6.0 Architectural Design Document

6.1. Introduction

6.1.1 System Objectives

6.1.2 Hardware, Software, and Human Interfaces

6.2. CSCI Descriptions

6.2.1 Peer-to-peer network

The nodes on the system communicate using a bidirectional streams. The initial entry point can be the address of any node on the network or (more typically) a load balancer with a static ip. The new node is then provided with a list of nodes the existing node is communicating with. This list can be checked by repeating the process with another target node and comparing the lists. Message payloads require encryption using the Elliptic Curve Digital Signature Algorithm. Nodes can check message integrity by using the ecdsa on the message signature, this process verifies the sender and that the message contents have not been tampered with. This functionality is enabled by the ecdsa go package.

6.2.2 Blockchain

The debased blockchain implementation follows the industry standards laid out by bitcoin and ethereum. This guarantees the blockchain fulfills the key characteristics regarding block creation, checking the chain integrity, and checking block integrity. The implementation being used is a modified version of the coral health blockchain. The coral health implementation was used as a base from which the code has been cleaned to fit our style standards and functionality was added to support storing data.

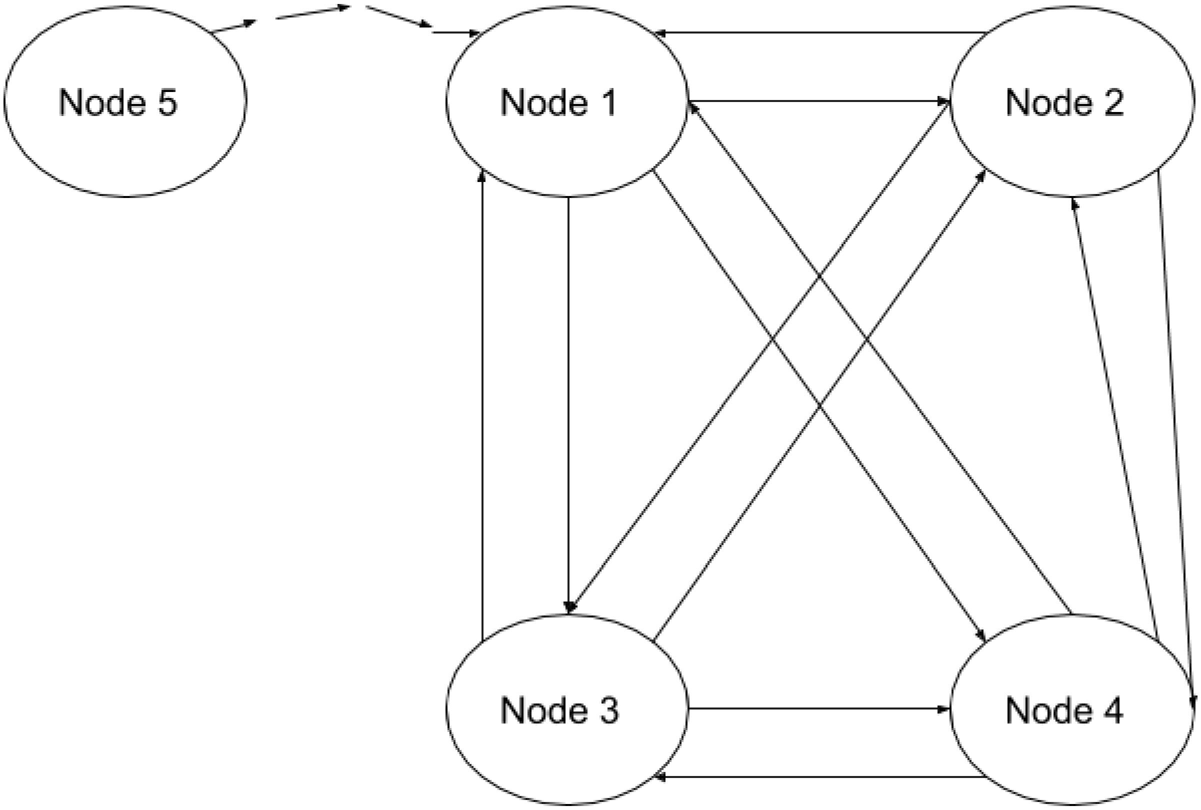
6.2.3 Proof of stake

