COSC1176/1179 – Network Programming (NP) Assignment (2019 Semester 1)

Aims: Apply the learnt networking concepts to a practical application

Instructions

- This assignment contributes 25% to your final marks.
- There are two options, **Task A** and **Task B**. You only need to **choose either** A **or** B.
  - Task A (includes A1 and A2) is an individual work.
  - Task B is a group based (max. 3 members).
- Due date: 23:59pm 18th May 2019.
- Please register in the following spreadsheet to show which option you have selected.
https://docs.google.com/spreadsheets/d/124gejcRYOcJN2iFY_oYyhy1vXJGSRp_AoOGxg8VquYA/edit?usp=sharing
- Your program will be marked on the Netprog1 and Netprog2 server. You must demonstrate your solution in the lab to get it marked (No demo, no marks).

Submission

- Submission via Canvas. Please zip all your documents, and use your student number, e.g., s3255601, as the file name for submission.
- Your submission should consist of following parts:
  (i) All Java source files.
  (ii) A sequence diagram to indicate the communications between clients and server.
  (iii) User manual with screenshot of execution of each step in netprog1 and netprog2 to describe
    - how to compile and run the program in Netprog servers.
    - how the program operates.

Note: You need to insert comments into your program with correct code practice specified in lab 1, as appropriate.

Late submission policy

- There will be a late submission period of 5 working days for this assignment.
- Late submissions will incur a penalty of 10% per day, unless prior arrangements are made with the tutor regarding an extension.
- Submissions that are received after the late submission period expires will not be assessed and will hence receive no mark.

Task A Develop a simple game played via the network.

The game is implemented with separate client and server machines. The player is sitting in front of the client machine (one player per client), and the server is administering the game. You are required to implement the game in two stages:

- **Stage 1**: Single-player game: one client communications with the server. When you have completed stage 1, save a copy of it for submission.
- **Stage 2**: Multiplayer game: you need to modify the server so that it can handle several (up to 6) clients simultaneously.

A.1 Single player version

It is a simple guessing game. The server generates a number between 0 and 9. The client's task is to guess the number (up to a maximum of 4 guesses).

- If the client correctly guesses the number, i.e. the guess number X matches the generated number, the server announces “Congratulation”.
- If the client fails to guess the number. For each incorrect guess, the client gets a clue:
    - The client’s guess number X is bigger than the generated number, or
    - The client’s guess number X is smaller than the generated number,
    - After 4 attempts incorrect guess, the server announces the answer.

A.2 Multiplayer version

The server maintains a lobby queue. Clients are required to register with the server (using their first name) to be added to the lobby queue before they can play. Clients can register at any time.

The game is played in rounds. At the start of each round, the server takes the first three clients from the lobby (or all the clients if there are less than three clients in the lobby), and starts the round.

- First the server announces the name of 3 participants.
- Each player can guess at any time (with their number of guesses tracked by the server).

Once all plays have either:

- Correctly guessed the generated number,
- Reached their maximum guess of 4,
- Chosen “e” to exit from the guess.

The server announces to all the clients the number of guesses for each client.

- Players can then choose to play again (p), or quit (q).
- The players that choose to play again are added back into the end of the lobby queue, and the process repeats with a new round.

Program components

You must use Java for programming. The communication is via sockets. You need to choose the appropriate socket for each task and explain why that socket is the most suitable.

- **The client**

It connects to the server, signs up for each game round, guesses numbers and forwards them to the server. It displays the server’s messages to the player.

- **The server**

It maintains a lobby queue, accepts players’ (clients’) requests to sign up, and selects the players for each round. It also manages the game.

- First it announces the number of players and generates a random number between 0 and 9.
- In the guessing phase, it takes guesses from players, compares it to the generated number and sends the feedback to the player while updating their guess count.
- After all players have completed their game (by correctly guessing/reaching maximum guesses/choosing to escape), the server announces the final rankings based on number of guesses, and allows players to either play again by entering “p” or quit by entering “q”.

Miscellaneous

- A simple text user interface will suffice. The user (player) should be notified about events and messages sent by the server. This includes information sent by the server about other players in the multiplayer game.
- In the multiplayer version, make sure that clients are handled correctly when signing up for the game and starting the game.

Task B Extend the Chat Server Application

This is a group-based task, aims to provide you with project experience.

The Chat Server Application could be found at: http://144.138.102.124/~xiali/rmit/bna/p2papp/

The application contains 3 components

- Chat Server
- Presentation Serve
- Peer – Serv*ent*: a combination of server and client

You may focus on Centralised P2P chat service, and add an advanced function to the Chat Server. The example of the advanced functions may include, but not limited to, as long as it is related to Network Programming.

1) **Checksum** for integrity check on the message and files sending.

   The tasks include:
   - Design a mechanism to implement checksum (e.g., CRC) to ensure the message and file sent between peers has not been altered.
   - Present the checksum number of each message between peers into the presentation server
   - Flag the result of checksum between peers (matched or error)

2) **Proxy server** for the Chat Server being used outside the local network.

   The tasks include:
   - Make the connection passing through http proxy in RMIT
   - Implement the function of proxy server

3) **Cryptographic protection** to encrypt the files sent between peers.

   The tasks include:
   - Investigate one of encryption algorithms
   - Design an architecture which can plug in the chosen encryption algorithm to the Chat Server
   - Fit the encryption algorithms into the Chat Server application.

You may start from:

- Download the application and understand how the it is executed {screenshot each steps)
- Draw class diagram and sequence diagram for the application
- A fully executable version of any of above tasks may get maximum of 5 marks bonus
- Groups need to demonstrate their progress in week 9-11 lectures for claiming bonus.

The End