Deepika Rajani
MN 64573

CMSC 621: Advanced Operating System

Centralized Multi-User Concurrent Bank Account Manager

DESIGN APPROACH:

- The project is implemented through socket communication to connect multiple clients with the server.
- The server indefinitely calls accept system call to accept the client requests.
- When a client connection is accepted, server creates a thread for that client. One thread is created for each client.
- The thread created continuously reads on the client socket for upcoming data from the client.
- Client after connecting with the server sends a transaction request (either a deposit / withdraw) to the server.
- Server receives the request and performs validation checks like insufficient balance before withdrawing and after performing the action, sends the response back to the client.
- All the transactions sent by the client and received at the server end are properly logged on server and client console.
- Client waits after sending each transaction request for the time difference between two consecutive requests timestamp multiplied by the timestep.
- Multiple clients when executed together may try to perform operation on the same account which may lead to inconsistent balance in the account. To avoid such situation, locking is applied on valid accounts.
- Before any operation of withdraw or deposit is performed for any account, the section of critical code is locked for other transactions on the same account. Lock is released after completion of the operation, so that next operation on that account can start.
- Two transactions on different accounts can happen simultaneously.
- When client do not have any more data to send, the thread created at server for that client ends.
- Server has a thread for calculating interest on each account periodically. The interest is calculated at the rate of .0001% every 10 minutes. Interest calculation is also protected by mutex lock.

COMPILATION AND EXECUTION

- Following are the steps of execution through the makefile.
- **make clean**: it cleans the .out file by executing rm -rf *.o compile
- **make compile** : Complies the server and the client source codes creating the executable files server.out and client.out
- Run server : **./server <port> <path_to_records_file>**
- Run client : **./client.out <machine_addr> <port> <timestep> <path_to_transaction_file>**

Format of the input files:
Records.txt -> <account_number> <account_holder> <balance_in_account>
e.g., 101 Chang 10000
Transaction.txt file format -> <timestamp> <account_number> <operation> <amount>
e.g., 5 101 d 1200

