# Public Ledger for Auctions

Daniela Tomás
up202004946@edu.fc.up.pt

Diogo Almeida
up202006059@edu.fc.up.pt

Diogo Nunes
up202007895@edu.fc.up.pt

***Abstract*—This report explores the architectural choices and implementation details of a public blockchain for auction transactions, focusing on security and trust. The system aims to provide a decentralized environment by using a secure ledger that supports Proof-of-Work (PoW) and Delegated Proof-of-Stake (DPoS), a robust peer-to-peer (P2P) layer that relies on S/Kademlia for efficient data transmission and trust mechanisms, and an auction system that is integrated with the blockchain.**

***Index Terms*—Kademlia, P2P, PoW, DPoS, Blockchain**

## I. Introduction

Blockchain technology has emerged as an innovation in digital transactions and decentralized systems. A public ledger is a very good example of what a blockchain can be used for, as it records transactions between entities so that everyone can see and agree, and the blockchain can provide the records in a transparent, secure, non-centralized, and non-repudiable manner.

To achieve this implementation over a network, a protocol is required, and Kademlia was chosen for this task because it provides a way to assemble a peer-to-peer (P2P) decentralized network while also providing methods to find and store key-value pairs that can be used to store blockchain elements and ledger transactions.

This report offers a full overview of the development process, obstacles faced, and solutions implemented. The subsequent sections delve into specific components of the system, detailing the distributed ledger mechanisms, P2P networking, and the auction system.

## II. System Architecture

The lowest level of our solution begins with a somewhat modified implementation of Kademlia, which will be addressed in greater detail later in this report, built in Java using the Netty framework for handling communication between nodes.

A layer above that, interfacing with the Kademlia layer for communication needs through the necessary methods without knowing how it works internally, is the blockchain, which serves to store all transactions that have occurred or are awaiting verification or storage before being recorded on the next to-be-mined block.

Finally, the Public Ledger layer interfaces with the blockchain layer for record keeping and visualization, as well as the Kademlia layer for communication throughout the auctioning process.

We were unable to construct the Pub/Sub system using Google Cloud Platform. Therefore, we chose to do it internally, utilizing the Public Ledger layer and Kademlia's node notification implementation techniques.

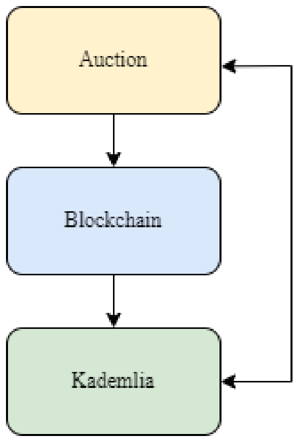

Fig. 1. System architecture and interactions.

## III. Distributed Ledger

### A. *Proof-of-Work (PoW)*

Proof-of-Work is a way of validating transactions in a secure manner. Different nodes in a network compete to be the first to ”mine” a block, normally for a reward, to incentivize good behavior. ”Mining” consists of computing a complex mathematical problem that is hard to solve but easy to verify. Once the problem is solved, any pending transactions are added to the newly ”mined” block, and the block itself is appended to the blockchain. This makes the blockchain very secure, as it's essentially impossible to tamper with, and it also offers decentralization as there is no single authority over the blocks, anyone can participate. This also has the drawback of being environmentally unfriendly, as the process of ”mining” requires a lot of power and can also be hard to scale, as with too many ”miners” the puzzles will quickly rise in difficulty, making new block creation take longer.

### B. Delegated Proof-of-Stake (DPoS)

Delegated Proof-of-Stake is another mechanism for participants to reach agreement on transaction validity. It presents itself as a more democratic and environmentally friendly alternative to PoW. Participants stake tokens, locking them up in the network to show others you have a stake in the success of a validation. DPoS has a voting system where token holders vote for delegates who are responsible for validating transactions and adding new blocks to the blockchain. These delegates are incentivized to behave correctly, as their stake is on the line and could be forfeited if they misbehave. So DPoS is faster and less wasteful of energy, which in turn makes it much more scalable than PoW. It is also more democratic, as computing power isn't nearly as important as in PoW. But in turn, it introduces the potential to be less decentralized, which can in turn lead to a concentration of power.

## IV. Peer-to-peer (P2P)

### A. Kademlia

Kademlia is a peer-to-peer distributed hash table (DHT) that efficiently locates nodes (peers) responsible for storing data [1].

*1) Node:* A node consists of:

- <ID, IP, Port>: Nodes are identified by 160-bit UIDs based on SHA-1. Kademlia uses the XOR metric to calculate the distance between two node IDs.
- *Routing table (k-bucket)*: Contains information about other nodes in the network.
- *Storage*: Store key,value pairs relevant to blockchain transactions.

