Assignment -1 : Simulation of Blockchain

Deadline : 27th May 11:59 pm (Tuesday).

There are **two objectives** for this assignment. You are required to do both.

Github Submission Guidelines:
https://github.com/pclubiitk/Lord-of-Chains-25.git

1) Fork this repository on Github.
2) Clone your fork to your local machine.
3) Work on your assignment in that particular assignment's folder (if it's the first assignment, work in Assignment-1 directory), **create a folder inside the assignments folder** (Naming convention: <your name>-<your roll no.>-<assignment number> Ex: Srijani-231033-1) and commit all changes with clear messages.
4) Push your changes to your fork.
5) Create a pull request from your fork's branch to the original repo.

Objective 1: Implement a simulation of a blockchain following the PoW consensus mechanism using OOPs. You must have **clear comments in your code.**

Guidelines for Objective 1:

1) Your program must consist of multiple nodes out of which few are miners.
2) When the program is run, it must prompt for the user's public and private key input. (This is actually not the case in real-life as a node stores its private key internally usually via a wallet, this will be explained. The user is not required to type in their private key).

3) Next, it prompts for the recipient's public key (address) and an amount. This information together makes up a transaction. It is required that once this transaction is entered, the terminal displays the digital signature.

4) You may ask for multiple transactions and once all the transactions are entered, all these transactions will be verified against by their respective sender's public key (show on terminal) and added into temporary storage.

5) Each miner takes up the transactions from the storage and will compute to find a nonce for the proposed block that the transaction consists of. All transactions do not go into one block. They must get verified via multiple blocks in one program run. Extra points for showing the miners take up different subsets of the transactions and competing for the nonce of their blocks.

6) This block must be added onto each node's copy of the blockchain after verifying the hash of the block. (You may use a file system to load the blockchain data during the program run and copy the updated blockchain back into files. The blocks must have the previous block's hash to ensure immutability as discussed in class.

7) Keep a track of the balances of each node during the process. During the mining process, add a reward transaction to the list of transactions for the miner.

8) Extra points if you are able to demonstrate the calculation of balances using the transactions on the blockchain.

Other Instructions:

1) Make sure to define classes for different entities involved, like nodes, miners, block, transaction, signature, blockchain and any more that you might need.
2) You may choose to build on the guidelines provided or modify them if however you may deem necessary. Your reasoning must be reflected via comments on your code.
3) Comments are necessary along with a README.md file with clear instructions on how to use your program along with a requirements.txt file if necessary.

Objective 2: Use randomized selectors as nodes and simulate the following consensus mechanism. Implement based on your understanding.

Avalanche Consensus Mechanism and Its Accuracy

The Avalanche consensus mechanism is a revolutionary approach to achieving consensus in distributed networks, primarily utilized by the Avalanche blockchain. Introduced by a group of researchers in 2018 and further developed by Ava Labs, the Avalanche protocol is designed to provide high throughput, low latency, and scalability while maintaining security and decentralization. This essay provides a detailed examination of the principles underpinning the Avalanche consensus mechanism, its operational accuracy, and its implications for blockchain networks.

Principles of Avalanche Consensus

Avalanche differs from traditional consensus protocols like Proof of Work (PoW) and Proof of Stake (PoS) by utilizing a probabilistic consensus mechanism that is divided into four phases: Slush, Snowflake, Snowball, and Avalanche. This multi-stage process enhances the accuracy of consensus by progressively increasing the confidence level of network decisions through repeated sampling.

1. **Slush:** A node selects a small, random subset of nodes to query about a specific state ("0" or "1"). The node adopts the majority state observed during the sampling process. This stage ensures that nodes gather preliminary consensus data.

2. **Snowflake:** The node continues to sample nodes iteratively. If the node observes the same state for a predefined number of consecutive samples, it becomes more biased toward that state. This stage reduces ambiguity and increases the probability of accurate consensus.

3. **Snowball:** The node maintains counters for each state, tracking the number of consecutive observations of each state. The state with the highest count becomes the preferred state, further solidifying the network's agreement.

4. **Avalanche:** In the final stage, the preferred state is propagated throughout the network, and consensus is achieved when a sufficient number of nodes adopt the same state. This stage is critical for preventing double-spending and ensuring data immutability.

Accuracy of Avalanche Consensus

The accuracy of the Avalanche consensus mechanism is primarily derived from its probabilistic sampling approach. Unlike deterministic mechanisms, Avalanche employs repeated sampling to reduce the probability of incorrect consensus. This method significantly lowers the likelihood of network forks and enhances network reliability.

Additionally, Avalanche employs mechanisms to mitigate Sybil attacks by requiring nodes to participate in multiple rounds of sampling, making it computationally prohibitive for attackers to influence the network outcome. The consensus threshold is set to a level where the probability of malicious nodes successfully subverting the network is extremely low.

Operational Implications and Accuracy Assessment

Avalanche's accuracy is further reinforced by its dynamic nature, allowing nodes to rapidly adjust to changing network conditions. This adaptability reduces the risk of consensus delays and stale data propagation. Moreover, the protocol's reliance on multiple phases of sampling ensures that consensus decisions are robust against transient network fluctuations and malicious attempts to manipulate consensus.

Empirical studies on Avalanche's implementation have demonstrated that the protocol can achieve consensus within seconds while maintaining a low error rate, even in the presence of Byzantine nodes. This accuracy is a crucial factor in maintaining blockchain integrity and preventing double-spending attacks.

Conclusion

The Avalanche consensus mechanism introduces a multi-phase, probabilistic consensus protocol that emphasizes accuracy through iterative sampling and state validation. Its layered approach, encompassing Slush, Snowflake,

Snowball, and Avalanche, ensures that consensus is achieved efficiently and accurately without sacrificing network scalability. As blockchain technology evolves, Avalanche's accuracy and resilience against attacks position it as a promising consensus model for decentralized applications and large-scale distributed networks.