**Group:** Livar Cunha, Luciano Mandryk, Sergio Clemente

# Compiling & Executing

Just run ./compile.sh or ./execute.sh

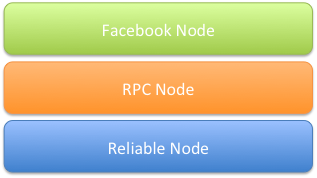
If you ran into any issues make sure you set your java home to something like the following:

|  |
| --- |
| [~] JAVA\_HOME=/Library/Java/JavaVirtualMachines/1.7.0.jdk/Contents/Home/bin  [~] export JAVA\_HOME |

Also, make sure the classes folder exist.

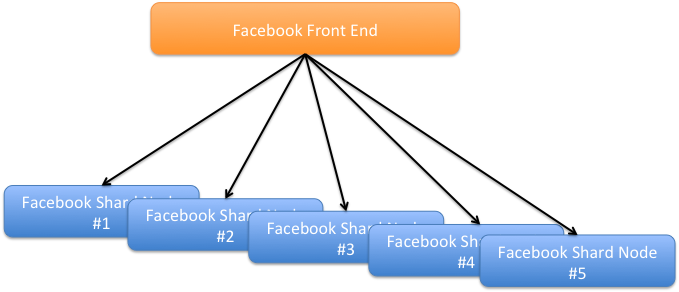
# Facebook System

We reused some of the code of assignment1. For example, the reliable message delivery system was kept intact. We changed our RPC node in order to have a cleaner API for returning values (We noticed we were duplicating too much code to handle exceptions or return values). The basic class architecture of the node is given below.



The main idea is that the reliable node provides reliable message delivering. The RPC node allows to define and call functions with a well defined API. The Facebook Node contains the logic of this assignment.

Let’s digg more into the Facebook Node which contains the logic of the assignment. The facebook nodes are divides into two roles as can be seen below:



* Front End: Contains logic that routes the request to the proper shard. For example, in our assignment we hardcoded the number of shards to 5.
  + Shard#1: Responsibles for storing the information of users that starts with letter a-e
  + Shard#2: Responsible for the range f-j
  + Shard#3: Responsible for the range k-o
  + Shard#4: Responsible for the range p-t
  + Shard#5: Responsible for the range u-z

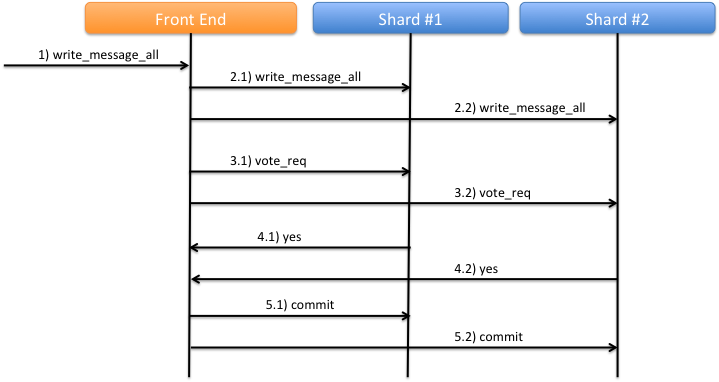
Node: We are assuming the users are evenly distributed. We could use a hashcode of the userId + some salt to make the distribution more independent to the login bug it was not required per the specification.

The front end node also contains the logic to coordinate the 2 phase commit protocol (More details below).

* Shard Node: Contain a slice of the data (Given a particular range)

# Two Phase Commit

The two phase commit logic was written outside of the node and both the frontend node and shard nodes call them separately. We just implemented the two phase commit in the write\_message\_all operation. The details is as follows:



When the write\_message\_all is received, the frontend starts the 2pc context and sends the write\_message\_all to all shards. After all of them reply saying that the write either succeeded or failed, the frontend sends either the commit or abort to all of them.

PS. To make the system correct, we needed to implement the 2pc also in the friend\_request (since it touches two shards).

# Example of running

The below example creates the frontend and shard nodes (As we mentioned below, they are hardcoded from 0-5) and adds users a,k and z. adds some friends and them broadcasts some message.

|  |
| --- |
| start 0  start 1  start 2  start 3  start 4  start 5  0 create\_user a a  0 create\_user k k  0 create\_user z z  0 login a a  0 login k k  0 login z z  0 add\_friend a;1234 z  0 add\_friend a;1234 k  0 accept\_friend z;1234 a  0 accept\_friend k;1234 a  0 write\_message\_all a;1234 message to k and z |