Department of Computer Science

FAST – National University of Computer & Emerging Sciences Chiniot-Faisalabad

Computer Networks – Semester Project Fall 2025

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

Team

Nimrah Shahid	23F-0734
Aqsa Ishaq	23F-0839
Ayesha Rauf	23F-0807

Date of Submission: 12/8/2025

Submitted to

Mr. Hassan Sami

Table of Contents

1. Introduction

This project implements a complete multi-campus information exchange system for FAST-NUCES, consisting of two integrated components:

- **Application Layer:** A C/C++ socket-based client-server system enabling inter-campus communication
- **Network Layer:** A WAN topology in Cisco Packet Tracer connecting multiple campus LANs

The system demonstrates real-world networking concepts including TCP/UDP protocols, concurrent server handling, WAN design, IP subnetting, and routing configuration.

2. Application Development

2.1 System Architecture

Our system follows a centralized client-server model:

- **Central Server:** Acts as the communication hub, managing all campus connections
- **Campus Clients:** Represent individual FAST-NUCES campuses (Lahore, Karachi, Peshawar, CFD, Multan, Islamabad)
- **Hybrid Protocol:** TCP for reliable messaging, UDP for heartbeats and broadcasts

2.2 Protocol Design

TCP Usage (Reliable Communication):

- Client authentication with the Central Server
- Direct inter-campus message routing
- Format: FROM:<campus>;TO:<target>;DEPT:<department>;MSG:<message>

UDP Usage (Lightweight Communication):

- Periodic heartbeat packets (every 10 seconds)
- Format: HEART|Campus:<name>|TS:<timestamp>
- Server-to-all broadcast announcements

- Format: ANNOUNCEMENT:<message>

2.3 Implementation Details

2.3.1 Central Server Module (server.cpp)

Key Features:

- **Multi-threaded Architecture:** Each client connection runs in a dedicated thread
- **Authentication System:** Hard-coded credentials validated on connection
  - Lahore: NU-LHR-123
  - Karachi: NU-KHI-123
  - Peshawar: NU-PSH-123
  - CFD: NU-CFD-123
  - Multan: NU-MTN-123
  - Islamabad: NU-ISB-123

Core Functionality:

1. **TCP Server (Port 9000):**
   - Accepts multiple concurrent connections
   - Authenticates each campus client
   - Routes messages between campuses
   - Handles graceful disconnections
2. **UDP Listener (Port 9001):**
   - Receives heartbeat packets from all campuses
   - Updates last-seen timestamps
   - Maintains campus status information
3. **Admin Console:**
   - status - Displays all campus connection states and last heartbeat times
   - announce <message> - Broadcasts UDP announcement to all campuses
   - exit - Exits admin console (server continues running)

Message Routing Logic:

Client A → Server (receives message)

→ Server (parses destination)

→ Server (forwards to Client B's TCP socket)

→ Client B (receives message)

Data Structures:

- map<string, int> campusSockets - Maps campus names to TCP socket descriptors
- map<string, time_t> lastSeen - Tracks last heartbeat timestamp
- mutex mtx - Synchronizes access to shared data

2.3.2 Campus Client Module (client.cpp)

Key Features:

- **Multi-threaded Design:**
    - Main thread: User interface and message sending
    - Thread 1: TCP listener for incoming routed messages
    - Thread 2: UDP heartbeat sender (10-second intervals)
    - Thread 3: UDP listener for server announcements

Functionality:

1. **Startup Sequence:**
    - User enters campus name
    - Establishes TCP connection to server
    - Sends authentication credentials
    - Waits for AUTH_OK response
2. **User Menu:**
    - Option 1: Send message to another campus
        - Prompts for: Target Campus, Target Department, Message
        - Formats and sends via TCP

- Option 2: Logout and exit
    - Sends LOGOUT; signal
    - Closes all connections gracefully

3. **Background Operations:**
    - Continuously listens for incoming TCP messages
    - Sends UDP heartbeat every 10 seconds
    - Listens for UDP broadcast announcements

2.4 Compilation and Execution Instructions

Compilation:

```
# Compile Server
cd ~/NU-Information-Exchange/server
g++ -std=c++11 server.cpp -o server -pthread

# Compile Client
cd ~/NU-Information-Exchange/client
g++ -std=c++11 client.cpp -o client -pthread
```

Execution:

```
# Terminal 1: Start Server
./server

# Terminal 2: Start Lahore Campus
./client
# Enter: Lahore

# Terminal 3: Start Karachi Campus
./client
```

```
# Enter: Karachi


# Repeat for other campuses...
```

Testing the System:

1. Start the server
2. Launch multiple client instances (different terminals)
3. Authenticate each client with a campus name
4. Use admin console to check status
5. Send messages between campuses
6. Broadcast announcements from admin console
7. Observe heartbeat logs on server

2.5 Code Quality

Our implementation demonstrates:

- **Clean Code Structure:** Modular functions with clear responsibilities
- **Comprehensive Comments:** Every major section documented
- **Error Handling:** Proper socket error checking and connection validation
- **Thread Safety:** Mutex-protected shared data structures
- **Resource Management:** Proper socket closure and thread cleanup

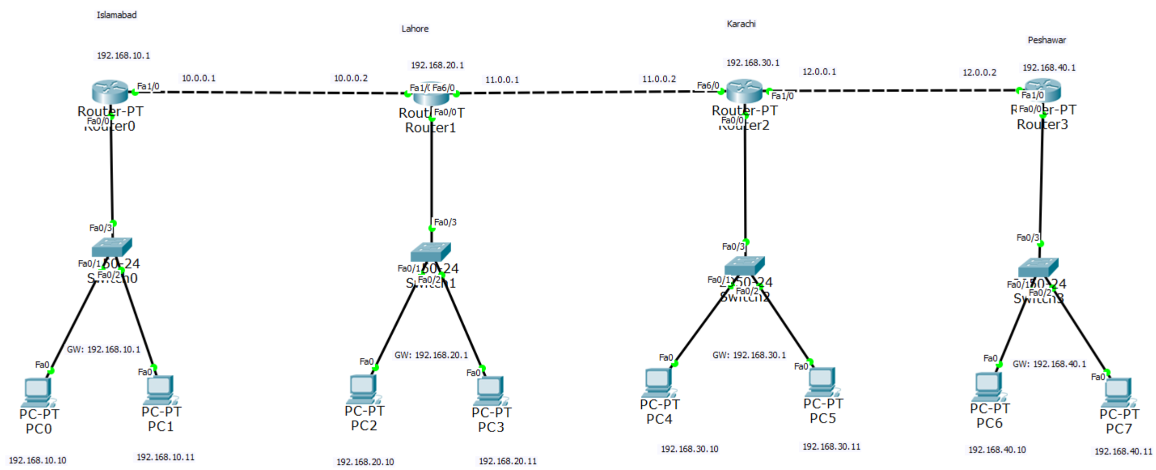
3. Network Architecture

3.1 Topology Overview

Our WAN design connects four FAST-NUCES campuses:

1. **Islamabad Campus** - Central hub
2. **Lahore Campus**
3. **Karachi Campus**

4. **Peshawar Campus**



3.2 Network Design

3.2.1 Campus LAN Structure

Each campus follows this architecture:

- **1 x Router** (Cisco 2911)
- **1 x Switch** (Cisco 2960-24TT)
- **2 x End Devices** (PCs representing departments)

3.2.2 WAN Connections

Routers are interconnected via serial/Ethernet links forming a linear topology:

Islamabad ↔ Lahore ↔ Karachi ↔ Peshawar

3.3 IP Addressing Scheme

Campus LANs:

Campus	**Network**	**Subnet Mask**	**Gateway**	**PC Range**
Islamabad	192.168.10.0/24	255.255.255.0	192.168.10.1	.10 - .11
Lahore	192.168.20.0/24	255.255.255.0	192.168.20.1	.10 - .11
Karachi	192.168.30.0/24	255.255.255.0	192.168.30.1	.10 - .11

Campus	Network	Subnet Mask	Gateway	PC Range
Peshawar	192.168.40.0/24	255.255.255.0	192.168.40.1	.10 - .11

WAN Links:

Link	Network	Router A IP	Router B IP
Islamabad ↔ Lahore	10.0.0.0/30	10.0.0.1	10.0.0.2
Lahore ↔ Karachi	11.0.0.0/30	11.0.0.1	11.0.0.2
Karachi ↔ Peshawar	12.0.0.0/30	12.0.0.1	12.0.0.2

3.4 Routing Configuration

3.4 Routing Configuration

We implemented **RIP (Routing Information Protocol) Version 2** for dynamic routing and automatic route discovery.

Routing Strategy:

RIP is a distance-vector routing protocol that automatically shares routing information between routers. This ensures:

- **Automatic Route Discovery:** Routers learn about remote networks without manual configuration
- **Dynamic Adaptation:** Network adjusts automatically if links go down
- **Simplified Management:** No need to manually configure every route on every router
- **Hop-Count Metric:** RIP uses hop count to determine the best path (maximum 15 hops)

RIP Configuration Process:

Each router was configured with RIP version 2 using the following commands:

Step 1: Enable RIP

Router(config)# router rip

Step 2: Specify RIP Version 2

Router(config-router)# version 2

Step 3: Advertise Connected Networks

Router(config-router)# network [network-address]

Step 4: Disable Auto-Summary (for proper subnet handling)

Router(config-router)# no auto-summary

Why RIP Version 2?

