NU-Information Exchange System

Course: Computer Networks Lab– Fall 2025

Instructor: Sir Hassan Ahmad

Submitted By:

Student 1: Asadullah (23F-0693)

Student 2: Ali Hasan (23F-0610)

Student 3: Ishwa Zainab (23F-0793)

Section: BS(CS)-5C

Date of Submission: December 7, 2025

Introduction

The **NU-Information Exchange System** is a computer network application designed to facilitate reliable and real-time communication between multiple university campuses, known as **Campus Clients**, managed by a central **Central Server**. The system uses a **hybrid networking architecture** employing both the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) for different communication needs.

Project Overview and Implementation Details

The project is structured around a Central Server that acts as a router and state manager, and multiple Campus Clients that connect to the server.

1. Central Server (`central_server.cpp`)

The server is the core of the system, responsible for managing client connections, authentication, message routing, and administrative tasks.

- **Networking:** It listens for new Campus Client connections on **TCP port 8080** and receives heartbeats on **UDP port 8081**.
- **Concurrency:** It is highly multithreaded.
    - The main thread handles new TCP client **`accept`** calls.
    - **`handleCampusClient`** is launched in a new thread for *each* connected client to manage its dedicated TCP session and listen for messages.
    - **`handleUDPHeartbeat`** runs in a separate thread to passively listen for client status updates.
    - **`adminModule`** runs in a thread to provide the server operator with a menu for monitoring and broadcasting.
- **State Management:** The server uses a `map<string, CampusClient>` called **`connectedClients`** to store the connection status, TCP socket, last heartbeat time, and the latest UDP address of each active campus.
- **Synchronization: Mutexes** (`clientMutex`, `consoleMutex`) are crucial for thread-safe access to the shared `connectedClients` map and synchronized console output.

2. Campus Client (`campus_client.cpp`)

The client application allows a campus to connect to the central server and communicate with other campuses.

- **Connection:** It initiates a **TCP connection** to the server's port 8080 for authentication and sending reliable messages.

- **Multithreading:** The client also uses multiple threads for parallel operations:
  - **`listenForMessages`**: Dedicated thread for receiving reliable TCP messages (point-to-point) and ACKs/ERRORs from the server.
  - **`sendHeartbeat`**: Dedicated thread that periodically sends a UDP packet to the server to maintain an active status.
  - **`listenForBroadcasts`**: Dedicated thread that binds a local UDP port (dynamically assigned) to receive server-initiated announcements.
- **Hybrid Communication:** It uses TCP for data (messages) and UDP for status/announcements. It binds a **dynamic UDP port (0)** and communicates this port number via the heartbeat to the server, allowing the server to directly target it for UDP broadcasts.

Custom Protocol Design

The communication within the system is governed by three simple, text-based application-layer protocols:

1. Authentication Protocol (TCP)

This protocol is used during the initial TCP setup phase to verify the client's identity.

- **Direction:** Client → Server
- **Format:** `Campus:CampusName,Pass:Password`
- **Example:** `Campus:Lahore,Pass:NU-LHR-123`

2. Point-to-Point Message Protocol (TCP)

This is the core communication protocol for reliable, directed messaging between campuses. It is designed to be easily parseable.

- **Direction:** Client ↔ Server ↔ Client (Routed)
- **Format:** **`TARGET:CampusName|DEPT:DeptName|FROM:SourceCampus|MSG:MessageBody`**
- **Separators:** A **pipe (`|`)** is used to delimit the header fields, and a colon (`:`) separates the key from the value (e.g., `FROM:Lahore`).

3. Heartbeat and Broadcast Protocol (UDP)

These protocols leverage UDP for low-latency status updates and announcements.

Protocol	Direction	Format	Purpose

Heartbeat	Client → Server	`HEARTBEAT:CampusName:ClientUDPPort`	To update the server on the client's live status and current UDP address.
Broadcast	Server → Client	`BROADCAST:AnnouncementMessage`	To send system-wide announcements to all active campuses.

User Guide

This guide details the steps to compile and run the project, assuming the server and client code are in separate files (`central_server.cpp` and `campus_client.cpp`).

Step 1: Compilation

The project requires the C++ standard library, a network library (Winsock on Windows, standard sockets on Unix-like systems), and thread support.

