THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

Hong Kong Polytechnic University

Department of Computing

COMP 5325 Distributed Computing

Assignment Four

LEI Haiwen 16105237g

Wong King Yu 17001226g

Siu Yui Wing 17001615g

Wu Zongheng 17077449g

Description of Requirements

This project is to develop a simulation of a reliable client/server system that provides fault-tolerant services. Invocations of servers are carried out via remote procedure call or remote method invocation. The fault tolerance is achieved by employing standby redundancy: the main server will be replaced by a standby copy when the main server fails.

Function module design

System architecture

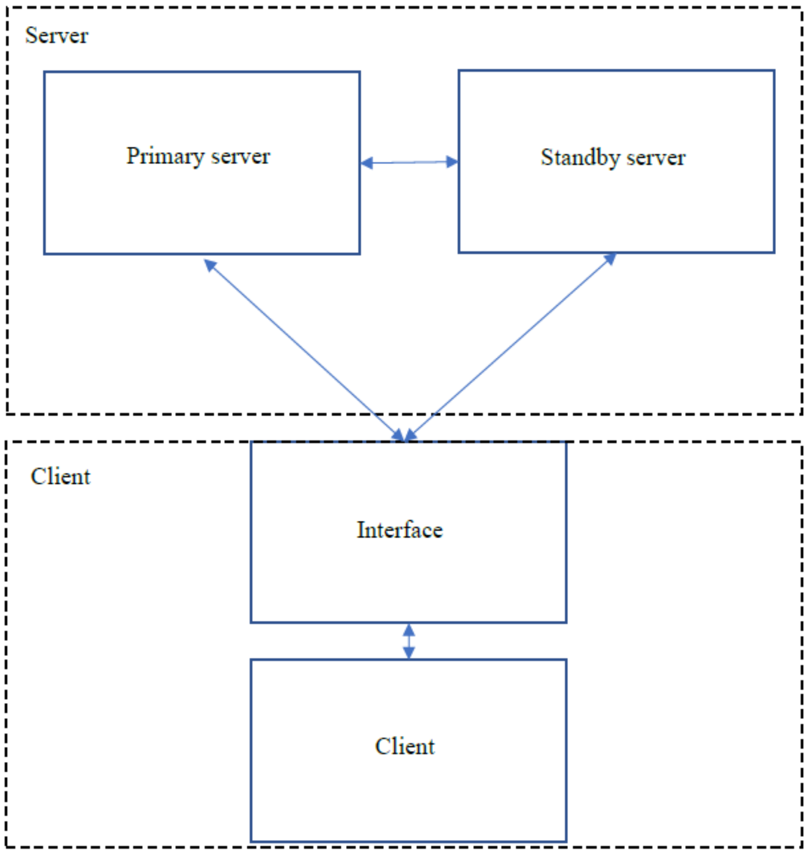


Fig. 1

In our project, an interface hides the server details from the clients. A service is a set of well-defined operations, and a server is an instance of a particular service running on a single machine. When requesting a service, a client only needs to specify the nature of the service through an attributed name. The exact location of the service and the server that provides such a service will be determined by the interface.

In server side, two servers are maintained; one is called the primary server, and the other is called the standby server as shown in Fig. 1. The primary and standby are located on different machines to provide the fault-tolerant service. In order to guarantee fault-tolerant service, at least one server is alive. When a client requests a service, the request is always first sent to the primary server if it is alive and response to the client; otherwise, the client switches to connect to the standby and selects the standby server to a primary server. The result returned by either copy of the server will be passed back to the client.

Server fault-tolerant function

Initially, both copies servers have the identical state which the data is same and the action id is zero. The primary server periodically sends synchronization messages with newest action id to the standby server, which then updates its own state to keep it consistent with the state of the primary.

Two servers maintain a file of account balance and allow the clients to access it. The account balance file contains lines of the format: account-id, balance. Account-id is a numerical number and balance is a non-negative integer value. The amounts of money in withdraw and deposits are a non-negative integer value.