- **VLSM Support:** Supports Variable Length Subnet Masking (unlike RIPv1)
- **Subnet Mask Information:** Sends subnet mask with routing updates
- **Classless Routing:** Better support for modern IP addressing
- **Multicast Updates:** Uses multicast address 224.0.0.9 (more efficient than broadcast)

Router-Specific Configurations:

Islamabad Router (Central Hub):

Router(config)# router rip

Router(config-router)# version 2

Router(config-router)# network 192.168.10.0

Router(config-router)# network 10.0.0.0

Router(config-router)# no auto-summary

- Advertises: Campus LAN (192.168.10.0/24) and WAN link to Lahore (10.0.0.0/30)

Lahore Router:

Router(config)# router rip

Router(config-router)# version 2

Router(config-router)# network 192.168.20.0

Router(config-router)# network 10.0.0.0

Router(config-router)# network 11.0.0.0

Router(config-router)# no auto-summary

- Advertises: Campus LAN (192.168.20.0/24), WAN link to Islamabad (10.0.0.0/30), and WAN link to Karachi (11.0.0.0/30)

Karachi Router:

Router(config)# router rip

Router(config-router)# version 2

Router(config-router)# network 192.168.30.0

Router(config-router)# network 11.0.0.0

Router(config-router)# network 12.0.0.0

Router(config-router)# no auto-summary

- Advertises: Campus LAN (192.168.30.0/24), WAN link to Lahore (11.0.0.0/30), and WAN link to Peshawar (12.0.0.0/30)

Peshawar Router:

Router(config)# router rip

Router(config-router)# version 2

Router(config-router)# network 192.168.40.0

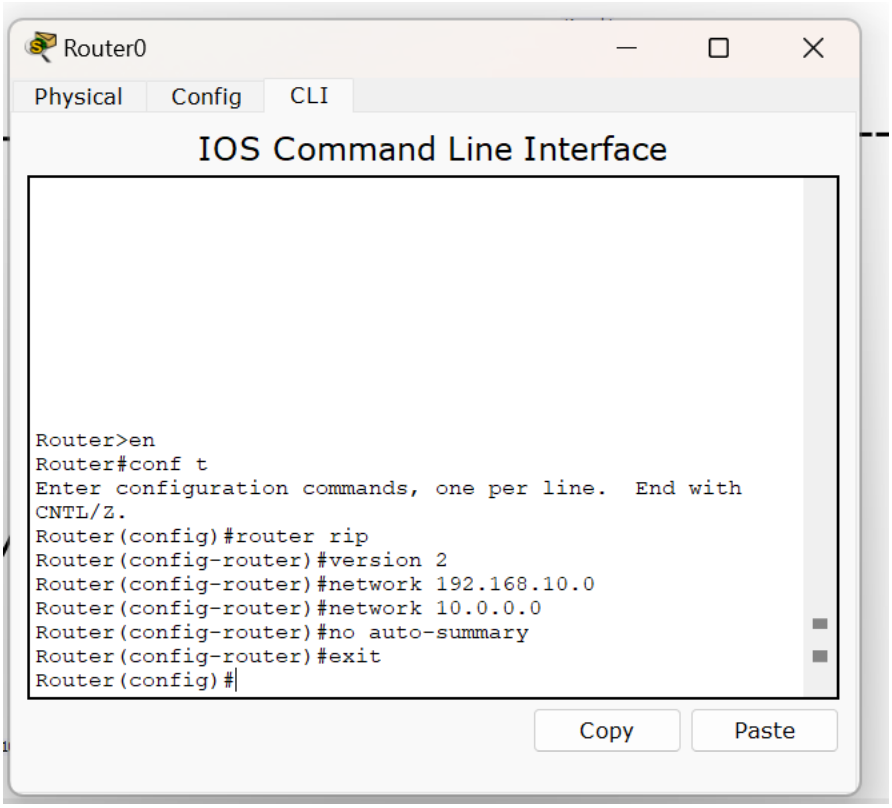
Router(config-router)# network 12.0.0.0

Router(config-router)# no auto-summary

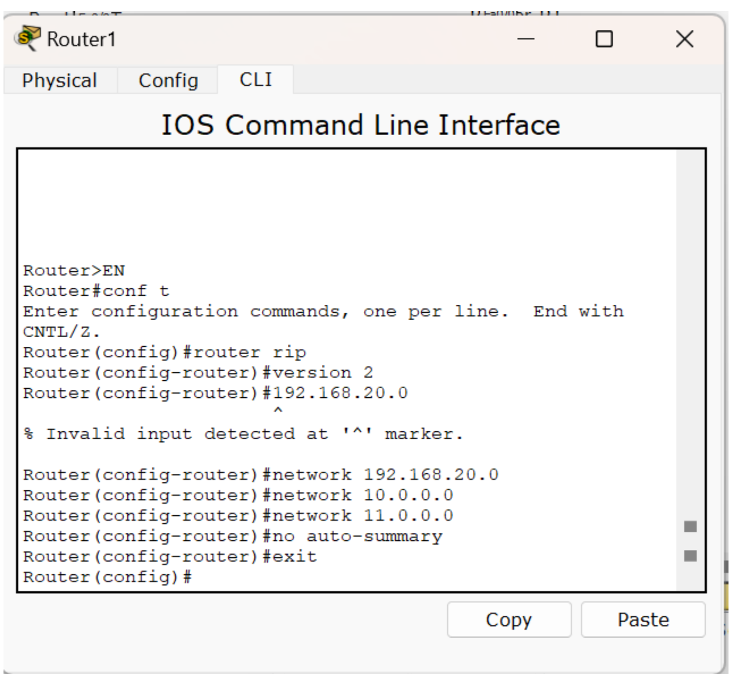
- Advertises: Campus LAN (192.168.40.0/24) and WAN link to Karachi (12.0.0.0/30)

Router Configuration Screenshots:

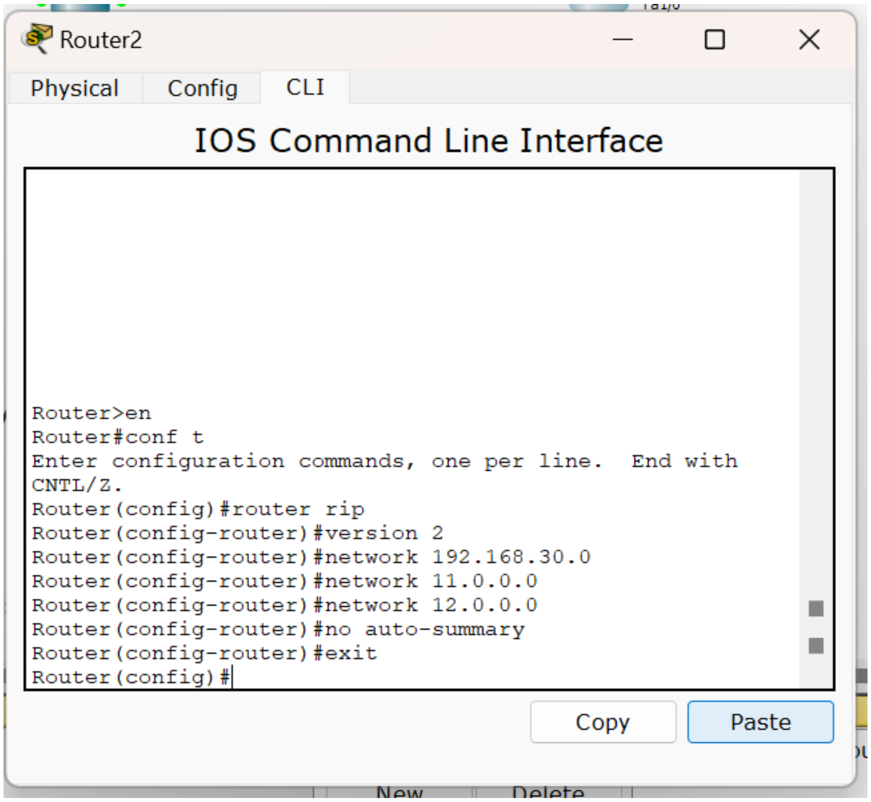
Islamabad Router RIP Configuration:



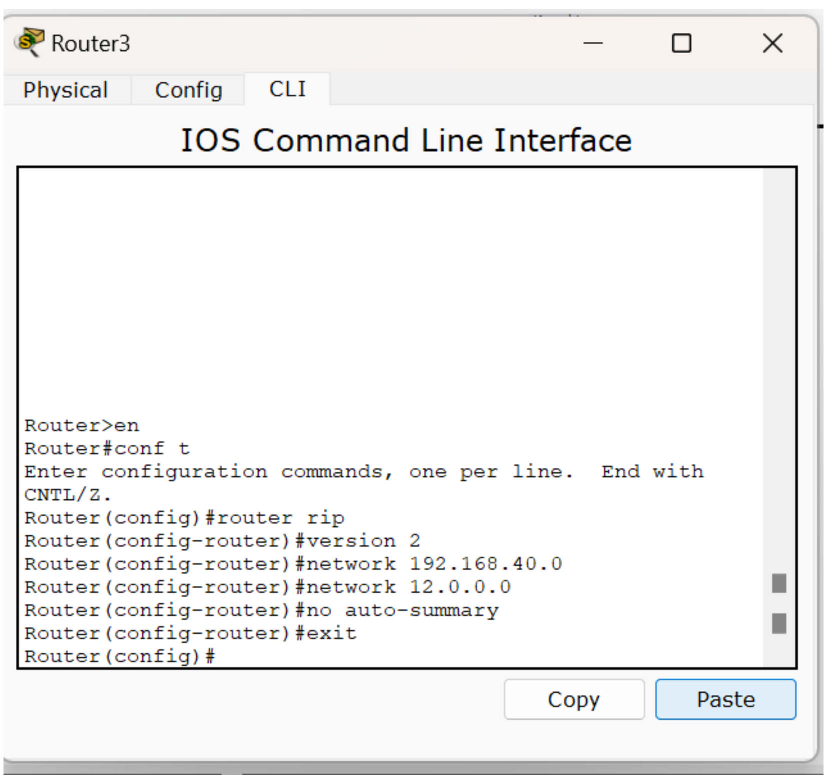
Lahore Router RIP Configuration:



Karachi Router RIP Configuration:



Peshawar Router RIP Configuration:



RIP Routing Table Verification:

After RIP converges (typically 30-90 seconds), each router learns about all remote networks automatically. You can verify this using:

Router# show ip route

The routing table will show entries marked with "R" (indicating RIP-learned routes) for all remote campus networks.