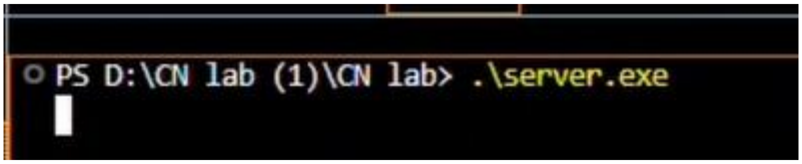
Use a modern C++ compiler (like g++):

```
# Compile Server
g++ central_server.cpp -o central_server.exe -std=c++11 -pthread

# Compile Client
g++ campus_client.cpp -o campus_client.exe -std=c++11 -pthread
```

Step 2: Running the Central Server (Must Run First)

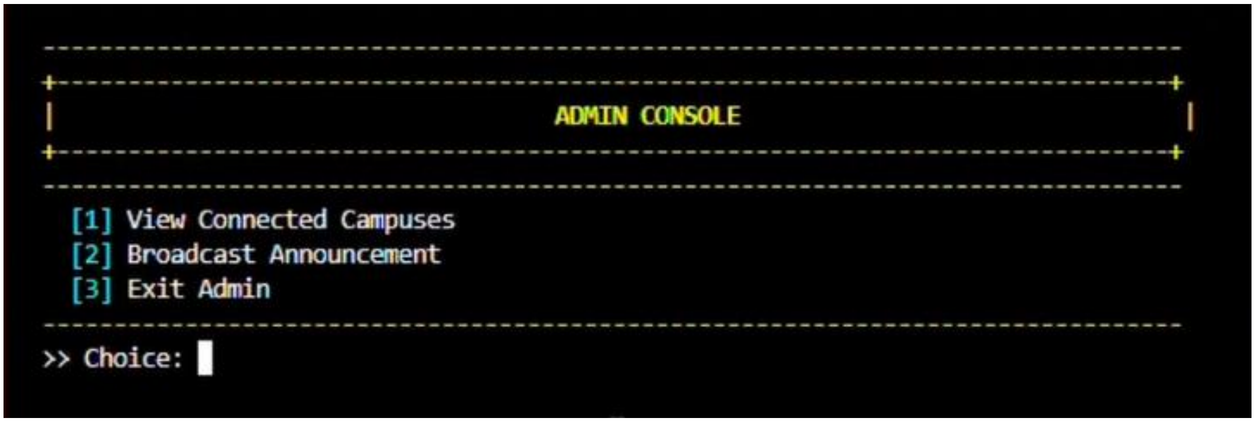
1. Open your first terminal (e.g., PowerShell, Linux Terminal).
2. Execute the server:

```
.\central_server.exe
```



```
○ PS D:\CN lab (1)\CN lab> .\server.exe
```

3. The server will initialize and begin listening on TCP port 8080 and UDP port 8081.
4. Press **Enter** as prompted to open the **Admin Console**.



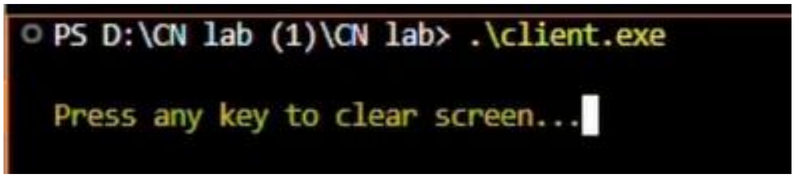
```
--------------------------------------------------------------------------------
+------------------------------------------------------------------------------+
|                                 ADMIN CONSOLE                                |
+------------------------------------------------------------------------------+
--------------------------------------------------------------------------------
  [1] View Connected Campuses
  [2] Broadcast Announcement
  [3] Exit Admin
--------------------------------------------------------------------------------
>> Choice:
```

Step 3: Running the Campus Clients

Open at least two separate terminals for the clients to demonstrate inter-campus communication.

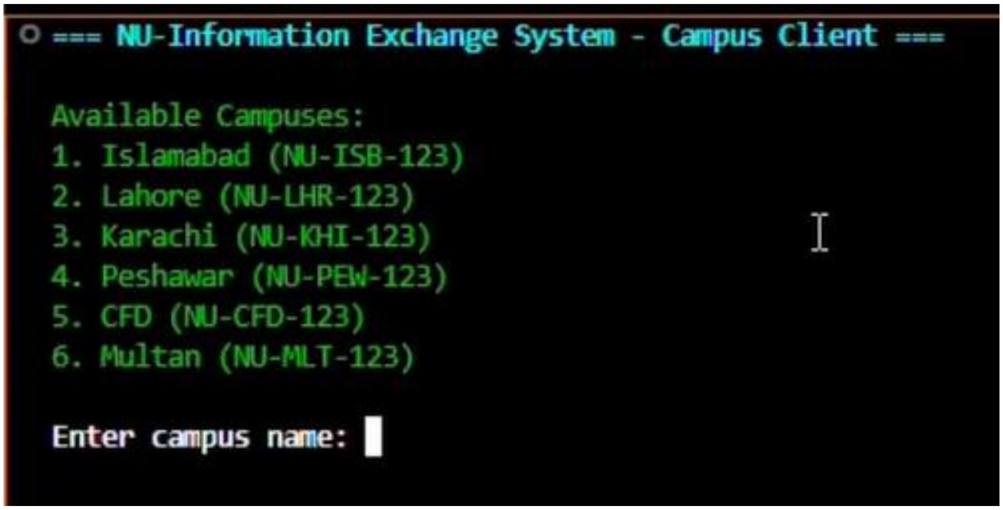
1. In a second terminal, execute the client:

```
.\campus_client.exe
```



