ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

****

BÁO CÁO

Bài tập thực hành :

Chương 7: Thiết kế Ap

plication Protocol

Học phần: Thực hành Lập Trình Mạng

Giảng viên hướng dẫn	: Trần Hải Anh
Mã lớp	: 151907
Sinh viên thực hiện	: Phạm Vân Anh
Mã số sinh viên	: 20214988

Hà Nội, tháng 10 năm 2024

Link github: github

Phần 1: Các Bước Thiết Kế Ứng Dụng

1.1 Xác Định Vấn Đề

- MOVE (0x02): Được gửi từ client để gửi nước đi (hàng và cột).
- STATE_UPDATE (0x03): Được gửi từ server cho cả hai client để cập nhật bảng trò chơi sau mỗi nước đi.
- RESULT (0x04): Được gửi từ server để thông báo kết quả trò chơi (thắng/thua/hòa).
- TURN_NOTIFICATION (0x05): Được gửi từ server để thông báo cho client rằng đến lượt họ đi.
- ANNOU (0x06): Được gửi từ server để thông báo bằng text.

1.2 Thiết Kế Giao Thức Truyền Thông

- Loại thông điệp (1 byte đầu tiên): Xác định loại thông điệp.
- Payload: Dữ liệu bổ sung cho mỗi thông điệp.
    - MOVE: Byte thứ 2 chứa thông tin hàng, byte thứ 3 chứa thông tin cột.
    - STATE_UPDATE: 9 bytes (từ byte thứ 2) mỗi byte chứa 1 trạng thái của bàn cờ (0x01: trống, 0x02: X, 0x03: O).
    - RESULT: Byte thứ 2 chứa kết quả (0x00: thắng, 0x01: thua, 0x02: hoà).
    - ANNOU: Từ byte thứ 2 chứa chuỗi thông báo.

1.3 Xác Định Luồng Trò Chơi và Vai Trò

1. Server bắt đầu: Server chờ hai client kết nối.
2. Khi client kết nối, server sẽ gửi TURN_NOTIFICATION để thông báo lượt chơi.
3. Khi client nhận được TURN_NOTIFICATION, sẽ yêu cầu người dùng nhập nước đi hợp lệ, sau đó gửi MOVE tới server.
4. Server nhận được MOVE, cập nhật trạng thái bàn cờ và gửi STATE_UPDATE tới tất cả các client. Quá trình này lặp lại cho đến khi có người thắng hoặc hòa.

5. Khi có kết quả, server sẽ gửi RESULT tới tất cả các client.

1.4 Cấu Trúc Dữ Liệu và Xử Lý Thông Điệp

Phần 2: Thiết Kế Server Từng Bước

2.1 Thiết lập kết nối

```
    listen(server_socket, 3);

    printf("Server is listening on port %d\n", PORT);

    for (int i = 0; i < NUM_PLAYERS; i++) {
        if ((players[i].socket_fd = accept(server_socket, (struct
sockaddr *)&server_addr, (socklen_t*)&address_length)) < 0) {
            perror("Connection acceptance failed");
            exit(EXIT_FAILURE);
        }
        printf("Player %d connected\n", i + 1);

        players[i].player_role = current_role;

        memset(message_buffer, 0, BUFFER_SIZE);
        sprintf(message_buffer, "%cYou're playing as %c\n",
NOTIFICATION, current_role);
        send(players[i].socket_fd, message_buffer,
strlen(message_buffer), 0);

        current_role = PO;
    }
```

2.2 Luân phiên lượt chơi

- Server luân phiên giữa 2 client.
- Server gửi thông báo cho player hiện tại, chờ nước đi của họ, và cập nhật trạng thái trò chơi.

```
    while (1) {
        for (int i = 0; i < NUM_PLAYERS; i++) {
            sleep(1);
            memset(message_buffer, 0, BUFFER_SIZE);

            message_buffer[0] = TURN_NOTIFICATION;
            send(players[i].socket_fd, message_buffer, 1, 0);
```

```
            while (read(players[i].socket_fd, message_buffer,
BUFFER_SIZE) <= 0);

            assert(message_buffer[0] == MOVE);

            move_x = message_buffer[1];
            move_y = message_buffer[2];

            memset(message_buffer, 0, BUFFER_SIZE);

            if (place_symbol(game_board, move_x, move_y,
players[i].player_role) == -1) {
                memset(message_buffer, 0, sizeof(message_buffer));
                sprintf(message_buffer, "%cInvalid move\n",
NOTIFICATION);
                send(players[i].socket_fd, message_buffer,
strlen(message_buffer), 0);

                i -= 1; // Retry the same player
                continue;
            }

            update_board_buffer(game_board, message_buffer);
            send_to_all(message_buffer, STATE_SIZE, players);

            if (is_board_full(game_board) || (game_winner =
determine_winner(game_board)) != NONE) {
                goto END_GAME_LOOP;
            }
        }
    }
```

2.3 Xác Thực Nước Đi và Cập Nhật Trạng Thái Trò Chơi

- Server nhận được nước đi từ người chơi hiện tại → kiểm tra nếu nước đi hợp lệ (tức là ô còn trống).
- Nếu hợp lệ, server cập nhật bảng trò chơi và gửi trạng thái mới cho hai người chơi, nếu không, server gửi thông báo lại một lần nữa cho người chơi hiện tại kèm thông báo nước đi không hợp lệ.

```
if (place_symbol(game_board, move_x, move_y, players[i].player_role) ==
-1) {
memset(message_buffer, 0, sizeof(message_buffer));
sprintf(message_buffer, "%cInvalid move\n", NOTIFICATION);
send(players[i].socket_fd, message_buffer, strlen(message_buffer), 0);
```

```
i -= 1; // Retry the same player
continue;
            }
```

