Multi-threaded chat server

Distributed Systems Paradigms
Lab Guide 1

Consider a simple multi-threaded chat server using Java and NIO sockets, where lines sent by any client are broadcast to all currently connected clients.

Steps

1. Implement the server using a simple thread-per-connection strategy and use `nc` as a client.
2. Implement a non-interactive client to generate load (*bot*) that sleeps a configurable amount of time between sending or receiving messages. Run clients with different delay configurations.
3. Reconsider threading strategy to avoid blocking writers: Structure the server as passive shared state and reader/writer pairs for each connection. Re-test with different delay configurations.

Questions

1. How does one client affect other clients?
2. How do clients affect server memory usage as observed with `jconsole`?

Learning Outcomes Recall basic distributed systems programming with Java, sockets and threads. Relate interactive performance and memory usage with server programming. Use shared buffers to reduce memory usage.