Birzeit University
Faculty of Engineering and Technology
Department of Electrical and Computer Engineering
ENCS3320 – Computer Networks (Term 1241)
Project #1 (*Socket Programming*) – Due Thursday, November 28, 2024

A) Objectives

This project aims to deepen your understanding of socket programming and computer networking. Specifically, the objectives are as follows:

- Gain familiarity with fundamental network commands.
- Understand how to create Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) sockets.
- Learn the basics of HyperText Markup Language (HTML) and web server configuration.
- Develop teamwork skills by collaborating on project tasks.

B) Requirements and Deliverables

This project includes ***three tasks***. As a team of *three students (from any section)*, you are responsible for completing **all** tasks and submitting both your source code and a project report. Please consider the following requirements:

- **Implementation Guidelines:** For Tasks 2 and 3, you may choose *any programming language* (e.g., Python, Java, C, etc.). Note that libraries should not be used for implementing socket functionalities; *only the standard socket library is allowed*. The implementation should reflect *your own work* and be adaptable for future expansion with minimal effort (i.e., your implementation should be as generic as possible).
- **Team Coordination:** Each team member should actively participate across all project stages, including research, design, implementation, testing, and report preparation. To support collaboration, we recommend using GitHub for version control and collaborative coding, and Overleaf for seamless, real-time document editing. Consider dividing responsibilities clearly, sharing regular feedback, and tracking progress closely to ensure a unified and successful project outcome.

Please submit the following **files** as *a single compressed folder* (**.zip**) named ***Project1_TeamName.zip*** through **ITC** (https://itc.birzeit.edu/course/view.php?id=30941):

1) **Project Report (*report.pdf*):** A detailed report in PDF format, documenting your solutions to each task. See *Section D (Report and Grading Criteria)*, for further guidelines.
2) **Task 2 Folder:** A folder named ***Task2*** containing **well-documented** *source code and files* for Task 2. This folder should include: (i) The main server code (e.g., *server.py*) and (ii) Subfolders named *html*, *css*, and *imgs* containing the necessary HTML, CSS, and image files for the server, respectively. Include as many subfolders as needed to make your implementation well organized.
3) **Task 3 Folder:** A folder named ***Task3*** with **well-documented** *source code* for running the server (e.g., *server.py*) and client (e.g., *client.py*) for Task 3.

Each team should submit **one** final version. The submission ***deadline*** is November 28, 2024.

C) Project Tasks

1) Task 1 – Network Commands and Wireshark

a. In your own words, provide a brief (two-sentence maximum) explanation of each of the following commands: (i) ipconfig, (ii) ping, (iii) tracert, (iv) telnet, and (v) nslookup.

b. Ensure your computer is connected to the internet, then perform the following actions:

- Run the **ipconfig /all** command on your computer and identify the IP address, subnet mask, default gateway, and Domain Name System (DNS) server addresses for your primary network interface.
- **Ping** a device within your local network (e.g., from your laptop to a smartphone on the same Wi-Fi network).
- **Ping** discover.engineering.utoronto.ca. Based on the results, briefly explain whether you believe the response originates from Canada.
- Run **tracert** on discover.engineering.utoronto.ca.
- Use **nslookup** to retrieve the DNS information for discover.engineering.utoronto.ca.
- Attempt to connect with **telnet** to discover.engineering.utoronto.ca.

c. Use the Wireshark packet analyzer to capture a DNS query and reply for any hostname of your choice. Note: You can download the Wireshark packet analyzer from the following link: https://www.wireshark.org/download.html.