The debased system uses a totally proprietary proof of stake implementation that is comprised of seven key subsystems: block generation bidding, block validation, transaction creation, transaction fulfillment, data storage, transaction validation, and hunting. Block generation bidding is a process in which nodes place bids on the system, the best bid is then chosen algorithmically. After the block is created, it is sent to the other nodes for the validation process. The nodes each vote on the block. The system quickly comes to consensus on the validity of the block. If the block is not valid repeat the bidding process, if the block is valid each node adds the block to its chain. Transaction creation is the process outline in the user interface section. It allows users to interact with the system and their data on it. Fulfilling transaction is carried out by the node generating the block, it is one of the key checks executed during the voting process. Data storage refers to the process of rewarding nodes for storing older blocks. This process occurs periodically, during which, the nodes prove what data they have stored. Their claim is then validated by the system and they are awarded accordingly. Query validation is the process of validating queries that do not need to be recorded on the blockchain (read operations). This process is a miniaturization of the block validation process. The hunt system allows nodes to present the network with proof of a nodes' wrongdoing. This allows the system to take the offending nodes stake. Each of these subsystems is outlined in greater detail in architecture folder.

6.2.1 Concept of Execution

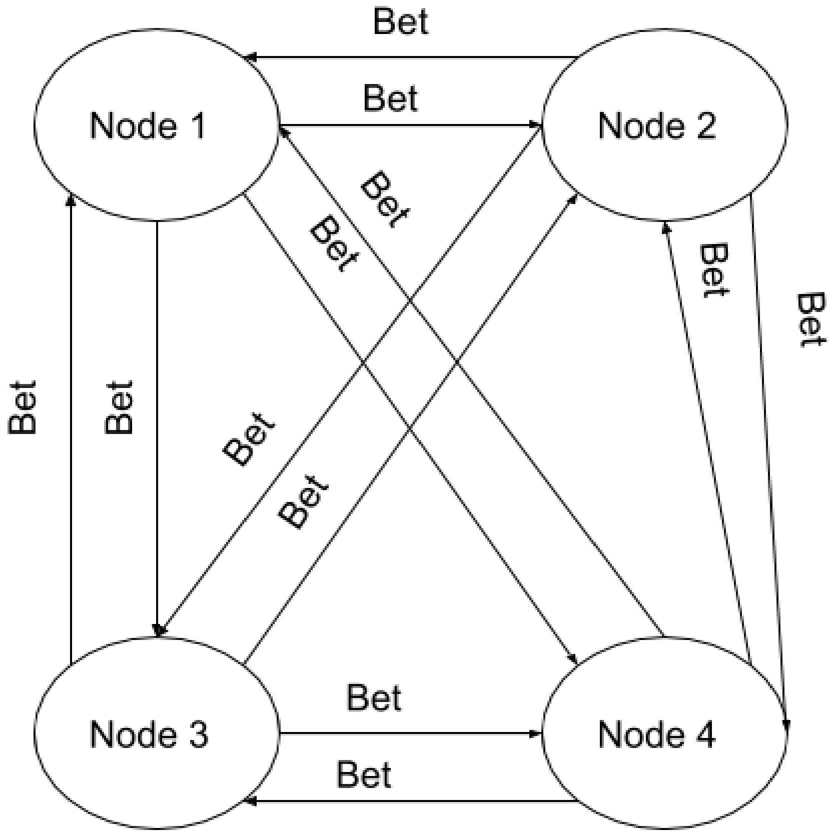
Here will be the descriptions of various use cases for the debased project. A detailed high level overview will be provided for each use case and scenario that can come up.

