SALIM HABIB UNIVERSITY

Multi-Threaded Lightweight HTTP Server

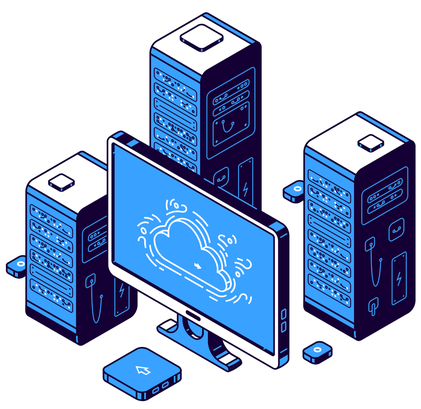
An Optimized and Scalable Solution for Concurrent Client Handling

Group Members: M Maaz, Bilal Jawaid, Hamza, Zeeshan | Course Instructor: Miss Sumra

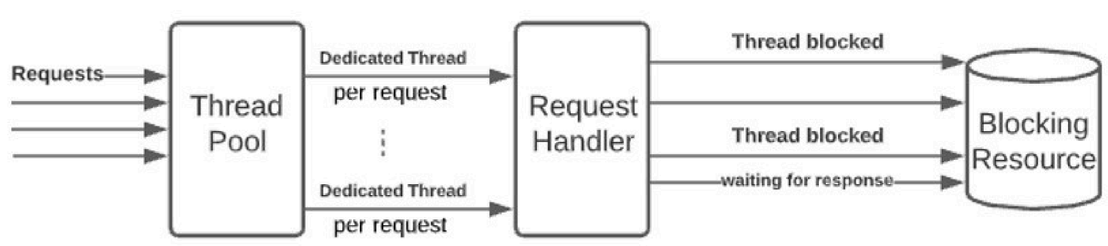
Department of Computer Science, Faculty of Information Technology

Introduction

This project demonstrates the implementation of a lightweight, multi-threaded HTTP server designed to efficiently handle client requests in a concurrent environment. The server supports static file hosting, request caching for optimized performance, and real-time metrics tracking for monitoring server activity.



System Architechture



The server uses a thread pool to handle multiple client requests efficiently, ensuring high performance and scalability. A request queue manages incoming traffic during high loads, preventing the server from being overwhelmed. Additionally, a caching mechanism stores frequently accessed files in memory, reducing disk access and improving response times. These features work together to enhance server reliability and ensure smooth operation under varying workloads.

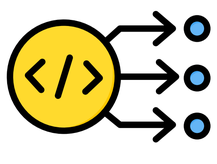
Features

Static File Serving

Supports HTTP GET for serving files

Multi-threading

Handles multiple requests concurrently



Caching

Speeds up responses for frequently accessed files.

Request Queuing

Prevents overload and manages high traffic

Metrics Tracking

Logs success, failure, and rejection rates

Stress Tester

Evaluates server performance under different load conditionses

Testing And Results

Test Case	Success Rate	Avg. Response Time (s)	Observations
Normal Load	95%	0.02	Efficient response times.
High Load	90%	0.10	Slightly increased latency, but stable.
Failure Simulation	50%	0.05	Handled errors gracefully.

Tools and Technologies

Programming Language

- socket for network communication
- threading for multithreading
- queue for request management
- os for file handling
- datetime for logging

Future Enhancements

- Add POST request support for dynamic interactions.
- Implement HTTPS with SSL/TLS for secure communication.
- Use LRU caching to optimize memory usage.
- Create a dashboard for real-time server metrics.
- load balancing for distributing requests across servers.

NC-24, Deh Dih, Korangi Creek , Karachi 74900 (+92-21) 111 248 338 03162754504