HTTP Server Code Review Exam

Interviewer Name

April 6, 2025

1 Architecture and Design (30 points)

1. What threading model does this HTTP server implement? Explain the roles of:
   - The listener thread
   - The worker threads
2. Why does the server use epoll for I/O multiplexing instead of:
   - select()
   - poll()

   What are the advantages of epoll in this context?
3. The server uses a round-robin approach to distribute connections to worker threads.
   - What are the pros and cons of this approach?
   - What alternative load balancing strategies could be used?
4. Explain the purpose of the `EventData` structure and how it's used in the event handling cycle.

2 Implementation Details (40 points)

5. The `HandleEpollEvent` method handles both EPOLLIN and EPOLLOUT events.
   - Describe the complete lifecycle of a single HTTP request through this method
   - Why does it need to manage memory allocation/deallocation of EventData structures?
6. The server uses non-blocking sockets with edge-triggered epoll.
   - What modifications would be needed to switch to level-triggered mode?
   - What are the implications of this change?
7. Examine the error handling in the `ProcessEvents` method:
   - What conditions cause a socket to be closed?
   - Why does it check for both EPOLLHUP and EPOLLERR?
8. The `control_epoll_event` helper function handles three epoll operations:
   - What are the race conditions that could occur when modifying epoll sets?
   - How does the current implementation prevent them?

3 Performance and Optimization (20 points)

9. The server implements an exponential backoff mechanism using `sleep_times_`.
   - When is this used and why?
   - How could this be improved?
10. The current implementation copies HTTP response data twice:
    - Where do these copies occur?
    - How could you reduce this to a single copy?

11. The server uses a fixed-size thread pool.
    - What are the limitations of this approach?
    - How could you make it dynamically scalable?

4 Advanced Questions (10 points)

12. How would you modify this server to support HTTPS/SSL connections?
    - What components would need to change?
    - How would this affect performance?
13. The current URI-based routing is simple. How would you implement:
    - Path parameters (e.g., `/users/:id`)
    - Middleware support
14. How could you add connection timeout handling?
    - For idle connections
    - For slow request/response cycles

Scoring Guide

- 90-100: Expert-level understanding
- 70-89: Strong working knowledge
- 50-69: Basic understanding with some gaps
- Below 50: Needs significant study