# Computer Networks – Semester Project Fall 2025

## NU-Information Exchange System (FAST-NUCES Multi-Campus Network)

### 1. Objective

The primary objective of this project is to design and implement a system that models the information exchange between multiple FAST-NUCES campuses. This project is divided into two distinct parts:

* **Application Development:** Implement a multi-client communication system using C/C++ socket programming to handle inter-campus communication.
* **Network Architecture:** Design the underlying Wide Area Network (WAN) topology in Cisco Packet Tracer that connects the various FAST-NUCES campuses across Pakistan.

### 2. Learning Outcomes

Upon successful completion of this project, you will be able to:

* Implement a client-server architecture using C/C++ sockets.
* Utilize both TCP (for reliable, connection-oriented communication) and UDP (for connectionless, low-overhead broadcasts).
* Analyze the trade-offs between TCP and UDP for different application needs.
* Manage multiple concurrent client connections on a server.
* Design a multi-site WAN topology in Cisco Packet Tracer.
* Configure IP addressing, subnets, and basic routing (static or dynamic) to enable end-to-end connectivity between different networks.

### 4. Application Development (Lab - 20 Marks)

This part involves creating software for the NU-Information Exchange System using C/C++ socket programming. The system will be console-based (GUI can also be implemented).

#### 4.1. Application Architecture

The system will follow a client-server model designed to represent the multi-campus structure of FAST-NUCES.

* **Central Server (e.g., Islamabad Campus):** A single server application that acts as the central hub. It manages connections with all other "Campus Clients."
* **Campus Clients (e.g., Lahore, Karachi, Peshawar, CFD, Multan):** Each of these is a separate client application. From the server's perspective, they are "clients." From a user's perspective, they are the local "server" for that campus.
* **Department Users:** We will simulate users from departments (Admissions, Academics, IT, Sports) by having them interact with their local Campus Client via its console.

#### 4.2. Protocol Design

You must use a hybrid TCP/UDP approach:

* **TCP (Reliable Stream):** Used for all critical, one-to-one communication.
  + Campus Client authentication with the Central Server.
  + Sending direct messages from one campus to another (e.g., Lahore Admissions to Karachi Academics).
  + Receiving administrative commands.
* **UDP (Connectionless Datagram):** Used for non-critical, broadcast, or status-update messages.
  + Campus Clients periodically send a UDP "heartbeat" or "online status" packet to the server.
  + The Central Server (via the Admin module) broadcasts system-wide announcements to all campuses using UDP.

#### 4.3. Core Modules (C/C++ Implementation)

##### Central Server Module

* **Protocol Handling:** Must listen for TCP connections and UDP datagrams on different, designated ports.
* **Connection Management:** Must handle multiple Campus Clients connecting via TCP concurrently (e.g., using std::thread for each client or a select()/poll() loop).
* **Authentication:** When a Campus Client connects via TCP, it must send a simple, hard-coded credential (e.g., Campus:Lahore,Pass:NU-LHR-123). The server must validate this.
* **Message Routing:** Parse incoming TCP messages and route them to the correct destination Campus Client. For example, a message from Lahore intended for Karachi should be received by the server and forwarded to the Karachi client's TCP socket.
* **Status Monitoring:** Receive and display the UDP "online status" heartbeats from all campuses.
* **Logging:** Print all major events (connections, disconnections, message routes) to the console.

##### Campus Client Module

* **Connection Handling:** On startup, establish a TCP connection to the Central Server and perform authentication.
* **Hybrid Protocol Usage:**
  + Maintain the TCP connection for sending/receiving direct messages.
  + Create a separate UDP socket to periodically (e.g., every 10 seconds) send an "online" status packet to the server.
* **Console Interface:** Provide a simple text-based menu for a simulated department user (e.g., "Press 1 to send a message," "Press 2 to view received messages").
* **Functionality:**
  + Allow the user to send a message to a specific department at another campus (e.g., send <TargetCampus> <TargetDept> <Message>).
  + Listen for and display incoming messages from the server.
  + Listen for and display system-wide UDP broadcasts.

##### Admin Module (Can be part of the Server Module)

* A console interface on the Central Server.
* **System Monitoring:** Display a real-time list of all connected campuses and their last-seen UDP status.
* **Broadcast Announcements:** Allow the admin to type a message to be broadcast via UDP to all connected campuses.

#### 4.4. Assessment (Part 1 - Lab)

|  |  |  |
| --- | --- | --- |
| **Component** | **Description** | **Marks** |
| **TCP & UDP Socket Implementation** | Correctly initializes and uses both TCP and UDP sockets for their intended purpose. | 5 |
| **Server Concurrency** | Server can robustly handle multiple Campus Clients connecting and communicating at the same time. | 5 |
| **Inter-Campus Message Routing** | The server correctly routes a TCP message from one client (e.g., Lahore) to another (e.g., Karachi). | 5 |
| **Hybrid Protocol Use** | TCP is used for reliable messages, and UDP is correctly used for status/broadcasts. | 3 |
| **Code Quality & Readability** | Code is well-commented, organized, and follows good C/C++ practices. | 2 |
| **Total** |  | **20** |

### 6. Project Deliverables

* **Source Code (Part 1):** Fully commented C/C++ source code for the Central Server and the Campus Client.
* **Technical Report (Combined):** A single PDF document that includes:
  + **Application Section:** A brief explanation of your application's design, the custom protocol you used (e.g., how messages are formatted), and instructions on how to compile and run your code.
  + **Student Details:** Your name, roll number, and section.
  + **Self-Evaluation Form:** Attached with this document.

### 7. Advanced Challenges (Optional Bonus Tasks)

* **File Transfer:** Implement functionality to transfer a simple text file from one campus to another using your TCP connection.
* **Dynamic Routing:** If you used static routing for Part 2, implement a dynamic protocol like RIPv2 or OSPF in your Packet Tracer design instead.
* **Multi-Department Clients:** Instead of a single "Campus Client," make the server handle connections from individual department clients and route messages based on Campus+Department (e.g., Lahore-Admissions to Karachi-IT).

### 8. Group Structure

* Every group must be compromised of 03 students.
* Across the section groups are not allowed.