6.2.1.1 Adding a node to the decentralized system



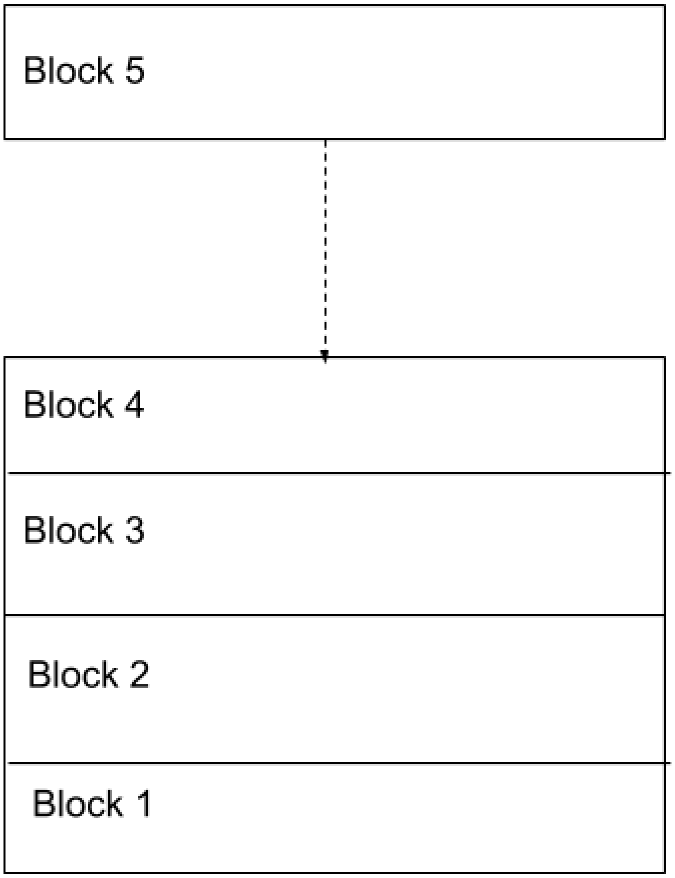
When adding a node to a debased system, the prospective node will need to be supplied with a static IP address that references any node within the system. By design, any node in the decentralized system can act as an entry point for new nodes. As long as the given IP address is valid, the prospective node will make a request to join the decentralized system. When the entry point node sees this request, it will instantiate a vote in a the decentralized system in order to determine whether or not the prospective node should be added. If the prospective node is approved, then it will now be added to the decentralized system. The entry point node will create a bidirectional stream between the prospective node and itself. With this stream, the entry point node will send over a list of IP addresses that are the other nodes in the system. The prospective now will now make a unique stream with each IP address (node) it is given. Since the prospective node has been approved at this point, there is no need to perform another voting process to approve of the new streams. Once all of the bidirectional streams have been established, the prospective node will be given an account, identity, and a certain amount of preliminary tokens. The node has now been safely added to the decentralized system and can act as any other node.

6.2.1.2 Voting process for some arbitrary transaction



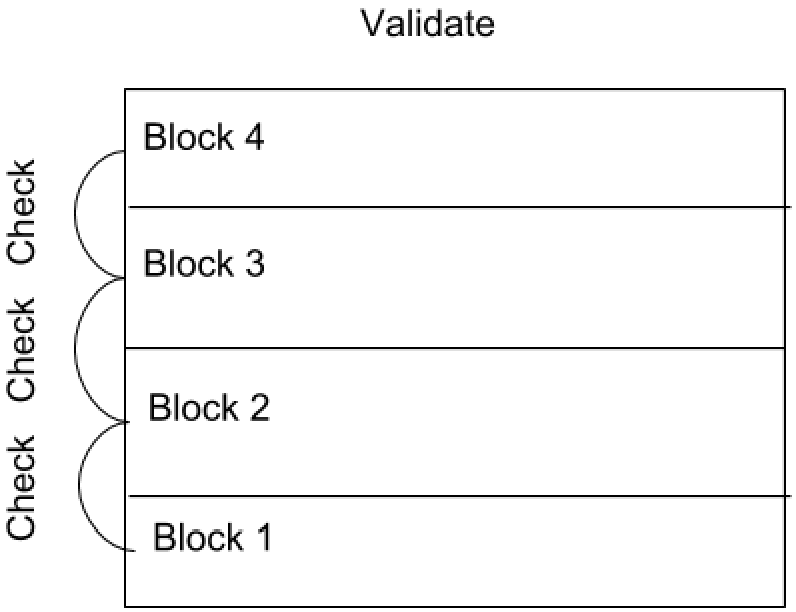
The voting process occurs whenever there needs to be some decision made in the decentralized system. These actions can range from adding a new block to the blockchain or performing a witch hunt on a specific node. In order to properly run a voting process, a node needs to instantiate a vote and notify the system that a vote is happening. Here the node uses the system seed value to determine its voting window. While waiting for the nodes voting window, the node will check as many pieces of the block as possible. This is important due because the node needs to analyze as much of the blockchain as possible in order to make an educated vote. Things that are checked during this checking phase include things like ensuring that all block generation rules were followed, verifying the confidence level this node has with the node that has instantiated the vote, and so on. Once it is time, the node will place a Bet at the start of the window. Bet is an object that contains the respective fields: Stake_bet, Block_id, Position, Confidence, Bet_round, and Signature. If all of the nodes come to a consensus, then the voting process is successful and either the requested action is performed or denied. However if the vote fails to reach an agreement, the whole voting process is repeated. The voting process will be repeated over and over again until a consensus is reached among the nodes in the system.