Advantages of Our RIP Implementation:

1. **Plug-and-Play:** New campuses can be added by simply configuring RIP
2. **Fault Tolerance:** If a link fails, RIP automatically finds alternate paths
3. **Easy Maintenance:** No need to update every router when topology changes
4. **Consistency:** All routers maintain synchronized routing information

Additional Router Configuration Verification:

Realtime

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
	Successful	PC0	PC1	ICMP		0.000	N	0	(edit)
	Successful	PC2	PC3	ICMP		0.000	N	1	(edit)

Realtime

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
	Successful	PC4	PC5	ICMP		0.000	N	2	(edit)
	Successful	PC6	PC7	ICMP		0.000	N	3	(edit)

Realtime

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
	Successful	PC0	PC2	ICMP		0.000	N	4	(edit)
	Successful	PC0	PC3	ICMP		0.000	N	5	(edit)

Realtime

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
	Successful	PC1	PC2	ICMP		0.000	N	6	(edit)
	Successful	PC1	PC3	ICMP		0.000	N	7	(edit)

Realtime

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
	Successful	PC2	PC4	ICMP		0.000	N	8	(edit)
	Successful	PC2	PC5	ICMP		0.000	N	9	(edit)

Realtime

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
	Successful	PC3	PC4	ICMP		0.000	N	10	(edit)
	Successful	PC3	PC5	ICMP		0.000	N	11	(edit)

Realtime

Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit
	Successful	PC4	PC6	ICMP		0.000	N	12	(edit)
	Successful	PC5	PC7	ICMP		0.000	N	13	(edit)

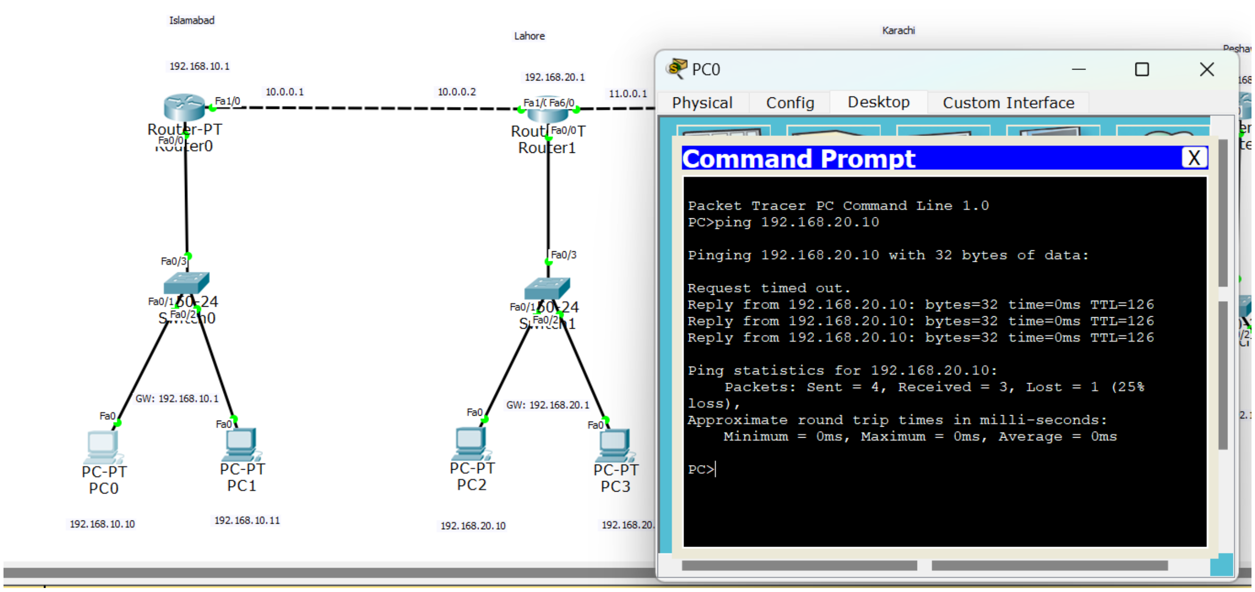
3.5 Verification and Testing

Connectivity Tests:

Test 1: Islamabad PC0 → Lahore PC2

PC> ping 192.168.20.10

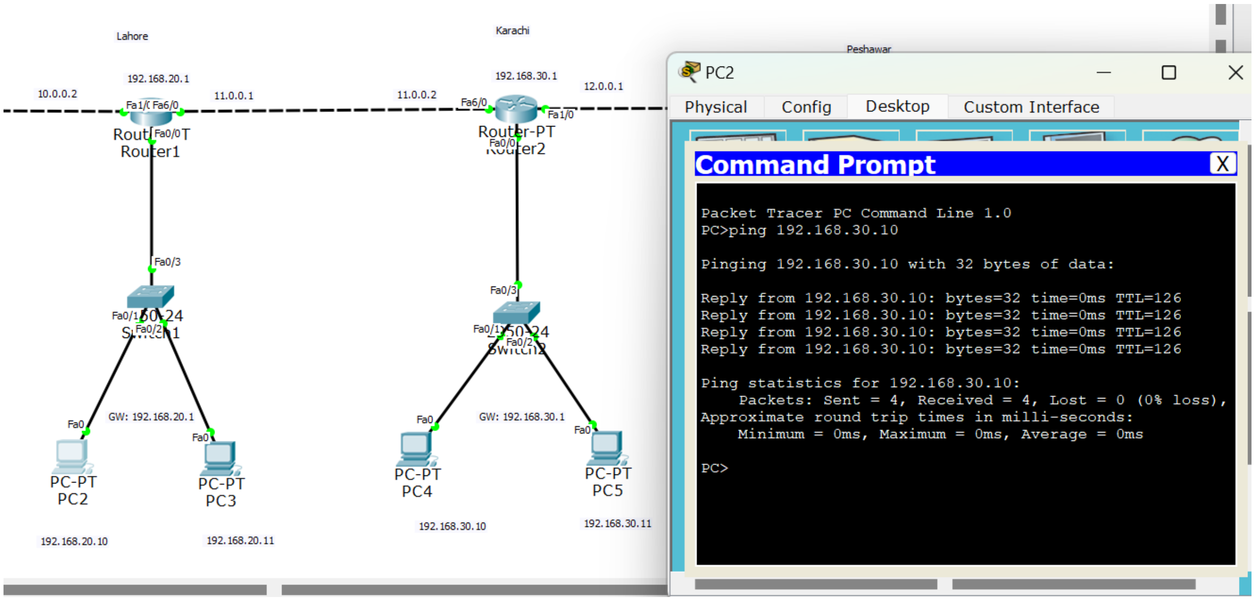
Result: Success (Reply received)



Test 2: Lahore PC2 → Karachi PC4

PC> ping 192.168.30.10

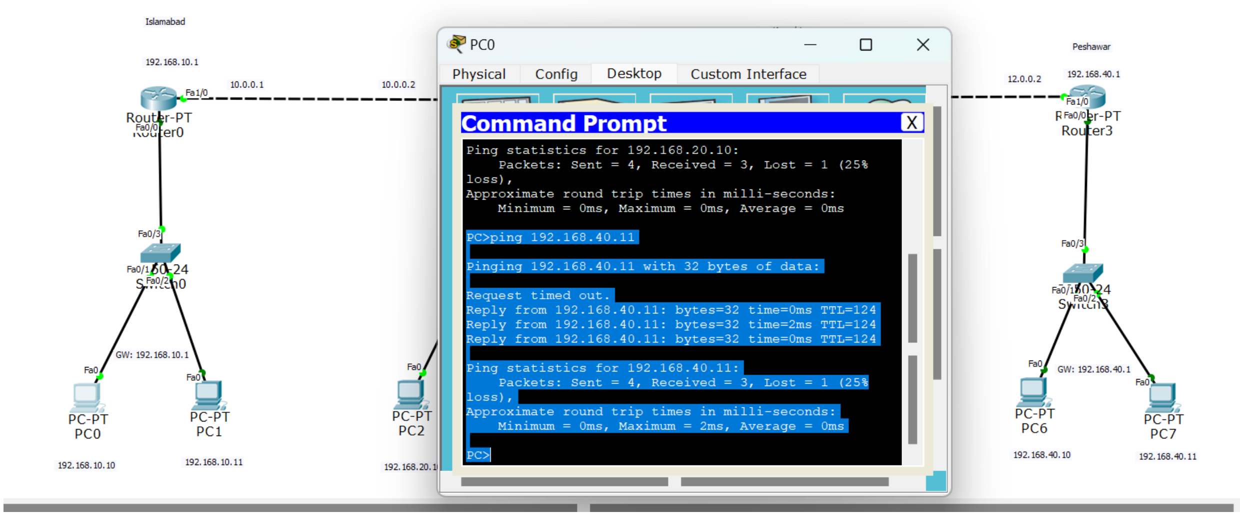
Result: Success (Reply received)



Test 3: Islamabad PC0 → Peshawar PC7

PC> ping 192.168.40.11

Result: Success (Reply received)



All end-to-end connectivity tests passed, confirming proper:

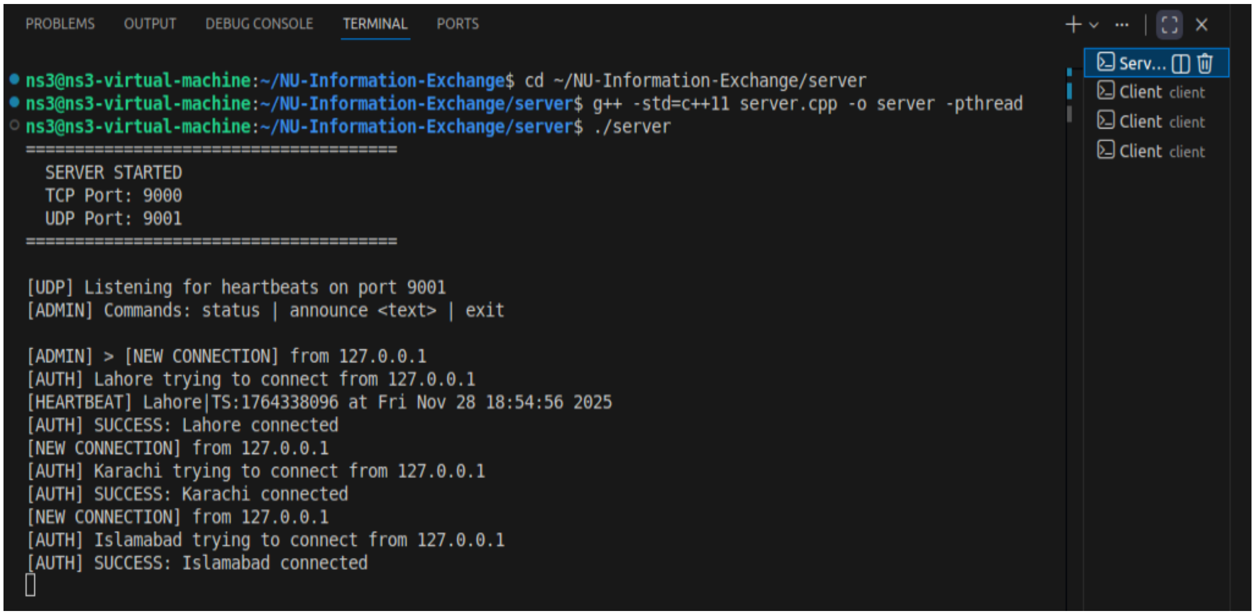
- IP addressing
- Default gateway configuration
- RIP dynamic routing setup
- Inter-campus communication

