

## PROJECT 3 (CMSC 621)

By: Bhanu Phanindra Gorantla

In this project I designed distributed banking service that supports multiple concurrent operations using pthreads and fault tolerance using redundancy of data. As required three process are created as servers that run on same machine(local) and capable of handling multiple requests (multiple clients) in parallel. A separate process called front end is created and used to communicate with clients that is it acts as intermediate process between servers(back end database) and clients.

Two Phase Commit(2pc) protocol is used to maintain atomicity between three servers. Changes are made to data when commit is issued successfully. True concurrency is achieved by creating one thread for each client in front end and front end in turn creates three threads for each client to backend servers. Mutex variable is created for every record. Therefore threads are blocked only when same account is updated in parallel. Instead of using separate acknowledgement signal output generated when given command is executed is taken as acknowledgement. To test fault tolerance part, during execution of program one program can be stopped by using (ctrl+z) keys. When new command is issued client is notified that his transaction has been aborted and can be retried if necessary and when retried will be successful if no other process is stopped.

Note: Use only (**ctrl+z**) to stop the process. Using close like ctrl+c will terminate program and bring up socket exceptions. Port numbers of three servers are hard coded in front end servers. If port number already in use error is displayed try closing and reopening the terminal.

The following are contents of make file, which shows how to compile each program.

#This target is used to compile cpp files

compile:

```
g++ back.cpp -o back -pthread -std=c++11 -w
```

```
g++ frontEnd.cpp -o front -std=c++11 -pthread -w
```

```
g++ client.cpp -o client -std=c++11 -pthread -w
```

#This target is used to clear object files

clean:

```
rm -rf *.out back front client
```

Make compile is used to compile and generate all object files. Make clean is used to remove all object files. First backend processes must be created then front end followed by clients must be created.

Arguments taken by each process is shown as below.

//creating backend servers

./back [port no] [client\_count]

