Birzeit University
Faculty of Engineering and Technology
Department of Electrical and Computer Engineering
ENCS3320 – Computer Networks (Term 1244)
Project (*Socket Programming*) – Due Thursday, September 18, 2025

A) Objectives

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

- Understand how to create TCP and 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 ***two tasks***. As a team of three students (*from any section*), you are responsible for completing **all** tasks and submitting all tasks' files that are required to run the tasks. Please consider the following requirements:

- **Implementation Guidelines**: For **Tasks**, 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, and testing. To support collaboration, we recommend using GitHub for version control and collaborative coding. Consider dividing responsibilities clearly, sharing regular feedback, and tracking progress closely to ensure a unified and successful project outcome.

Please submit the task **files** as *a single compressed folder* (**.zip**) named ***TeamName.zip*** (TeamName is given in the excel sheet, e.g., team name for group#1 is **T001 → T001.zip**) through **ITC** (https://itc.birzeit.edu/course/view.php?id=35183):

1) **Task 1 Folder**: A folder named ***Task1*** containing well-documented source code and files for Task1. 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.
2) **Task 2 Folder**: A folder named ***Task2*** with well-documented source code for running the server (e.g., *server.py*) and client (e.g., *client.py*) for Task 2.

Each team should submit one final version. The submission deadline is end of **Sep. 18, 2025**.

C) Project Tasks

Task 1 – Implement a Simple File-Sharing Web Server Using Socket Programming

Objective:

In this task, you will implement a basic web server using socket programming that serves static files and displays team/student information. The server must listen on a port number derived from your student ID. You will create both English and Arabic versions of the main website.

Port Number Rule:

Use the last 3 digits of your student ID + 5000. **Example**: If the student ID is 1240310, the port number = (310 + 5000) = 5310.

Server Requirements:

1. **Core Web Server**
   - The server must accept incoming HTTP requests over TCP sockets.
   - It should parse the request and respond with the correct HTTP response.
   - For each request, the server must print in the terminal:
     - Client IP address and port.
     - The requested resource.
     - The status code sent back (200, 403, 404).
2. **Website Structure**

   You must create two versions of the website:

   A. English Main Page (main_en.html)

   This page should include:

   - Title in browser tab: "ENCS3320-Webserver" (in red).
   - Page header: "Welcome to ENCS3320 - Computer Networks Webserver" (in blue).
   - A team section in a table-like format, where each student has:
     - Full name, student ID
     - A photo (.png format)
     - Hobbies (e.g., football, programming, music)
     - Number of finished credit hours
   - A short description about each student (e.g., skills, projects completed).
   - A small networking-related topic summary (from chapter 1 slides) with:
     - A short paragraph description
     - One .jpg image
     - One ordered list and one unordered list
   - Links to:
     - Textbook website
     - Birzeit University website

- A link to the local file request page (TeamNumber_STDID_en.html). Where TeamNumber from the excel sheet and STDID is the student ID for one of the students in the team.

B. Arabic Main Page (main_ar.html)

- A mirror version of main_en.html but in Arabic.
- Same student details translated into Arabic.
- Same design but right-to-left layout for Arabic text.
- Link to the Arabic local file request page (TeamNumber_STDID_ar.html).

3. Local File Request Pages

A. English Version (TeamNumber_STDID_en.html)

- A form where the user can enter the name of a file (e.g., notes.txt, report.pdf).
- If the file exists → server sends it with 200 OK.
- If the file name contains "private" → server responds with 403 Forbidden and displays:
  - "Access Denied – This file is private." in a styled HTML page.
- If the file does not exist → server responds with 404 Not Found and displays a custom HTML page with:
  - Browser tab title: "Error 404"
  - age body: "The file is not found." (in red)
  - Client IP and port displayed at the bottom.

B. Arabic Version (TeamNumber_STDID_ar.html)

- Same functionality as English version.
- Messages and form in Arabic (e.g., "أدخل اسم الملف").
- 403 and 404 error pages should also be in Arabic.

4. Default Responses

- For requests to /, /index.html, /main_en.html, or /en, **return** main_en.html. For example: http://localhost:5910/.
- For requests to /ar or /main_ar.html, **return** main_ar.html. For example: http://localhost:5910/ar.