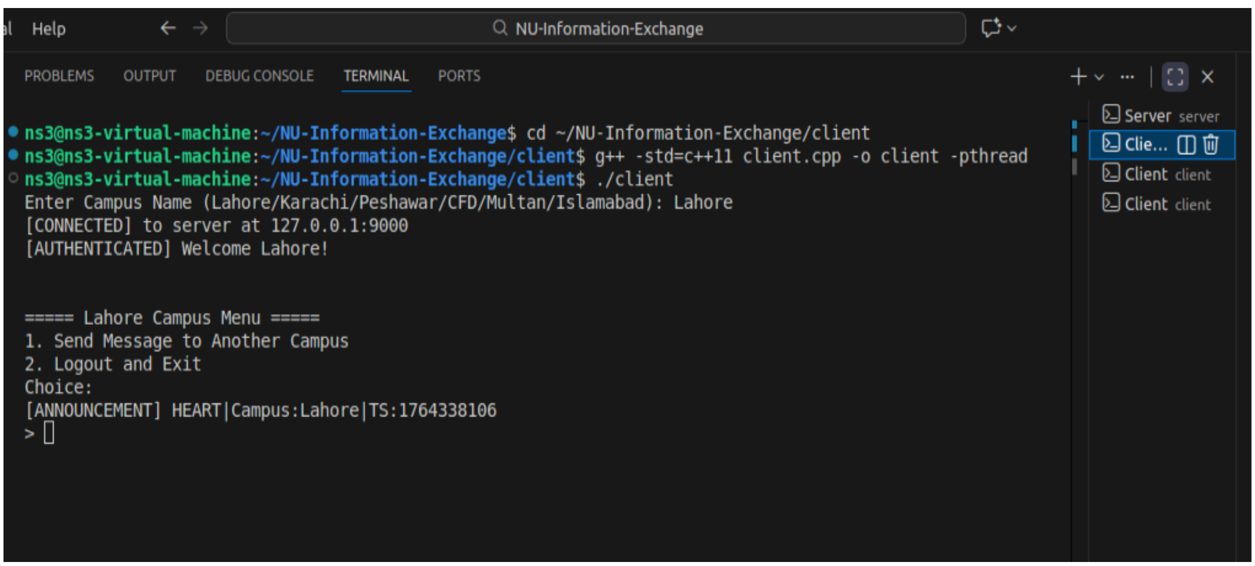
4. Testing and Validation

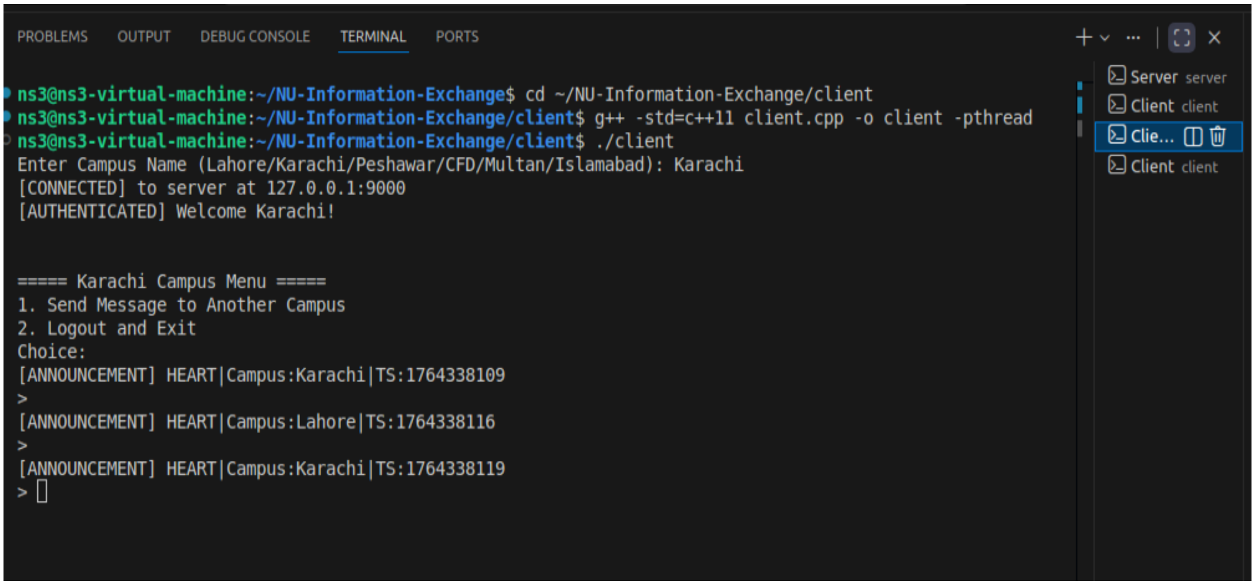
4.1 Application Testing

Test Case 1: Multi-Client Authentication

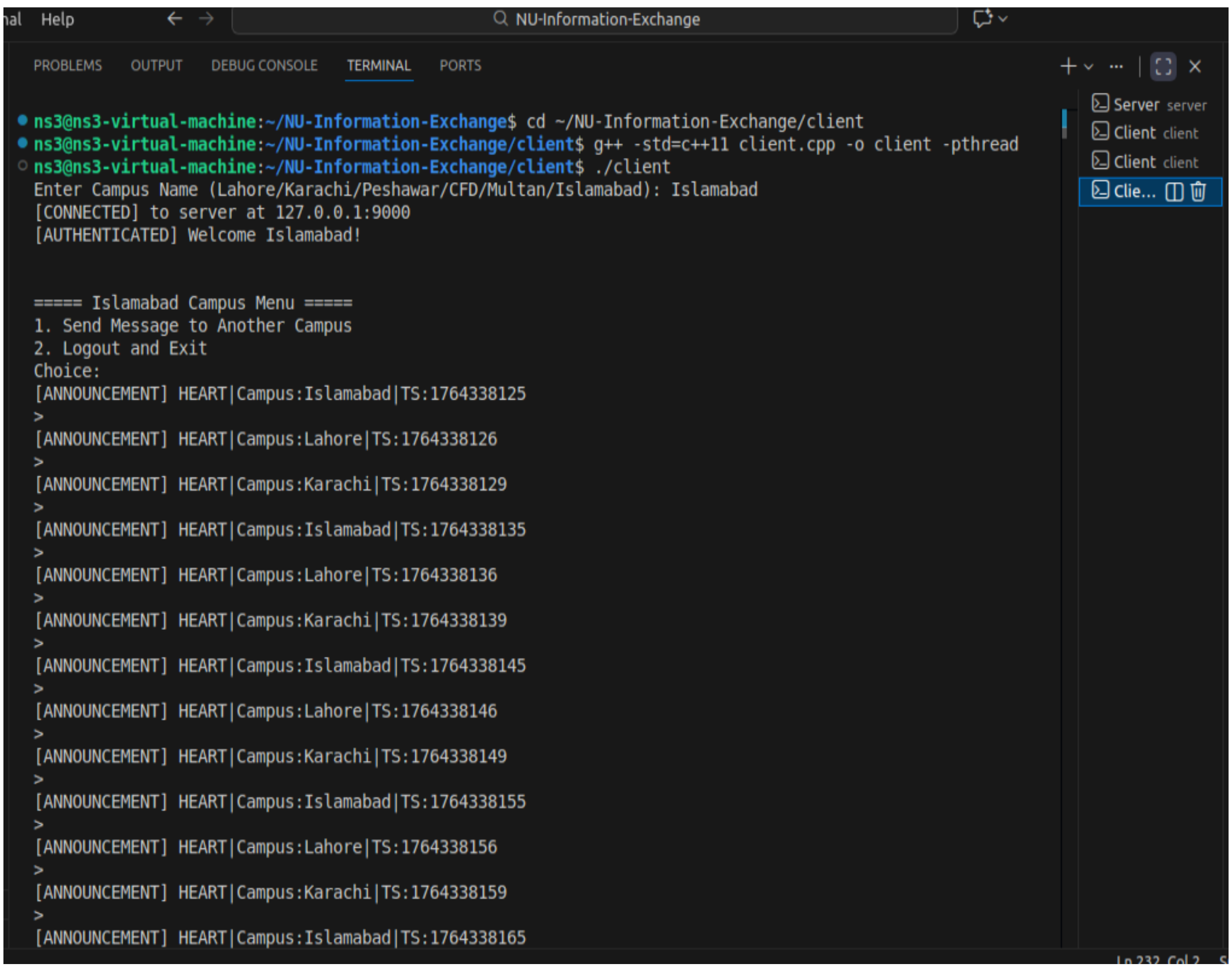
- **Objective:** Verify server handles multiple simultaneous connections
- **Procedure:**
    1. Start server
    2. Connect 3 campus clients simultaneously
    3. Authenticate each with correct credentials
- **Result:** ✓ All clients authenticated successfully
- **Evidence:**


