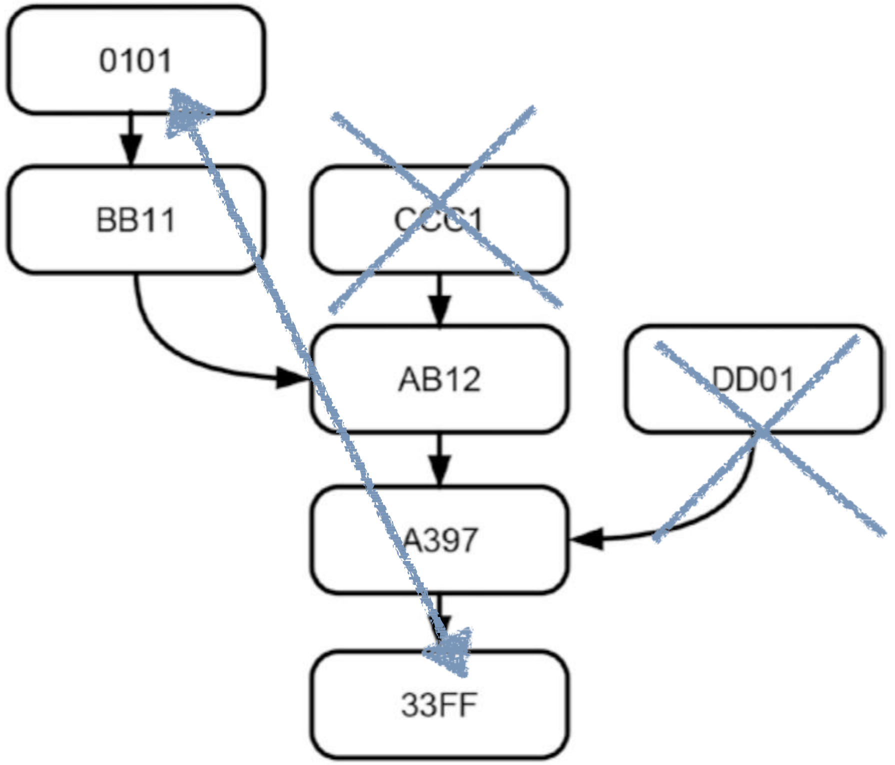
The longest-chain-criterion

Eventually, one new block arrives that refers to block BB11 as its predecessor, which yields the data structure as below:



The longest-chain-criterion

The blockchain-data-structure now contains many conflicting versions of the transaction history, but the **longest-chain-criterion** yields one unambiguous result that is the chain consisting of the blocks **0101—>BB11—>AB12—>A397–>33FF**. The majority of nodes and eventually all nodes of the system will use that chain for clarifying ownership-related requests.



The heaviest-chain-criterion

The idea of selecting a transaction history based on the computational effort that was spent for creating it has led to the following two criteria:

