Cryptocurrencies and Blockchain

Sheet 2, submission by Konstantin Bork

Assignment 1: Bitcoin Various Topics

1. All the steps involving reference checks in both the transaction pool and the main branch are computational expensive. This includes steps 8, 9, 10 and 12 which increase in computation cost with the increase of the number of transactions and blocks. The cost for each of these steps is in O(n * x) with n being the number of blocks and transactions, and x being the cost to find information in the block or transaction, which can be in O(n) if information is stored in a list.
   A good data structure to speed up verification is the binary hash tree in which the hashes of the transactions as its leaves. This structure allows the user to not have to load all transactions of a block for the verification process and, due to the tree characteristic, the search has a cost in O(log n).

2.

Extra question: more forking

1. The fork lasted for 24 blocks when one branch finally pulled ahead. Therefore, the 24 blocks of the other branch have been abandoned. Source: https://bitcoinmagazine.com/articles/bitcoin-network-shaken-by-blockchain-fork-1363144448/

2. 94 blocks have been orphaned because of this soft-fork as they include an unconfirmed transaction which might include an incorrect pay-to-script-hash. Source: https://bitcoin.stackexchange.com/questions/9678/what-is-script-hash-address-exactly-and-how-does-it-work

Assignment 2: Transaction fees

1. As Alice would have to pay fees, we can assume that any of the conditions for not paying a fee is not fulfilled. Furthermore, we know Alice wants to create one coin only, meaning the number of outputs is 1. Because of the latter knowledge, we can reduce the transaction size to following:
   $transSize = 148N_{inputs} + 44$
   Now, we calculate the fees with our current knowledge:
   $fees = (148N_{inputs} + 44)Byte * \frac{0.0001BTC}{1000Byte} = 0.0000148N_{inputs}BTC + 0.0000044BTC$
   We know the fees equal the sum of all coins with value v:
   $N_{inputs} * v = 0.0000148N_{inputs}BTC + 0.0000044BTC$
   We now solve the equation to v:
   $v = 0.0000148BTC + \frac{0.0000044BTC}{N_{inputs}}$

The higher N_{inputs} becomes the less important the second summand becomes. Therefore, we can assume v is 0.0000148 BTC.

2. To avoid any transaction fee, all three conditions must be fulfilled. Let's assume the transaction output is above 0.01 BTC. To meet this condition, Alice must have at least 676 coins (0.01/0.0000148). With this amount of coins, the first condition can never be fulfilled as the transaction size exceeds the limit of 1000 Bytes for a free transaction. This also means we do not have to check the priority of the transaction at all.

3. The current structure encourages users and services to prioritize transactions which have "old" bitcoins or which include a fee for the transactions as a reward. This strategy prevents spam transactions in the network which would hinder more useful transactions from being processed. Thus, the chance of the network staying healthy increases compared to a structure which does not factor the input age.

Assignment 3: Multi-signature wallet

1. It is one solution to create a new multi-signature wallet with both non-compromised keys and a new key. Then, all Bitcoins are transfered from the old wallet to the new one. This transaction is signed by both old non-compromised keys. When the transaction succeeds, the old wallet can be abandoned and is practically unusable for Mallory. It is even more secure to create new private keys when creating the new wallet and revoke all old private keys. This step minimises the risk that Mallory signs transactions from the old wallet in case of gaining the other keys, too.

2. At first, they should backup the seeds of both keys and store them in a secure place. The compromised and potentially deleted key can then be restored from this seed. After restoring the keys, they should create a new wallet with new keys and transfer all Bitcoins to the new wallet.

Assignment 4: Node in a blockchain

The source code can be found in the src folder.
For the given tests in Test.java, the blockchain works as intended. There are several try-catch-blocks missing to catch exceptions when transactions or blocks coming from the user are malformed. Furthermore, the UTXOPools are not updated, so double-spending might be possible. Last but not least, the code is not tested with concurrent requests, therefore there might be some flaws here as well.