There is a possibility that the primary crashes in the middle of its computation before it completes the service of the request.

In case, the primary sends its action id, which represent its state with the balance content to the standby server before it goes down. As the standby receives the message with balance content, it compares its action id with the one in the message, if the incoming actionID is larger, it updates it to the new one, also the balance content.Meanwhile, the interface cannot receive the response from the primary, it will try to connect to the standby, the standby becomes the primary.

For example, the balance content is like "{'acct1':100,'acct2':200}", with initiated action id 0, the interface sends request to withdraw acct2 by 50 with initiated action id 0. Primary server will process withdrawal. In this time, balance will be update to "{'acct1': 100,'acct2':150}" with action id 1 in server, which will be sent back to interface as response and standby server as checkpoint .After the standby server receives the message, it will compare the actionID with the one contains in the message, if its larger, it will update its balance content to "{'acct1':100,'acct2':150}", also, the actionID to 1. After interface receives response, it updates the action id to the one response contained.

Afterwards, primary server is down and interface send another withdraw request as same as previous one with a updated action id 1. Since interface could not receive response from primary server, it will send such request to standby server without notifying the client.

While standby server receives that request, standby server will compare if the client's action id is greater then itself. If client's action id is larger, it means that the standby server does not complete the synchronization yet. Therefore a reject message is responded to interface until it completes its synchronization process. In opposite, when the client's action id is smaller or equal, they are under the consistency state and the withdrawal will be process. In this time, balance will be updated to "{'acct1':100,'acct2':100}" with action id 2 in standby server, which will be sent back to interface as response.

The flow chart is shown in Fig. 2.

