# Chat Application

Sukriti Gupta(2016CS50084) Chinmay Rai(2016CS50615)

## 1 Handling Arbitrary Disconnection

In the Scenario of the user disconnecting arbitrarily by pressing Ctrl+C, we can use a Runtime method, addShutdownHook, which provides a way of trapping a shutdown of the java virtual machine. The method expects one argument and that is a Thread object. This thread will be started when the virtual machine terminates. This will be called whether the program exits normally or with Ctrl+C, we will need to add some way to distinguish between the two scenarios. A simple if check on a flag variable can work (flag set to true on normal exit at the end of UNREGISTER)
The Following snippet shows how to handle Interrupt caused by a signal.

```
// A.java
import java.lang.System;
import java.lang.Runtime;

public class A {
    public static void main(String args[]) {
        Runtime.getRuntime().addShutdownHook(new Thread() {
          public void run() {
              System.out.println("Exited!: Cleanup code can be executed here");
          }
       });

        try{
            Thread.currentThread().sleep(1000);
       }
        catch(InterruptedException ie){}
      System.out.println("Normal Exit");
    }
}
```

The above mentioned solution is language specific. A more generic solution could be the use of **heartbeat** messages. These heartbeat messages will be sent by the server to the (online) client at regular intervals. The client will keep on acknowledging the heartbeats. As long as the server hears the heartbeats it knows that the client is online. If there is no response, the server can close the socket.

## 2 Handling offline Users

In order to deal with offline users we can have a sign-up procedure that initializes an Inbox data structure for each user which will contain information like:

```
Struct msg{
    String receiver;
    String Header;
    String content;
}

Struct inbox{
    String username;
    boolean is_online;
    Socket incoming_socket, outgoing_socket;
    Buffered_Reader inFromServer, inFromClient;
    DataOutptuStream outToServer, OutToClient;
    msg [] toBeDelivered;
}
```

This data structure is saved in a registry of users irrespective of their offline/online status. When the user is comes online (log-in), we look up his Inbox from the registry and update information like the is_online, sockets and the I/O streams. We also deliver all the outstanding messages that are pending in his Inbox.
When the user goes offline (log-out), is_online is set to false, the sockets and the streams are set to NULL. Messages for a user who is offline simply stored in the msg list 'toBeDelivered' of the user's Inbox.
When the server receives a message for a specific user, it will first check if that user is registered:

- If yes, then it checks if the user is online:
  - If Yes then the message is directly forwarded to the user.
  - If No then the message is stored in Inbox of the recipient.
- If No, then it sends an error to recipient: "INVALID RECIPIENT".

## 3 2-Factor Acknowledgement

The purpose of single tick in whats App is the indication of successful delivery of message to the server, while the double ticks indicate the receipt of message by the receiver. The Current form of acknowledgement that we are using is equivalent to a double tick (since we assume the users are always online). In order to implement single tick acknowledgement, we will add an extra ACK signal into the protocol, which will be sent by the server to the sender, after it successfully reads in the message that is to be forwarded. This means the sender will receive two acknowledgements:

- Single Tick : The first acknowledgement will be sent by the server, after it receives the message from sender
- Double Tick : The Second acknowledgement will be sent by the receiver (through server), on receipt of message. This form of acknowledgement is already implemented (if users are always online). In case users may go offline, we send this ACK when the receiver comes online and the toBeDelivered message queue is emptied. As and when the msg is dequeued, the sender can be sent an ACK.

# 4 Base64 encoding usage

Many text-based protocols use Base64 out of convenience: It's an established standard and it's efficient enough for most applications. For base 64, the general strategy is to choose 64 characters that are common to most encodings and that are also printable. This combination leaves the data unlikely to be modified in transit through information systems, such as email, that were traditionally not 8-bit clean. The base64 mode allows us to encode arbitrary binary data so that intermediate mail agents do not mingle them up.
There are systems that may corrupt the raw source bytes. This also lends readability to the data.

# 5 Corner Case Handled

No sender is kept waiting infinitely. The way the server is designed, in case the message was not sent, an immediate response to the sender is sent by the server