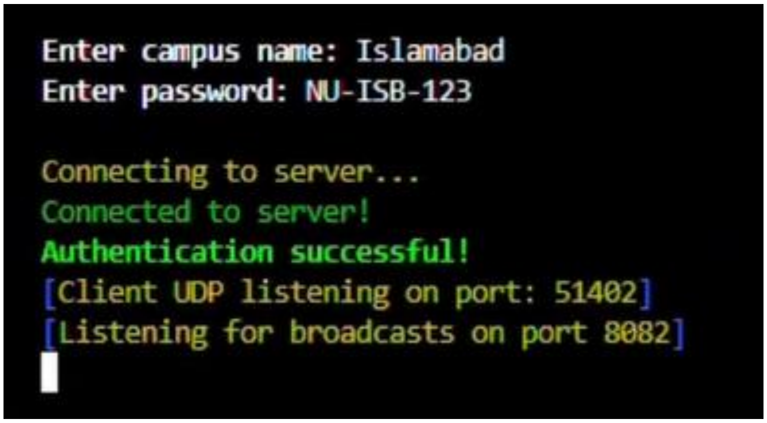
```
○ PS D:\CN lab (1)\CN lab> .\client.exe

  Press any key to clear screen...
```

2. The client will display the available campuses.



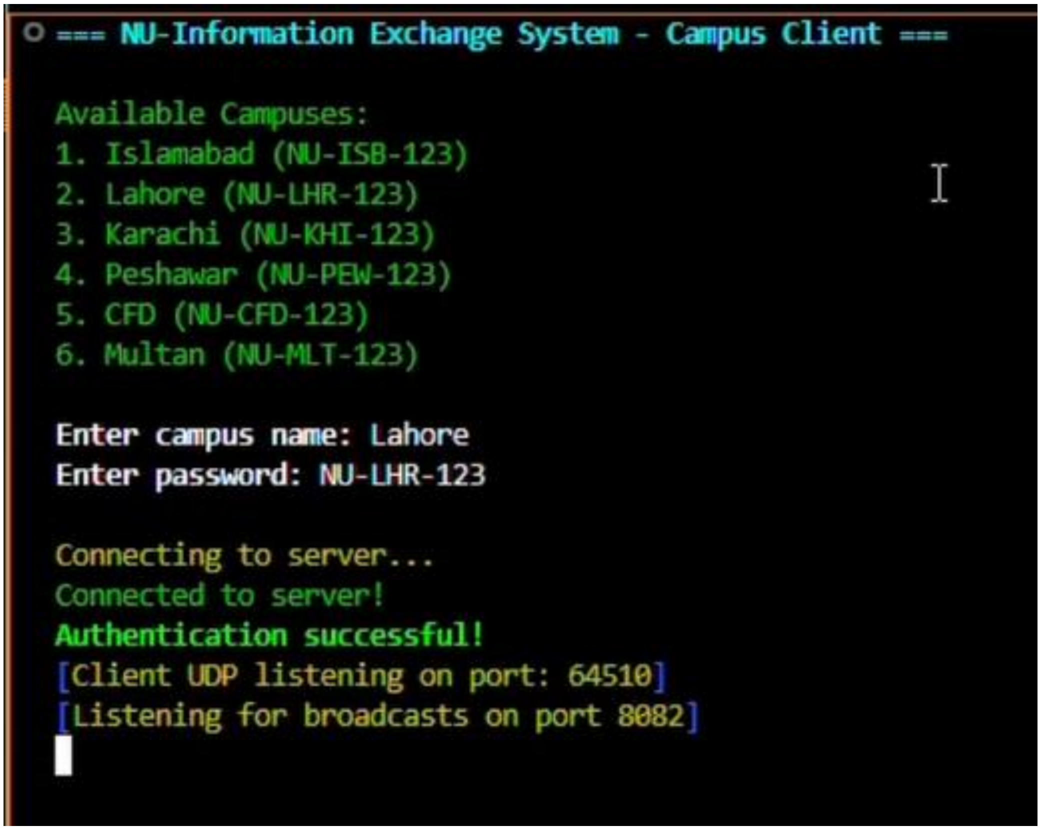
```
=== NU-Information Exchange System - Campus Client ===

Available Campuses:
1. Islamabad (NU-ISB-123)
2. Lahore (NU-LHR-123)
3. Karachi (NU-KHI-123)
4. Peshawar (NU-PEW-123)
5. CFD (NU-CFD-123)
6. Multan (NU-MLT-123)

Enter campus name: 
```