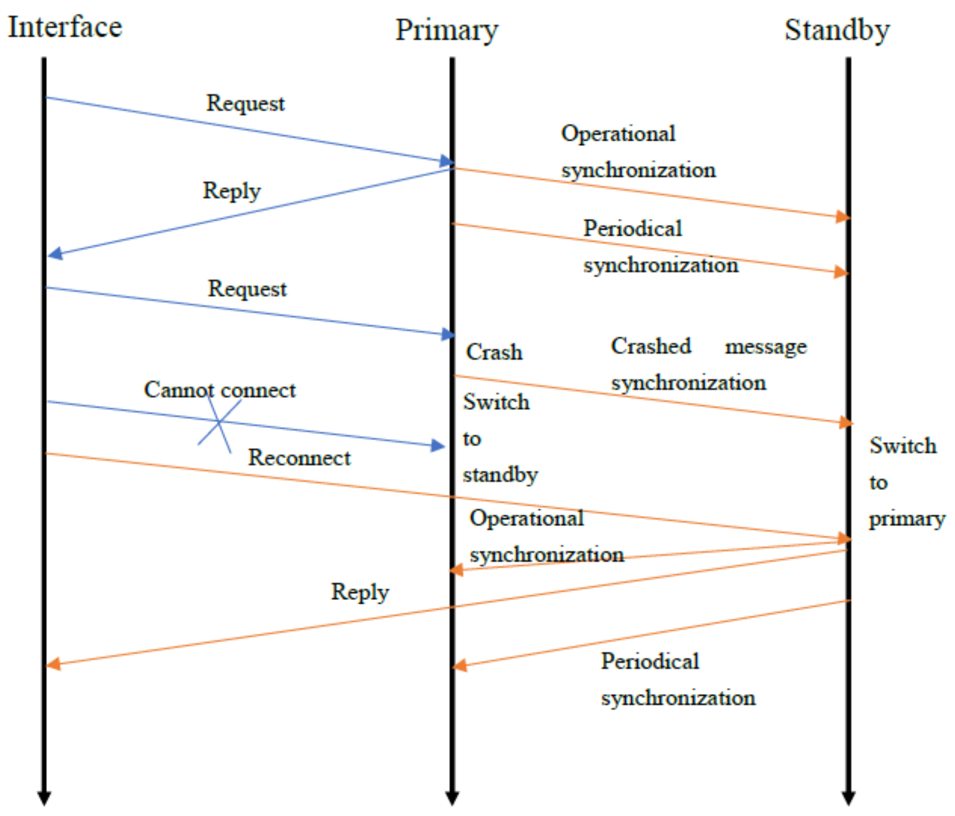

Interface
Primary
Standby
Request
Operational synchronization
Reply
Periodical synchronization
Request
Crash
Crashed message synchronization
Cannot connect
Switch to standby
Reconnect
Switch to primary
Operational synchronization
Reply
Periodical synchronization

Fig. 2

Interface function

The interface program repeatedly displays the following menu of options on the screen:

1. Look up balance
2. Withdraw an amount of money
3. Save an amount of money
4. Terminating program

For option 1, the client inputs account id in interface and interface send it to server. The server opens the account balance file and searches the account id. If it finds a match, the server sends result to the interface. The interface prints the corresponding balance. If the server searches the entire file without finding a match, it returns a message indicating that the account is not in the file and the interface prints 'Account not found'.

For option 2 and 3, the client inputs account id and an integer value in interface and interface send them to server. The server opens the account balance file and searches the account id. If it finds a match, the server updates the balance accordingly by the client's request, and then sends result to the interface. The interface prints the corresponding balance. If the server searches the entire file without finding a match, it returns a message indicating that the account is not in the file and the interface prints 'Account not found'. The balance of an account must be non-negative all the time. If the execution of an operation violates this property, the operation should be aborted.

After execution, the interface program clears the screen and redisplays the master menu. This cycle continues until the client chooses the last option, which terminates the program.

Deployment detail

Network requirement: Within the same subnet network.

Software requirement: Python3 and gRPC must be installed.

Install the grpcio-tools package:

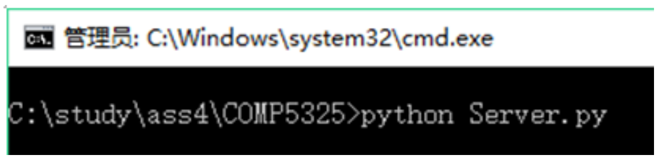
pip install grpcio-tools

User manual

Server configuration

In each server, serverhost. config must be configured to {"s1":"this server IP address", "s2":"the other server IP address"}.

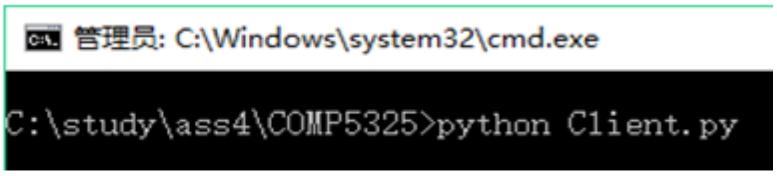
After configuration, run ***python Server.py*** in COMP5325 folder in each server.



```
管理员: C:\Windows\system32\cmd.exe
C:\study\ass4\COMP5325>python Server.py
```

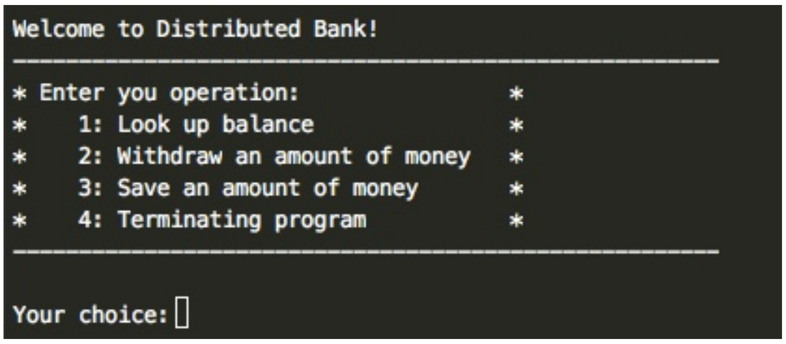
Run interface

In client's computer, run ***python Client.py*** in COMP5325 folder.



```
管理员: C:\Windows\system32\cmd.exe
C:\study\ass4\COMP5325>python Client.py
```

Choose option in menu.

