University of Bahrain
College of IT
Department of CE

ITCE320: Network Programming, S2 2023-2024
Term Project (group work)

Multithreaded News Client/Server Information System

Course instructor: Dr. Mohammed Almeer

Office: S40-2116

E-mail: malmeer@uob.edu.bh

Due: Code, documentation, and demonstration (Sun. 19 May 2024)

Late submission policy: Late submissions are subject to penalties. Two marks will be deducted for missing the deadline, and two marks for every additional day.

Assignment

In this project, learners should build a client-server system that exchanges information about current news. The emphasis in this project is on client/server architecture, network communication, multithreading, APIs, and applying good coding practices.

The system should consist of at least two Python scripts: the server and the client. The server should retrieve news updates from NewsAPI.org via the appropriate APIs, extract the necessary information from the retrieved data, manage connections with multiple simultaneous clients, and respond to client requests.

On the other hand, the client will establish a connection with the server script to retrieve lists of news and sources, and request details about them.

The Server Script

The server should start and be ready to accept TCP connections from clients. It should be able to handle at least three simultaneous connections. Upon establishing a connection with a client, the server will be ready to receive different requests from the client. Based on the client's request, the server will communicate with the appropriate API endpoint and retrieve the data to fulfill the client's requests.

The server should provide a list of records with brief details about the results, and then the client can request further details about one of the given results from the server.

The server should perform the following tasks:

1. Once the server starts up, it should be ready and waiting for clients to connect.
2. It should be capable of accepting connection requests from clients and handling at least three connections simultaneously.
3. Upon accepting a connection, the server should store and display the client's name sent by the client.
4. It should be ready to receive requests from the client.
5. Based on the option selected by the client:
    a. The server should connect to the appropriate endpoint to retrieve the data.
    b. It should save the data in a JSON file named "<group_ID>_<client_name>_<option>.json" for testing and evaluation purposes.
    c. The server should send a list of results to the client, giving the user on the client side the ability to choose one item and get its complete details.
6. Once the client selects a specific result, the server will provide the corresponding details.

The server should clearly display the following details on the server screen:

1. The acceptance of a new connection, including the client's name.
2. The requester's name, type of request, and request parameters.
3. The disconnection of a client, including its name.

The Client Script

The client script should establish a connection with the server and remain active until the user selects the "Quit" option. The client should present multiple options provided by the server, including different requests, the ability to go back to the previous menu, and the option to quit. The client script should be able to send various types of requests, and to receive and display the responses accordingly. It should also prioritize user-friendliness by neatly displaying the available options and presenting the results in an organized manner.

The client should remain connected and ready to send new requests until the user decides to quit.

The client program should conduct the following tasks:

1. Establish a connection with the server and send the client's username to identify itself.
2. Display the main menu to the user.
3. Show different screens of option menus and lists of results based on the selected option. Please refer to the details below in Table 1 for more information about the options and responses.
4. The user will navigate between different screens until selecting the "Quit" option, which will terminate the connection and close the client.

Note: The retrieved information should be displayed in a clear and organized manner.

Menus and responses

The following subsections represent the different menus, responses from the server, and actions that can be taken on those responses.

Menus and options

There will be three main menus: the main menu, the headlines menu, and the sources menu. Figure 1 illustrates the three menus.

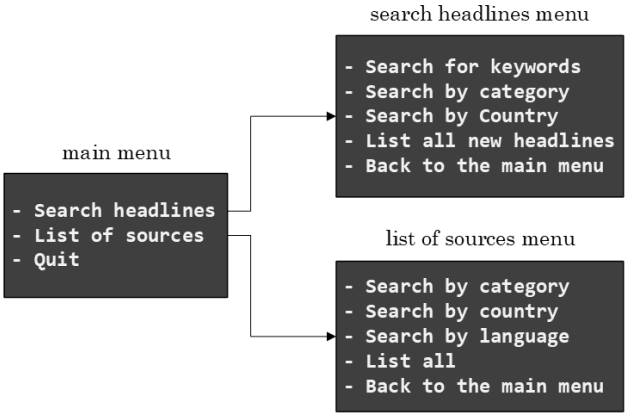

Figure 1: Client menus

Table 1 provides an overview of the different options available on each menu, along with a brief description of each option's functionality. This table aims to assist users in understanding the purpose and capabilities of each menu item.

