Bài tập thực hành

Môn học: TH Lập Trình Mạng

Chương 6: UDP sockets

1. MỤC TIÊU

Mục tiêu của bài thực hành này là giúp sinh viên hiểu sâu hơn về lập trình socket UDP bằng cách triển khai một trò chơi chat nhiều người chơi có mã hóa. Sinh viên cần thực hiện các công việc sau:

- Sử dụng socket UDP (socket(), sendto(), recvfrom()).
- Xử lý nhiều client và các thông điệp.
- Triển khai xác minh server bằng cách sử dụng hàm memcmp().
- Sử dụng hàm connect() để tối ưu hóa giao tiếp UDP.
- Triển khai timeout cho các thông điệp.
- Thêm mã hóa đơn giản cho việc trao đổi dữ liệu giữa client và server.

2. YÊU CẦU

- Kiến thức cơ bản về lập trình C
- Kiến thức cơ bản về Mạng máy tính
- Máy tính cài đặt Hệ điều hành Linux (khuyến khích distro Ubuntu)
- Sử dụng các hàm trong bộ thư viện liên quan lập trình mạng

3. BÀI THỰC HÀNH

Tổng quan về ứng dụng:

Trong bài thực hành này, sinh viên sẽ triển khai một trò chơi chat nhiều người chơi, trong đó server xử lý nhiều client. Việc giao tiếp sẽ được mã hóa bằng cách sử dụng một thuật toán XOR đơn giản, và client sẽ xác minh danh tính của server bằng cách sử dụng hàm `memcmp()`.

- **Server:** Gửi các thông điệp đã mã hóa đến nhiều client, nhận phản hồi từ họ và xử lý chúng.

- **Client:** Kết nối đến server, nhận các thông điệp đã mã hóa, giải mã chúng, gửi lại phản hồi và xác minh rằng thông điệp đến từ server đúng bằng cách sử dụng `memcmp()`.

Phần 1: Thiết lập UDP Sockets với mã hóa

1.1 Chương trình Server (Xử lý nhiều client, các hàm cơ bản, mã hóa)

1. Tạo một server UDP lắng nghe kết nối từ nhiều client trên một địa chỉ IP và cổng cụ thể.
2. Server gửi các thông điệp đã mã hóa đến các client và nhận phản hồi của họ.
3. Mỗi client nhận được một thông điệp khác nhau, và server phản hồi với một thông báo xác nhận.

Cần giải quyết các vấn đề sau:

- Xử lý nhiều client với các thông điệp khác nhau.
- Mã hóa và giải mã các thông điệp bằng cách sử dụng thuật toán XOR đơn giản.
- Triển khai timeout cho thông điệp bằng cách sử dụng select().

Hướng dẫn:

- Sử dụng socket() để tạo socket UDP cho server.
- Sử dụng bind() để liên kết socket với một địa chỉ IP và cổng.
- Sử dụng recvfrom() và sendto() để giao tiếp với các client.
- Sử dụng thuật toán XOR để mã hóa và giải mã các thông điệp.
- Sử dụng select() để xử lý timeout cho các thông điệp.

Mã nguồn tham khảo cho hàm mã hóa và giải mã:

```
void xor_cipher(char *data, char key) {
    for (int i = 0; data[i] != '\0'; i++) {
        data[i] ^= key;
```

```
    }
}
```

1.2 Chương trình Client (Xác minh địa chỉ server bằng memcmp(), mã hóa)

1. Client kết nối đến server, nhận thông điệp mã hóa, giải mã nó và gửi lại câu trả lời.
2. Client xác minh danh tính của server bằng cách sử dụng memcmp().

Cần giải quyết các vấn đề sau:

- Giải mã thông điệp bằng thuật toán XOR.
- Xác minh rằng thông điệp đến từ server mong đợi.
- Xử lý timeout cho thông điệp bằng cách sử dụng select().

Hướng dẫn:

- Sử dụng socket() để tạo socket UDP cho client.
- Sử dụng connect() để đơn giản hóa giao tiếp.
- Sử dụng recvfrom() và sendto() để giao tiếp.
- Sử dụng memcmp() để so sánh địa chỉ server.
- Triển khai mã hóa đơn giản bằng XOR.

Phần 2: Cải tiến

2.1 Nhiệm vụ bổ sung

- Thêm tính năng xử lý nhiều client trong server.
- Cải thiện cơ chế mã hóa để an toàn hơn.
- Triển khai xử lý lỗi tốt hơn cho timeout của thông điệp và phản hồi từ server.

Yêu cầu

Mỗi sinh viên phải nộp:

- Giải thích ngắn gọn về mã nguồn đã viết.
- Ảnh chụp màn hình hoặc nhật ký của việc giao tiếp giữa client và server.
- Thảo luận về cách triển khai mã hóa, xác minh server bằng memcmp(), và sử dụng connect().

Chú ý:

- Sinh viên phải nộp cả mã nguồn của server và client.
- Một báo cáo bao gồm giải thích và ảnh chụp màn hình của kết quả chương trình.