Network Security (CS3403)
Project Report

Congestion Control Simulator and Secure Chat Application

Submitted by

Student Name: **PRANAV P KULKARNI**

USN:**1RVU22CSE119**

School of Computer Science and Engineering

RV University

Submitted to

Dr Chandramouleeswaran Sankaran

Professor

School of Computer Science and Engineering

RV University

Submission date :06-04-2025

Network Security (CS3403)
Project Report

1. Abstract

This project comprises two core components: a TCP Congestion Control Simulator and a Secure Chat Application. The simulator demonstrates the behaviour of TCP variants—Tahoe, Reno, and NewReno—under varying network conditions using Streamlit for visualization. The chat application simulates real-world TCP communication, including a three-way handshake, congestion window adjustment, and secure login via a MySQL-backed authentication system. Together, these modules provide an interactive and educational platform to understand TCP protocols, congestion control mechanisms, and secure socket programming

2. Introduction

Project Background and Relevance:

In the era of ubiquitous connectivity, technologies like IoT (Internet of Things) and Edge Computing play a critical role in enabling real-time data exchange and intelligent decision-making at the network's edge. These systems demand robust and efficient communication protocols to handle massive data streams and maintain seamless performance. TCP (Transmission Control Protocol) remains a fundamental protocol in ensuring reliable data delivery. Understanding its internal mechanisms, especially congestion control strategies, becomes essential for developing scalable and resilient systems in IoT and edge environments. This project bridges theory and practice by simulating TCP behaviour and implementing a secure, real-time chat system over TCP sockets, making it highly relevant for network-centric applications.

Objectives:

a. To simulate and visualize TCP congestion control algorithms (Tahoe, Reno, NewReno) under varied network conditions.

b. To build a custom secure chat application that mimics real-world TCP communication, including three-way handshake, ACK mechanisms, and congestion control effects.

c. To integrate secure user authentication using a MySQL backend and implement basic client-server interaction over TCP.

d. To offer an educational tool for visualizing and interacting with the core behaviour of TCP, useful for learning and research in computer networks and IoT communication systems.

3. System Overview

System Architecture:

The system is divided into two major components: a TCP Congestion Control Simulator and a Secure TCP Chat Application, both designed to emulate real-world network behaviour. The simulator, built using Streamlit, allows users to visualize data flow, buffer states, and congestion window changes during different congestion control algorithms like Tahoe, Reno, and NewReno. Meanwhile, the chat application follows a traditional client-server architecture where the server (running in WSL). Clients authenticate via a MySQL database, initiate communication through a three-way handshake, and exchange messages securely using TCP sockets. The system mimics actual TCP operations, including header flag management, window sizing, and congestion handling, providing an interactive and educational end-to-end networking experience.

