# Assignment 2: Multi Client Chat System with Performance Analytics

CS 3205

## Overview

Design and implement a multi-client chat application with features including private messaging, broadcast capabilities, user management, centralized backup, and a DNS-like discovery service. Conduct performance benchmarking of the system.

## 1 System Architecture

Your implementation must consist of the following components:

### 1.1 Chat Server (`chat_server`)

- Handle broadcast messages i.e., send to all connected clients [2]
- Support private messaging between any two clients through itself [4]
- Support concurrent connections from multiple clients (minimum 10 simultaneous clients).
  - Handle multiple client connections concurrently using multiple processes (using fork system call) [2]
  - Handle multiple client connections concurrently using multiple threads (using pthreads) [2]
  - Non-blocking server that can manage total 10 clients Using select()/poll() system call/ epoll API [2]
- Implement user authentication using username/password [2]
- Maintain active user list [2]
- Implement proper connection handling with graceful disconnection [2]

### 1.2 Discovery Server (`discovery_server`)

- Acts as a DNS-like service that maps username to IP addresses and port [4]
- Supports user registration with credentials (username, password, port number) [2]

### 1.3 Chat Client (`chat_client`)

- Register with discovery server on startup [2]
- Connect to chat server using credentials [2]
- Provide interactive CLI for user operations:
  - Send broadcast messages [2]
  - Send private messages to specific users [2]
  - Request and view list of online users [2]

## 2 Protocol Design

- Design a custom application-layer protocol for communication
- Define message formats for different operations (login, broadcast, private message etc.)
- Use TCP for reliable message delivery
- Implement proper message framing to handle variable-length messages
- Document your protocol specification in the README

## 3 Bonus Question

- Chat history is stored at server in a structured format (you might consider JSON or custom binary format) with timestamp [4]
- A client can view its own chat history [2]
- A client can change status (available/busy/away) [2]

## 4 Performance Benchmarking

Conduct performance monitoring and reporting

- **Server Metrics:**
  - CPU usage (percentage)
  - Memory consumption (PSS, VmRSS)
- **Client Metrics:**
  - Message delivery time from send to receive

  Design and implement the following test scenarios:

  - **Load Test:** Simulate 10 concurrent clients sending messages
  - **Stress Test:** Gradually increase load until system degradation

Create a separate monitoring module/thread that collects metrics. Log metrics to files at regular intervals (e.g., every 5 seconds). Generate summary reports at the end of tests. Create visualization scripts to create comparison plot for

- Message delivery time distribution for fork based vs thread based vs non-blocking server. [2]
- CPU and memory usage vs. number of clients for fork based vs thread based vs non-blocking server. [3]

# 5 Deliverables

- Submit a single tar file with naming rollno-assignment2.tar.gz containing the following:
- Source code for all components with comprehensive comments
- README with: [2]
  - System architecture overview
  - Protocol specification
  - Compilation and execution instructions
  - Testing guide
- Test scripts [2]
- Performance analysis report with [4]
  - Methodology
  - Test results with graphs
  - Analysis and observations
  - Bottlenecks identified
  - Potential optimizations
- Learnings.txt file - The lessons learned and challenges faced during this assignment