Gaukhar Dauzhan

**REPORT**

The server has a functionality to forward messages to registered clients. Clients can send messages and register for getting messages from other clients (either with some specific hashtags or all messages).

***Structure design:***

1. Storage of tags and registered clients

2. Basic structure explanation

a) server

b) client

3. Specific situations handling

4. Race conditions

**1. Storage of tags and registered clients**

In the server there is a linked list of tags sorted alphabetically. For each linked list, there is a file descriptor set of registered clients. There is a separate file descriptor set regAll for the clients who want to get all messages. There is also a separate file descriptor for the clients that are working with images (sending or receiving).

**2. Basic structure explanation**

***a) Server***

Server starts from reading the first 5 letters from the client in order to handle first situations where bytecount is provided by the user: MSGE and IMAGE

a) When a client sends an encrypted message, server reads first until “/” to identify tag(if it exists) and bytecount. Then server calculates number of bytes in total that he needs to accept (its bytecount + bytes before the message) and stores them in a buffer. After that it searches for all the clients registered to tag in the linked list (LL), sets client’s fds to 1 and does the same in a separate fd set of clients registered to all (regAll).

b) When a client sends an image, server creates a thread for handling it (images take much longer time to be processed than text messages). In a thread, as with MSGE, server identifies number of image bytes and then reads them in a loop – 2048 bytes at most in iteration into the big buffer (size=imgbytes+preceding bytes). It stops reading when the amount of bytes read is equal to the expected value. Then it sends the image to the interested clients.

Server has an fd set of clients working with images (fdlock). If it is set to 1, no other messages can be accepted from this client.

If it’s not MSGE or IMAGE, then it just reads until \r\n

c) If it is MSG, it just forwards the message to interested clients (the way is already described)

d) If it is REGISTER tag, first server searches for the right place in the linked list, then it sets corresponding bit to 1 indicating that the client will now get all the messages with this hashtag

e)If it is DEREGISTER tag, server searches for the tag in the linked list and sets corresponding bit to 0

f)If it is REGISTERALL, server just activates its fd in the regAll fd set

g)If it is DEREGISTERALL, server deactivates its fd in the regAll fd set

***b) Client***

a) A client adds /r/n to the end of all correct messages from user except MSGE

b) If the message starts from MSGE, it counts number of bytes in msg, adds this to the message and encrypts msgbytes

b) There is a thread waiting for messages. If MSGE comes, it identifies number of bytes, reads the whole msg and then decrypts it

**3. Specific situations**

a) When a client is currently in the status of registered to all, but sends REGISTER sometag message, server ignores this message. Only deregisterall can deactivate REGISTERALL status

b) When a client is currently in the status of registered to some particular tags and sends REGISTERALL, server first deletes it from the fdset of the tags he was registered for, then adds to a separate fd set regAll

**4) Race conditions**

In order to prevent several users to change linked list structure at the same time, there is a mutex lock that prevents other users to access linked list while others modify it. There