CS765 Project Part-1

Simulation of a P2P Cryptocurrency Network

Design Document

Saksham Rathi (22B1003), Kavya Gupta (22B1053), Mayank Kumar (22B0933)

Department of Computer Science,

Indian Institute of Technology Bombay

Flowchart of Event Execution

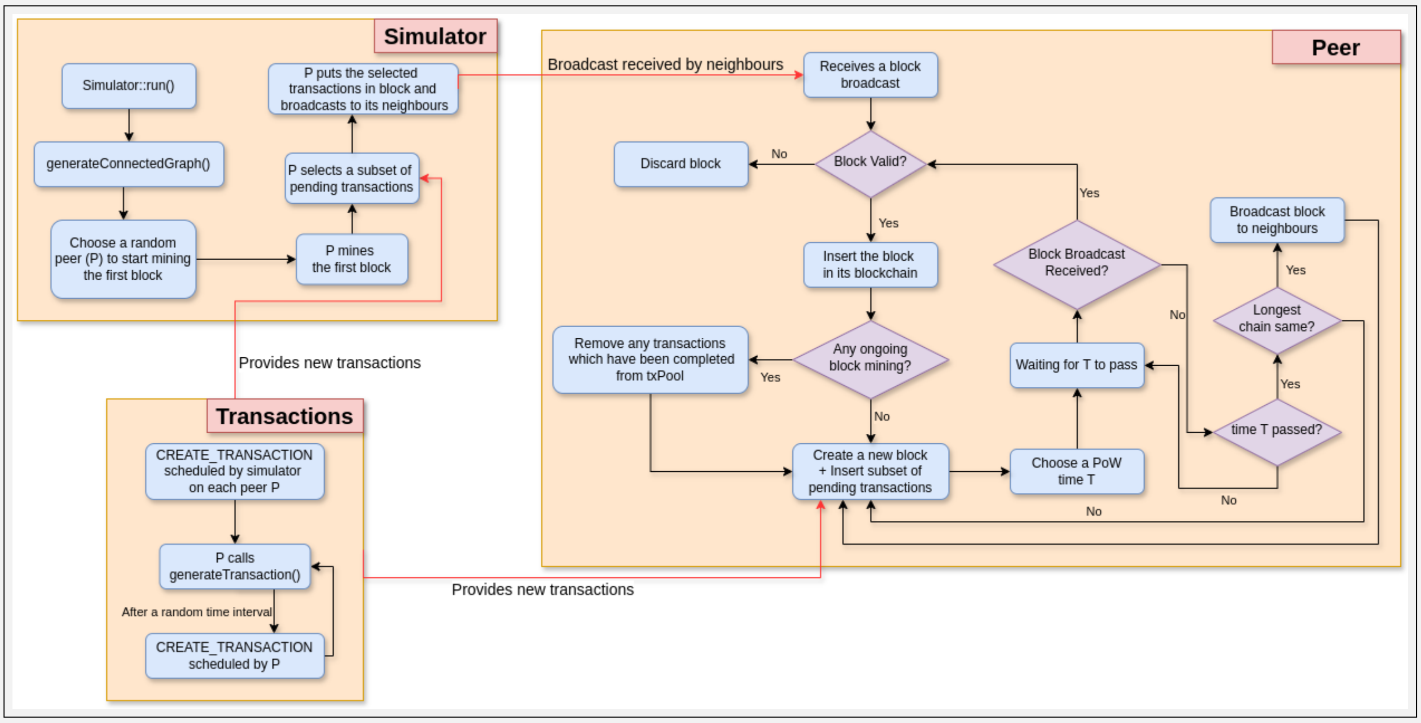
Simulator
Simulator::run()
generateConnectedGraph()
Choose a random peer (P) to start mining the first block
P mines the first block
P selects a subset of pending transactions
P puts the selected transactions in block and broadcasts to its neighbours
Provides new transactions
Transactions
CREATE_TRANSACTION scheduled by simulator on each peer P
P calls generateTransaction()
After a random time interval
CREATE_TRANSACTION scheduled by P
Provides new transactions
Peer
Broadcast received by neighbours
Receives a block broadcast
Block Valid?
No
Discard block
Yes
Insert the block in its blockchain
Any ongoing block mining?
Yes
Remove any transactions which have been completed from txPool
No
Create a new block + Insert subset of pending transactions
Choose a PoW time T
Waiting for T to pass
Block Broadcast Received?
Yes
No
time T passed?
Yes
Longest chain same?
Yes
Broadcast block to neighbours
No
No

Figure 1: Flow of events

Simulator

- The simulator calls the `run` function to start the simulation.
- It first calls the `generateConnectedGraph` function to create a connected graph of nodes.
- Each node then creates a genesis block using the `createGenesisBlock` function.
- The simulator schedules the creation of transactions by calling `scheduleEvent` with an inter-arrival time, event type, and node index.
- A random peer is selected to start mining by scheduling a `MINING_START` event.
- The simulator processes events from the event queue by updating the current time and handling each event accordingly.