2.4 Kiểm tra thắng hoặc hòa

- Server thoát vòng lặp và gửi thông báo kết quả tới 2 người chơi

```
END_GAME_LOOP:

    sleep(1);

    memset(message_buffer, 0, BUFFER_SIZE);
    message_buffer[0] = RESULT;

    if (game_winner == NONE) {
        message_buffer[1] = DRAW;
        send_to_all(message_buffer, 2, players);
    } else {
        for (int i = 0; i < NUM_PLAYERS; i++) {
            if (players[i].player_role == game_winner) {
                message_buffer[1] = WIN;
            } else {
                message_buffer[1] = LOSE;
            }
            send(players[i].socket_fd, message_buffer, 2, 0);
        }
    }
```

Phần 3: Thiết Kế Client từng bước

3.1 Thiết lập kết nối

```
int setup_connection(struct sockaddr_in *address) {
    int socket_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (socket_fd < 0) {
        printf("\nSocket creation error\n");
        return -1;
    }

    address->sin_family = AF_INET;
    address->sin_port = htons(SERVER_PORT);

    if (inet_pton(AF_INET, "127.0.0.1", &address->sin_addr) <= 0) {
        printf("\nInvalid address / Address not supported\n");
```

```
        return -1;
    }

    return socket_fd;
}
```

3.2 Nhận thông báo lượt và Gửi nước đi

- Client chờ tới khi nhận được thông điệp thông báo từ server.
- Client thực hiện lượt chơi của mình bằng request _turn(). Thủ tục request_turn() yêu cầu người chơi nhập toạ độ điểm đánh và gửi thông điệp MOVE tới server.

```
    while (1) {
        handle_server_response(socket_fd, buffer, game_board);

        // Break the loop if RESULT message is received
        if (buffer[0] == RESULT) {
            break;
        }
    }
```

```
void request_turn(int socket_fd, char buffer[BUFFER_CAPACITY]) {
    int move_x, move_y;
    printf("Your turn (x,y): ");
    scanf("%d,%d", &move_x, &move_y);

    memset(buffer, 0, BUFFER_CAPACITY);
    sprintf(buffer, "%c%c%c", MOVE, (char) move_x, (char) move_y);
    send(socket_fd, buffer, strlen(buffer), 0);
}
```

3.3 Nhận cập nhật trạng thái trò chơi và kết quả

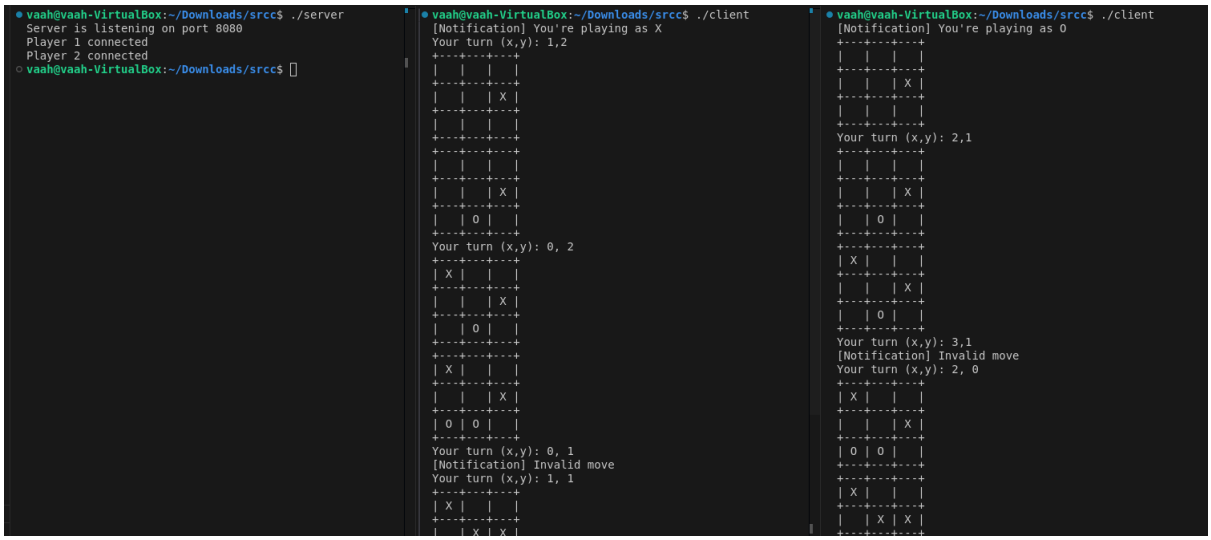
```
    case STATE_UPDATE:
        update_board_from_buffer(game_board, buffer);
        display_board(game_board);
        break;

    case RESULT:
        if (buffer[1] == WIN) {
            printf("You win!\n");
        } else if (buffer[1] == DRAW) {
            printf("Draw!\n");
        } else {
            printf("You lose :))\n");
```

```
        }
        break;
```

Phần 4: Kiểm tra ứng dụng

- Thông báo người chơi

- Kiểm tra nước đi



- Thông báo kết quả: