Problem 6: Transaction Broadcaster Service

Overview:

1) API endpoint(broadcast_transaction): Used to receive transaction information.
2) Transaction Handler: Processes incoming post requests, signs them, returns a signed transaction, and broadcasts the signed transaction to an EVM-compatible blockchain system
3) Transaction Tracker: Tracks the status of existing transactions and provides an intractable interface for all transactions and their status.
4) Admin Interface: allows administrators to manually retry a failed broadcast
5) Restart Handler: deals with the transactions that are interrupted by the broadcaster system restarting unexpectedly.

API Endpoint

- Used to receive transaction information.
- Returns HTTP 200: Successfully transaction
- Return HTTP400-500: Encountered an error or failed transaction.

Transaction handler

- Validate the post request made to /broadcast_transaction and update the Transaction Handler
- Extract the transaction data
- Sign the transaction using a private key that is unique to the broadcaster's service
- Returns a signed transaction
- Broadcast the signed transaction to an EVM-compatible blockchain network via an RPC
- Update the Transaction Handler and request to the blockchain node.
- Retrieve response from the blockchain node
- If successful, mark the transaction as successful and update the Transaction tracker
- If unsuccessful, mark the transaction as failed and notify the Transaction Tracker
    - Run the Restart Handler

- To ensure that the handler doesn't bottleneck due to the large volume of transactions, we can distribute the load across multiple instances of the Transaction Handler.
- Using load balancing, evenly distribute the requests among the different instances of the Transaction Handler.

Transaction Tracker

- Keeps track of the different transactions.
- Provides an interface for users to keep track of the status of their transactions.

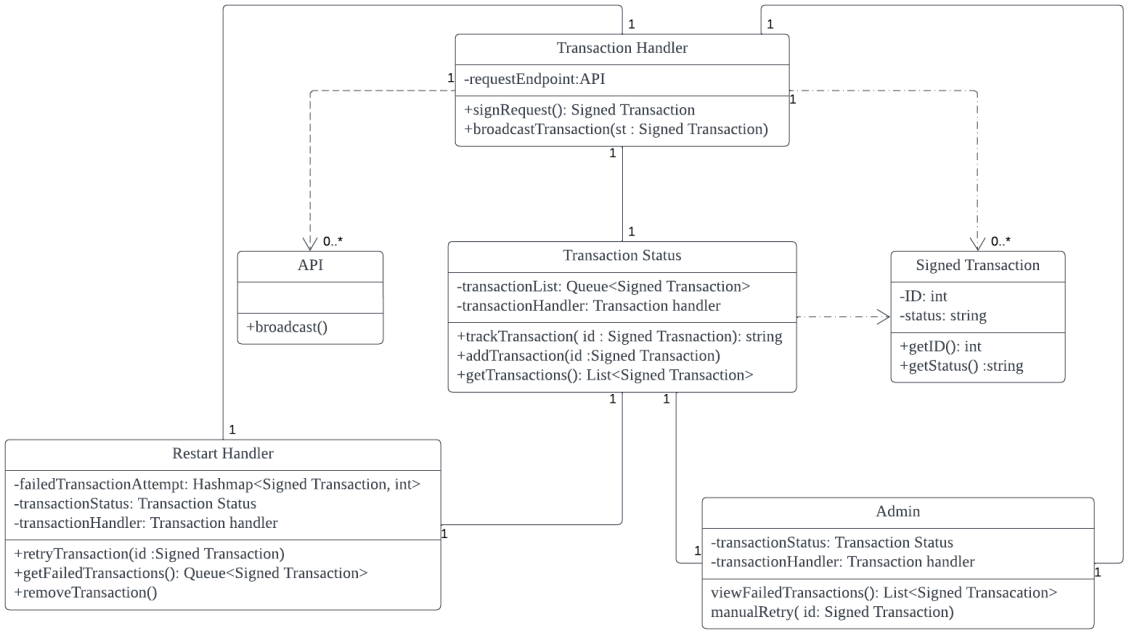
- Store the results of different transactions
- Provides an interface for the Administrators to identify and manage failed transactions

Admin Interface:
- Manually retry the failed transaction
- Provides a list of all failed transactions that can be retrieved from the Transaction Tracker
- Ability to view error logs and messages on failed transactions.
- Identify transactions that took longer than the 20-30s mark to highlight any outliers.

Restart Handler:
- Failed transactions are queued up to run again
- Each failed transaction ID is stored in a hashmap with the number of retry attempts to limit the same transaction from occupying the system for too long.
- Manages the restart interval
- Calls the Transaction Handler to run the first transaction in the queue.
- Once a failed transaction runs and is successful, it is removed from the Hashmap, and the status of the transaction is updated by notifying the Transaction Tracker.

Performance Conditions:
- Implementing parallel processing for the signing and broadcasting so that multiple instances of the Transaction Handler can run more efficiently.
- Optimizing the retry mechanism to handle failures optimally without introducing any unnecessary delay.
- Setting up monitoring tools to track the server's uptime, response time, and relevant metrics to analyze performance.

Monitoring and Logging:
- Used to keep track of the health and performance of the broadcaster service
- Log each transaction aligned with its details so that we can analyze and debug if any error occurs.
- Implementing notifications in the event of system crashes or service downtime
- Use monitoring tools to track the server's uptime, response time, and relevant metrics to analyze performance.


Transaction Handler
-requestEndpoint:API
+signRequest(): Signed Transaction
+broadcastTransaction(st : Signed Transaction)
API
+broadcast()
Transaction Status
-transactionList: Queue<Signed Transaction>
-transactionHandler: Transaction handler
+trackTransaction(id : Signed Trasnaction): string
+addTransaction(id :Signed Transaction)
+getTransactions(): List<Signed Transaction>
Signed Transaction
-ID: int
-status: string
+getID(): int
+getStatus() :string
Restart Handler
-failedTransactionAttempt: Hashmap<Signed Transaction, int>
-transactionStatus: Transaction Status
-transactionHandler: Transaction handler
+retryTransaction(id :Signed Transaction)
+getFailedTransactions(): Queue<Signed Transaction>
+removeTransaction()
Admin
-transactionStatus: Transaction Status
-transactionHandler: Transaction handler
viewFailedTransactions(): List<Signed Transacation>
manualRetry(id: Signed Transaction)
1
0..*