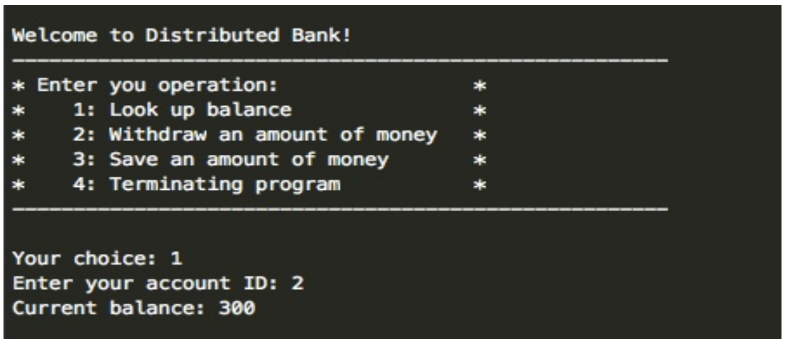


```
Welcome to Distributed Bank!
------------------------------------------------------
* Enter you operation:                          *
*     1: Look up balance                        *
*     2: Withdraw an amount of money            *
*     3: Save an amount of money                *
*     4: Terminating program                    *
------------------------------------------------------

Your choice:
```

Look up balance

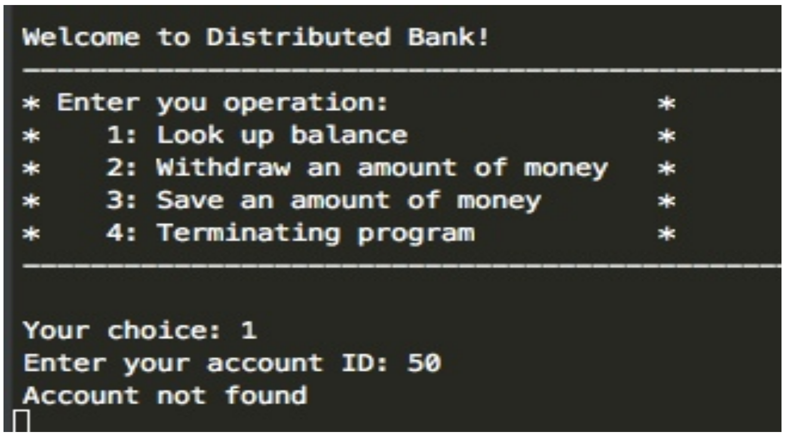
If you want to look up balance, choose 1 and then input your account id.



```
Welcome to Distributed Bank!
------------------------------------------------------
* Enter you operation:                      *
*     1: Look up balance                    *
*     2: Withdraw an amount of money        *
*     3: Save an amount of money            *
*     4: Terminating program                *
------------------------------------------------------

Your choice: 1
Enter your account ID: 2
Current balance: 300
```

If you input an account id not in the database, it will output ' Account not found '.

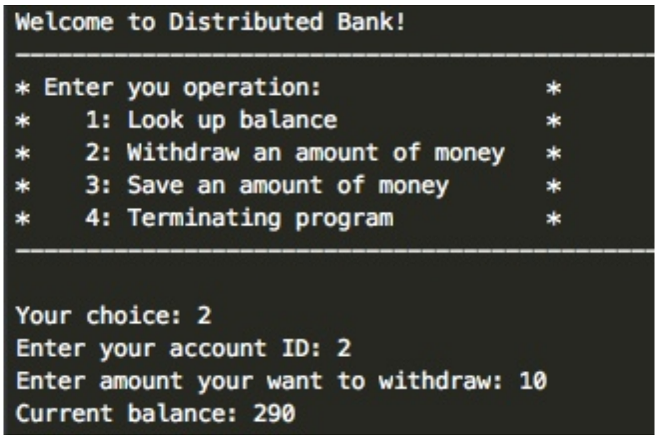


```
Welcome to Distributed Bank!
------------------------------------------------------
* Enter you operation:                      *
*     1: Look up balance                    *
*     2: Withdraw an amount of money        *
*     3: Save an amount of money            *
*     4: Terminating program                *
------------------------------------------------------

Your choice: 1
Enter your account ID: 50
Account not found
```

