Submission Info Project 3

Sketch of Implementation for Integrating EVM into the Decentralized Platform

Steps Required to Set Up the EVM

Select and Integrate an EVM Library:

- Choose one of the provided EVM libraries (e.g., Go-Ethereum's EVM).
- Integrate the chosen library into the existing system.

Initialize EVM Environment:

- Define the initial state of the EVM, including setting up genesis accounts with pre-allocated Ether.
- Implement methods to initialize the EVM context for each transaction, including the gas limit, sender, recipient, and input data.

Modify Block Structure:

- Update the block data structure to include the `state_digest` field, which stores the Keccak-256 hash of the current EVM state after processing the block's transactions.
- Replace the `messages` field with a `transactions` field containing a list of Transaction objects.

Deploying a Smart Contract

Check Transaction for Contract Deployment:

- Check a transaction created by tests and deploy it to the EVM.
- Check the required fields such as sender, nonce, amount, signature, and public_key.

Process Deployment Transaction:

- When a deployment transaction is included in a block, the block proposer initializes the EVM with the transaction context and executes the contract creation.
- The EVM generates a new contract address, deploys the bytecode, and returns the contract address.

Update State and Store Contract:

- Update the EVM state with the newly deployed contract for all other nodes.

Handling Transactions

Transaction Validation:

- Verify the validity of each transaction, including checking the signature and nonce.
- Ensure the sender has sufficient balance to cover the amount and potential gas fees (if enabled).

Execute Transactions:

- If `to` is empty, handle it as a contract creation. Otherwise, treat it as a function call to an existing contract.
- Execute the transaction and save the result.

Handling State and Forks

State Management:

- Maintain a state database that keeps track of all account balances, contract storage, and other relevant state data.
- Ensure that all nodes update their state databases consistently after processing each block.

Fork Handling:

- While fork resolution is not required for this project, our blockchain handles forks in some way.
- To resolve forks, nodes would need to re-evaluate the competing chains based on predefined consensus rules (e.g., lowest leader value) and revert any state changes from the discarded chain.

Steps to Resolve Forks (Conceptual ideal way)

Detect Fork:

- Monitor for blocks that extend different branches of the chain.

Evaluate Chains:

- Use consensus rules to evaluate which chain should be considered canonical (e.g., longest chain with the most work).

Revert Non-Canonical Chain:

- Revert state changes caused by transactions on the non-canonical chain.

Reprocess Canonical Chain:

- Apply transactions from the blocks in the canonical chain to ensure the state is updated correctly.

By following these steps, the integration of the EVM into your decentralized platform can be achieved, enabling the execution of arbitrary smart contract code and ensuring consistent state across the network.