Alexandria National University
Faculty of Computing and Data Science
Computer Networks: 00207N
Lab 2 - Fall 2025

Multi-Client String Processing Server

Objective

This lab aims to strengthen your understanding of socket programming by creating a multi-client string processing server.
You will:

- Implement multiple concurrent client connections using TCP sockets.
- Design and use a simple command-based text protocol.
- Capture and analyze communication using Wireshark.

Overview

In this lab:

- The server listens to multiple client connections on a TCP port.
- Each client can send text commands like:
  UPPER: hello or REVERSE: world.
- The server processes the message based on the command and sends back the result.
- Each client runs independently, and the server distinguishes between them by their socket connection.

Every client connection is handled by a separate thread, allowing multiple clients to interact with the server simultaneously.

Task Description

1. **Server Requirements:** Create a Python (or Java) server that:
   A. Listens for connections on port 5000.
   B. Accept multiple clients using threads.
   C. For each connected client:
      - Reads a line from the client.
      - Parses the command and message.
      - Performs the required operation.
      - Send the result back to the client.
      - Logs the communication (client name + command + result) to the console.
   D. If a client sends QUIT, close that client's connection gracefully.
   E. Continue serving other clients even when one disconnects.

Instructors: Dr. Mohamed Rezk & Dr. Ahmed Eltrass
Teaching Assistants: Eng. Ahmed Ashraf / Eng. Miar Mamdouh / Eng. Salma Magdy

2. **Client Requirements:** Create a Python (or Java) client that:
   A. Connects to the server using TCP on port **5000**.
   B. Reads user input from the console.
   C. Send the input message to the server.
   D. Displays the server's response.
   E. Ends the session when the user enters QUIT.

3. **Communication Protocol:**
   Each client's message must follow this format: COMMAND: MESSAGE

   Supported Commands

Command	Description	Example Input	Expected Server Response
UPPER	Converts message to uppercase	UPPER: hello	HELLO
LOWER	Converts message to lowercase	LOWER: HELLO	hello
REVERSE	Reverse the message	REVERSE: hello	olleh
COUNT	Counts characters in message	COUNT: network	7
QUIT	Disconnects client	QUIT	(server closes connection)

   If client1 sends a message that doesn't match one of the valid commands (which means an invalid command is received):
   The server replies with: ERROR: Unknown command from client_number.
   Example:
   Client1 sent: HELLO WORLD
   Server response: ERROR: Unknown command from Client1.
   Notes:
   - The QUIT command is the only one that doesn't require a message.
   - Each client must be handled in a separate thread, allowing multiple clients to interact simultaneously without blocking each other.
   - The command and message should be separated by a single colon (:).

4. **How the Server Differentiates Clients**
   A. Every time a new client connects, the server assigns it a unique client name, such as: Client1, Client2, Client3, ...
   B. The server keeps a counter that increments with each new connection.
   C. When a client connects, the server prints something like:
   Client1 connected from /127.0.0.1:50236

Output

1. Sample Output (Server Console)

```
Server started on port 5000
Client1 connected from 127.0.0.1: 60123
Client1 sent: UPPER: hello → HELLO
Client2 connected from 127.0.0.1: 60124
Client2 sent: REVERSE: world → dlrow
Client1 sent: COUNT: network → 7
Client1 disconnected.
```

2. Sample Output (Client Console)

```
Connected to server.
Enter command (UPPER/LOWER/REVERSE/COUNT/QUIT):
> UPPER: hello
Response from server: HELLO
> COUNT: network
Response from server: 7
> QUIT
Connection closed.
```

Wireshark Analysis

After running your server and multiple clients:

- You are required to provide a screenshot of Wireshark showing the traffic between your server and clients' requests.

Bonus

Implement one or more:

1. Add command VOWELS: → Count number of vowels in the string.
   Example: VOWELS: Hello → 2
2. Add command TIME: → Return current server time.
3. Add command DATE: → Return current server date.
4. Log all requests/responses into a file named server_log.txt.

Deliverables

- Source Code for both client and server.
- A comprehensive technical report documenting your implementation, usage instructions, dependencies, and testing results (with supporting screenshots).

Policy:

- The deadline for delivery is your lab time.
- A discussion will be held in your lab.
- You must work in groups of 4.
- Your code should be clean, readable, and the report should be supported with screenshots.
- Cheating will be severely penalized.
- No late submission is allowed.