Name: Darshit Bimal Gandhi
Roll No: 22M0824

CS765: Introduction of Blockchains, Cryptocurrencies & Smart Contracts

Project Title: Building a layer-2 DAPP application

Instructor: Prof. Vinay Ribeiro

- **Project Features:**
  - **a.** Developed a **DApp** using **Solidity** on top of the **Ethereum Blockchain**. This project enables users to **register themselves**, **form accounts, transact funds with each other** and **terminate their accounts.**
  - **b.** Each user's contribution is tracked in the joint account.
  - **c.** Transactions can occur between users who are directly connected and those connected through intermediate users.
  - **d.** The network of users and joint accounts forms a graph structure.

- **Smart Contract Development:**
  - i. Developed **smart contracts using Solidity**, implementing functions for **user registration, joint account creation, transactions, and account termination**.
  - ii. Ensure efficient event handling and logging.

- **Ethereum Node Setup:**
  - i. Set up **an Ethereum node using Truffle/Ganache** for local deployment and testing.
  - ii. **Deployed** smart contracts onto the **Ethereum node**.
  - iii. Establish communication between the DApp and the blockchain.

- **User Interaction Script:**
  - i. Developed a **Python script** for user interaction with the DApp and Ethereum node.
  - ii. **Automate** registration, account creation, transactions, and account termination.

- **Set of Functions in the Smart Contract:**
  - **a.** registerUser(id, user name): Register users and add them to the available user list.
    - i. Users can register by providing unique user IDs and usernames.
    - ii. User information is stored on the blockchain.

  - **b.** createAcc(user id 1, user id 2, balance): Create a joint account between two users and track individual contributions.
    - i. Users can form joint accounts, specifying their individual contributions.
    - ii. The DApp tracks individual contributions within joint accounts.

  - **c.** sendAmount(user id 1, user id 2): Transfer an amount between users, considering their joint account balances.
    - i. Users can initiate transactions between connected accounts.
    - ii. Transactions can occur through intermediate users.
    - iii. Transaction execution is conditional on account balances.

  - **d.** closeAccount(user id 1, user id 2): Terminate a joint account.
    - i. Users can close joint accounts, removing the corresponding edge from the network.
    - ii. Account data and history are securely cleared.

- **Setup Instructions:**
  a. Follow the instructions provided in the https://docs.google.com/document/d/1IfIwdF6vhf4KYP1OYTwFmgqtdHO6nT7m0JaMH1xCIYo/edit?usp=sharing to set up an Ethereum node on your local machine using Truffle/Ganache.

  b. Deploy the smart contract (DApp.sol) using Remix or the Ethereum node.

  c. Use the provided python script (`client.py`) or your preferred language to interact with the deployed smart contract. The script should be used to register users, create joint accounts, initiate transactions, and close accounts.

- **Usage:**
  a. Compile and deploy the smart contract.

  b. Run the client.py script to simulate interactions with the DApp. The script will register users, create joint accounts, and initiate transactions.

