Distributed Ledger Simulator

Prakhar Agrawal (2014CS10207)
Harsh Arya (2014CS10221)

May 8, 2017

Abstract

This report presents the overall design, implementation details and important design decisions made in the implementation (submitted along with this report) of the simulator for simulating a distributed ledger system (like the one used in many P2P networks). We also present a brief description of the files and what they stand for or are used for during the simulation of the distributed system. Though the implementation was used in the simulator, the code in many parts of the implementation is generic enough to be used in a real time implementation one such network with a appropriate changes of course. The report has been divided into sections for ease of navigation and understanding.

1 Overall design

The implementation has been divided into two packages. To be precise: 1. Distributed Ledger part (package node) 2. The Simulator (package simulator). The package node implements all the necessary interfaces and logic required by the distributed ledger (hereafter referred as DL) while the Simulation or the simulator logic has been separately kept in the simulator package. This has been done for modularity of code and ease of logical understanding.

2 Implementation details

The implementation details are discussed briefly in this section. Below is the structure of the go packages showing the classes in the individual packages that make up the underlying implementation:

package node
- *node.go*
- *transaction.go*
- *message.go*

com.simulator
- *Simulator.go*

Now we briefly discuss what each the function of each class is:

- **Node**: Node is the representative of an individual computer system connected by network to a distributed system setting. Each individual node is independently run on a separate thread. These threads are spawned by the main thread that runs the simulator of the program. It contains basic node information.
- **Message**: This file defines a struct that defines the structure of the message sent between the nodes.
- **Transaction**: This file defines a struct transaction which states the form of transactions that two nodes perform.
- **Simulator** This program helps us run long simulations and watch the results stabilize. It will start multiple nodes and then can also force quit them if we do not want wish the nodes to stop.
- **Checker**: This program helps us verify whether output is correct by checking the transaction order at each node.

3 References

- Slides by Prof. S.R. Sarangi: `http://privateweb.iitd.ac.in/~srsarangi/docs/dist/ft.pdf`

- https://bitcoin.org/en/how-it-works
- https://en.wikipedia.org/wiki/Virtual_synchrony
- https://en.wikipedia.org/wiki/Blockchain