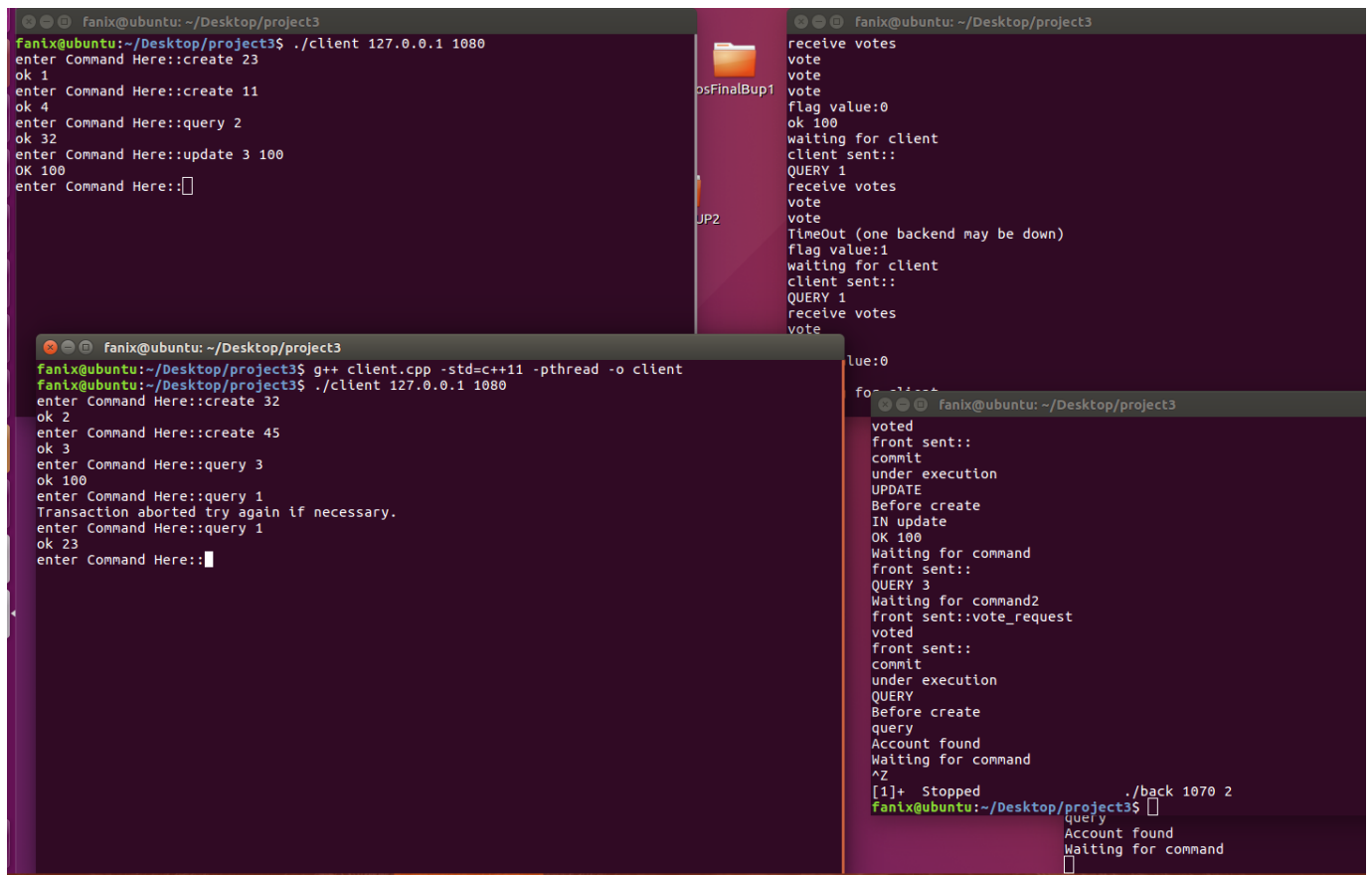
Eg: ./back 1050 1

./back 1060 1

./back 1070 1

./front 1080 1

./client 127.0.0.1 1080 (client takes port number of front end and additionally takes local host ip address as argument)



```
fanix@ubuntu: ~/Desktop/project3
fanix@ubuntu:~/Desktop/project3$ ./client 127.0.0.1 1080
enter Command Here::create 23
ok 1
enter Command Here::create 11
ok 4
enter Command Here::query 2
ok 32
enter Command Here::update 3 100
OK 100
enter Command Here::

fanix@ubuntu:~/Desktop/project3$ g++ client.cpp -std=c++11 -pthread -o client
fanix@ubuntu:~/Desktop/project3$ ./client 127.0.0.1 1080
enter Command Here::create 32
ok 2
enter Command Here::create 45
ok 3
enter Command Here::query 3
ok 100
enter Command Here::query 1
Transaction aborted try again if necessary.
enter Command Here::query 1
ok 23
enter Command Here::

fanix@ubuntu:~/Desktop/project3$ ./back 1070 2
receive votes
vote
vote
vote
flag value:0
ok 100
waiting for client
client sent::
QUERY 1
receive votes
vote
vote
Timeout (one backend may be down)
flag value:1
waiting for client
client sent::
QUERY 1
receive votes
vote
vote
voted
front sent::
commit
under execution
UPDATE
Before create
IN update
OK 100
Waiting for command
front sent::
QUERY 3
Waiting for command2
front sent::vote_request
voted
front sent::
commit
under execution
QUERY
Before create
query
Account found
Waiting for command
^Z
[1]+  Stopped                  ./back 1070 2
fanix@ubuntu:~/Desktop/project3$
query
Account found
Waiting for command
```

Fig: Screenshot of programs in execution

The above shows different cases that arises in project. The left two terminal at top and bottom are clients. There are two clients. Firstly account 1 is created in terminal1(top left in above picture) and account 2 and 3 are created in terminal2. But it could be seen that when account 2 is queried in terminal1, it successfully returns value. Also after backend with port 1070 is stopped using ctrl+z combination(bottom right in above picture) and query 1 command is sent we get response saying “transaction aborted try again if necessary” and when tried again it successfully executed.