Submission Requirements:

- Submit a single folder named by **Task1** and contains:
  - Source code (.py, .c, or .java)
  - HTML, CSS, and media files (.html, .css, .png, .jpg)
- No report is required.

Grading Criteria (Weight 5%):

Component	Weight
Correct port number derived from student ID	**5%**
Server accepts connections & logs request info	**10%**
English main page (layout, student details, links, design)	**15%**

Arabic main page (layout, translated content)	**15%**
File request pages (form + functionality)	**15%**
Correct **200 OK** response with existing files	**10%**
Correct **403 Forbidden** response for “private” files	**10%**
Correct **404 Not Found** error page	**5%**
Default response handling (/, /en, /ar)	**5%**
Code readability, comments, structure + Files	**10%**

Task 2 – TCP/UDP Hybrid Client-Server Game Using Socket Programming

Objective:

In this task, you are required to develop a multiplayer quiz game using both TCP and UDP sockets. The server coordinates player registration, quiz questions, scoring, and results. The design uses a hybrid approach:

- TCP → for reliable connection setup, player registration, and final results.
- UDP → for fast-paced answer submission during quiz rounds.

Port Number Rule:

Each student must customize the server port numbers using one student ID from the team:

- Example: For Student ID = 1240310
  - TCP Port = 310 (last 3 digits + 3000)
  - UDP Port = 124 (first 3 digits + 6000)

Game Features:

- Multiplayer mode: At least 2 players (your partners), maximum 4 (if more 2 add friends).
- Player Registration (TCP): Clients connect via TCP, register with unique usernames, and wait for the game to start.
- Hybrid Gameplay:
  - Server sends quiz questions (via TCP).
  - Players submit answers quickly (via UDP).
- Scoring:
  - Correct answer = +1 point.
  - Wrong or no answer = 0 points.
- Winner Announcement (TCP):
  - After all rounds, the server sends final scores and announces the winner.

Workflow:

Phase 1 – Registration (TCP)

1. Server listens on TCP_PORT.
2. Clients connect and send: JOIN <username>.
3. Server acknowledges unique usernames.

4. Once the minimum number of players join, the server broadcasts Game Start and rules via TCP.

Phase 2 – Quiz Rounds

Step 1 – Question Broadcast (**TCP**):

- Server sends a question to all clients, for example:

```
Q1: What does HTTP stand for?
a) Hyper Transfer Text Protocol
b) HyperText Transfer Protocol
c) High Text Transmission Protocol
d) None of the above
```

Step 2 – Answer Submission (**UDP**):

- Clients send their answer (e.g., **Player1: b**) via UDP to the server's UDP port.
- Server checks correctness and updates scores.

Step 3 – Feedback (**UDP**):

- Server sends fast feedback to each player:
  - "Correct"
  - "Wrong"

Phase 3 – Results (TCP)

- After all questions (e.g., 5 rounds), the server sends final scores to all players via TCP:

```
Final Results:
Player1: 3 points
Player2: 2 points
Winner: Player1
```

Initial Settings:

- TCP Port = last 3 digits of your student ID + 3000
- UDP Port = first 3 digits of your student ID + 6000
- Min players = 2
- Max players = 4
- Time per question = 10 seconds
- Number of questions = 5
- Allowed libraries: {socket, threading, random, time}

Submission Requirements:

- Submit a single folder named by **Task2** and contain:
  - Source code for server (.py, .c, or .java)
  - Source code for client (.py, .c, or .java)
  - Readme file (.txt) on how run the task.

- No report is required.

Grading Criteria (Weight 5%):

Criteria	Details	Weight
Port Customization	Correct use of Student ID to set TCP and UDP ports.	**10%**
Server Implementation	Handles player registration via TCP, sends questions via TCP, receives answers via UDP, provides feedback, tracks scores, and announces results.	**30%**
Client Implementation	Connects and registers via TCP, submits answers via UDP, receives feedback and final results correctly.	**30%**
Protocol Usage (Hybrid)	Proper use of TCP for control and reliability, UDP for fast-paced answer submission.	**15%**
Code Quality & Execution + Files	Code runs without errors, uses only allowed libraries, is well-structured, and contains clear comments.	**15%**

D) Grading

This project is worth **15%** of the total grade: **10%** will be allocated to the source codes (Task1 (**5%**) and Task2 (**5%**)), while the remaining **5%** will be based on short-answer questions in the exam related to this project.