2) Task 2 – Web Server

Using socket programming, implement a simple but complete web server that listens on port 5698. The server should support the following HTML web pages:

a. Main English Webpage (*main_en.html*):

This page should include:

- The text "ENCS3320-Webserver" displayed on the web browser tab.
- The title "Welcome to ENCS3320 - Computer Networks Webserver" at the top of the page.
- Team members' photos (in **.png** format) displayed alongside their names and IDs, arranged in a table-like structure. Organize each member's information within separate boxes on the page.
- A brief description of each team member, covering details like projects completed in various courses, skills, hobbies, etc.
- A well-presented topic from the first two chapters of the textbook, including elements such as paragraphs, images (in **.jpg** format), and ordered and unordered lists. Use appropriate headings, font styles, colors, and text formatting (bold/italic).
- A link to a local HTML file (*supporting_material_en.html*), discussed below.
- Links to external resources, including:
  - The textbook website (https://gaia.cs.umass.edu/kurose_ross/index.php)
  - Ritaj website (https://ritaj.birzeit.edu/)
- A separate CSS file to style the page attractively.

b. Supporting Material Page (*supporting_material_en.html*):

This page should contain a form where users can request an image or video related to the computer networking topic discussed in *main_en.html* by entering the name of the file in an input box. If the requested file is unavailable, the server should redirect the client (web browser) with a "307 Temporary Redirect" status code, depending on the file type:

- **Images:** Redirect to a Google search URL that filters results to images based on the user's input (e.g., https://www.google.com/search?q=http+protocol&udm=2 for HTTP protocol).
- **Videos:** Redirect to a YouTube search URL that filters results to videos based on the user's input (e.g., https://www.youtube.com/results?search_query=http+protocol for HTTP protocol).

c. Arabic Versions:

Provide Arabic versions of the *main_en.html* and *supporting_material_en.html* files, named *main_ar.html* and *supporting_material_ar.html*, respectively.

Server Functionality:

When the server receives an HTTP request, it should:

- Print the HTTP request details to the terminal.
- Generate and send an HTTP response that includes the requested content and specifies the correct Content-Type based on the file type (e.g., text/html for HTML, text/css for CSS, image/png for PNG, and video/mp4 for MP4).

Default and Error Responses:

- For requests targeting /, /en, /index.html, or /main_en.html (e.g., http://localhost:5689/, http://localhost:5689/en, http://localhost:5689/index.html, or http://localhost:5689/main_en.html), the server should respond with *main_en.html*.
- For requests targeting /ar or /main_ar.html (e.g., http://localhost:5689/ar or http://localhost:5689/main_ar.html), the server should respond with *main_ar.html*.
- If the client requests an invalid URL (i.e., a file not found on the server), the server should return a simple HTML error page with:
  - The status line "HTTP/1.1 404 Not Found"
  - The text "Error 404" in the browser tab
  - The message "The file is not found" displayed in red text in the body
  - The client's IP address and port number.

Organize the server files as outlined in *Section B (Requirements and Deliverables)*.

3) Task 3 – UDP Client-Server Trivia Game Using Socket Programming

In this task, you are required to implement an interactive multiplayer trivia game using UDP socket programming to test players' (i.e., clients') knowledge in a fun and competitive environment. The server orchestrates the overall flow of the game, which consists of a series of rounds. The primary responsibilities of the server and client, along with their respective terminal outputs, include:

a. Server Responsibilities

- **Client Connection Management:** The server listens on port 5689 for incoming client connections and maintains a list of active clients, i.e., newly registered clients and those who participated in the previous round by answering at least one question. To uniquely identify each client, the server uses the client's IP address and port number.
- **Round Initialization:** Each round begins only if there is a minimum number of active clients (set to 2). Once enough clients are connected, the server broadcasts a welcome message to notify clients that the round is about to start.
- **Question Broadcast:** In each round, the server broadcasts *N* randomly selected questions (e.g., 3) from a predefined database to all connected clients. The server pauses for a specified period (e.g., 60 seconds) before broadcasting each question, allowing players time to prepare.

- **Answer Collection and Time Management:** The server waits a specified amount of time (e.g., 90 seconds) for players to respond. During this period, clients can submit their answers, which the server checks for correctness and assigns points accordingly (points may vary, with more points awarded to earlier correct responses). If a client submits multiple answers, the server will only accept the first *received* answer.
- **Score Tracking and Winner Announcement**: At the end of each question, the server broadcasts the correct answer and the current scores to all active clients. After each round, the server announces the current standings and the leading player (i.e., the client who has won the most rounds).
- **Pause Between Rounds:** The server briefly pauses between rounds, allowing players time to review the leaderboard and prepare for the next round.

