Distributed System project: Global Snapshot

May 17, 2016

Ardino Pierfrancesco Alex D.e Biaso

In this project we were asked to implement a distributed bank application capable of transferring money between remote branches of a bank. Each branch has a TCP connection with the others branches. Each branch has to send a certain amount of money, randomly chosen, to another random branch. The project consists in these files:

- `start_bank.py`: it accepts as parameter the IP address of the host machine. His role is to create a process for each branch of the machine declared in the file **hostlist**
- `host.list`: contains the list of the branches, every host has the same file in order to have a global view of the network. Every branch is identified by a tuple composed by the IP address of the machine, the socket port of the machine and whether he has to start the global snapshot.
- `bank_main.py`: this script simply start a branch instance passing to **bank.py** the information of the branch taken from `host.list`.
- `bank.py`: this class is responsible for both money and snapshot handler, provides functions to start the snapshot and decides the amount of money to be transferred and to which branch send the money.
- `bankinterfaceout.py`: this class provides the interface with another branch, sends the money to a given branch and checks whether the money has been delivered with *at most once* semantics. If transfer fails for three times, it will report to `bank.py` that somethings gone wrong and so to not withdraw the money.
- `clean_logs.sh`: this script remove the folder `logs` and its content.
- `check_logs.sh`: this script checks and prints the content of the logs.

1 Implementation

1.1 Initialization

At the beginning, the branch is launched by the `start_banks.py` script with the IP address of the machine as argument. This script create a new process that instantiates a new `Bank` class for each different branch on the machine. After a period of initialization each branch is connected to all the others branches. Each branch will attempt to connect with the others branches every two seconds, until all the network is connected.

1.2 Money exchange

After the initialization procedure, each branch, after a period of time governed by a uniform variable between 2 and 5 seconds, spawns a thread which contains an infinite loop. In this loop it tries to acquire a lock on the amount of money owned by the branch. After having successfully acquire the lock it chooses randomly an amount of money and a branch, then it tries to send

this amount to the other branch through the `bankinterfaceout` class. If the operation finishes successfully the thread withdraws the money from the total amount otherwise it does not and finally releases the lock.

1.3 Global snapshot

The branch that has to start the Global Snapshot, spawns also another special thread. This threads periodically starts a new Global Snapshot. It tries to acquire the lock on the total amount and once it succeed, it records the local state and it sends to every branch a snapshot token. All the other branches process this token following the Global Snapshot protocol described during the lectures.

1.4 Safety guarantees

Using Locks in the functions that modify the money of a branch guarantees that a branch can not send and receive money in the same time. The same happens if a branch wants to start a new snapshot. It will acquire the lock, saves the current amount of money of the branch and then releases the lock. Tcp guarantees also FIFO ordering and the automatic retransmission of a packet and the management of acknowledgments. The money are withdrawn from the account of a branch only if the Tcp socket has successfully sent the packet. Using the option `MSG_WAITALL` on the receiver side, guarantees that the receiving socket waits until all the bytes will be received.

2 Testing

2.1 Network failures

Without network failures, the snapshot will eventually finish. In presence of network failures during a snapshot, it never terminate since a branch will wait forever for the delivery of a token from the others branch. The snapshot will continue when the failures will be fixed.
Network failures were simulated by removing the Ethernet for a couple of seconds and then reconnecting the cable and finally checking the log of the snapshot in order to check that no money had been lost.

2.2 Crash Failures

In the context processes crashes, the application was not designed to tolerate them. In presence of crash failures the snapshot will never terminate. The software on each branch has to restarted and unless there are snapshots already completed the money will be lost.

3 Launching example

3.1 Single machine

In this execution each branch is running on the same machine, the content of `host.list` is the following:
127.0.0.1:8000:S
127.0.0.1:8001
127.0.0.1:8002
127.0.0.1:8003
Thus four branch with the first that will start the snapshot.
The application is launched with the following command:
./start_bank.py 127.0.0.1

3.2 Multiple machines

Following there is an example with two machines, one with IP 10.0.0.1, the other with 10.0.0.2, the content of `host.list` is the following:
10.0.0.1:8000:S
10.0.0.1:8001
10.0.0.1:8002
10.0.0.2:8003
10.0.0.2:8004
10.0.0.2:8005
The application is launched on the two machines with the following commands:
Launch start_bank on host 1 with:
./start_bank.py 10.0.0.1
Launch start_bank on host 2 with:
./start_bank.py 10.0.0.2