Introduction

This task will describe how a transaction broadcaster service API can be designed with the following constraints:

1. The transaction broadcaster service should sign the data before broadcasting it.
2. The transaction broadcaster service should retry broadcasting a transaction if it fails.
3. The transaction broadcaster service should track all broadcasted transactions' status.
4. The transaction broadcaster service should provide a web UI to view the status of all broadcasted transactions.

Additional Requirement:

5. An admin is able to, at any point in time, retry a failed broadcast.

The transaction broadcaster service can be broken down into three parts for ease of explanation:

1. Transaction Validation:
    a. The transaction validator would validate the transactions before they are broadcasted. The transaction validator would check the following:
        i. The transaction is well-formed.
        ii. The sender has sufficient funds to pay for and process (gas fees) the transaction.
        iii. The transaction is not a duplicate.
2. Transaction Signer:
    a. If a transaction is validated, it will move on to the signing step, where it is signed, and a record of the unsigned transaction is made on the database.
    b. Another entry is made on a table with the signed transaction and the transaction status set to pending.
    c. A background transaction monitoring service queues all pending transactions, reads each one and makes an RPC call to broadcast the transaction.
3. RPC:
    a. The RPC will be designed to have a callback function to notify of the transaction status once the HTTP Post request is made to broadcast a transaction marked as pending.

    b. The callback function can notify the transaction monitoring service if a posted transaction has failed or been completed.
    c. The background monitoring service can then notify the client about the completion or failure of the transaction.
        i. In the case of failure, the transaction is again queued to be broadcasted. If this repeats a set threshold number of times, it is marked as failed and added to a table that the admin can monitor.
        ii. The client is notified of the transaction failure and the associated error.
        iii. If the transaction succeeds, the RPC callback returns a transaction hash which is then added to the database, with the status marked as successful.

Additional Requirement:

An admin is able to, at any point in time, retry a failed broadcast. To implement this, we need some additional steps to authenticate an admin, post a failed transaction, and a web UI to view all posted transactions with their status.

1. Authentication Service:
    a. The authentication service will make sure a user logged in is an admin or not.
2. Web UI:
    a. If the user logged in is an admin, they can view the list of posted transactions along with their statuses.
    b. The web UI will display the following:
        i. The transaction ID.
        ii. The sender and recipient addresses.
        iii. The number of tokens transferred.
        iv. The transaction status.
        v. The block number in which the transaction was included.

The admin can repost a failed transaction at any given time. This transaction is processed as follows.

1. The unsigned transaction from the database is fetched, which is again validated and signed. The admin is notified of the error if any of these steps fail.
2. The transaction is marked as pending and tracked by a background monitoring service, which queues it to be posted using the RPC.
3. The RPC's call back notified the admin of the status of the transaction:

a. If the transaction fails, it is retried a set number of threshold times again before being posted as a failed transaction.
b. If the transaction is successful, the client and admin are notified about its successful posting, and the database is updated to mark the transaction as successful.