University of Bahrain

College of IT

Department of CE

جَامِعَةُ البَحْرَيْن
كُلِّيَّةُ تِقْنِيَّةِ المَعْلُومَات
قِسْمُ هَنْدَسَةِ الحَاسُوب

ITCE320: Network Programming, S2 2020-2021
***Term Project** (group work)*

Multithreaded Flights arrival Client/Server Information System

Course instructor: Dr. Mohammed Almeer

Office: S40-2116

E-mail: malmeer@uob.edu.bh

Due: Report and code (May 20, 2021) and Demo (May 24, 2021 at lab time)

Late submission policy: Late submissions are subject to penalty. Two marks will be deducted for missing the deadline, and other two marks per day.

Assignment

In this project, the learners should create a client-server system for enquiring information about flights at a certain airport. The emphasis in this project is on the client/server architecture, network communication, and applying good coding practices.

The system should consist of two Python scripts (the server and the client). The server should be able to manage connections with multiple clients at the same time, retrieve the flight data from aviationstack.com via a proper API, and extract the required information from the retrieved data to send it to the client.

The Client Script

The client script should connect to the server, send different types of requests, and receive and display the responses. The client should be user-friendly and display the options and results neatly. The client should stay connected and ready to send new requests until the user chooses to quit.

The client program should conduct the following tasks:

1. Establish a connection with the server and send a username to identify itself.
2. Sends one of the following four request types:
    a. **Arrived flights:** the client should display the flight code (IATA), departure airport, arrival time, terminal, and gate.
    b. **Delayed flights:** the client should display the flight code (IATA), departure airport, departure time, estimated arrival time, terminal, and gate.
    c. **All flights coming from a specific city:** (the client should display the flight code (IATA), departure airport, departure time, estimated arrival time, terminal, and gate.
    d. **Details of a particular flight:** the client should display the flight code (IATA), date, departure (airport, gate, and terminal), arrival (airport, gate, terminal), status, scheduled departure time, scheduled arrival time, estimated arrival time, and delay.
3. Close the connection and leave when the user selects the Quit option.

Note: the retrieved information should be displayed clearly and neatly.

The Server Script

The server script should handle the clients and their requests. The server should retrieve the information about flights of a specific airport over an API. Then it should extract the needed information from the proper retrieved records and send it to the client.

The server should conduct the following tasks:

1. Once the server starts up, it should retrieve the flights' information.
2. It should ask the user to enter the airport code (arr_icao).
3. Uses the proper API to retrieve 100 records of the flights at the entered airport (use avaitionstack.com).
4. Store the retrieved data in a JSON file called "group_ID.json" (for testing and evaluation)
5. Wait for clients' requests to connect (should be able to accept three connections simultaneously).
    a. Accept the connection.
    b. Store the client’s name and display it on the terminal.

6. Wait for clients' requests, search the retrieved data for matching, and send a reply with the matching information.
    a. All arrived flights (return the flight no., estimated time of arrival, and terminal number).
    b. All delayed flights (return the flight no., original time, and estimated time of arrival).
    c. All flights from a specific city (return the flight no., original time, and status).
    d. Details of a particular flight (return flight no., original time, estimated time, status, and terminal number).

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

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

What you will learn

In this project, the learners will build a simple, centralized, client/server, multi-threaded, connection-oriented Python application (system) with the use of available online API. By doing so, the learners will learn a basic framework that can be used when creating such a system, using techniques that work well in many situations. You will examine some of the limitations of this framework and explore ways of getting around them.

Why create from scratch

There are many systems available that can take care of many of the networking details for you. In many cases, the best real-world solution is to use an existing framework because it often provides useful features such as fault-tolerance, load-balancing, and sessioning. It is, nevertheless, crucial to understand how these things work at the lowest level. No existing solution is perfect for all problems, and existing solutions often have minor flaws that your code must workaround. Merely choosing the right pre-packaged solution takes a discriminating eye that has been educated as to the tradeoffs inherent in various techniques.