Table 1: Client menus

ID	Menu	Menu option	Description
1	Main	Search headlines	Takes the user to the Headlines menu
1.1	Headlines	Search for keywords	The user can search in the news for a keyword in the news
1.2	Headlines	Search by category	The user can select the news category
1.3	Headlines	Search by country	The user can select news by country
1.4	Headlines	List all new headlines	The user can select news with no specific preference
1.5	Headlines	Back to the main menu	Return to the main menu
2	Main	List of Sources	Takes the user to the Sources menu
2.1	Sources	Search by category	The user can select the sources category
2.2	Sources	Search by country	The user can select sources country
2.3	Sources	Search by language	The user can select sources language
2.4	Sources	List all	The user can select sources with no specific preference
2.5	Sources	Back to the main menu	Return to the main menu
3	Main	Quit	Terminates the connection and client

The following table presents the included parameter values for the countries, languages, and categories available for search.

Table 2: Parameters

Parameter	Included Values
Countries	au, nz, ca, ae, sa, gb, us, eg, ma
Languages	ar, en
Categories	business, entertainment, general, health, science, sports, technology

Responses

The client should display the received list of results, and then the user can request details for a specific result, which the server should provide accordingly. Table 3 presents the details of the received response, which should be displayed on the client side along with the corresponding details of a selected item from the received list.

Table 3: Details of responses

Response type	List details	Selected item details
Headlines	Source name, author, title	Source name, author, title, URL, description, publish date, and publish time.
Sources	Source name	Source name, country, description, URL, category, and language

Note: To simplify the output for evaluation, each list should be limited to a maximum of 15 results.

Project learning outcomes

In this project, the learners will construct a Python networked application (system) that utilizes an online API. The application will be centralized, client/server-oriented, and capable of handling multiple threads for efficient connections. Through this project, the learners will acquire a fundamental understanding of building such a system, using techniques that are effective in various scenarios. Additionally, they will analyze the limitations of the framework and explore potential solutions to overcome them.

Why create from scratch?

Building a programming project from scratch offers several advantages. It provides a deep understanding of software development principles, encourages problem-solving skills, fosters creativity, and promotes collaboration. Students gain hands-on experience, develop critical thinking abilities, and enhance their adaptability to different programming environments. Additionally, starting from scratch allows students to customize the solution according to their needs, enabling them to explore different approaches, technologies, and design choices. This customization fosters a sense of ownership and empowers students to create a project that aligns with their unique vision and requirements.

Recommended Design Guidelines

1. Networking

Begin by implementing a simple Python server that can accept incoming TCP connections, and echo data sent on the connections back to the sender. You can test your server by using the "telnet" program. Furthermore, you can use Wireshark to follow the exchange packets between the client and the server. Then implement the client and test the connection with the server.

2. Retrieving the data from the online source

Implement the API to retrieve the data from the online source. Then implement the extraction of wanted information.

3. Threads

Extend the networking code you wrote in part 1 to accept multiple connections by creating a thread to deal with each incoming connection. Make sure that you put the functions in their appropriate location (main thread, loop, thread body)

Evaluation

The assignment will primarily be graded on:

1. The overall design of the system (no unneeded repetition of code).
2. How closely does your code implement the described protocol (system)?
3. How much of the described functionality do you provide, and how correct your implementation is?
4. Coding style (such as neat code, and proper variable names).
5. Usage of **GitHub** (for the entire process of project script development) and the use of **Teams private channel** for meetings and sharing documents and resources.

Design and style are essential components of this project. A significant lack of documentation (comments) or elegance within your code will affect your mark. Furthermore, a lack of participation in GitHub and Teams will affect your mark.

Your grade will be based on the following

1. **Correctness and completeness (30%):** To get some credit, you must, at the minimum, be able to connect properly to the API endpoint (15%). The next 10% for proper responses and the display of them. Another 5% for handling all other control messages. Failing to properly handle a server/client failure costs 5%. Your implementation must match the user interface specified above exactly. If your code does not compile, you will not receive any marks for correctness and completeness.

