IoT programming HW#3
Lab2-2 분석 결과

AI융합학부 20223092

1. uecho / uecho_con / echo : client 파일의 **소스코드** 비교

	uecho_client.c	uecho_con_client.c	echo_client.c
프로토콜	UDP - 전송 순서 보장 X	UDP - 전송 순서 보장 X	TCP - stream으로 취급
동시 접속	여러 클라이언트가 서버와 메세지 송수신 가능	여러 클라이언트가 서버와 메세지 송수신 가능	echo_server.c 코드 내 동시 접속 처리 기능이 없으므로, 선착순으로 한 클라이언트가 서버 점유
연결 방식	sendto()/recvfrom() 으로 매번 주소 지정	connect()로 주소 고정(to OS)	connect() 후 스트림처럼 통신 (3-way handshake)
송수신 함수	sendto() / recvfrom()	write() / read()	write() / read()
uecho vs. uecho_con	같은 UDP 통신이지만, 주소 지정 방식에서 차이가 있음. uecho_con은 주소를 고정해 사용하므로 매번 주소를 명시할 필요가 없어 코드의 가독성이 좋아짐. 그러나 여전히 UDP의 신뢰성을 가짐		
uecho_con vs. echo		둘다 connect()를 사용하지만, uecho_con은 상대주소를 고정할 뿐 실제 연결이 아님. 여전히 UDP이며 메시지 단위 통신, 전송 순서 미보장, 신뢰성 낮음의 특징을 가지고 있음. 반대로 echo는 느릴 수도 있지만 데이터를 놓치지 않음	

1. uecho / uecho_con / echo : client 파일의 **소스코드** 비교

uecho_client.c	uecho_con_client.c	echo_client.c

```
connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr));   -> uecho에는 없음

while(1)
{
        fputs("Input message(Q to quit): ", stdout);
        fgets(message, BUF_SIZE, stdin);
        if (!strcmp(message, "q\n") || !strcmp(message, "Q\n"))
                break;

        // sendto(sock, message, strlen(message), 0, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
        write(sock, message, strlen(message));
        // addr_size = sizeof(from_addr);
        // str_len = recvfrom(sock, message, BUF_SIZE, 0, (struct sockaddr*)&from_addr, &addr_size);
        str_len = read(sock, message, sizeof(message) - 1);

        message[str_len] = 0;
        printf("Message from server: %s", message);
}
```

주석처리된 부분

>>>uecho vs. uecho_con

uecho_client와 달리, uecho_con_client에선 while loop 전 connect로 주소를 지정하고, 이후 송수신 과정에서 write()/read()함수를 사용한다.

uecho_con vs. echo<<<

socket 생성시 소켓 타입이 DGRAM / STREAM으로 다름. 즉 통신 타입이 다른 것을 확인할 수 있음.

connect 이후 서버와의 송수신 과정에서 차이점:

- **uecho_con**: read() 호출 한번으로 한 메세지를 통째로 읽음
- **echo**: 루프 안에서 read()를 여러번 호출해서 누적된 데이터를 읽음 -> 보낸 길이 만큼 수신했다는 보장이 없는 TCP 통신이므로 len 비교를 기준으로 누적.

diff uecho_con echo로 비교한 결과

```
24c23
<     sock = socket(PF_INET, SOCK_DGRAM, 0);
---
>     sock = socket(PF_INET, SOCK_STREAM, 0);
33c32,35
<     connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
---
>     if (connect(sock, (struct sockaddr*) &serv_addr, sizeof(serv_addr)) == -1)
>         error_handling("connect() error");
>     else
>         puts("Connected...........");
38a41
>
42,46c45,54
<         // sendto(sock, message, strlen(message), 0, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
<         write(sock, message, strlen(message));
<         // addr_size = sizeof(from_addr);
<         // str_len = recvfrom(sock, message, BUF_SIZE, 0, (struct sockaddr*)&from_addr, &addr_size);
<         str_len = read(sock, message, sizeof(message) - 1);
---
>         str_len = write(sock, message, strlen(message));
>
>         recv_len = 0;
>         while (recv_len < str_len)
>         {
>                 recv_cnt = read(sock, &message[recv_len], BUF_SIZE - 1);
>                 if (recv_cnt == -1)
>                         error_handling("read() error!");
>                 recv_len += recv_cnt;
>         }
```

2. uecho / uecho_con / echo : client 파일의 **실행결과** 비교

<terminal 1: uecho / uecho_con / echo 순서대로 클라이언트 실행, 첫번째 클라이언트>

```
hakyung02@hakyung02-IdeaPad-Pro-5-14AHP9:~/2025-1/IoTprog/lab2$ ./uecho_client 127.0.0.1 9190
Input message(Q to quit): First client in uecho_client
Message from server: First client in uecho_client
Input message(Q to quit): q
hakyung02@hakyung02-IdeaPad-Pro-5-14AHP9:~/2025-1/IoTprog/lab2$ ./uecho_con_client 127.0.0.1 9190
Input message(Q to quit): First client in uecho_con_client
Message from server: First client in uecho_con_cliInput message(Q to quit): Message from server: ent
Input message(Q to quit): q
hakyung02@hakyung02-IdeaPad-Pro-5-14AHP9:~/2025-1/IoTprog/lab2$ ../lab1/echo_client 127.0.0.1 9190
Connected...........
Input message(Q to quit): First client in echo_client
Message from server: First client in echo_client
Input message(Q to quit): I'm still connected..
Message from server: I'm still connected..
Input message(Q to quit): And message has sent
Message from server: And message has sent
Input message(Q to quit): I'm done and let's see the client 2's terminal
Message from server: I'm done and let's see the client 2's terminal
Input message(Q to quit): q
```

<terminal 2: 순서대로 클라이언트 실행, 두번째 클라이언트>

```
hakyung02@hakyung02-IdeaPad-Pro-5-14AHP9:~/2025-1/IoTprog/lab2$ ./uecho_client 127.0.0.1 9190
Input message(Q to quit): Second client in uecho_client
Message from server: Second client in uecho_clientInput message(Q to quit): Message from server:
Input message(Q to quit): First and Second clients are connecting  at the same time!
Message from server: First and Second clients are Input message(Q to quit): Message from server: connecting at the same time!
Input message(Q to quit): q
hakyung02@hakyung02-IdeaPad-Pro-5-14AHP9:~/2025-1/IoTprog/lab2$ ./uecho_con_client 127.0.0.1 9190
Input message(Q to quit): Second client in uecho_con_client
Message from server: Second client in uecho_con_clInput message(Q to quit): Message from server: ient
Input message(Q to quit): First and Second clients are connecting at the same time!
Message from server: First and Second clients are Input message(Q to quit): Message from server: connecting at the same time!
Input message(Q to quit): q
hakyung02@hakyung02-IdeaPad-Pro-5-14AHP9:~/2025-1/IoTprog/lab2$ ../lab1/echo_client 127.0.0.1 9190
Connected...........
Input message(Q to quit): Second client in echo_client
Messages don't send to server.
Because client 1 is still connected.Message from server: Second client in echo_client
Input message(Q to quit): Message from server: Messages don't send to server.
Input message(Q to quit): The last message that start with "Because" is gone.
Message from server: Because client 1 is still connected.The last message that start with "Because" is gone.
Input message(Q to quit): okay, It come with the new last one.
Message from server: okay, It come with the new last one.
Input message(Q to quit): bye.
Message from server: bye.
Input message(Q to quit): q
```

-> TCP 통신을 사용한 echo_client는 echo_server에서 따로 동시 접속 처리를 하지 않아 클라이언트를 하나씩 처리하고 있음