Transactions

A peer creates a transaction by calling the `generateTransaction` function.

- The peer checks its balance using the `getPeerBalance` function.
- If the balance is greater than zero, it selects a random target peer.
- A transaction is created and added to the transaction pool.
- The transaction is then broadcasted to all neighboring peers.

After the simulator first schedules the event `CREATE_TRANSACTION` at each node:

- Each node generates a transaction and then a new time (`newTime`) is calculated by adding the current time to a value obtained from an exponential distribution using the function `getInterArrivalTime()`.
- The event `CREATE_TRANSACTION` is then recursively scheduled on the same node with the newly calculated `newTime`.

Peer

Receiving the Blocks

- The peer first checks if the block has not been seen before. If the block has not been seen, it stores the block's timestamp.
- Then, it checks if the parent of the received block is present in its blockchain. If the parent is not present, the block ID of the block is stored in a set.
- This block without a parent will be checked later when its parent arrives.

Validating the Received Block

- The peer validates the block by checking if the parent block exists in the blockchain.
- It initializes the balance of each peer to the initial balance and then iterates through the current block and its parent blocks, updating the balances based on the transactions.
- For each transaction, the sender's balance is decreased, and the receiver's balance is increased by the transaction amount.
- The miner of each block receives a mining reward added to their balance.
- If any peer's balance becomes negative during this process, the block is considered invalid.

Inserting the Block in the Blockchain

- Given the block is valid, it is inserted into the blockchain of the peer.
- The process of identifying the longest chain in the tree is then done and respective fields updated accordingly.
- Finally, it is checked if any children of the current block had arrived early on the peer. If so, then it is also inserted into the tree if it is still valid.

After all these steps, the information of the block is sent to the neighbors of the block. In case of the child blocks arriving before the parent blocks, their information is withheld with the peer till their parent arrives. *This ensures that the blockchain remains consistent and valid.*

Mining a Block

- **MINING_START:** In this event, the peer selects a subset of pending transactions from the transaction pool (txPool) and creates a block. The peer ensures that the sender has sufficient balance for each transaction before adding it to the block. The block is then prepared with the necessary details such as parent ID, miner ID, and height.

- **MINING_END:** This event is scheduled and added to the event queue after MINING_START. During this event, it is first checked if the longest chain is the same as when the MINING_START was scheduled. If the chain is unchanged, the block is inserted into the blockchain and sent to the peer's neighbors. If the chain has changed, the block is discarded and the mining process is started again.

Helper Functions

generateConnectedGraph

- **Initialize Peer IDs:** The function starts by creating a list of peer IDs from the peers array. The peer IDs are shuffled to ensure randomness in the graph structure.
- **Create Initial Chain:** The function creates a chain of connections between peers to ensure the graph is connected. This is done by iterating through the shuffled peer IDs and connecting each peer to the next one in the list.
- **Ensure Minimum Degree:** To ensure that each peer has at least 3 connections, the function adds edges between the current peer and a randomly selected peer, ensuring that the randomly selected peer has fewer than 6 connections and is not already connected to the current peer.

handleEvent

- **Event Handling:** The function processes different types of events such as mining start, mining end, transaction creation, transaction sending, transaction receiving, block sending, and block receiving. Each event type triggers specific actions performed by the peers.
- **Event Scheduling:** When an event is processed, the function calculates the time for the next event based on the current time and specific conditions such as block inter-arrival time, transaction inter-arrival time, or network latency.
- **Event Queue:** The calculated future event is then scheduled by adding it to an event queue. This queue ensures that events are processed in the correct order based on their scheduled times.
- **Simulation Flow:** The event queue drives the simulation flow by continuously processing and scheduling events, allowing the simulation to progress through different states and interactions between peers.

Additional Features

Handling of Blocks with Invalid Parent ID

- If the parent of the received block is not present in the blockchain, the block ID is stored in a set.
- Blocks without a parent will be checked later when their parent arrives.
- If any children of the current block had arrived earlier, they are also inserted into the blockchain if they are still valid.
- After these steps, the block information is sent to the neighboring peers.
- In case child blocks arrive before their parent blocks, their information is withheld until the parent block arrives.