Design Document -Fault tolerant Banking Application

In this system, a simple, distributed banking service that permits multiple concurrent operations and exhibits simple fault tolerance is designed and implemented. Three servers are run, each as a separate process on a machine. Each server is capable of handling multiple simultaneous requests, and any account is accessible from any server whether or not it was created on that server.

Additionally, the group of servers tolerate the loss of any one server in the group without impacting the overall service. A process is killed to emulate a crash. To handle this, the data is replicated between servers.

Servers maintains just enough state as to complete the transactions. For locking, P_thread's mutex locking mechanism is used. Two-phase commit protocol ensures Atomicity.

Server implementation uses a multi-tiered architecture. It includes a front-end process that the client always connects to. It is assumed that the front-end process never crashes. The front end process then accesses data or updates data on the three servers (those are the backend processes). The locking mechanism is implemented in the front-end process .

TRANSACTION PROTOCOL

SESSION

The protocol used to communicate with the servers is a TCP service on some specified port(s).

Upon accepting a new session, and at the completion of each client command, the server returns a string with the following format:

status output_string CRLF

status

Either "OK" on connection or success, or "ERR" if an error in processing occurs.

output_string

A carriage return (ASCII 0x13) followed by a line feed (ASCII 0x10).

COMMANDS

All commands entered by the client are case-insensitive and may be terminated by either a line feed or a carriage return followed by a line feed. The following commands are supported:

CREATE *initial _deposit*
Creates a new account containing the amount specified by *initial_deposit*, which should be a real value with a precision of no more than 2 digits. On success the output is a unique non-negative integer ID for the account. On error, the output is an error message.

UPDATE *acct_id value*

Sets the amount in the account identified by the non-negative integer *acct_id* to the value specified by *value*, a real value with no more than 2 digits of precision. On success, outputs the new value. On error, outputs an error string.

QUERY *acct_id*

Queries the current amount in the account specified by the non-negative integer *acct_id*. On success, returns the current amount as a real value with 2 digits of precision. On error, returns and error message.

QUIT

Terminates the session. No output string is needed.

SAMPLE SESSION

```
% ./client <frontend_server_port> <frontend_server_ip>
OK
CREATE 50.00
OK 100
QUERY 100
OK 50.00
UPDATE 100 150.00
OK 150.00
UPDATE 101 20.00
ERR Account 101 does not exist.


2
QUERY 100
OK 150.00
QUIT
OK
Connection closed by foreign host.
```