**Bài tập thực hành**

**Môn: TH Lập Trình Mạng**

**Chương 6: I/O 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.

1. **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

1. **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.

A computer screen with text

Description automatically generated

*Nếu socket không tạo được sẽ in ra lỗi và thoát chương trình.*

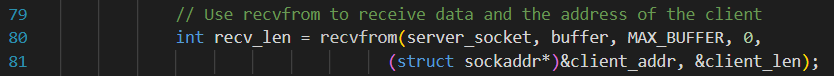
* Sử dụng bind() để liên kết socket với một địa chỉ IP và cổng.

A screen shot of a computer

Description automatically generated

*Nếu bind thất bại, server sẽ thoát ra.*

* Sử dụng recvfrom() và sendto() để giao tiếp với các client.



*Nhận dữ liệu từ client, lưu vào* ***buffer*** *và địa chỉ client lưu vào* ***client\_addr.***

* Sử dụng thuật toán XOR để mã hóa và giải mã các thông điệp.

A black background with white text

Description automatically generated

*Hàm lặp từng ký tự trong chuỗi và XOR với khóa* ***key****. Khi thực hiện XOR lần thứ 2 với cùng 1 khóa sẽ trả về giá trị gốc nên sử dụng hàm này để giải mã và mã hóa các thông điệp.*

* Sử dụng select() để xử lý timeout cho các thông điệp.

A computer screen with many colorful text

Description automatically generated with medium confidence

*Hàm* ***select()*** *sẽ chờ khi có dữ liệu hoặc timeout:*

* *Nếu có dữ liệu, sẽ trả về số lượng file descriptors sẵn sàng để đọc.*
* *Nếu không có dữ liệu, sẽ trả về 0 và hiển thị timeout.*

**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.

A computer screen with text

Description automatically generated

*Tương tự phía* ***server***

* Sử dụng connect() để đơn giản hóa giao tiếp.

A computer code on a black background

Description automatically generated

*Dùng để kết nối socket với địa chỉ và cổng của server. Do UDP không yêu cầu kết nối thực sự như TCP nên sử dụng* ***connect()*** *để đơn giản hóa giao tiếp giữa client và server (không cần chỉ định địa chỉ đích khi gửi, nhận dữ liệu).*

* Sử dụng recvfrom() và sendto() để giao tiếp.

***recvfrom()*** *và* ***sendto()*** *được sử dụng khi không kết nối socket trước (không dùng* ***connect()****), phù hợp khi server/client không duy trì một kết nối cố định. Vì đã sử dụng* ***connect()*** *nên ở đây sẽ dùng* ***recv()*** *và* ***send()*** *để giao tiếp:*

*A screen shot of a computer code

Description automatically generated*

*Hàm* ***send()*** *gửi dữ liệu với* ***buffer*** *là dữ liệu đã mã hóa.*

*A computer screen with text

Description automatically generated*

*Hàm* ***recv()*** *nhận dữ liệu qua socket và lưu vào* ***buffer****. Giải mã* ***buffer*** *với khóa* ***XOR\_KEY****, nếu khớp thì xác minh được thông điệp đến từ server đúng.*

* Sử dụng memcmp() để so sánh địa chỉ server.

A screen shot of a computer

Description automatically generated

*Hàm* ***memcmp()*** *so sánh nội dung địa chỉ server mà client nhận được với địa chỉ server mong muốn để đảm bảo tính toàn vẹn của kết nối.*

* Triển khai mã hóa đơn giản bằng XOR.

A black background with white text

Description automatically generated

*Tương tự phía* ***server***

*A screenshot of a computer program

Description automatically generated*

*Mã hóa thông điệp trước khi gửi bằng khóa* ***XOR\_KEY*** *được định nghĩa ở đầu.*

A screen shot of a computer

Description automatically generated

*Giải mã và hiển thị thông điệp nhận từ* ***server.***

***Kết quả:***

* *Server:*

A screenshot of a computer

Description automatically generated

* *Client 1:*

A screenshot of a computer screen

Description automatically generated

* *Client 2:*

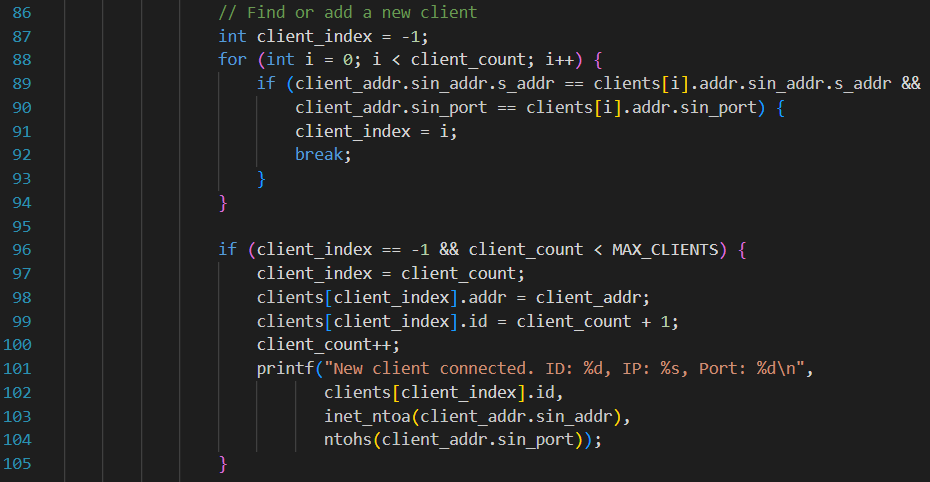
A screenshot of a computer screen

Description automatically generated

**Phần 2: Cải tiến**

**2.1: Nhiệm vụ bổ sung**

* Thêm tính năng xử lý nhiều client trong server.



*Trong file* ***server****, tạo danh sách client. Mỗi khi nhận thông điệp từ client, server sẽ kiểm tra địa chỉ đã tồn tại trong danh sách chưa. Nếu chưa, sẽ thêm vào danh sách, bao gồm địa chỉ và ID của client. Điều này cho phép server theo dõi nhiều client khác nhau, phân biệt dựa trên địa chỉ IP và cổng của client.*

* 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.

A computer screen shot of a program

Description automatically generated

*Hàm* ***send\_with\_timeout*** *dùng để gửi thông điệp qua socket thử lại nhiều lần. Nếu không thể gửi trong khoảng thời gian quy định, hàm sẽ gửi lại nhiều lần (tối đa là* ***MAX\_RETRIES****), đảm bảo việc gửi dữ liệu không bị treo khi có lỗi mạng hoặc server không phản hồi.*

**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.