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
ns3@ns3-virtual-machine:~/NU-Information-Exchange$ cd ~/NU-Information-Exchange/server
ns3@ns3-virtual-machine:~/NU-Information-Exchange/server$ g++ -std=c++11 server.cpp -o server -pthread
ns3@ns3-virtual-machine:~/NU-Information-Exchange/server$ ./server
======================================
SERVER STARTED
TCP Port: 9000
UDP Port: 9001
======================================
[UDP] Listening for heartbeats on port 9001
[ADMIN] Commands: status | announce <text> | exit
[ADMIN] > [NEW CONNECTION] from 127.0.0.1
[AUTH] Lahore trying to connect from 127.0.0.1
[HEARTBEAT] Lahore|TS:1764338096 at Fri Nov 28 18:54:56 2025
[AUTH] SUCCESS: Lahore connected
[NEW CONNECTION] from 127.0.0.1
[AUTH] Karachi trying to connect from 127.0.0.1
[AUTH] SUCCESS: Karachi connected
[NEW CONNECTION] from 127.0.0.1
[AUTH] Islamabad trying to connect from 127.0.0.1
[AUTH] SUCCESS: Islamabad connected
Serv...
Client client
Client client
Client client


Help
NU-Information-Exchange
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
ns3@ns3-virtual-machine:~/NU-Information-Exchange$ cd ~/NU-Information-Exchange/client
ns3@ns3-virtual-machine:~/NU-Information-Exchange/client$ g++ -std=c++11 client.cpp -o client -pthread
ns3@ns3-virtual-machine:~/NU-Information-Exchange/client$ ./client
Enter Campus Name (Lahore/Karachi/Peshawar/CFD/Multan/Islamabad): Lahore
[CONNECTED] to server at 127.0.0.1:9000
[AUTHENTICATED] Welcome Lahore!
===== Lahore Campus Menu =====
1. Send Message to Another Campus
2. Logout and Exit
Choice:
[ANNOUNCEMENT] HEART|Campus:Lahore|TS:1764338106
>
Server server
Clie...
Client client
Client client



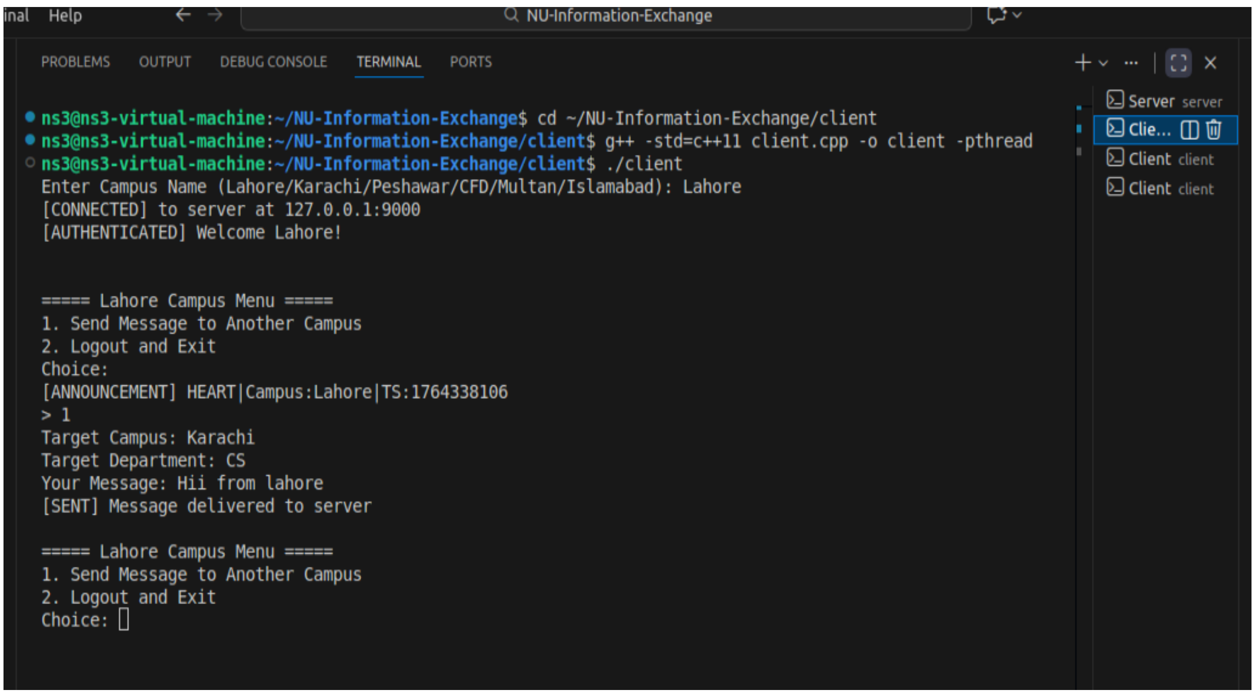
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
ns3@ns3-virtual-machine:~/NU-Information-Exchange$ cd ~/NU-Information-Exchange/client
ns3@ns3-virtual-machine:~/NU-Information-Exchange/client$ g++ -std=c++11 client.cpp -o client -pthread
ns3@ns3-virtual-machine:~/NU-Information-Exchange/client$ ./client
Enter Campus Name (Lahore/Karachi/Peshawar/CFD/Multan/Islamabad): Karachi
[CONNECTED] to server at 127.0.0.1:9000
[AUTHENTICATED] Welcome Karachi!


===== Karachi Campus Menu =====
1. Send Message to Another Campus
2. Logout and Exit
Choice:
[ANNOUNCEMENT] HEART|Campus:Karachi|TS:1764338109
>
[ANNOUNCEMENT] HEART|Campus:Lahore|TS:1764338116
>
[ANNOUNCEMENT] HEART|Campus:Karachi|TS:1764338119
>
```

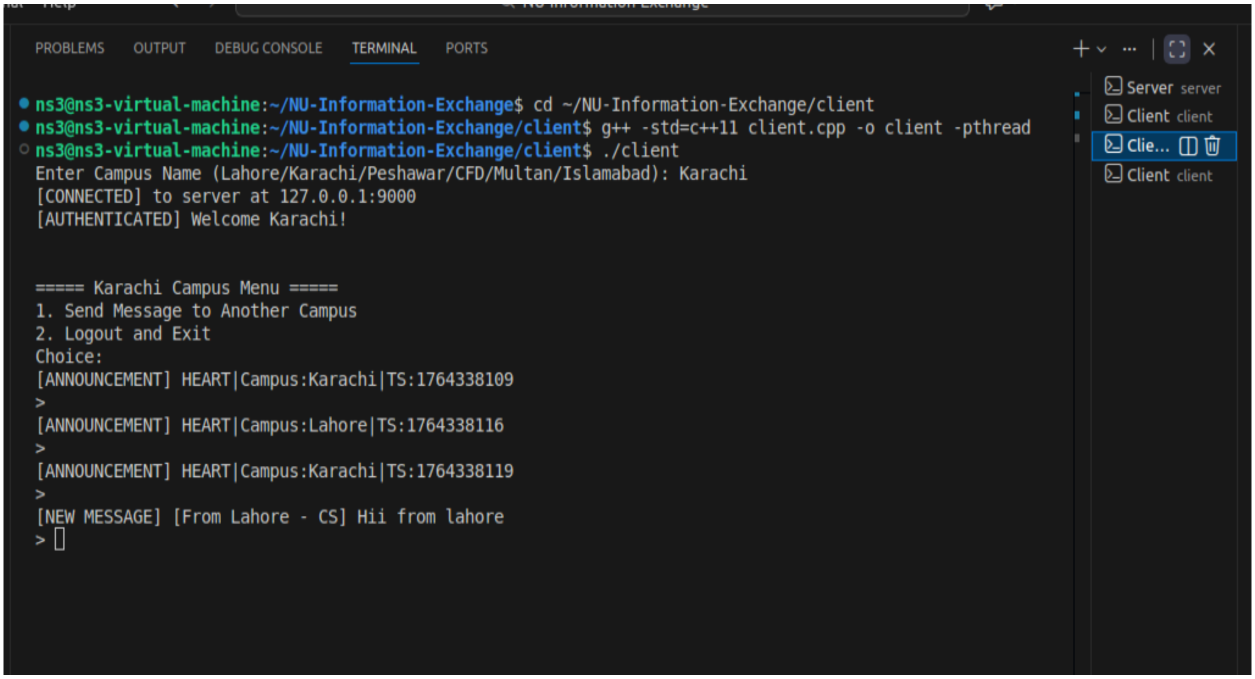



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
ns3@ns3-virtual-machine:~/NU-Information-Exchange$ cd ~/NU-Information-Exchange/client
ns3@ns3-virtual-machine:~/NU-Information-Exchange/client$ g++ -std=c++11 client.cpp -o client -pthread
ns3@ns3-virtual-machine:~/NU-Information-Exchange/client$ ./client
Enter Campus Name (Lahore/Karachi/Peshawar/CFD/Multan/Islamabad): Islamabad
[CONNECTED] to server at 127.0.0.1:9000
[AUTHENTICATED] Welcome Islamabad!


===== Islamabad Campus Menu =====
1. Send Message to Another Campus
2. Logout and Exit
Choice:
[ANNOUNCEMENT] HEART|Campus:Islamabad|TS:1764338125
>
[ANNOUNCEMENT] HEART|Campus:Lahore|TS:1764338126
>
[ANNOUNCEMENT] HEART|Campus:Karachi|TS:1764338129
>
[ANNOUNCEMENT] HEART|Campus:Islamabad|TS:1764338135
>
[ANNOUNCEMENT] HEART|Campus:Lahore|TS:1764338136
>
[ANNOUNCEMENT] HEART|Campus:Karachi|TS:1764338139
>
[ANNOUNCEMENT] HEART|Campus:Islamabad|TS:1764338145
>
[ANNOUNCEMENT] HEART|Campus:Lahore|TS:1764338146
>
[ANNOUNCEMENT] HEART|Campus:Karachi|TS:1764338149
>
[ANNOUNCEMENT] HEART|Campus:Islamabad|TS:1764338155
>
[ANNOUNCEMENT] HEART|Campus:Lahore|TS:1764338156
>
[ANNOUNCEMENT] HEART|Campus:Karachi|TS:1764338159
>
[ANNOUNCEMENT] HEART|Campus:Islamabad|TS:1764338165
```