2. **Design (20%):** Elegance, the robustness of design, and handling of error conditions and special cases will be considered.
3. **Coding style (10%):** You should develop your code in a good style. For instance, consider splitting your code into modules, utilizing header files, using meaningful variable names, and adding comments to explain design choices and non-intuitive concepts. Additionally, ensure proper code indentation and utilize empty lines to enhance code readability. Please refer to the guidelines provided on the Blackboard for further details.
4. **Documentation (10%):** Your GitHub README file (markdown language) should be properly formatted and contain the following:
    a. Project title
    b. Project Description
    c. Semester
    d. Group: write the group name, course code, section, student name, and student ID)
    e. Table of Contents
    f. Requirements: Provide a clear and detailed explanation of the necessary steps to set up and run the project in a local environment. This should include instructions for installing any required dependencies, libraries, or frameworks, and guidance on configuring the development environment.
    g. How to: Explains how to run the system, including the necessary configuration steps, as well as clear instructions on how to interact with the server.
    h. The scripts: Provide a brief description of the client-server scripts, including their main functionalities, utilized packages, functions, and classes. Include relevant code snippets to support the descriptions in this section.
    i. Additional concept: Describe any additional concepts used in the project and highlight the corresponding code and its functionality.
    j. Acknowledgments
    k. Conclusion
    l. Resources (optional)
5. **Demonstration (20%):** a 10-minute overview of your work. When talking, follow your notes, and stick to your structure. Take your time, for example, to illustrate concepts or else that you introduce; your talk should not be too dense or abstract, which makes it hard to follow. Finally, you should be able to answer questions related to your design and code.
6. **Additional concepts (10%):** Add a concept that we did not take in the class. Select one of the following topics, learn it, implement it in your code, and explain its basics in your report:
    1. TSL/SSL (security): should secure all your packet transactions. Use Wireshark to examine the exchanged packets. You must show that it is

working during the demonstration and include pieces of evidence in your report (screenshots of the packet data).

2. Object-Oriented Programming: use object-oriented programming in your coding. Describe, in your report, the basics of OO in Python, and your classes.
3. GUI: use a graphical user interface for the client. It should be easy to use and clear to read. The user should send all requests using the GUI.

7. **Penalties:** there will be penalties for:
    1. Poor engagement in the project
    2. Uneasy to read codes (spacing, ordering, and naming)
    3. Redundancy in codes.
    4. Unclear displayed outputs.
    5. Late/missing submissions.

Using GitHub

GitHub will be used to collaborate between both group members to develop the scripts and manage the development process. This process is evaluated. Therefore, every group from the beginning should:

1. Create a repository called "ITNE352 Project – Group X" where X is your group name.
2. Both group members should be in that repository.
3. Add malmeer@uob.edu.bh to the repository.
4. Both members should participate in the code
5. Update the repository step by step using the proper GitHub commands.
6. Managing the project is within GitHub.
7. Management of all technical issues should be done within GitHub.

Deliverables

Each group leader must submit one copy of the project scripts and the README document. You must submit the following over the Blackboard:

1. A project README.md and README.pdf
2. The .py scripts, and the JSON files (generated during the demonstration).

Note: README should be set up properly on the main page in the GitHub repository.

Plagiarism

The writing in this survey paper should be in your own words. All submissions are subject to strict checks for plagiarism (the Turnitin tool is used here for similarity checks). Copying from past submissions is also treated as plagiarism. Please note that we take this matter quite seriously. We will decide on appropriate penalties for detected cases of plagiarism. A penalty would be to reduce the project mark to ZERO.

All ideas and paraphrases of other people's words must be correctly attributed in the body of the paper, citing the references. Cut-and-paste from documents such as Wikipedia or existing tutorials freely available will receive Zero marks. You must paraphrase any information that you collect from external sources. You can use figures/images from other sources if they are acknowledged in your paper. Your first starting point should be to read a technical survey on a related topic that has been published in an IEEE or ACM publication. This will give you a clear idea of what a survey article reads like and how you should structure your paper. You are required to cite at least six credible references.