3. **Enter Campus Name:** Type a campus name (e.g., Islamabad).
4. **Enter Password:** Type the corresponding password (e.g., `NU-ISB-123`).



```
Enter campus name: Islamabad
Enter password: NU-ISB-123

Connecting to server...
Connected to server!
Authentication successful!
[Client UDP listening on port: 51402]
[Listening for broadcasts on port 8082]
```

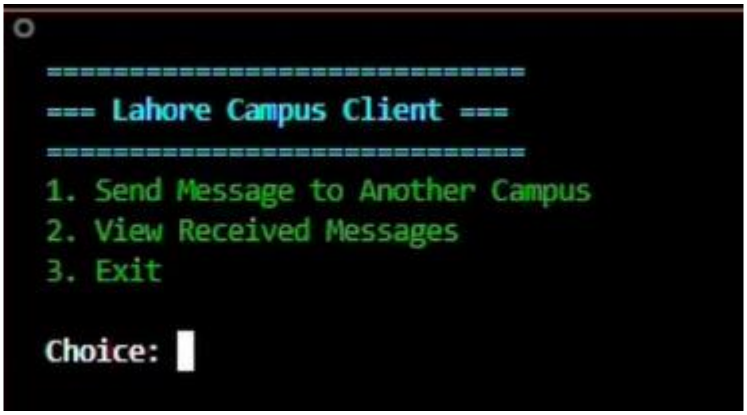
5. Repeat this process in a third terminal for another campus (e.g., Lahore).



```
=== NU-Information Exchange System - Campus Client ===

Available Campuses:
1. Islamabad (NU-ISB-123)
2. Lahore (NU-LHR-123)
3. Karachi (NU-KHI-123)
4. Peshawar (NU-PEW-123)
5. CFD (NU-CFD-123)
6. Multan (NU-MLT-123)

Enter campus name: Lahore
Enter password: NU-LHR-123

Connecting to server...
Connected to server!
Authentication successful!
[Client UDP listening on port: 64510]
[Listening for broadcasts on port 8082]
```