Test Case 2: Inter-Campus Messaging

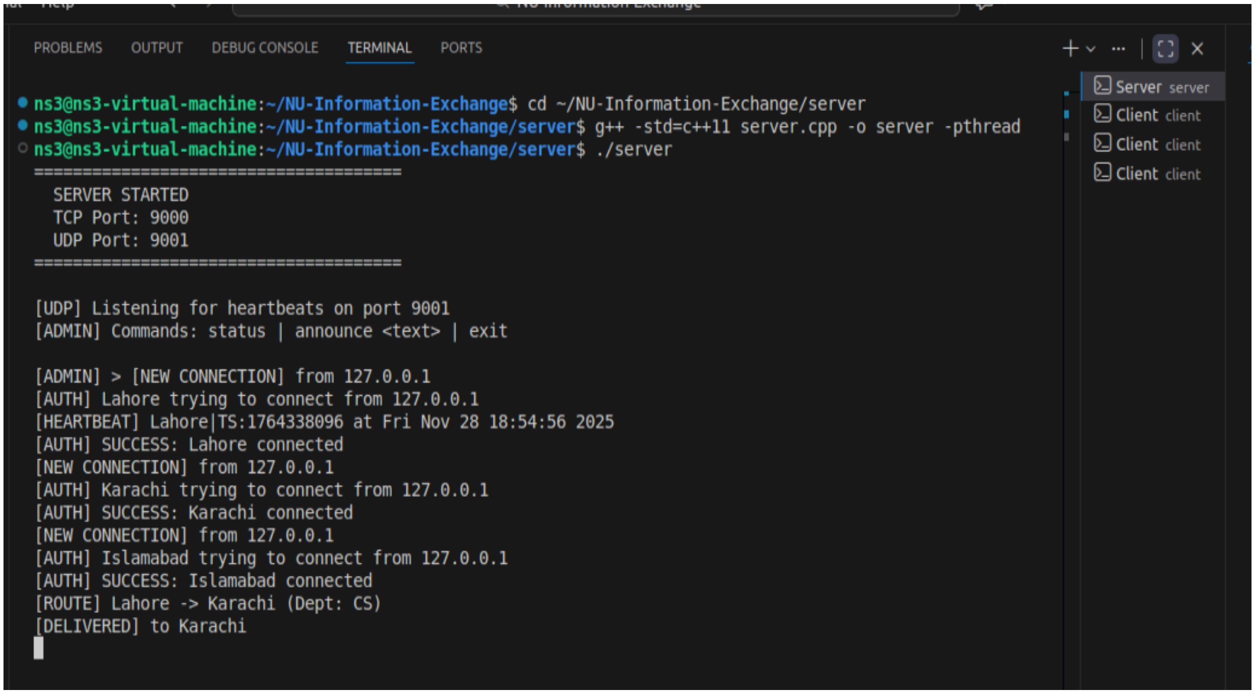
- **Objective:** Verify TCP message routing
- **Procedure:**
    1. Lahore sends message to Karachi's Admissions department
    2. Verify Karachi receives the message
- **Result:** ✓ Message routed and delivered correctly
- **Evidence:**





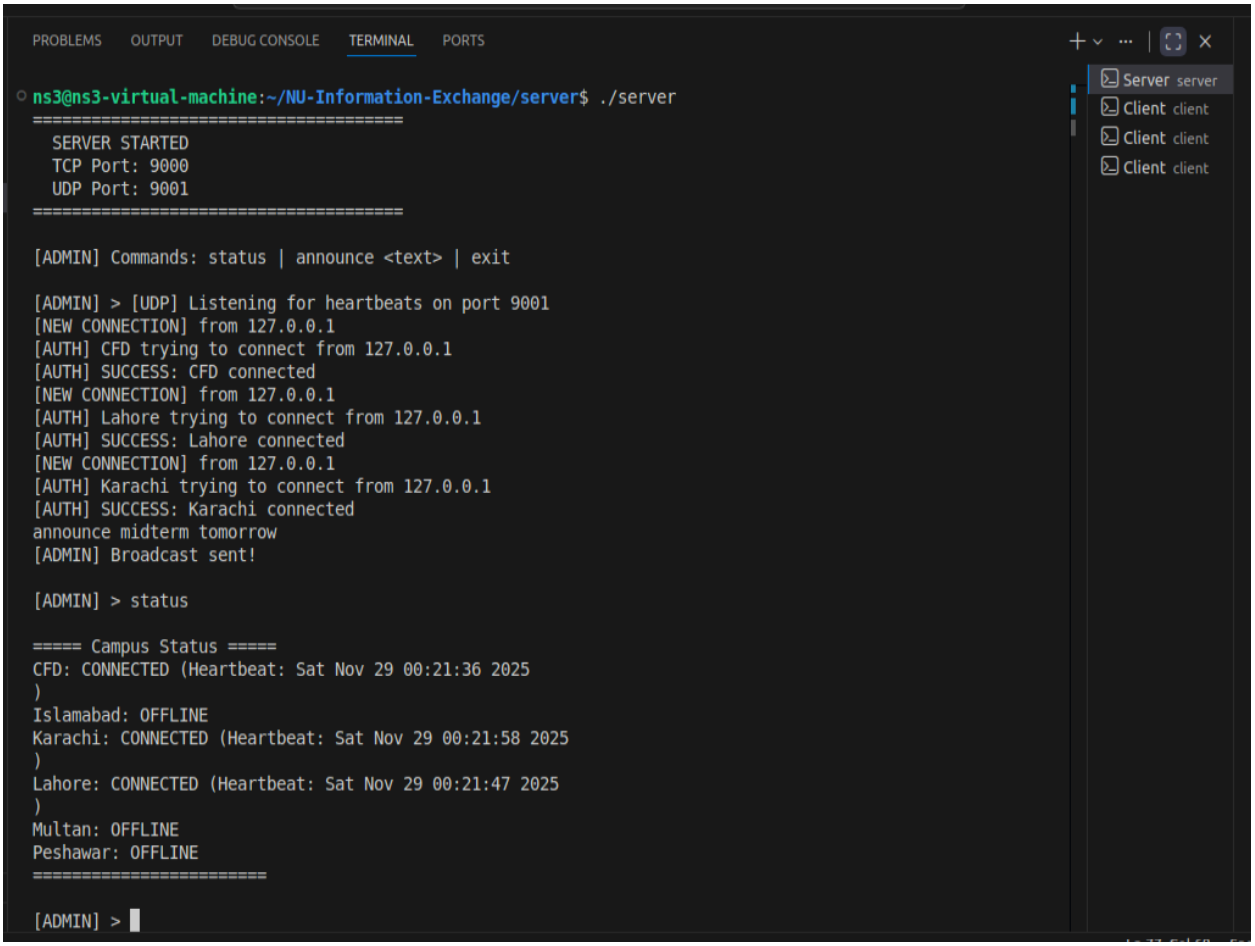
Test Case 3: UDP Heartbeat

- **Objective:** Verify periodic status updates
- **Procedure:**
  1. Connect a client
  2. Observe server console for heartbeat logs
  3. Check admin status command
- **Result:** ✓ Heartbeats received every 10 seconds
- **Evidence:**

