Distributed Systems MP2 CP2**Group 54**

Ambati, Teja(tambati2)   Sypereddi, Anoop(anoops2)

April 2019

URL: `TBD`
Hash: TBD

1 Runnning Instructions

mp2 files inside /home/anoops2/go/src/
Build Instructions: go install mp2
Running Instructions: mp2 $NAME $SERVICEIP $SERVICEPORT

2 Design Specifications

We took advantage of net.Conn in order to inplement our distributed node implementation.

2.1 Node Connectivity

2.1.1 Node Initializations

Upon initialization a Node will initially find an open port and make a connection with the service using net.Dial. After sending the connection information the service will start a loop which listens to the service and recieves messages. Upon recieving an INTRODUCE message from the service, the node will store the message in a set of known nodesv. It will also store that dialed connection in a set which may be used for multicasting. Additionally, the node upon initialization will start a listener thread which will listen to the open port which was initially found by the program. After recieving a connection the node will create a connection with this node and store it in the set and also start a thread which will listen for any messages sent by that node.

2.1.2 Node Discovery

Upon recieving a connection from a node the node will connect to that node and proceeds to send previous known connections information from the nodes set. Whenever a listener thread recieves a previous INTRODUCE message, it will check if it already exists in its own set. If it is a new connection it will store it in its nodes set and multicast the message to all its current connections. This way all connections are propogated amongst all nodes.

2.1.3 Node Failures

When a node fails by either QUIT or DIE it is handled the same way. The node will have several nodes listening to it. When the node's connection is lost is detected the listener threads will traverse through nodes and attempt to connect to any node which hasn't been declared as failed. If the node can not be connected to it will be assumed that the node is dead and it will be marked as dead and a DELETE message will be multicasted to let all nodes know that the node is dead. If a new connection is found it will be added to the connections array and the nodeListener will now listen to that node instead.

2.2 Handling Transactions

2.2.1 Propogating Transactions

When the service sends a transaction to any node, the node will call the node storage method and multicast the transaction to all other nodes. Any node which recieves the transaction will call the node storage method and send multicast the message to all connections. Additionally, upon connecting to a new connection the whole list of transactions are given to the new connection. This is to ensure new connections also have all transaction history. This however slows down the propogation process and resulted in some longer propogation delays.

2.2.2 Storing Transactions

Transactions are stored as transaction objects within a map as well as the current block. By using blocks for concensus ordering is guaranteed.

2.3 Blockchain Implementation

2.3.1 Initialization of the Blockchain

Whenever a node is instantiated a blockchain object is instantiated as well. The blockchain consists of block objects which hold transactions, a previoushash, a current has, the depth, and account balances. The chain holds the current block, the block being solved, all blocks, and a place for unbuffered transactions to be stored. The chain starts with a previous hash of 0.

2.3.2 Building Blocks

Whenever the blockchain is instantiated or a block is moved to solving, a new block is created. The block will hold all transactions which come in during this time. When the solve command is issued this block will stop recieving transactions. It will go through all stored transactions and approve or reject the transation using the previous block's account information. Previous block is determined by selecting the block with the highest depth or the longest chain. If there are multiple chains of the same length, the block will select the node with the most transactions or highest hash value. It will also remove any duplicate transactions. At the end the final transactions and account infomartion is turned into a hash and the block is moved to be solved. A solving command is issued and a new block is created.

2.3.3 Handling Solved Blocks

When a node recieves a solved block it will check if it already has the given block. If it does it will do nothing. Otherwise, it will first verify the given block using the block and hash. Upon recieving the verification it will store the block on its chain and multicast the block to its neighbors.