b. Server Terminal Output

- Print a message confirming that the server has started and is listening on a specific port.
- For each new client connection, display a message with the client's IP address and port.
- When enough clients have joined, indicate that a new round will begin.
- For each question, display the question number and text.
- As answers arrive, display each client's response with their identifier (IP and port) and whether it's correct or incorrect.
- At the end of a round, announce the winner of the round.
- Indicate the countdown before starting the next round.
- If the server is stopped, display a message confirming the shutdown.

c. Client Responsibilities

- **Connecting to the Server:** The client connects to the server using the provided IP address and port. Upon joining, the client submits their username for display on the leaderboard. Clients can join or leave the game at any time.
- **Receiving Game Notifications:** The client listens for server messages, including round start notifications, questions, and score updates. These messages keep the player informed about the game's progress and their standing.
- **Answer Submission:** Upon receiving a question, the client has a set period (e.g., 90 seconds) to submit their answer to the server.

d. Client Terminal Output

- Upon connecting to the server, print a confirmation message with the server's IP and port.
- Display any message or instruction received from the server, including notifications about round starts, question text, score updates, the current leaderboard standings, and notifications about the server disconnection.
- After the player submits an answer, confirm it in the terminal.

Demo Link: For a better understanding of how the system operates, a demo showcasing interactions between a server and three clients will soon be available at the following link: https://drive.google.com/drive/folders/1wzxue820O5uVIWt6NLfnM_TBogTmxqa9?usp=sharing.

D) Report and Grading Criteria

1) Project Report

The report should include the following:

a. Cover Page

- Include the university logo, department, course name and number, project title, names, and IDs of all team members along with their section, and submission date.

b. Theory and Procedure

- Explain the theoretical concepts relevant to each project requirement. Organize these ideas according to the specified titles for each section.
- List the components or methods used for each part of the project, including a brief explanation of why each one is necessary. Use diagrams or flowcharts to illustrate the solutions, with detailed descriptions and captions for all figures. Be sure to reference each figure in your descriptions.
- Cite sources for all theoretical concepts and ideas. Ensure all references are included at the end of the report.

c. Results and Discussions

- For each task listed in *Section C (Project Tasks)*, provide screenshots that show the outputs and results, along with a discussion. Screenshots of the code may be included to support your explanations, but avoid using screenshots alone without commentary.
- Discuss your approach and results in detail. For example, when redirecting to a URL (such as https://www.google.com/search?q=http+protocol&udm=2), explain the significance of each URL component and why it was included.
- For **Task 2**, provide browser screenshots that demonstrate the project's functionality for all required links. Test the project both from a browser on the same computer and from an external device (e.g., another computer or smartphone within your local network). Also, include a screenshot showing the HTTP request output printed on the command line.
- Ensure each screenshot includes your system's date and time.

d. Alternative Solutions, Issues, and Limitations

- Discuss any alternative approaches that could potentially be used to solve the project tasks, beyond those specified in the instructions.
- Describe any limitations, issues, or challenges you encountered, either individually or as a team.
- Highlight any parts of the project that did not work as expected, if applicable.

e. Teamwork

- Document the contributions of each team member using a chart that outlines the specific tasks completed by each member.

Ensure the report includes proper numbering, a table of contents, a list of figures, a list of tables, references, and appendices where needed.

2) Grading Criteria

This project is worth 12% of the total grade: 8% will be allocated to the source code and report, while the remaining 4% will be based on short-answer questions in the midterm exam related to this project. Performance on this project will also be partially competitive, with additional points awarded for the following distinguishing factors:

- Quality and completeness of functioning features.

- The cohesiveness of the integrated components in achieving the project objectives.
- Demonstrated understanding of implementation details.
- User-friendliness and ease of interfacing with the components.
- Deductions may apply if extensions or names deviate from those specified in *Section B (Requirements and Deliverables)* and *Section C (Project Tasks)*.

Late submissions are permitted up to three days past the deadline, with a 10% deduction for each day late.

Helpful resources to support your success on this project are available at the following link: https://drive.google.com/drive/folders/16AroRSIdt39aSl9ydGVxziP4Tqejl5yx?usp=sharing.

GOOD LUCK