PROJECT REPORT

ALGORAND

CS620: NEW TRENDS IN IT

DIPTYAROOP MAJI(183050016)

DEBAYAN BANDYOPADHYAY(183050034)

ABHISHEK VARMA(183050038)

02nd May 2019

**INTRODUCTION**

Algorand is a new cryptocurrency that confirms transactions in order of minutes unlike other cryptocurrencies like Bitcoin while at the same time preventing the occurrence of forks. It deals with cases where the network might be partitioned or malicious nodes exist in the network. It reaches consensus by using Byzantine Agreement function. They use randomness in the selection of block proposers while at the same time select random committee members who select the block to be added to the blockchain, thereby ensuring that denial-of-service attack does not take place. Lastly, the system ensures that there is no fixed central system ie. The system is fully distributed avoiding cases like a 51% attack.

**SYSTEM DESIGN & SETUP:**

**NETWORK**

* 256 Nodes
* Each node has 2 to 4 neighbours randomly chosen
* Delays
  + Block message: max{0, N (200 ms, 400 ms)}
  + Non-block message: max{N (30 ms, 64 ms)}
  + N (µ, σ) is a Gaussian distribution with mean=µ and standard deviation=σ

**CRYPTOGRAPHIC FUNCTION**

* Each node has a public-private key pair generated using ECDSA
* SHA-256 used for hashing blocks

**Simulator:** We used as a priority queue (heapq module in python) for making the simulator. Events are pushed into the priority queue, where priority is given based on the current timestamp. The events entered with the same timestamp are considered to be parallel (although they are executed one by one). Events once executed are removed from the queue.

**Events:** We created a class for events. Events have the following variables:

* timeStart: The time when the event starts
* src: The source for the event
* dest: The destination for the event
* timeEnd: The time at which the event would end.
* timeout: If any event is triggered whose timestamp is greater than the timeout value set, then it is simply discarded
* action: The action that needs to be done in the event
* msg: The message that will be sent in that event. It is an object of the Message class defined in code

**Global State:** We created this class for creating a global state that is common for all the nodes. It contains code for:

* Functions
  + terminate(self): Ends the program
  + startBAstar: Starts the BA\* event for all the nodes
  + cleanup: Clear the buffer
  + setNonBlockDelay: Sets the delay for non-block delay
  + assignInitialStake: Assigns initial stake to each of the nodes
  + storePublicKeys: Store the Public keys of the nodes in a common buffer
  + incrementSeqID: Increase the sequence id to deal with conflicting events with same start time
  + addTime: Adds value to current time
* Variables
  + globalState = "init"
  + seqID = 0
  + time = 0
  + numNodes = int(sys.argv[1])
  + blockDelay = []
  + nonBlockDelay = []
  + pubKeyDB = {}
  + genesisMsg = "We are building the best Algorand Discrete Event Simulator"
  + totalStake = 0.0
  + t\_proposer = 5
  + t\_step = 5
  + t\_final = 5
  + lambda\_proposal = 3
  + lambda\_block = 30
  + lambda\_step = 3
  + blockchain = []
  + blockchain.append(Block(self.genesisMsg, 0))
  + lastBlockIndex = 0
  + pubKey\_list = []
  + roundNum = 0
  + T = 2/3
  + lock = Lock()
  + blockInserted = False
  + failStopParameter = 10
  + byzParameter = 5

**Node:** This class contains the details for each of the simulated nodes in the network.

* Functions
  + send() - to send a message to the neighbors
  + recv() - receiving messages
  + sortition()
  + gossip()
  + checkIfCommitteeMember()
  + waitForPriorityProposals()

**Message:** This class contains the details about the message being sent with the class containing two variables:

* Message: The actual message
* NodesVisited: A set of nodes visited (It is used to prevent loops during the broadcast, getting the contents of the message and directing it where to go.)

**Work done**:

* Gossip protocol: Nodes can successfully exchange messages using the gossip protocol implemented. It adds send and receive events which allow gossiping to take place.
* PRG: PseudoRandomGenerator is implemented using SHA-512 using the randomness of the hash functions.
* Cryptographic sortition: Sortition functions as desired and the hash value generated by it can be successfully validated.
* Block Proposal: Proposer nodes can be selected easily using sortition functions and blocks can be proposed by the selected proposers.
* Committee election: Correct sortition ensures that committee is correctly elected.
* Voting: After the proposal, the committee can successfully vote for a block.
* BAStar: Reduction and Binary BAStar with maximum steps set to 10

Experiment 2.3: Included fail-stop adversary who does not propose block/take part in consensus

**Work to be done:**

1. Experiment 2.4 -> currently byzantine nodes only proposing 2 blocks at once, and communication delay within byzantine nodes is set to 0. Remaining part to be done
2. Majority of the blocks in the blockchain are empty blocks. Need to see why is it so and whether tuning any parameters would decrease the number of empty blocks. For example, increasing the no. of neighbors for each node have reduced the no. of empty blocks slightly.