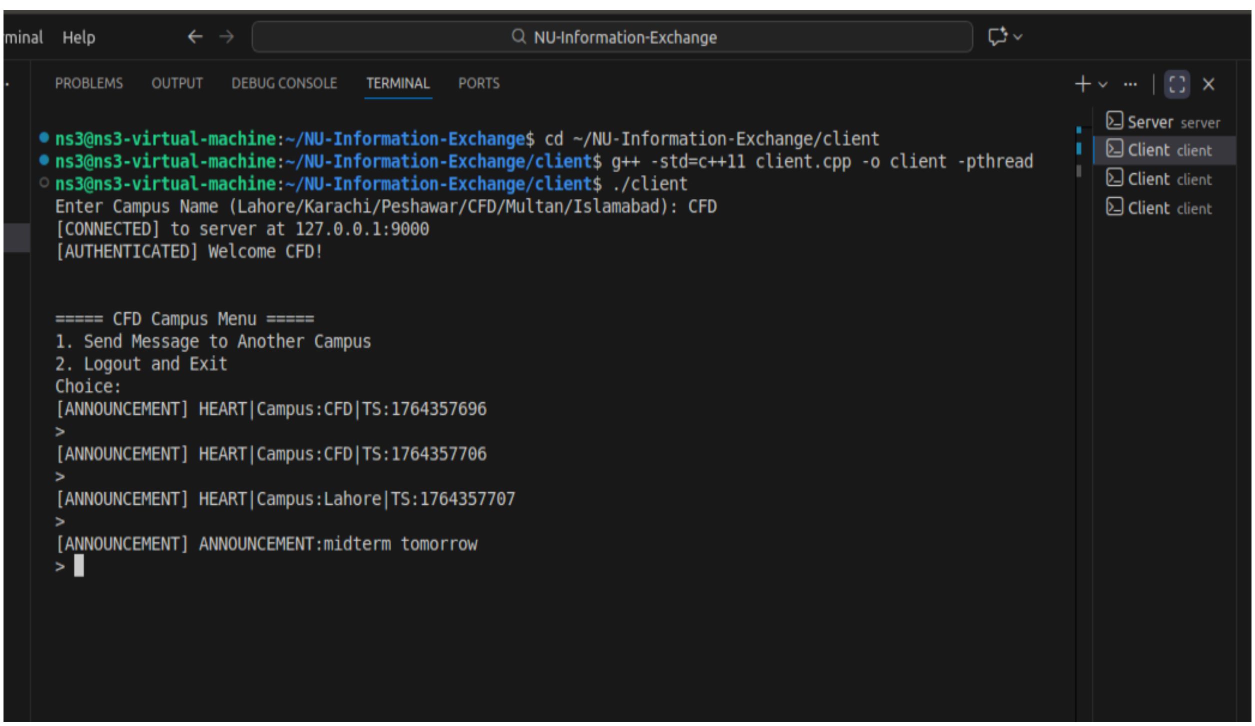


Test Case 4: Admin Broadcast

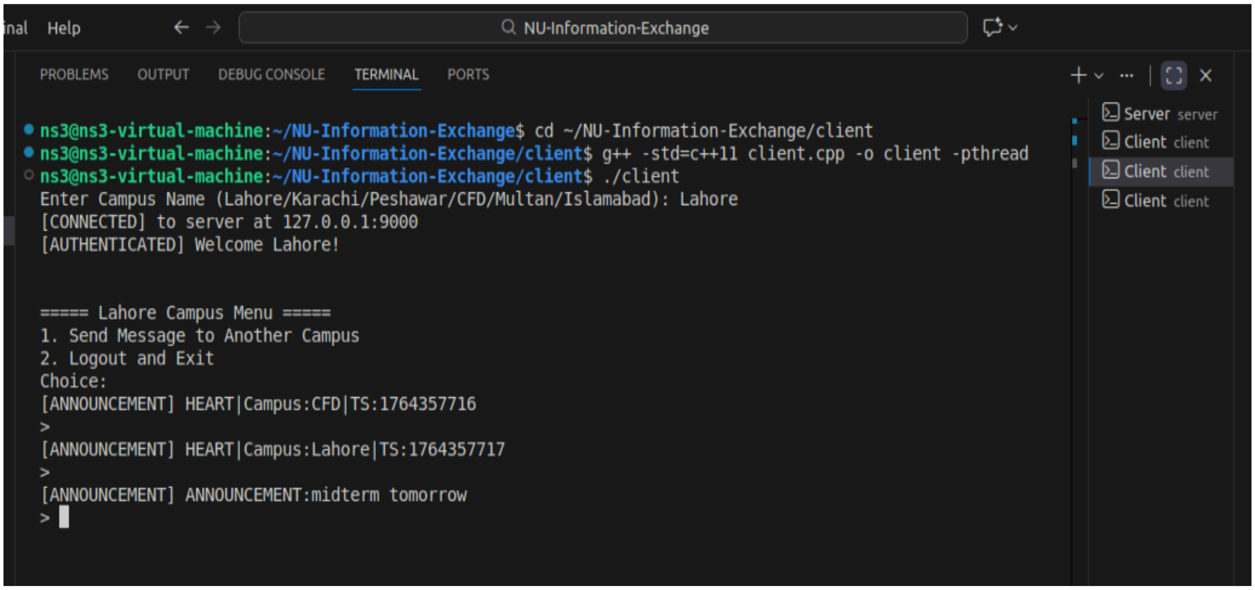
- **Objective:** Verify UDP announcements reach all clients
- **Procedure:**
    1. Connect 2+ clients
    2. Admin sends announcement
    3. Verify all clients display announcement
- **Result:** ✓ All clients received broadcast
- **Evidence:**

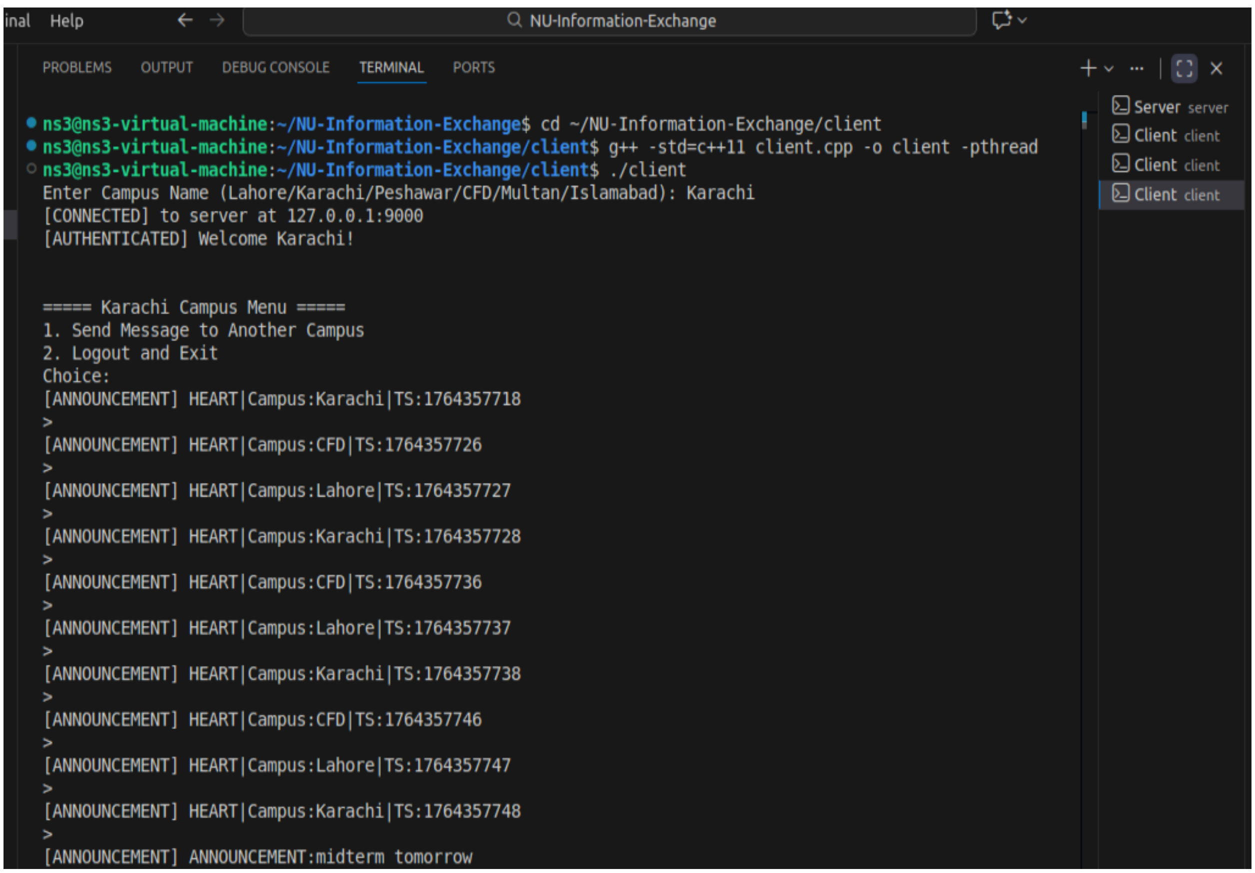



```
ns3@ns3-virtual-machine:~/NU-Information-Exchange/server$ ./server
======================================
  SERVER STARTED
  TCP Port: 9000
  UDP Port: 9001
======================================

[ADMIN] Commands: status | announce <text> | exit

[ADMIN] > [UDP] Listening for heartbeats on port 9001
[NEW CONNECTION] from 127.0.0.1
[AUTH] CFD trying to connect from 127.0.0.1
[AUTH] SUCCESS: CFD connected
[NEW CONNECTION] from 127.0.0.1
[AUTH] Lahore trying to connect from 127.0.0.1
[AUTH] SUCCESS: Lahore connected
[NEW CONNECTION] from 127.0.0.1
[AUTH] Karachi trying to connect from 127.0.0.1
[AUTH] SUCCESS: Karachi connected
announce midterm tomorrow
[ADMIN] Broadcast sent!

[ADMIN] > status

===== Campus Status =====
CFD: CONNECTED (Heartbeat: Sat Nov 29 00:21:36 2025
)
Islamabad: OFFLINE
Karachi: CONNECTED (Heartbeat: Sat Nov 29 00:21:58 2025
)
Lahore: CONNECTED (Heartbeat: Sat Nov 29 00:21:47 2025
)
Multan: OFFLINE
Peshawar: OFFLINE
========================

[ADMIN] >
```



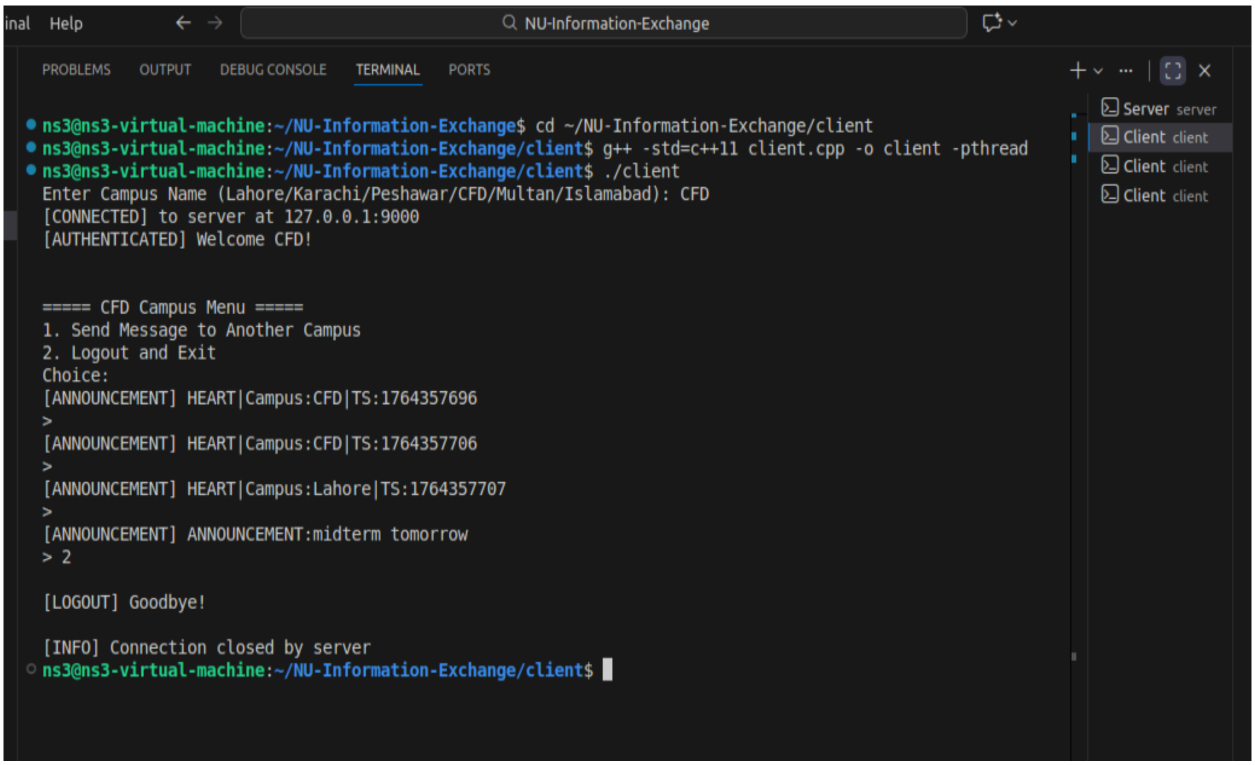

```
ns3@ns3-virtual-machine:~/NU-Information-Exchange$ cd ~/NU-Information-Exchange/client
ns3@ns3-virtual-machine:~/NU-Information-Exchange/client$ g++ -std=c++11 client.cpp -o client -pthread
ns3@ns3-virtual-machine:~/NU-Information-Exchange/client$ ./client
Enter Campus Name (Lahore/Karachi/Peshawar/CFD/Multan/Islamabad): CFD
[CONNECTED] to server at 127.0.0.1:9000
[AUTHENTICATED] Welcome CFD!


===== CFD Campus Menu =====
1. Send Message to Another Campus
2. Logout and Exit
Choice:
[ANNOUNCEMENT] HEART|Campus:CFD|TS:1764357696
>
[ANNOUNCEMENT] HEART|Campus:CFD|TS:1764357706
>
[ANNOUNCEMENT] HEART|Campus:Lahore|TS:1764357707
>
[ANNOUNCEMENT] ANNOUNCEMENT:midterm tomorrow
>
```