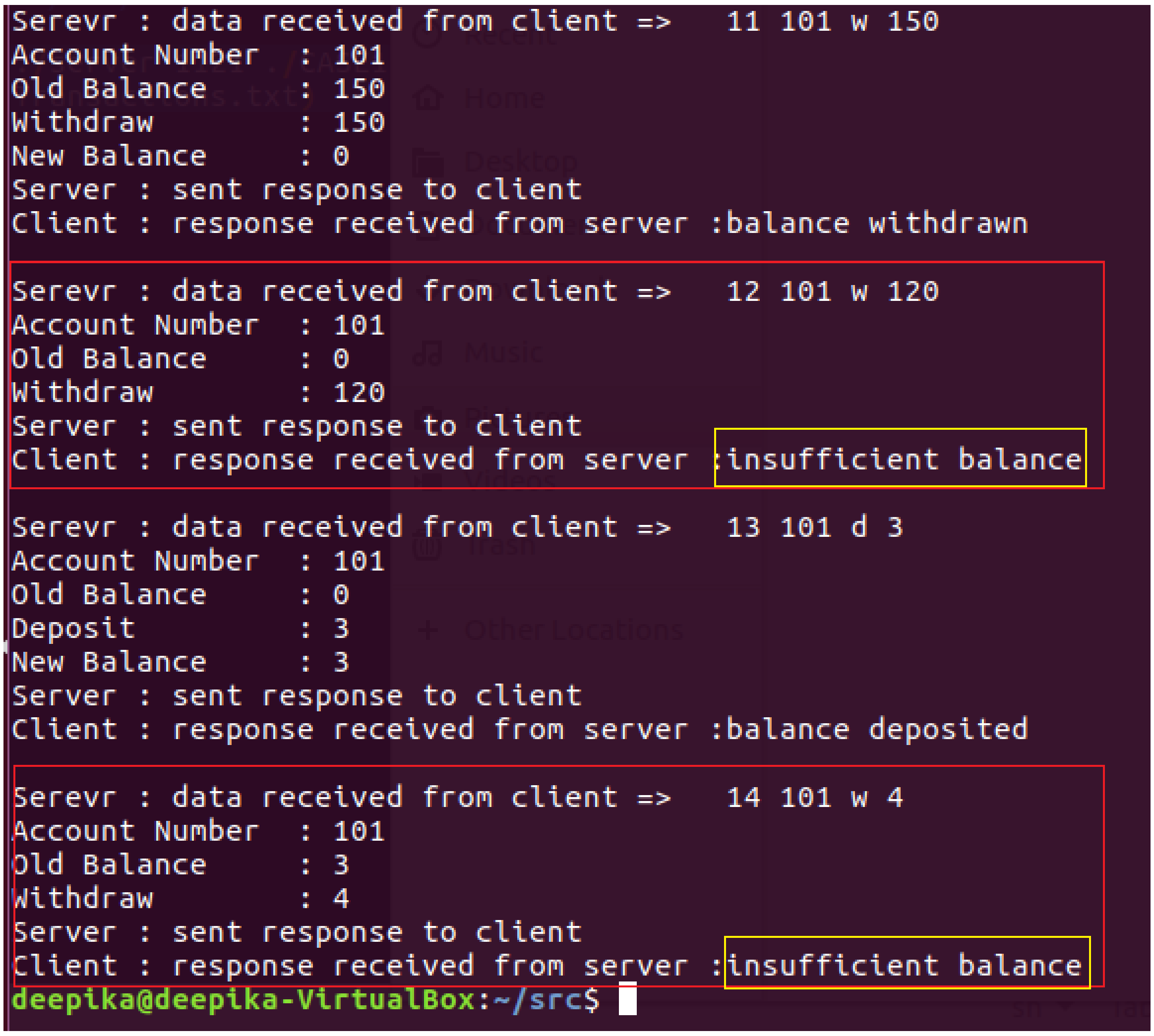
POSSIBLE ENHANCEMENT

- The project can be enhanced by providing some additional functionalities like new account creation.

- The data is currently not updated inside the file after balance is updated. As a future improvement, data can be persisted inside the file.
- Validation can be applied on the amount being withdrawn or deposited, like amount less than 1 cannot be deposited.

CORRECTNESS EVALUATION

- **Server processes transaction by deducting money only when available otherwise sends message as "insufficient balance"**



```
Serevr : data received from client =>   11 101 w 150
Account Number   : 101
Old Balance      : 150
Withdraw         : 150
New Balance      : 0
Server : sent response to client
Client : response received from server :balance withdrawn

Serevr : data received from client =>   12 101 w 120
Account Number   : 101
Old Balance      : 0
Withdraw         : 120
Server : sent response to client
Client : response received from server :insufficient balance

Serevr : data received from client =>   13 101 d 3
Account Number   : 101
Old Balance      : 0
Deposit          : 3
New Balance      : 3
Server : sent response to client
Client : response received from server :balance deposited

Serevr : data received from client =>   14 101 w 4
Account Number   : 101
Old Balance      : 3
Withdraw         : 4
Server : sent response to client
Client : response received from server :insufficient balance
deepika@deepika-VirtualBox:~/src$
```

- **Server accepts multiple clients request simultaneously, protects simultaneous access to the same account**.

