CNLINE Group 22 Report

0. Environment & Language

- Machine: Linux 64 bit
- Language: Python 3

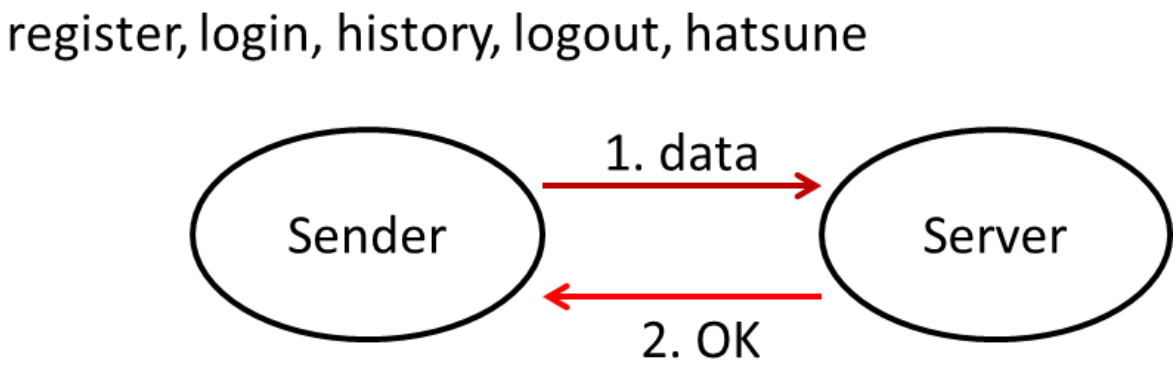
1. Members & Team Work

Student ID	Name	Work
B03902005	林日能	Basic server implementation, Miku chatting system implementation (Bonus) and the report
B03902107	鄭格承	Client implementation, multiple file transfer implementation, password encryption (Bonus)

2. Protocol Specification

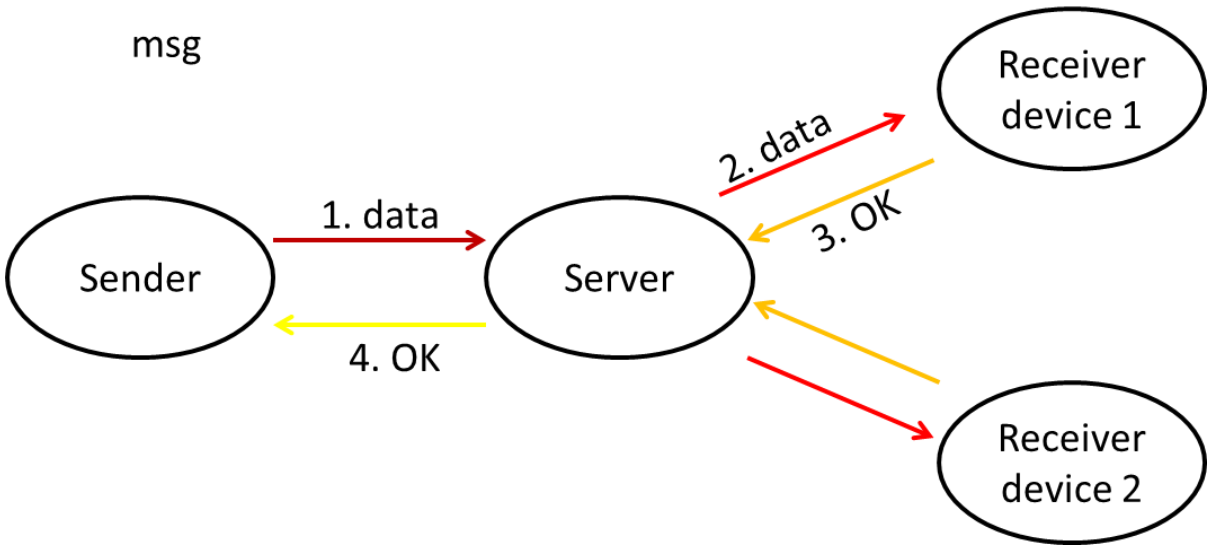
Our protocol between client and server is implemented through dictionaries. According to their behaviour, we divide our protocaol into 3 type:

Type 1: the protocol for register, log in, log history, log out and Hatsune Miku



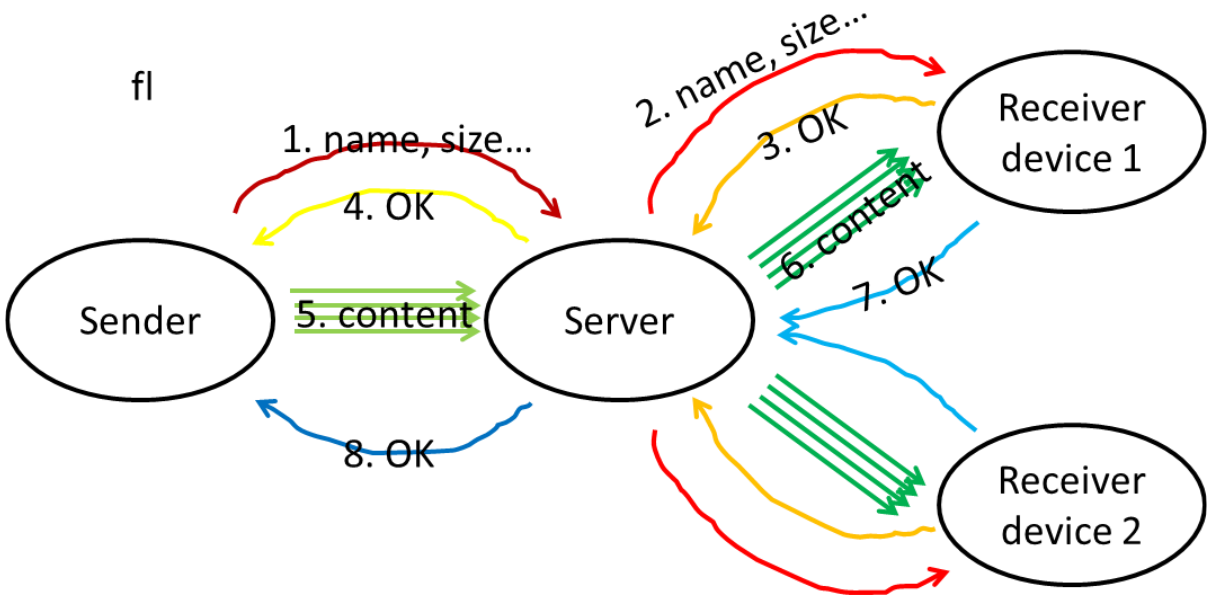
The sender sends a dictionary, which includes type of action (value of key ‘action’), the origin (value of key ‘from’), the destination (value of key ‘to’), the timestamp (value of key ‘time’) and the critical information (value of key ‘body’) to the server. The critical information for history is the desired log history, but is not applicable for other actions. The server then sends an acknowledgement back to the sender.

Type 2: the protocol for sending message



The sender sends a dictionary to the server. The message is packed in the body. Upon the reception, the sender directly transmit the data toward the receiver. Note that if an account is logged in multiple devices, the server needs to handle multiple receivers. After the receiver receives the message, it sends out an acknowledgement to the server, and the server transmits it back to the sender.

Type 3: the protocol for file tranmission



Phase I.
The sender sends a dictionary, which specifies the file's name (value of key 'name') and length (value of key 'length'). Upon the reception, the sender directly transmit the data toward the receiver. Note that if an account is logged in multiple devices, the server needs to handle multiple receivers. After the receiver receives the message, it sends out an acknowledgement to the server, and the server transmits it back to the sender.

Phase II.

At this phase, both the sender and the receiver knows how many bytes of data the file is. The sender sends exactly the content of data, 4,096 bytes at a teime, to the receiver. The receiver receives exactly the file's number of bytes of data, 4,096 bytes at a time, from the sender. Once finished, the receiver sends out an acknowledgement to the server, and the server transmits it back to the sender.

3. User & Operator Guide

A. Initialization

Before running a server for the first time, type the following command to clean up (set up) the storage and members' ID-password information at server's side:
sh initialize.sh

For the users, if you are connecting to the server for the first time, write the IP address and port to a file named 'ServerID' at client's root directory, like this:
echo XXX.XXX.X.X:XXXX > client/ServerID
(The IP address and port has printed out when a server was set up)

B. Execution

For the operator, type the following command to run a server:
python server/server.py

For a user, type the following command to run a client:
python main.py
And our command line UI will kindly guide you to use our CNLINE. No worries!

Hidden features:
After logged in, type
[smsg] miku: [your message here]
Hatsune Miku will show up and respond to you!

Moreover, type
[miku] [client ID here]
Then the message sent from that client ID will be perceived as that from Hatsune Miku! You may imagine you are talking with Miku-chan!
If you want to cancel this configuration, type the same command again.

C. Termination

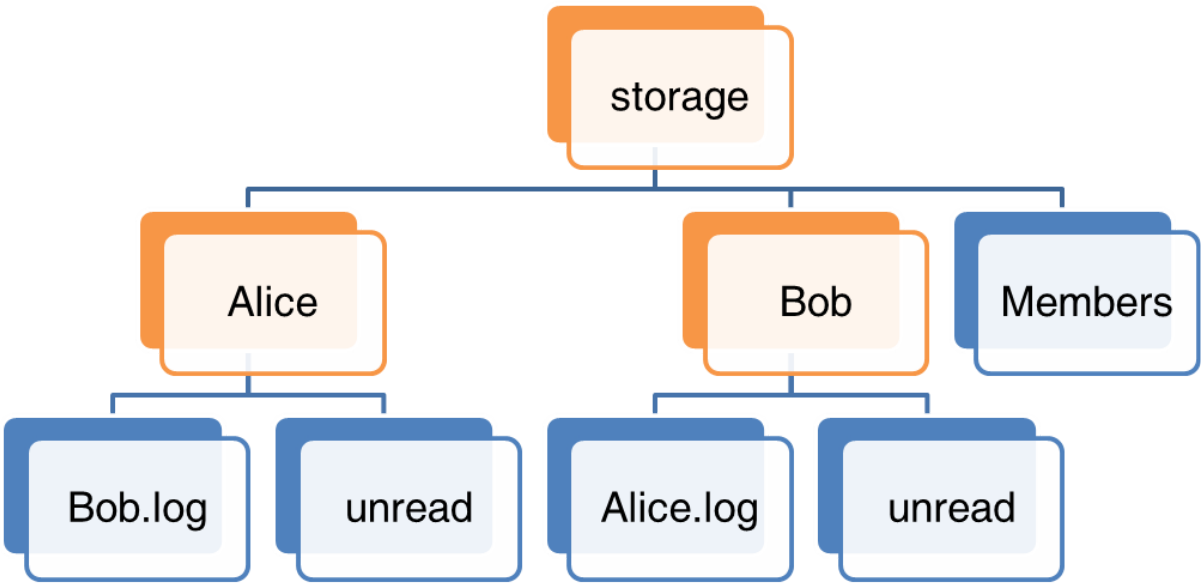
For a user, type 'logout' for logging out and 'leave' for closing the user process.
For the server, type ctrl + C to shut down the server.

4. System & Program Design

A. Socket Connection

The server is always-on, has a main thread handling incoming connections and distributing the requests to other threads to handle.
When a client sets up a socket connection with the server, the server remembers the socket which it logged in. The clients may create multiple sockets to parallelly transmit data to the server, but the server only sends back response messages to the first socket.

B. Storage Hierarchy



About our storage hierarchy at the server, we store a list of all the ID-password pair and a list of all members at the file storage/Members. Under the directory storage, a member has its own directory after registeration. Under a specific user's directory, there are logs between the user and others if they have had a conversation. An unread file is set up along with the user's directory, storing all messages that were sent during its offline period.

C. File Transfer System

About our file transfer system, we do not send files to clients which are offline. But if the client were online, it is forced to download the files whoever sent to them. If the client happens to have a file with the same name at its side, the new coming file is renamed to [name]_1, [name]_2, [name]_3… sequentially. And all the file transfer history are recorded in the log file, regardless of they were succeeded or not.

D. User Interface

Our UI is in command line. Although it may look simple, it is the best way to represent ASCII art. Our bonus part (Hatsune Miku) takes the form of ASCII art, actually.

Side note: Our system is capable of sending messages and files to yourself.

5. About Our Bonuses

We have two bonuses, adding a little fun to our project.

1. Auto-responding Agent

We all like to communicate with people. But if there are some times when nobody to chat with, how about chatting with a program? Hatsune Miku is always there to give you some heart-warming response messages, if you are willing to chat with her! Although we do not have enough time to develop an intelligent agent to chat with you, it is still nice to have some pre-scripted messages!

2. User Tranformation

There are always some circumstances that you do not want to chat with specific people. Instead of ignoring or cold-responding to their messages, why not transform those users into Hatsune Miku? You will have fun chatting with them then!

3. Password Encryption

We send, receive, save user's password under SHA224 ciphertext. We assure the security of our users while choosing their passwords!