30/11/2024

Connor Mallon

CSC4008 Distributed Computing

40295919

**Introduction**

The node.py script implements a foundational peer-to-peer network node for a distributed computing network or a P2P network.

Written in Python, this program establishes a communication framework using UDP sockets, allowing nodes to exchange messages, share transaction details, and maintain synchronized states across the network. With support for broadcasting transactions, Lamport clocks for event ordering, and peer discovery, the script demonstrates key elements of decentralized systems. This node serves as a building block for scalable, fault-tolerant networks used in applications like cryptocurrencies, distributed databases, and consensus algorithms.

At the core of the implementation is the Node class, which encapsulates functionalities like maintaining peers, managing transactions, and facilitating inter-node communication. Each node is uniquely identified by a UUID and initializes with a default balance of 1000 units. It can process transactions, broadcast them to peers, and maintain synchronization using Lamport clocks to resolve event ordering. The script employs UDP messaging for efficient, lightweight communication between nodes. Supported message types include transaction broadcasts, peer discovery, synchronization requests, and balance queries. The Transaction class provides a structured way to encapsulate and validate transaction data, ensuring the ledger remains consistent and secure. Additional utilities, such as formatted logging and error handling, enhance the maintainability and reliability of the code. Nodes can also join existing networks by initiating a discovery process, allowing dynamic scaling and network growth.

In conclusion,

The node.py script is a robust and extensible implementation of a decentralized node designed for distributed systems. By integrating peer-to-peer communication, transaction handling, and synchronization mechanisms, it highlights the fundamental concepts of distributed ledger technology. The use of Lamport clocks ensures consistent event ordering across asynchronous nodes, while the broadcast mechanism guarantees the dissemination of critical information. This codebase serves as an excellent starting point for developing and exploring decentralized systems and true Peer-to-Peer applications.