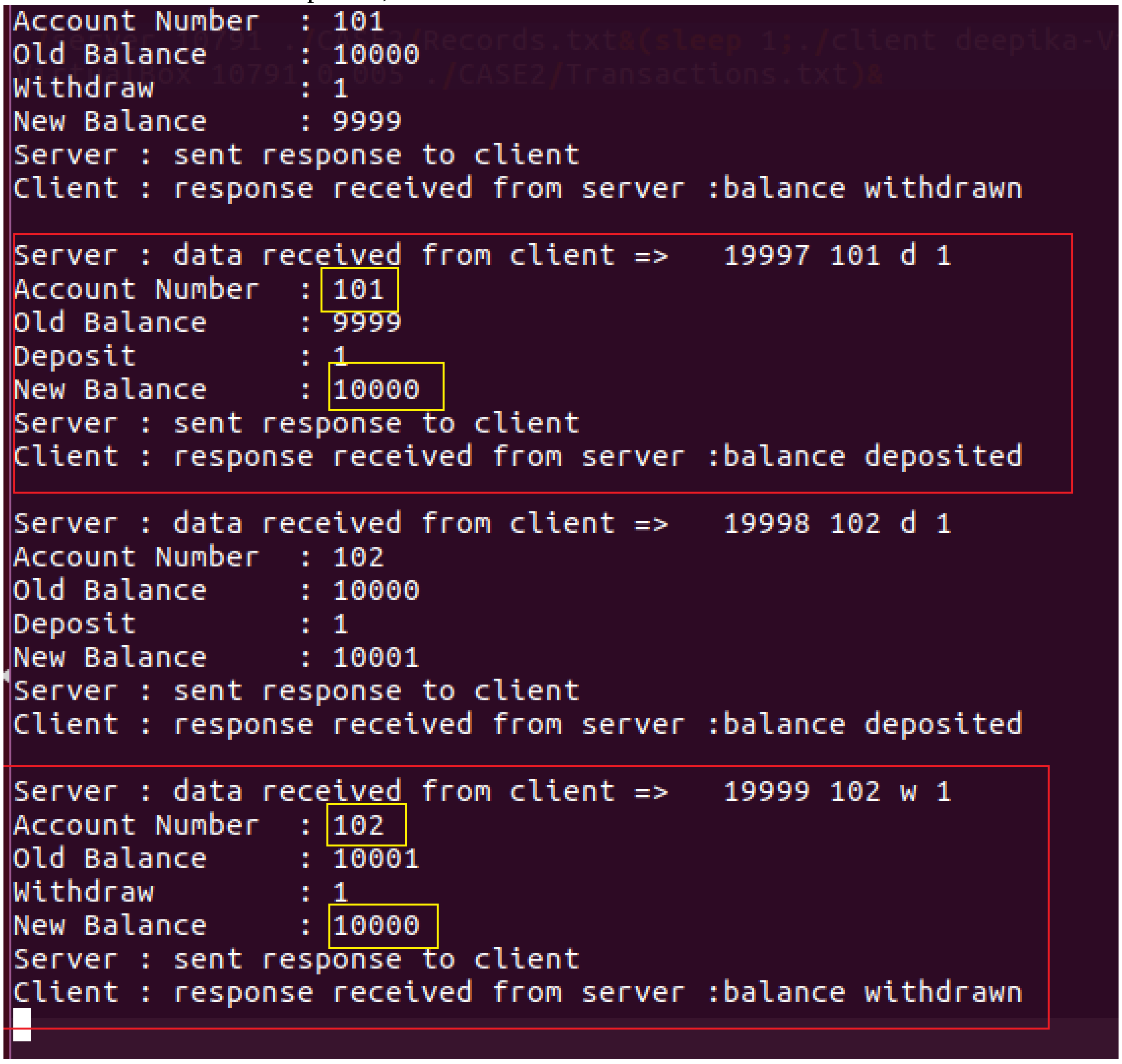
Records.txt:
101 Peter 10000
102 John 10000

Deepika Rajani
MN 64573

Executed **4 clients** each executing **20000 number of transactions**. Transactions file has equal number of withdraw and deposit transactions for both the account.

Expected: Account balance to remain same after client execution, as equal amount is being deposited and withdrawn from them.

Evaluation: Is same as expected, as shown below



```
Account Number   : 101
Old Balance      : 10000
Withdraw         : 1
New Balance      : 9999
Server : sent response to client
Client : response received from server :balance withdrawn

Server : data received from client =>   19997 101 d 1
Account Number   : 101
Old Balance      : 9999
Deposit          : 1
New Balance      : 10000
Server : sent response to client
Client : response received from server :balance deposited

Server : data received from client =>   19998 102 d 1
Account Number   : 102
Old Balance      : 10000
Deposit          : 1
New Balance      : 10001
Server : sent response to client
Client : response received from server :balance deposited

Server : data received from client =>   19999 102 w 1
Account Number   : 102
Old Balance      : 10001
Withdraw         : 1
New Balance      : 10000
Server : sent response to client
Client : response received from server :balance withdrawn
```