Sipna College of Engineering & Technology, Amravati.
Department of Computer Science & Engineering

Branch :- Computer Sci. & Engg. **Class :- Final Year**
Subject :-Block Chain Fundamentals Lab manual **Sem :- VII**

Teacher Manual

PRACTICAL NO 9

AIM: Implement the consensus mechanisms of Proof of Work (PoW)

S/W REQUIRED: Python

Proof of Stake (PoS) is a consensus mechanism used by blockchain networks to validate transactions and create new blocks. Unlike **Proof of Work (PoW)**, which requires computational power, PoS selects validators based on the amount of cryptocurrency they "stake" or lock up as a guarantee of honesty. The higher the stake, the higher the chance of being selected to validate the next block.

A typical PoS based mechanism workflow:

1. Nodes make transactions. The PoS algorithm puts all these transactions in a pool.
2. All the nodes contending to become validator for the next block raise a stake. This stake is combined with other factors like 'coin-age' or 'randomized block selection' to select the validator.
3. The validator verifies all the transactions and publishes the block. His stake still remains locked and the forging reward is also not granted yet. This is so that the nodes on the network can 'OK' the new block.
4. If the block is 'OK'-ed, the validator gets the stake back and the reward too. If the algorithm is using a coin-age based mechanism to select validators, the validator for the current block's has its coin-age reset to 0. This puts him in a low-priority for the next validator election.
5. If the block is not verified by other nodes on the network, the validator loses its stake and is marked as 'bad' by the algorithm. The process again starts from step 1 to forge the new block.

Implementation:

```
import random

# Define a Validator class
class Validator:
    def __init__(self, name, stake):
        self.name = name
        self.stake = stake  # stake in coins

    def __str__(self):
        return f"{self.name} (Stake: {self.stake})"

# Create a list of validators
validators = [
    Validator("Alice", 50),
    Validator("Bob", 30),
    Validator("Charlie", 15),
    Validator("Diana", 5),
]

# Build the staking pool (names are repeated proportional to stake)
```

```
def build_staking_pool(validators):
    pool = []
    for v in validators:
        pool.extend([v] * v.stake)
    return pool

# Select a block validator
def select_validator(pool):
    return random.choice(pool)

# Simulate multiple rounds of block selection
def simulate_pos_rounds(validators, rounds=10):
    pool = build_staking_pool(validators)
    print("Validators in the network:")
    for v in validators:
        print(f" - {v}")

    print("\n--- Block Validation Rounds ---")
    for i in range(rounds):
        winner = select_validator(pool)
        print(f"Round {i+1}: Block validated by {winner.name}")

# Run the simulation
simulate_pos_rounds(validators, rounds=10)
```

Output:

```
Validators in the network:
 - Alice (Stake: 50)
 - Bob (Stake: 30)
 - Charlie (Stake: 15)
 - Diana (Stake: 5)

--- Block Validation Rounds ---
Round 1: Block validated by Alice
Round 2: Block validated by Alice
Round 3: Block validated by Bob
Round 4: Block validated by Alice
Round 5: Block validated by Charlie
Round 6: Block validated by Bob
Round 7: Block validated by Alice
Round 8: Block validated by Bob
Round 9: Block validated by Alice
Round 10: Block validated by Charlie
```

CONCLUSION: This simple implementation of **Proof of Stake** demonstrates how validators are selected based on their stake