Computer Networks 2018 Final Project: CNLine

B05902083 資工三 余柏序
B05902073 資工三 張庭與

API:
Server:

- **Success: send “AC <additional info>”**
- **Fail: send “WA <error message>”**

Basic commands

Signup

- Client: send “signup <username> <password>”
- Server:
  - “AC”
  - “WA”

Login

- Client: send “login <username> <password>”
- Server:
  - “AC <session_token, 16-digit hexadecimal>”
  - “WA”

Objects:

- message_id: 16-digit hexadecimal
- message_content (json format):
  {
  from: sender username,
  to: receiver username,
  text: message text (heximal encoded, possible chars are: [0-9a-f])
  }
  Example:
  {from:burney,to:octopos_tea,text:e4bda0e79c9fe79a84e698afe58f88e99bb7e58f88e4a8bbe880b6efbc8ce8aab0e58fabe4bda0e784a1e8818ae8a7a3e9968be98099e5808be8a88ae681afe79a84efbc9f}

User-related command (Bonus Feature)

Online Users:

- Client: send “online_users <session_token>”
- Server:
  - “AC <number of users> <username 1> <username 2> ...”

Friends:

- Client: send “friends <session_token>”
- Server:
    - “AC <number of users> <username 1> <username 2> ...”

Add Friend:

- Client: send “add_friend <session_token> <friend username>”
- Server:
    - “AC”
    - “WA”

Remove Friend:

- Client: send “remove_friend <session_token> <friend username>”
- Server:
    - “AC”
    - “WA”

Messaging Commands

Send Message

- Client: send “send_message <session_token> <partner username> <message>”
- Server:
    - “AC <message_id>”

Check Message (用next_messages來模擬)

Get Last Message(older)

- Client: “last_message <session_token> <partner username>”
- Server:1
    - “AC <message_id>”

Get Next Messages(newer)

- Client:
    - “next_messages <session_token> <message_id> <desired number of messages to get>”
- Server:
    - “AC <number of messages, might be 0> <message_id 1> <message_id 2> ...”
    - “WA”

Get Previous Messages

- Client: “prev_messages <session_token> … (和next_messages一樣)”
- Server: (和next_messages一樣)

Get Message:

- Client: “get_message <session_token> <message_id>”
- Server:
    - “AC <message_content>”
    - "WA"

File Transmission Commands

Procedure:

1. If A want to transmit a file to B
2. A send file_request to server with file_size info
3. server waits for B to respond
4. B check whether there's a file request regularily
5. server received B's check_file_request
6. server generates a file_id for this transmission
7. server replied file_id to A and B
8. A send send_file request to server with file_id & entire file content
9. B send receive_file request to server with file_id
   =====file transmission started=====
10. server replied to B with entire file content, at the same time, server is still receiving file contents from A's send_file request
11. server received file_size bytes from A, replied "AC" to A and close A's socket
12. server transmitted file_size bytes of file content to B, close B's socket
    =====file transmission ended=====
13. Done

Objects:

- file_id: 16-digit hexadecimal

File Request:

- Client: "file_request <session_id> <receiver username> <file_size> <file_name>"
- Server:
    - "AC <file_id>"
    - "WA other_side_no_response"

Check File Request:

- Client: "check_file_request <session_id>"
- Server:
    - "AC <file_id> <sender username> <file_size> <file_name>"

Send File:

- Client: "send_file <session_id> <file_id> <entire file content>"
- Server:
    - "AC"
    - "WA"

Receive File:

- Client: "receive_file <session_id> <file_id>"
- Server:
    - "AC <entire file content>"
    - "WA"

Bonus Features:

Online Check:

- Client: "online_check <session token> <username>"
- Server:
    - "AC online"
    - "AC offline"
    - "WA" + errorMsg

Encryption

- Password (registration/login)
    - Simple hashing of password.
    - Prevents original password leakage.
    - But does not prevent MITM attack.
- Message
    - Client sends and receives encoded messages
    - Server processes encoded messages

Auto reconnect

- Client will stay slive and wait until reconnected to server.

Advanced File Transmission

- Saves resources of server

Client Structure

Main thread

- main (logout state main menu)
    - main_login (login state main menu)
        - friend list
        - online list
        - chat for each list
            - input messages and commands
    - login page
    - registration page

TCP sender thread

- tcpSender
    - queued requests, queued responses
    - communitation with main thread with queues and mutex locks

Chat thread

- send requests via TCP sender thread
    - get older/newer message
    - get message content
    - get current (last) message
    - render messages on screen

File handler thread

- handles file requests etc.
- file sending & receiving multiplexing

TCP sender thread for files

- TCP sender/receiver for file transmission only

Server Structure

Main thread

- Server's initialization, open socket for listening incoming requests
- Load saved user data if existed
- Accepts requests from multiple client, opens threads to server them

Save thread

- Started after server finishing initialization
- Save server's data to disk every second

Client thread

- Created and runs whenever any requests from clients are accepted
- Deal with clients' various requests and decide how to respond them
    1. signup
    2. login
    3. online_users
    4. friends
    5. add_friend
    6. remove_friend
    7. get_message
    8. send_message
    9. last_message
    10. next_messages
    11. prev_messages
    12. file_request
    13. check_file_request
    14. send_file
    15. receive_file
    16. online_check

- Responsible to maintain user's data so that they're all sound and up-to-date
- Deal with conflicting requests between clients
- Sync and interact between clients, if necessarily, by carefully controls the inter-client workflow
- Mutex is used to prevent race condition, while excessive locks should be avoided. Different requests may require different locking policies
- Sends detailed error messages back to those clients who requests something abnormal, while prevents leaking of sensitive infos by checking the permissions of the session_tokens they provides
- Generates access tokens for those clients who needs to access sensitive or temporary data

DataBase

- Stores user's data & temporary tokens
- Provides interface for Client thread to create, access, or modify user's data, temporary tokens, etc.
- Provides interface for Save thread to dump non-temporary data to disk, or recovery them from disk