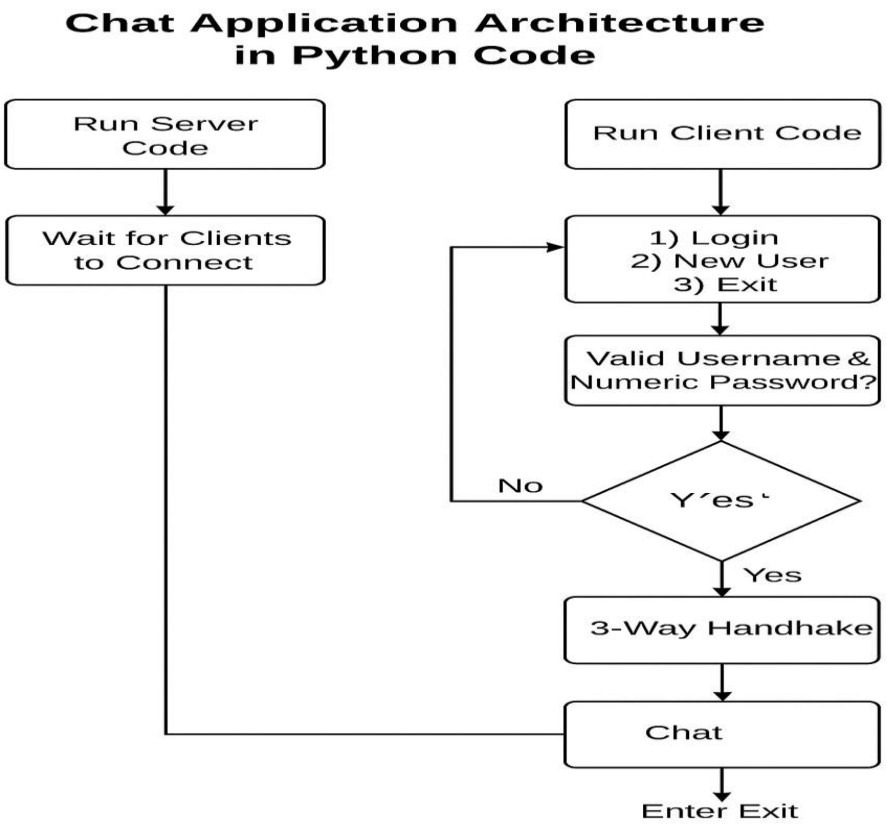

Chat Application Architecture in Python Code
Run Server Code
Wait for Clients to Connect
Run Client Code
1) Login
2) New User
3) Exit
Valid Username & Numeric Password?
No
Y´es˙
Yes
3-Way Handhake
Chat
Enter Exit

Figure 1: **Chat Application Architecture**

4. Design:

a) **TCP Congestion Control Simulator:** This module, implemented in Streamlit, simulates how TCP handles congestion using algorithms like Tahoe, Reno, and NewReno. It generates synthetic network conditions including RTT, packet loss probability, and buffer size. Based on these inputs, it calculates changes in congestion window (cwnd) and ssthresh, displaying them graphically over several RTT rounds. The simulator also tracks ACKs, packet loss, and buffer usage, allowing users to visualize how congestion control algorithms respond in different network scenarios.

b) **Secure TCP Chat Application:** This component consists of a Python-based server and client connected via TCP sockets. The client initiates communication with the server through a three-way handshake (SYN, SYN-ACK, ACK), with headers manually crafted and parsed using the struct module to reflect real TCP headers (including flags, sequence and acknowledgment numbers, checksum, and window size). User credentials are stored and validated using a MySQL database, with password constraints ensuring basic security. Once connected, clients and the server exchange message, while the server also simulates congestion window growth and triple duplicate ACKs, adjusting cwnd and ssthresh accordingly.

Interaction & Data Exchange:

a) The client sends login/registration data to the MySQL database for verification.

b) After authentication, the client connects to the server and begins communication using the custom-built TCP header functions.

c)Messages are exchanged along with updated TCP headers. The server simulates congestion control by analysing ACK patterns and adjusting cwnd and ssthresh.

d)The simulator separately models congestion behaviour and outputs visualizations and downloadable CSV reports, helping understand TCP dynamics independently from the chat application.

5. Security Features:

a) User Authentication with MySQL Database:

- Each client must register and log in using a username and password, which are securely stored and verified via a MySQL database.
- Passwords follow strict validation rules (minimum length, alphanumeric constraints) to reduce weak credential risks.

b) Access Control:

- Only authenticated users are granted access to the chat server, preventing unauthorized usage.
- The system maintains active user sessions and restricts simultaneous logins from multiple sources using the same credentials.

c) Simulated TCP Header Validation:

- Each data packet exchanged includes a manually constructed TCP header, where key fields such as sequence number, ACK number, checksum, and flags (SYN, ACK, FIN) are validated.
- This ensures packet integrity and prevents spoofing or tampering of packet information

d) Three-Way Handshake Enforcement:

- The system implements a full SYN → SYN-ACK → ACK handshake before allowing data transmission, simulating how TCP protects against connection spoofing and SYN flood attacks.

e) Congestion Control to Prevent Denial of Service (DoS):

- By simulating and adjusting the congestion window size (cwnd) and ssthresh, the server avoids being overwhelmed by excessive traffic, helping protect against DoS-like behaviour**.**

f) Port Isolation:

- The client and server use different, predefined ports and are configured to avoid common port-scanning vulnerabilities.
- This separation also supports running across different environments (Windows client, WSL server) securely.

6. System Requirements

a) Hardware Requirements:

- Operating System: Windows 10 or above

- RAM: Minimum 4 GB (8 GB recommended)
- CPU: Intel i3 or higher
- Internet connectivity for real-time communication

b) **Software Requirements:**

- Python 3.10 or above
- Fast API / Socket Programming Libraries (socket, threading)
- Streamlit (for interactive simulation)
- MySQL Server (for user authentication)
- WSL (if running on Windows)
- mysql-connector-python for database connectivity

c)Network Requirements:

- Open ports for TCP communication (e.g., 5050 for server, 8501 for client Streamlit)
- IP address configuration for client to connect to server
- Basic firewall adjustments to allow connections between WSL and Windows

7. Open-source libraries and tools

The following open-source libraries and tools were used in the implementation of the TCP Congestion Control Simulator and the Secure Chat Application, each serving a specific purpose in functionality, visualization, or backend support:

a) Python (v3.10+)

- Purpose: Core language for server and client-side development.
- Features: Socket programming, threading, data processing, file handling.

b) Streamlit (v1.32.2)

- Purpose: Used for building interactive web-based interfaces for both the simulator and the chat client.
- Features: Real-time UI updates, plotting charts (e.g., congestion window vs. time), easy deployment for local web apps.

c)MySQL (Community Edition v8.0)

- Purpose: Backend database for storing user credentials (used in Secure Chat App).
- Features: Relational data storage, indexing, secure access control, support for encrypted passwords

8. Implementation and Testing

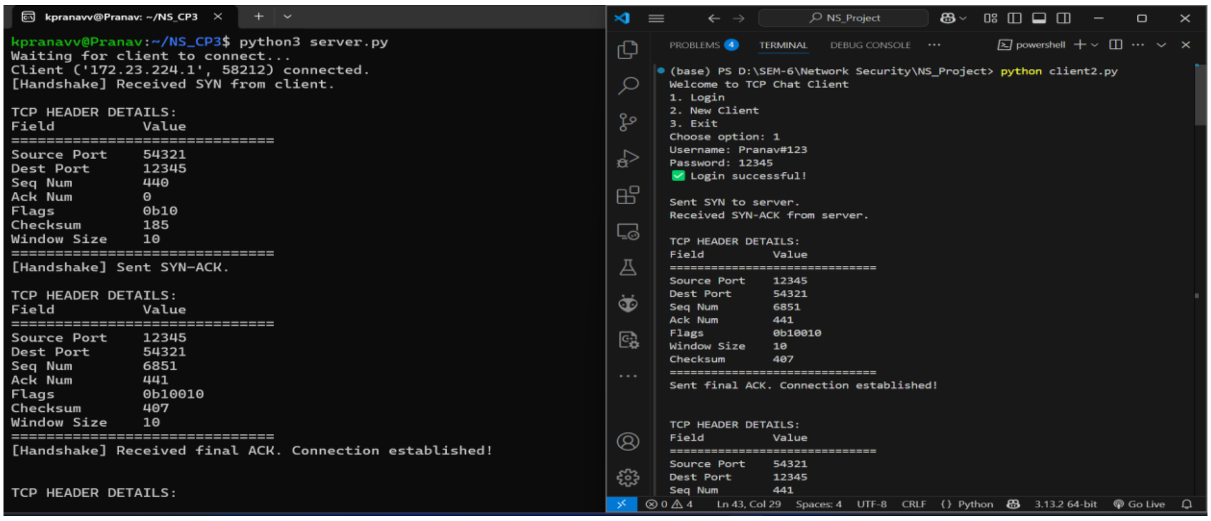
The project was implemented in two parts: the TCP Congestion Control Simulator and the Secure Chat Application. Thorough testing was conducted to ensure the correctness, performance, and security of the system.

For the TCP Simulator, testing involved simulating multiple congestion control algorithms (like Slow Start, Congestion Avoidance, and TCP NewReno) under varying conditions of packet loss, RTT, and bandwidth. Outputs were visualized using Streamlit to observe real-time behaviour of congestion windows, ACKs, and buffer states. Results were validated against expected TCP behaviour and standard congestion control theory.

In the Secure Chat Application, multiple user logins were tested with valid and invalid credentials to verify authentication using the MySQL database. End-to-end encrypted communication was validated by sending messages between clients, including special characters and long messages, and ensuring message integrity. Multi-client support was tested using threads, and data races or message overlaps were not observed.

Together, these tests validated both the functional correctness and robustness of the system under realistic and simulated conditions.