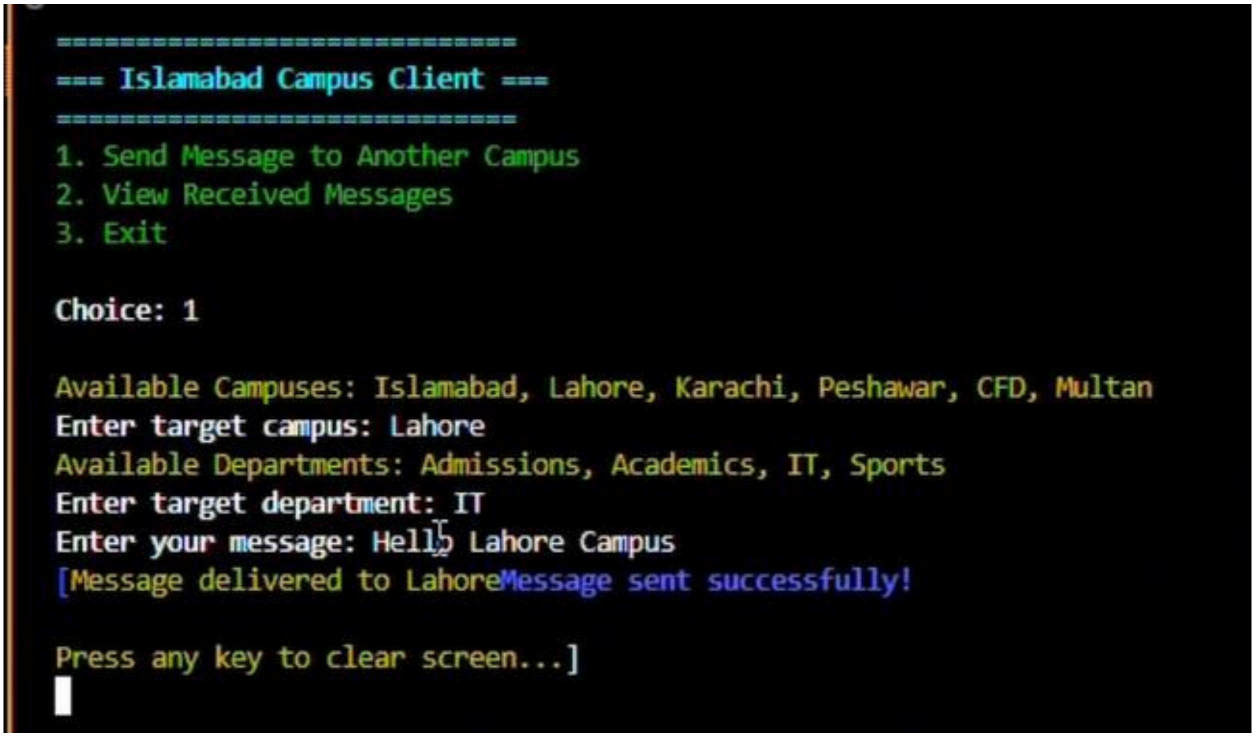
Step 4: Testing Communication

1. In one of the **Campus Client** terminals (e.g., Lahore), select menu option **1** (**Send Message to Another Campus**).



```
==============================
=== Lahore Campus Client ===
==============================
1. Send Message to Another Campus
2. View Received Messages
3. Exit

Choice:
```

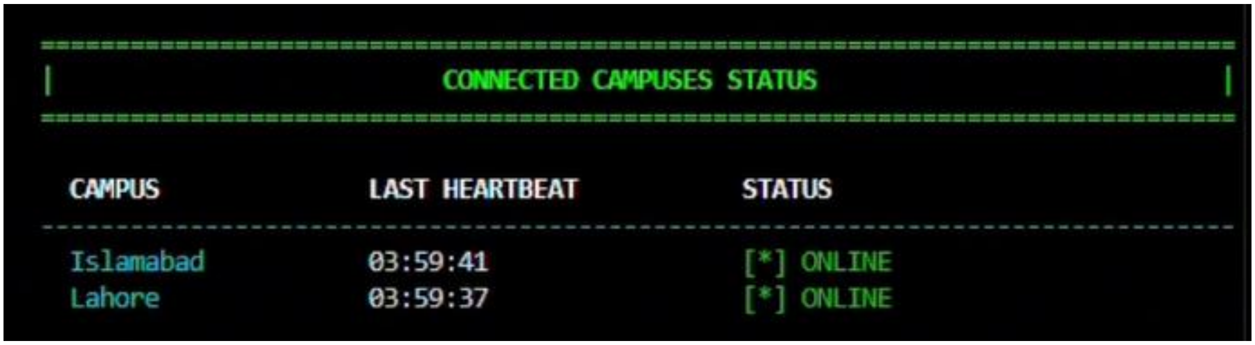
2. When prompted:
   a. Enter **target campus** (e.g., `Karachi`).
   b. Enter **target department** (e.g., `IT`).
   c. Enter your **message** (e.g., `Check the new schedule`).
3. The message will be sent via the Central Server.
4. The message will instantly appear on the **Karachi** client terminal, confirming successful routing.



```
==============================
=== Islamabad Campus Client ===
==============================
1. Send Message to Another Campus
2. View Received Messages
3. Exit

Choice: 1

Available Campuses: Islamabad, Lahore, Karachi, Peshawar, CFD, Multan
Enter target campus: Lahore
Available Departments: Admissions, Academics, IT, Sports
Enter target department: IT
Enter your message: Hello Lahore Campus
[Message delivered to LahoreMessage sent successfully!

Press any key to clear screen...]
```

Step 5: Testing Administration

1. In the **Central Server Admin Console**, select option **1** (**View Connected Campuses**) to see the active status of Lahore and Karachi.
2. Select option **2** (**Broadcast Announcement**) to send a message that will be received simultaneously by all connected clients via UDP.



```
==============================================================================
|                          CONNECTED CAMPUSES STATUS                         |
==============================================================================

  CAMPUS              LAST HEARTBEAT          STATUS
------------------------------------------------------------------------------
  Islamabad           03:59:41                [*] ONLINE
  Lahore              03:59:37                [*] ONLINE
```