The main functionalities of the Kademlia protocol include the following:

*2) Join Network:* For a new node to join the network, it must connect with a bootstrap node to obtain information about other nodes. The node iteratively finds and updates its routing table with nodes closer to its own ID, ensuring a well-connected network.

*3) PING RPC:* Check if a node is available.

*4) FIND_NODE RPC:* Given a target node ID, it finds and returns the closest nodes to this target ID.

*5) FIND_VALUE RPC:* If a node hasn't received a store for the requested key, it responds just like it would to a *FIND_NODE* RPC. If the node has received a store for the key, it responds with the value.

*6) STORE RPC:* key,value pairs are stored on the node that has the ID closest to that key.

*7) NOTIFY and LATEST_BLOCK RPCs:* To ensure that all nodes in the network are synchronized with the most recent block added to the blockchain, we created two new RPCs to broadcast (*NOTIFY*) and request (*LATEST_BLOCK*) the latest block hash. *LATEST_BLOCK* RPC is useful during the join network process to synchronize a new node with the latest block hash in the network. *NOTIFY* RPC informs nodes about a new block hash and determines whether the new block hash differs from the current *latestBlockHash*, which is an attribute in the Kademlia class that stores the hash of the latest block in the blockchain. If so, it updates *latestBlockHash* and notifies all nodes in the routing table, who then repeat the process.

To protect against address forgery, all RPC requests to another node will contain a random 160-bit RPC ID, and the other node must send the response containing the same RPC ID. The node that sent the initial RPC request checks if it matches.

To implement these functionalities, we used Netty for UDP communication.

### B. Resistance to Sybil and Eclipse attacks

S/Kademlia [2] extends Kademlia to resist to attacks.

Eclipse attacks can be prevented, at least partially, by limiting nodes' power over their given IDs. To aid with this, we generate node IDs by concatenating certain node information with a random number generator and applying a hash function to the result. This method, while not flawless, makes the assault more difficult to perform.

Furthermore, it has been demonstrated [3] that Sybil attacks cannot be entirely negated, just delayed by utilizing signatures, which in turn will assist with eclipse assaults. However, we will not go into specifics, as we did not implement these procedures.

## V. Auction System

### A. Transactions

Each transaction contains relevant data such as the sender and receiver's public keys, the value being transferred from the sender to the receiver, and a cryptographic signature generated by the sender's private key to authenticate the transaction and serve as a proof of authenticity and integrity. Before being included in a block, each signature is verified with the sender's public key.

Buyers submit their bids by creating a transaction. Valid bids are recorded on the blockchain, creating an immutable record of all bids placed during the auction. New bids are propagated through the network, ensuring subscribers and the node that has the auction stored are aware of the latest bid and bidder. When the auction duration expires, the system automatically selects the highest bid amount as the winner.

### *B. Publisher/Subscriber System*

To support the auction system, particularly the real-time aspects of bidding, a Publisher/Subscriber system facilitates communication among the entities involved, including auctioneers, bidders, and subscribers.

The system allows users to create auctions, place bids, and be notified about auction updates if they are subscribers to a specific auction. We aimed to integrate a Pub/Sub with Google Cloud Pub/Sub [4] in the auctions system, but unfortunately it was not possible. Our implementation leverages a custom peer-to-peer communication mechanism using Netty for auction-related messages, but it has a lot of limitations. Due to the direct connections between clients, it lacks the scalability and fault tolerance of a more robust Pub/Sub system like Google Cloud Pub/Sub.

We introduced two custom RPCs on Kademlia tailored to the auction system's needs:

*1) NEW_AUCTION RPC:* Broadcast the auction ID of a new auction to all nodes in the network.

*2) AUCTION_UPDATE RPC:* Notify nodes about updates to an auction. It propagates updates about bids and other changes in the auction to all relevant nodes, particularly subscribers and the node storing the auction.

## VI. Conclusion

Overall, this report describes a public blockchain system for auction transactions that combines security, decentralization, and trust. The system uses the Proof-of-Work mechanism for secure transactions and S/Kademlia for efficient data transmission. Despite challenges with the Pub/Sub system, we developed an internal communication method using Netty that provides real-time updates.

Future work could include improving communication layer scalability, implementing a Delegated Proof-of-Stake mechanism, creating a Pub/Sub system with the Google Cloud platform, and implementing security measures against Sybil and Eclipse attacks.

## References

[1] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, pages 53–65, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[2] Ingmar Baumgart and Sebastian Mies. S/kademlia: A practicable approach towards secure key-based routing. In *2007 International Conference on Parallel and Distributed Systems*, pages 1–8, 2007.

[3] John R. Douceur. The sybil attack. In Peter Druschel, Frans Kaashoek, and Antony Rowstron, editors, *Peer-to-Peer Systems*, pages 251–260, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[4] Google Cloud. Quickstart: Publish and receive messages in pub/sub by using a client library. https://cloud.google.com/pubsub/docs/publish-receive-messages-client-libraryjava.