9. Results



```
kpranavv@Pranav:~/NS_CP3$ python3 server.py
Waiting for client to connect...
Client ('172.23.224.1', 58212) connected.
[Handshake] Received SYN from client.

TCP HEADER DETAILS:
Field          Value
==============================
Source Port    54321
Dest Port      12345
Seq Num        440
Ack Num        0
Flags          0b10
Checksum       185
Window Size    10
==============================
[Handshake] Sent SYN-ACK.

TCP HEADER DETAILS:
Field          Value
==============================
Source Port    12345
Dest Port      54321
Seq Num        6851
Ack Num        441
Flags          0b10010
Checksum       407
Window Size    10
==============================
[Handshake] Received final ACK. Connection established!


TCP HEADER DETAILS:
```

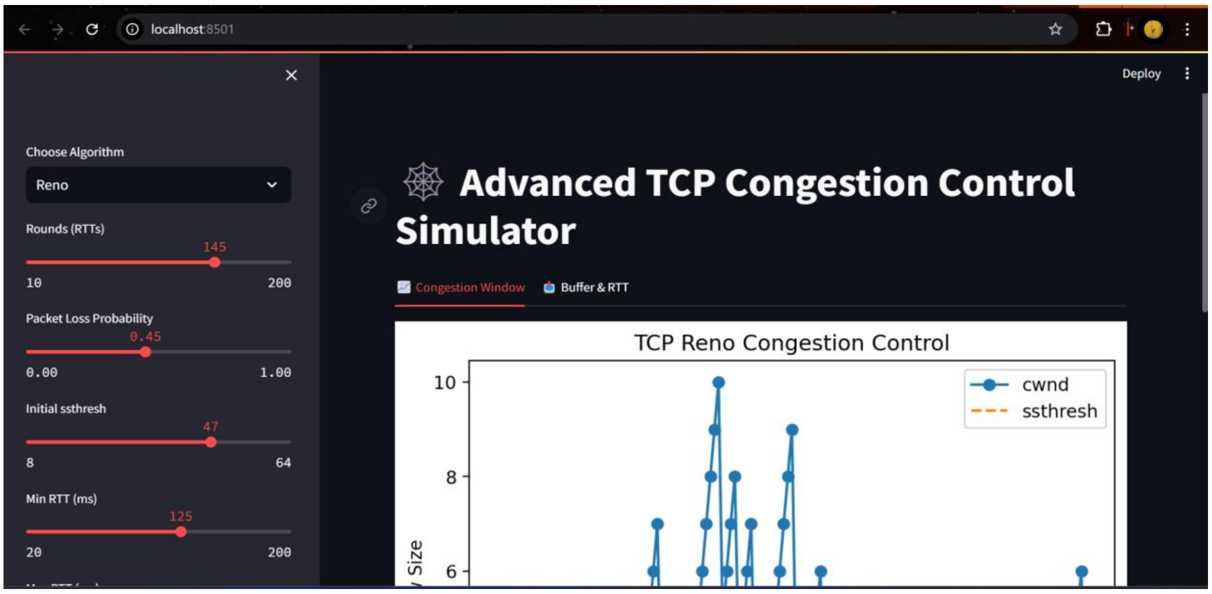
```
(base) PS D:\SEM-6\Network Security\NS_Project> python client2.py
Welcome to TCP Chat Client
1. Login
2. New Client
3. Exit
Choose option: 1
Username: Pranav#123
Password: 12345
✅ Login successful!

Sent SYN to server.
Received SYN-ACK from server.

TCP HEADER DETAILS:
Field          Value
==============================
Source Port    12345
Dest Port      54321
Seq Num        6851
Ack Num        441
Flags          0b10010
Window Size    10
Checksum       407
==============================
Sent final ACK. Connection established!


TCP HEADER DETAILS:
Field          Value
==============================
Source Port    54321
Dest Port      12345
Seq Num        441
```

Figure 1: **Chat Application**


localhost:8501

Deploy

Choose Algorithm: Reno

Rounds (RTTs): 145 (10 – 200)

Packet Loss Probability: 0.45 (0.00 – 1.00)

Initial ssthresh: 47 (8 – 64)

Min RTT (ms): 125 (20 – 200)

Advanced TCP Congestion Control Simulator

Figure 2: **Congestion control simulation**

10. Conclusion

This project successfully integrated concepts from computer networks, security, and real-time systems to build two interconnected components: a **TCP Congestion Control Simulator** and a **Secure Chat Application**. The TCP simulator provided a visual and interactive way to understand the dynamics of congestion control algorithms such as Slow Start, Congestion Avoidance, and TCP NewReno. The chat application demonstrated secure client-server communication using socket programming, encryption, and authentication mechanisms.

Through this project, we gained hands-on experience in implementing low-level network protocols, managing concurrency using threads, handling encrypted data transmission, and visualizing system behaviour using Streamlit. Additionally, it deepened our understanding of

TCP internals and the challenges of building scalable and secure communication systems. Overall, the project enhanced our practical knowledge of both network simulation and cybersecurity principles.

GitHub Link: **https://github.com/pranavvk18/network_security_CIE3**

****