6.2.1.3 Generating a block



In order to generate a block, there are several rules that must be followed. Firstly, all transactions must come from pending transactions and receipts that exist within the decentralized system. The reason for this is due to ensuring that there are not any conflicting transactions within the system. Following that, all transactions that are present within the system must be at or above market cost. Additionally, it is important that all transactions have been verified to be valid. With these conditions maintained, this means that the blockchain has been left in a valid state and can be signed by a generation bid winner. After all of these checks, a vote will be initiated to attempt to add the current block to the blockchain. If the vote passes and a consensus is reached, the system will either add the new block or deny it from being added to the blockchain. After that, each node will now start with a new block and start from the beginning.

6.2.1.4 Validating a block



There are various ways for the debased system to validate a block. Firstly, validating a block is performed for a block is added to the blockchain. This ensures that all of the blocks on the blockchain are not corrupt or contain any malicious data. One condition to check for validity is to ensure that all of data on the blockchain is matching the data in the blockchain of other nodes. After running these checks, the nodes can assume that the block is valid.

6.2.2 Interface Design

The debased system is reliant on the interfacing between the p2p network, blockchain, and the proof of stake subsystems.

6.2.2.1 Interface Identification and Diagrams

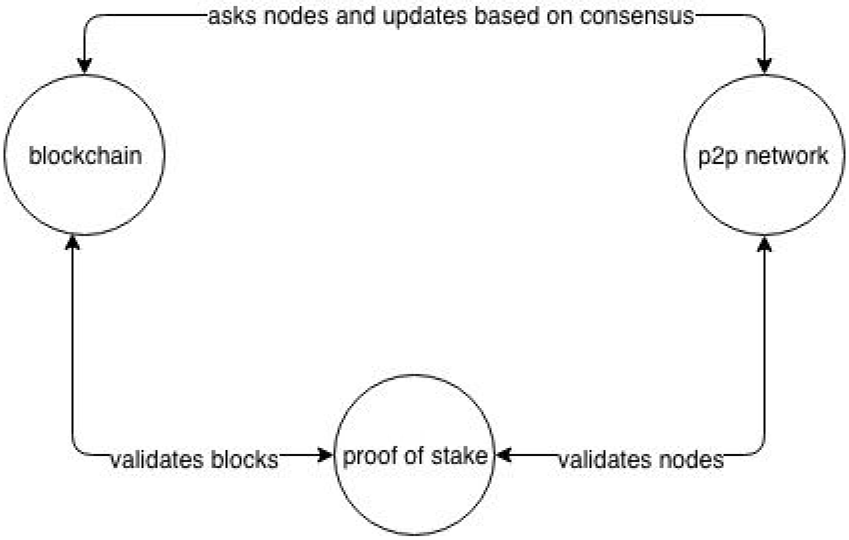
The only actor in our system is the user. From any user perspective, the interface between the user and the debased system is through the program CLI. The user will be able to write to the command line interface to execute the program and then interface with the blockchain through the program by reading and writing text through the debased program's CLI. The other subsystems of the debased system are listed in the chart below:

Subsystem	**Interface Description**
program CLI (user)	- Program's command line interface. - Interfaces with the user's blockchain.

blockchain	- Data structure that the user can manipulate through the Program CLI. - Interfaces with the p2p network and the proof of stake.
P2P network	- Refers to communication between nodes within the debased system. - Interfaces with the blockchain and the proof of stake.
proof of stake	- The implementation of blockchain and node validation. - Interfaces with the blockchain and the P2P network.

6.2.2.2 Project Interactions

The different subsystems that make up the debased system are best illustrated interfacing with each each other below. More information regarding the interactions between the different subsystems is described below the diagram:



Proof of stake interactions: The proof of stake is the implementation for validating nodes and their activity in the p2p network. It also asserts validation of new blocks that would be added to the blockchain.

Blockchain interactions: The blockchain interfaces with the p2p network by updating based consensus between the nodes on the network. The blockchain uses proof of stake to validate new blocks on the blockchain.

P2P network interactions: The p2p network interfaces with the blockchain of each user and updates the blockchain according to consensus. The p2p network interfaces with proof of stake by implementing it in the validation of changes to the blockchain and validation of nodes in the network.