Withdraw money

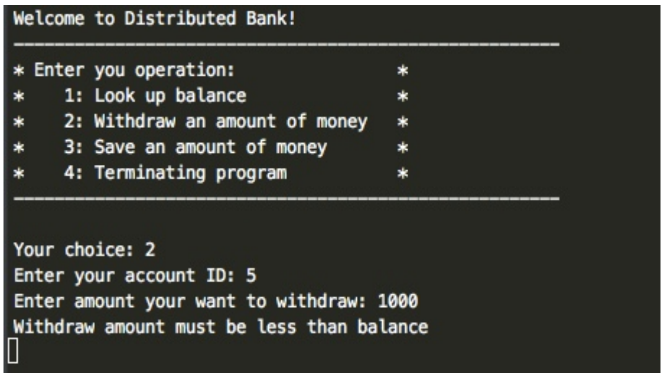
If you want to withdraw money, choose 2 and then input your account id, then input the amount you want to withdraw. After finishing withdraws, the interface will output the current balance.



```
Welcome to Distributed Bank!
------------------------------------------------------
* Enter you operation:                      *
*     1: Look up balance                    *
*     2: Withdraw an amount of money        *
*     3: Save an amount of money            *
*     4: Terminating program                *
------------------------------------------------------

Your choice: 2
Enter your account ID: 2
Enter amount your want to withdraw: 10
Current balance: 290
```

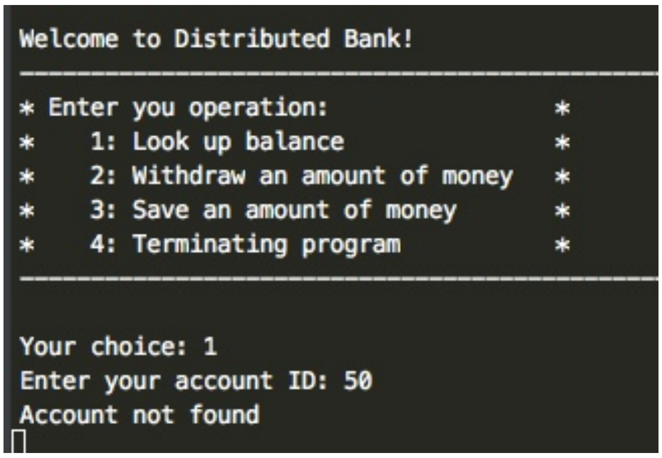
If you input amount of money more than balance, it will output ' Withdraw amount must be less than balance '.



```
Welcome to Distributed Bank!
--------------------------------------------------
* Enter you operation:                  *
*    1: Look up balance                 *
*    2: Withdraw an amount of money     *
*    3: Save an amount of money         *
*    4: Terminating program             *
--------------------------------------------------

Your choice: 2
Enter your account ID: 5
Enter amount your want to withdraw: 1000
Withdraw amount must be less than balance
```

If you input an account id not in the database, it will output 'Account not found'.

