Fall17 CS271 Project2

October 21, 2017

For project 2, we would like you to implement the Distributed Snapshot Algorithm. We all have bank accounts and do transactions everyday. When you start a transfer to another account, the money will be on the fly. It has been deducted from your account but has been not posted on the other account yet. Your application should be able to take a snapshot of the global state, which means all the money in the accounts and on the fly.

1 Application Component

We will assume multiple clients. Each client holds an account and have 2 operations:

- Transfer money.
- Start a snapshot.

2 Implementation Detail Requirements

1. Each client should start with some amount money, say 1000. You can implement the transfer function as an Automatic Transfer. For example, you can set up something like every 10 seconds, each client has 0.2 probability to send a random amount of money to any of the other clients by sending a message to the other client. In order to detect messages in transit, incorporate some delay after the send comand is issued.

2. Each client should be able to start a snapshot at anytime. A snapshot should be initialized by entering a command.

3 User Interface

1. When starting a client, it should know the ip and port of all other clients (You can keep clients connected all the time or connect them when they need to talk). You should have a configuration file that contains all the

ip, port and account balance (To be simple, all clients can start with the same balance)

2. You should log all necessary information on the console for the sake of debugging and demonstration, e.g. Client XX sent $ZZ dollar to Client YY. Client YY received $ZZ dollar from Client XX. Also, output the current balance when sending or receiving any money.
   When a snapshot has terminated, output the balance of each account, the money on the fly ($XX is being sent from Client XX to Client YY) and the total amount of money (sum the account balance and the money on the fly). The sum should always add up to the same amount for all snapshots, since the overall money in the system is constant.

3. You should add some delay (e.g. 5 seconds) when transferring money (starting a snapshot does not need any delay). This simulates the time for message passing and makes it easier for demoing concurrent events.

4. Use message passing primitives TCP/UDP. You can decide which alternative and explore the trade-offs. We will be interested in hearing your experience.

4 Demo case

For the demo, you should have 3 to 5 clients. Initially, each client should output their name and balance:

Mark, $1000

Then each client should start to transfer money to each other with a certain probability periodically.

You should be able to start a snapshot from any client by typing a command. For example :

snapshot

5 Deadlines, Extension and Deployment

This project will be due 11/06/2017. We will have a short demo for each project For this project's demo, you can deploy your code on several machines. However it is also acceptable if you just use several processes in the same machine to simulate the distributed environment.

6 Teams

Projects should be done in team of at most 2. You can use Piazza to form teams.