- The longest-chain-criterion
- **The heaviest-chain-criterion**

The heaviest-chain-criterion

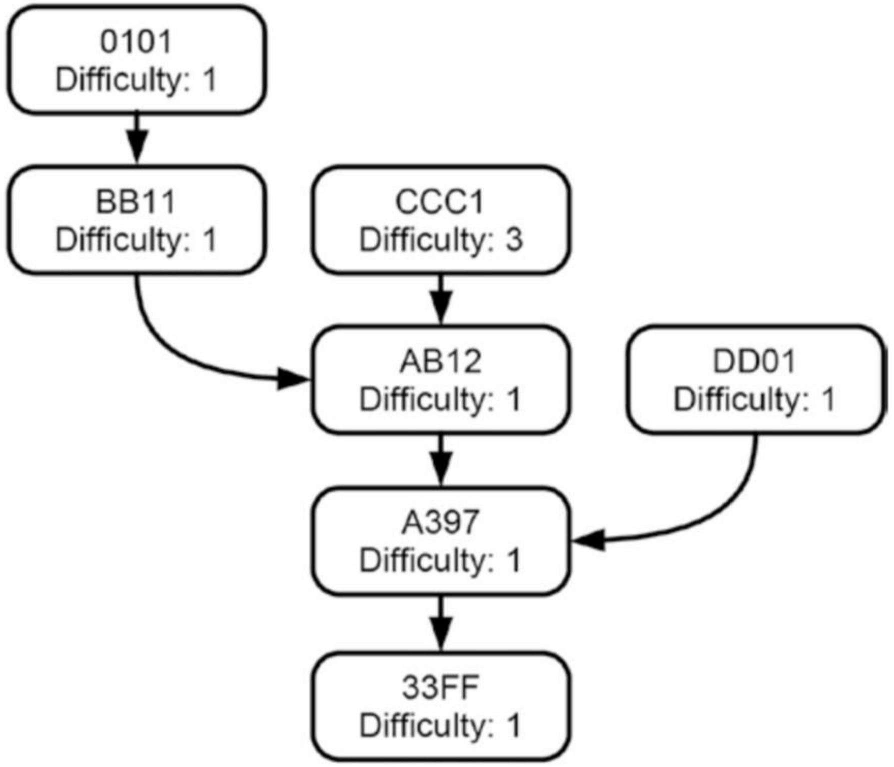
Blockchain applications **rarely utilize a constant difficulty level** for the hash puzzle to be solved for adding a new block to the blockchain-data-structure.

- (Instead, they typically determine the difficulty level dynamically, which causes the blocks to differ with respect to the computational effort that was spent for adding them to the blockchain-data- structure.)

Therefore, the **longest-chain-criterion** is NOT the one that represents the **most computational effort**.

The heaviest-chain-criterion

Therefore, the **longest-chain-criterion** is NOT the one that represents the **most computational effort:**



The heaviest-chain-criterion

As a result, blockchains that determine the difficulty level dynamically:

- They utilize **the heaviest-chain- criterion**.

In the case that the difficulty level is identical for all blocks

- The longest path is identical with the heaviest path and both **the longest-chain-criterion** and **the heaviest-chain-criterion yield the identical result**.

Consequences of Selecting One Chain

Consequences of Selecting One Chain:

1. Orphan blocks: those blocks in the tree-shaped data structure **that are not part of the authoritative path** are abandoned by the nodes.

2. Reclaimed reward: Orphan blocks are useless for the purpose of clarifying ownership, as they do not contribute to the authoritative chain. As a result, **the reward given to the node that created and submitted them is reclaimed**.

3. Clarifying ownership: Only those transactions that are part of the authoritative chain are considered to have happened and are used to clarify ownership-related requests.

4. Reprocessing of transactions: Transaction data that unluckily ended up in orphan blocks needed to be reprocessed and later added to the blockchain-data-structure.

Consequences of Selecting One Chain

Consequences of Selecting One Chain:

5. Eventual consistency: The deeper down the authoritative chain a block is located:
   - The less it is affected by random changes of the blocks that belong to the longest chain
   - The less likely it will be abandoned
   - The more accepted it is by the nodes of the system
   - The more anchored it is in the common history of the nodes

The fact that certainty concerning the inclusion of blocks in the authoritative chain increases as time goes by and more blocks are added eventually is called **eventually consistency**.

Consequences of Selecting One Chain

Consequences of Selecting One Chain:

6. Robustness against manipulations

- As long as **honest nodes** own the **majority of computational resources** of the whole system, the path maintained by them will grow fastest and outpace any competing paths.

- In order to manipulate an inner block, an attacker would have to **redo the proof of work** of that **block** and subsequently **redo the hash puzzle of all blocks after it** and then catch up with and overtake the path maintained by the honest nodes.

- However, **establishing a new path** by catching up with and overtaking the path maintained by the majority **is impossible for any attacker that controls less computational power than the majority**.