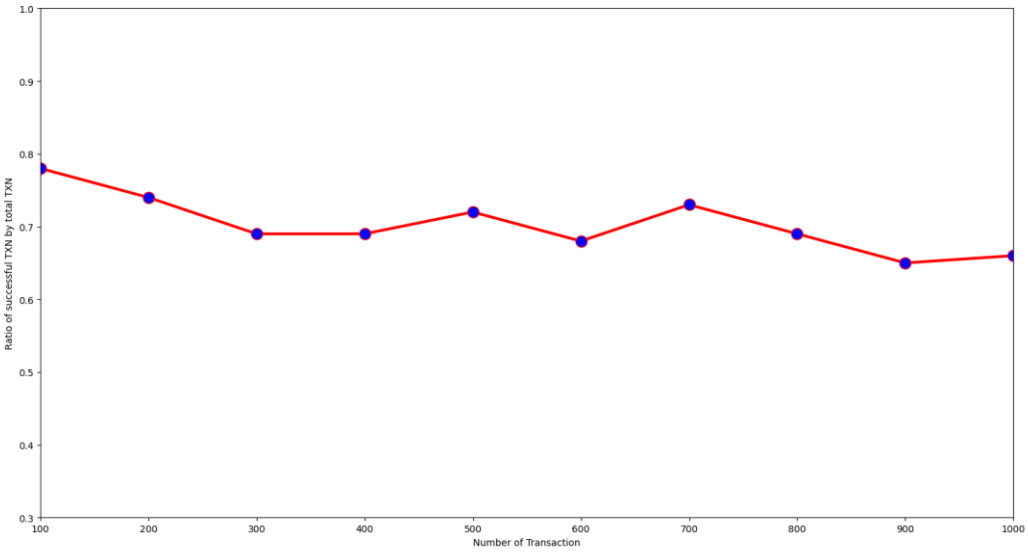
- **Why power law?**

  Power law degree distribution, also known as scale-free distribution, is often observed in many real-world networks, such as the World Wide Web, social networks, biological networks, and citation networks. The power law degree distribution is characterized by a few nodes with a high degree (often called "hubs") and many nodes with low degrees.

  Robustness: Networks with power law degree distribution are more robust to random node removal than networks with other degree distributions. This is because the hubs in a power law distribution have many connections, and their removal has a smaller effect on the overall structure of the network.

  Efficiency: Networks with power law degree distribution can have a higher efficiency in terms of information transfer than other types of degree distributions. This is because information can travel quickly through the hubs, which have many connections, and reach the rest of the network.

- **Observation:**



We see that as the number of total transactions increases, our overall number of failed transactions increases and hence our efficiency decreases. This result is expected because once the transactions start utilizing the links, the balance on those links decreases and hence those links are not able to fund more transactions. Also, we have to find alternate paths to carry out the transactions, which also decreases over time and hence a larger number of transactions fail.

- **Conclusion:**
    a. Decentralization and Trustlessness: The project highlighted the core tenets of blockchain technology—decentralization and trustlessness. By utilizing a distributed network and smart contracts, we demonstrated how users can transact and interact without relying on a central authority.

    b. Smart Contract Development: The implementation of the DApp's functionalities using the Solidity programming language deepened our understanding of smart contract development. We learned to define contract structures, implement logic, and manage interactions among users.

    c. Ethereum Network Setup: Setting up an Ethereum node using Truffle/Ganache introduced us to the practical aspect of blockchain deployment. This experience provided insights into the complexities of configuring a local environment for testing and development.

    d. Realistic Simulation: The simulation of a user network with power-law degree distribution and exponential balance distribution offered a practical glimpse into the potential real-world applications of our project. This observation highlighted how blockchain can model complex financial relationships efficiently.

    e. Transaction Handling and Consistency: Ensuring accurate transaction handling and maintaining consistent account balances were vital challenges. This project reinforced the importance of robust error handling and the need to consider various edge cases in a decentralized environment.

- **Future Prospects:**
    a. Enhanced User Interface: Developing an intuitive user interface for interacting with the DApp could enhance user experience and accessibility.

    b. Security Auditing: Conducting thorough security audits to identify vulnerabilities and ensure the DApp's resilience against potential attacks is crucial for its robustness.

    c. Scaling Solutions: Exploring layer-2 scaling solutions could improve the DApp's scalability, allowing it to handle a larger number of users and transactions.

    d. Integration with Real-World Applications: Extending the project's concepts to real-world scenarios, such as shared financial management or supply chain tracking, could showcase the practicality of decentralized systems.

- **Final Thoughts:**

  This project has not only deepened our technical skills but also expanded our perspective on the potential of blockchain in revolutionizing various industries. As we conclude this endeavor, we reflect on the journey that led us to design, develop, and implement a functional layer-2 DApp. With the newfound knowledge and experience gained, we are well-equipped to contribute to the ongoing evolution of blockchain technology and its transformative impact on the world.