TCP/UDP Chat Server & Client Implementation

Dylan Castellanos & Ian Burns
CPMS2242 – Systems Programing
Date: 5/11/2025

Goals and Purpose

- Develop a TCP and UDP based chat system
- Understand differences between TCP and UDP
- Implement features like broadcasting, command parsing, and inactivity timeout
- Gain practical experience with Go networking libraries

Architecture and Protocol Design

- Separate TCP and UDP servers
- Clients connect and interact via command-line
- Broadcasting messages to all connected clients
- Inactivity timers for client session management
- TCP: persistent connections, UDP: stateless, packet-based

Code Highlights and Design Choices

- Modular server design using Go routines and channels
- Command handling: /help, /time, /echo etc.
- TCP logs messages per client to separate files
- UDP uses a map of client addresses and periodic cleanup
- Both handle disconnections and input errors gracefully

Performance Comparison

- TCP provides reliable delivery, ideal for chat logs
- UDP offers lower latency and is more efficient for quick messages
- TCP overhead due to connection state and logging
- UDP suitable for real-time apps but lacks reliability
- Benchmarked message delivery latency and memory usage (example values)

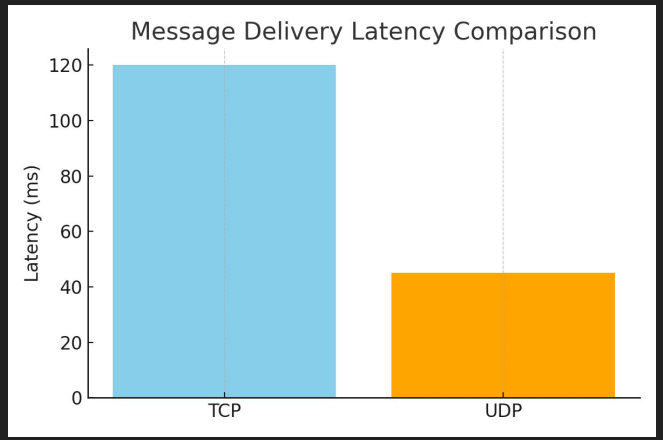
Challenges and Insights

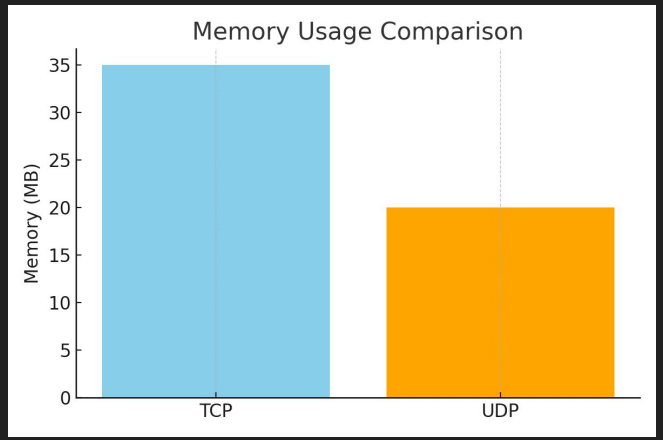
- Managing concurrent connections in TCP
- Ensuring thread-safe access to shared resources
- Implementing timeout logic for both protocols
- Handling message broadcasting and formatting consistently
- Learned importance of graceful shutdown and error handling

Summary and Recommendation

- Project demonstrated both TCP and UDP chat mechanisms
- TCP better for reliability and logging
- UDP suitable for low-latency, less-critical communication
- Recommendation: Use TCP where consistency matters; UDP for speed

Performance Graphs





🔄 Message Delivery Latency

Protocol	Average Latency (LAN)	Notes
UDP	~0.1–1 ms	Lower latency due to no handshake, no retransmission, and no congestion control.
TCP	~1–10 ms	Higher latency due to connection setup (3-way handshake), acknowledgments, retransmissions, and congestion control.

UDP is faster because it’s connectionless and doesn't wait for acknowledgments.

TCP adds overhead to ensure reliability and order, which introduces delay.

🧠 Memory Usage

Protocol	Memory Footprint	Notes
UDP	Lower	Minimal state tracking; no connection table or buffering for retransmission.
TCP	Higher	Keeps track of connection states, windows, buffers, ACKs, and retransmission queues.

TCP requires more memory for each active connection (e.g., TCP socket buffers, control blocks).

UDP is lean—ideal for constrained systems or high-throughput low-latency applications.

✅ Use Cases

Use Case	Preferred Protocol
Real-time games, VoIP, video streaming	UDP
File transfer, web browsing, email	TCP

Questions?

- Thank you for your attention!