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 your code implements the described protocol (system).
3. How much of the described functionality you provided, and how correct your implementation is.
4. Coding style (such as neat code, and proper variable names).

Design and style are essential components of this project. A significant lack of documentation (comments) or elegance within your code will affect your grade.

Your grade will be based on the following

1. **Correctness and completeness (20%):** To get some credit, you must, at the minimum, be <u>able to join the system and list matching flight records (15%)</u>. Another 10% for <u>handling all other control messages</u>. Detection and recovery from server/client failure worth the final 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. **Documentation (10%):** You should develop your code in a good style. You should, for example, <u>split up code into modules</u>, make use of header files, <u>use reasonable variable names</u> and <u>comment portions of your code that reflect design choices and non-intuitive ideas</u>, properly indent your code and use empty lines to <u>make your code more readable</u>.
4. **Report (20%):** Your written documentation must include everything you think is worth mentioning about your implementation.
   - Describe your design and what has and hasn’t been done. Marks will be given for the clarity of this documentation –organize it neatly, use proper English, and spellcheck!

- Discuss the purpose of the main elements (Listener class, Thread class, While-Read/Write loop (server-side), Removing dead connections, etc.) of your framework.
- Discuss some of the limitations of your framework and explore ways of getting around them, if possible
- Now that you have some experience in Python, what do you think of it? Is it a worthwhile language for developing distributed systems? Discuss two pros and two cons, and at least one extension you would like to see made to the language.

Your report should also include comments on the main design decisions (evaluation of the design) you have made:

- Messages formats
- How does your code deal with concurrent connection requests?
- How does the chat server deal with client failures?
- How do clients deal with chat server failures?
- Any other decision you feel influenced your design.
- etc.

5. **Online presentation (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 have to show that it is working in the online presentation and include pieces of evidence in your report (screenshots of the packets 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. Uneasy to read codes (spacing, ordering, and naming)
    2. Redundancy in codes.
    3. Unclear displayed outputs.

Deliverables

Each group leader must submit one copy of the project scripts and the documents. You must submit the following over MS Teams:

1. A project cover page listing learners' names and IDs.
2. A HowTo.txt file, detailing how to run the server, and the client.
3. The report as discussed above (Word document).
4. The client, server .py scripts, the JSON file.
5. A Presentation.ppt file, containing your in-class presentation slides.

Requirements

Your report must comply with the following paper specifications:

- Page length is limited to 12 single-column pages (including Title page and References **NOT** the appendices), double-spaced, 1" margins.
- Font: Arial, 12 point.
- A minimum of four sources (references) from scholarly publications or technical journals. You may find articles in books, journals from the library, on-line, or other sources.

Resources

Some of the following resources will prove useful in searching for related papers:

- IEEE Explore: http://ieeexplore.ieee.org/Xplore/dynhome.jsp
- ACM Digital Library: http://portal.acm.org/dl.cfm
- Elsevier Science Direct: http://www.sciencedirect.com/
- Google Scholar: http://scholar.google.com/

Reference Format

Note: You must reference the original source of a work, not a secondary source

List and number all bibliographical references in 10-point Times, single-spaced, at the end of your paper. When referenced in the text, enclose the citation number in square brackets, for example: [1]. Where appropriate, include the name(s) of editors of referenced books. The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in "[3]"—do not use "Ref. [3]" or "reference [3]". Do not use reference citations as nouns of a sentence (e.g., not: "as the writer explains in [1]").

Unless there are six authors or more give all authors' names and do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as “unpublished” [4]. Papers that have been accepted for publication should be cited as “in press” [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

Plagiarism

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

All ideas, paraphrases of other people's words must be correctly attributed in the body of the paper, citing the references. Cut-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 as long as 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 on what a survey article reads like and how you should structure your paper. You are required to cite at least six credible references.