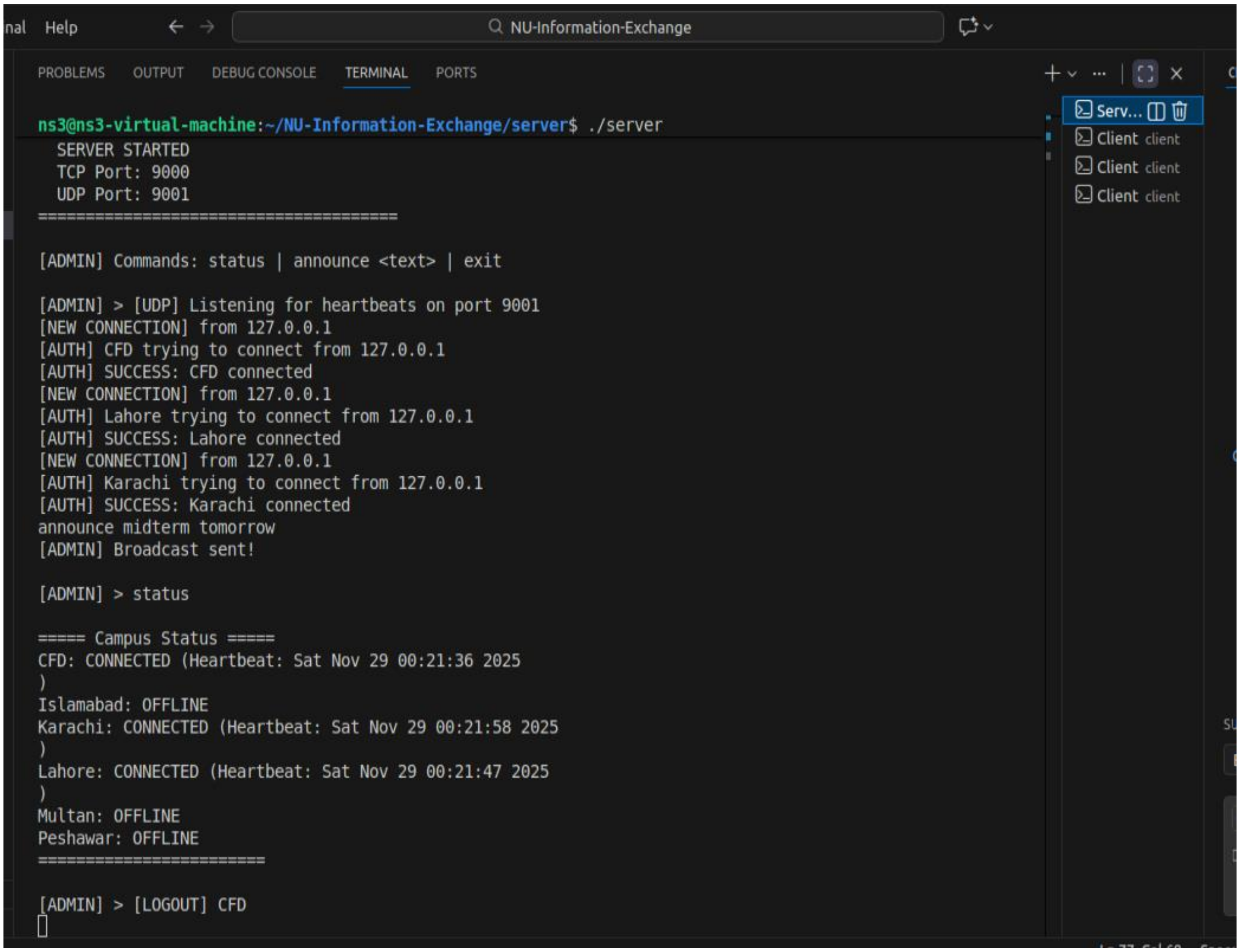




Test Case 5: Graceful Disconnect

- **Objective:** Verify proper cleanup on logout
- **Procedure:**
    1. Client selects "Logout & Exit"
    2. Check server removes client from active list
- **Result:** ✓ Server detected disconnect and cleaned up
- **Evidence:**





4.2 Network Testing

Ping Test Matrix:

Source	Destination	Result	RTT (ms)
Islamabad PC0	Lahore PC2	Success	<1ms
Islamabad PC0	Karachi PC4	Success	<1ms
Islamabad PC0	Peshawar PC7	Success	<1ms
Lahore PC2	Karachi PC4	Success	<1ms

Source	Destination	Result	RTT (ms)
Lahore PC2	Peshawar PC7	Success	<1ms
Karachi PC4	Peshawar PC7	Success	<1ms

5.Self-Evaluation Form

<table>
<tr><td>Name: Nimrah Shahid</td><td>Name: Aqsa Ishaq</td><td>Name: Ayesha Rauf</td></tr>
<tr><td>Reg. No.: 23F-0734</td><td>Reg. No.: 23F-0839</td><td>Reg. No.: 23F-0807</td></tr>
<tr>
<td>Contribution to the Project’s development:
Developed the following modules:
Complete Campus Client Implementation (client.cpp)
TCP connection and authentication logic
UDP heartbeat sender thread
UDP announcement listener thread
User interface and menu system
Client-side message formatting and parsing
Multi-threaded client architecture</td>
<td>Contribution to the Project’s development:
Developed the following modules:
Central Server TCP handling
Socket creation and binding
Multi-threaded client acceptance
Per-client authentication handler
Message routing engine
ParsedMessage structure
Message forwarding logic</td>
<td>Contribution to the Project’s development:
Developed the following modules:
Server UDP components
UDP heartbeat listener thread
Heartbeat parsing and campus status tracking
UDP broadcast functionality
Admin console module
Status command implementation
Announce command implementation
Real-time campus monitoring</td>
</tr>
</table>

4. Packet Tracer topology design (collaborative)	3. Campus socket management and synchronization 4. Packet Tracer topology design (collaborative)	3. Server logging and error handling 4. Packet Tracer topology design (collaborative)

Note: All three team members collaborated on the Cisco Packet Tracer topology design, IP addressing scheme, routing configuration, and testing.

Project completed (in %): 100% - All required features implemented and tested

Project Modules Successfully Implemented:

1. All

2. All

3. All

Modules That Were Not Implemented:

1. None

2. None

3. None

Challenges faced & how they were overcome:

Challenge 1: UDP Port Binding Conflicts

- **Problem:** Multiple clients on localhost couldn't bind to UDP_PORT 9001
- **Solution:** Added SO_REUSEADDR and SO_REUSEPORT socket options to allow multiple processes to share the UDP port for receiving broadcasts

Challenge 2: Thread Synchronization

- **Problem:** Race conditions when multiple clients connected/disconnected simultaneously

- **Solution:** Implemented mutex locks around shared data structures (campusSockets, lastSeen using mtx mutex)

Challenge 3: Message Parsing

- **Problem:** Extracting structured data from string messages
- **Solution:** Designed a custom protocol with delimiters (; and :) and created a ParsedMessage structure with helper functions

Challenge 4: Graceful Shutdown

- **Problem:** Threads continued running after client logout
- **Solution:** Implemented global running flag and proper socket shutdown sequence (shutdown() → close())

Challenge 5: RIP Routing Configuration

- **Problem:** Understanding RIP convergence time and ensuring all routers learned routes properly
- **Solution:** Verified routing tables using 'show ip route' command and waited for RIP to converge (30-90 seconds) before testing connectivity

Learnings from the Project:

1. Socket Programming:
    - TCP vs UDP trade-offs
    - Non-blocking vs blocking I/O
    - Socket lifecycle management
2. Concurrent Programming:
    - Multi-threading with std::thread
    - Mutex synchronization
    - Thread-safe data structures
3. Network Protocols:
    - Protocol design principles
    - Message formatting and parsing
    - Client-server communication patterns

4. Network Architecture:
    - WAN topology design
    - IP subnetting and VLSM
    - RIP dynamic routing configuration
    - Layer 2 vs Layer 3 concepts
5. System Integration:
    - Connecting application layer to network layer
    - End-to-end testing methodologies
    - Debugging distributed systems

Comments (if any):

____
____

6. Conclusion

This project successfully implements a complete multi-campus information exchange system, demonstrating both software engineering and network architecture principles. The application layer provides robust, concurrent client-server communication using hybrid TCP/UDP protocols, while the network layer establishes proper WAN connectivity between campuses.