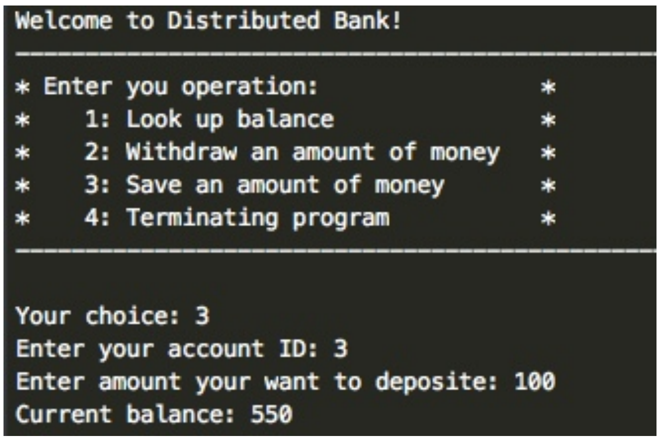


```
Welcome to Distributed Bank!
--------------------------------------------------
* Enter you operation:                  *
*    1: Look up balance                 *
*    2: Withdraw an amount of money     *
*    3: Save an amount of money         *
*    4: Terminating program             *
--------------------------------------------------

Your choice: 1
Enter your account ID: 50
Account not found
```

Save money

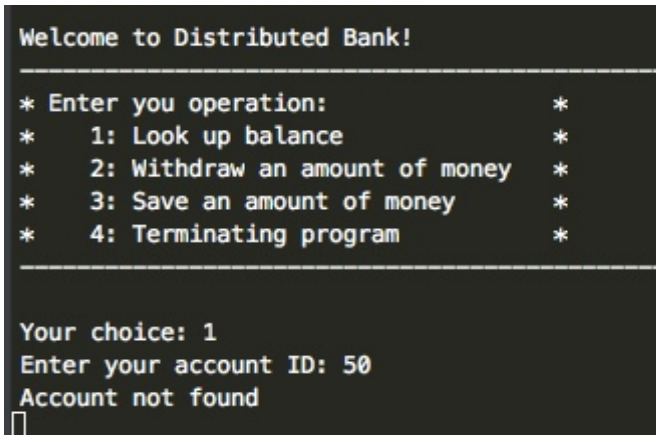
If you want to save money, choose 3 and then input your account id, then input the amount you want to save. After finishing deposit, the interface will output the current balance.



```
Welcome to Distributed Bank!
--------------------------------------------------
* Enter you operation:                  *
*    1: Look up balance                 *
*    2: Withdraw an amount of money     *
*    3: Save an amount of money         *
*    4: Terminating program             *
--------------------------------------------------

Your choice: 3
Enter your account ID: 3
Enter amount your want to deposite: 100
Current balance: 550
```

If you input an account id not in the database, it will output 'Account not found'.

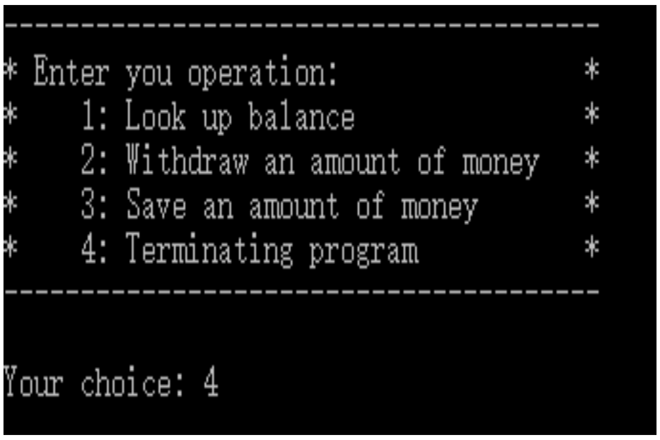


```
Welcome to Distributed Bank!
-----------------------------------------------
* Enter you operation:                        *
*     1: Look up balance                      *
*     2: Withdraw an amount of money          *
*     3: Save an amount of money              *
*     4: Terminating program                  *
-----------------------------------------------

Your choice: 1
Enter your account ID: 50
Account not found
```

Exit interface

If you want to exit, choose 4.



```
-----------------------------------------
* Enter you operation:                  *
*     1: Look up balance                *
*     2: Withdraw an amount of money    *
*     3: Save an amount of money        *
*     4: Terminating program            *
-----------------------------------------


Your choice: 4
```