Computer Network #02

Programming Assignment #01

21800409 신지영

21800510 이민규

1. becho_clinet.c 와 becho_server.c에서 송수신 방식과 동일하게 TCP program을 작성하여 실행하시오. TCP와 UDP 실행 결과에 어떤 차이가 있는 가를 비교하고, 그 이유를 설명하시오.

- **기존 UDP program 실행 결과**

```
[s21800510@localhost ~]$ gcc becho_server.c -o server
[s21800510@localhost ~]$ ./server
Usage : ./server <port>
[s21800510@localhost ~]$ ./server 1106
수신 번호 : 0
Hello
수신 번호 : 1
World
수신 번호 : 2
Everybody!
```

UDP_server.c

```
[s21800409@localhost Network_practice]$ ./client.out 203.252.112.25 1106
서버로부터수신된0차메시지: Hello
서버로부터수신된1차메시지: World
서버로부터수신된2차메시지: Everybody!
[s21800409@localhost Network_practice]$
```

UDP_client.c

- **위 프로그램과 동일한 송수신 방식의 TCP program 실행 결과**

```
[s21800510@localhost ~]$ ./server 1106
nf: 4    ss: 3
수신 번호 : 0
I Love Network S2 S2 S2!
```

TCP_server.c

```
[s21800409@localhost Network_practice]$ ./client.out 203.252.112.25 1106
서버로부터수신된0차메시지: I Love Network S2 S2 S2!
```

TCP_client.c

차이: UDP 방식의 경우 0차부터 2차까지 실행결과가 메시지 하나를 받으면 하나를 출력하는 형태를 취하지만 TCP방식의 경우 메시지를 한번에 여러 개 보내서 한번에 여러개를 출력하는 형태를 갖습니다.

이유: UDP는 connection setup과정이 없으니 packet을 보내는 즉시, 패킷마다 확인을 하는 과정없이 바로 전송을 하고 반면에 TCP의 경우 packet들을 받으면 다 queue에 저장을 하고 확인을 한 후 일괄적으로 보내줍니다.

- 같은 실행 결과를 보이도록 수정한 TCP program

```
    //write(sock, MSG1, strlen(MSG1));
    //write(sock, MSG2, strlen(MSG2));
    //write(sock, MSG3, strlen(MSG3));

    for(i=0; i<3; i++){
        write(sock, MSG[i], strlen(MSG[i]));
        sleep(1);
        str_len = read(sock, message, BUFSIZE);
        message[str_len]=0;
        printf("서버로부터수신된%d차메시지: %s \n", i, message);
    }
    close(sock);
    return 0;
}
```

Code Modified

```
[s21800510@localhost ~]$ ./server 1106
nf: 4    ss: 3
수신 번호 : 0
I love
수신 번호 : 1
Network
수신 번호 : 2
S2 S2 S2
```

TCP_server.c

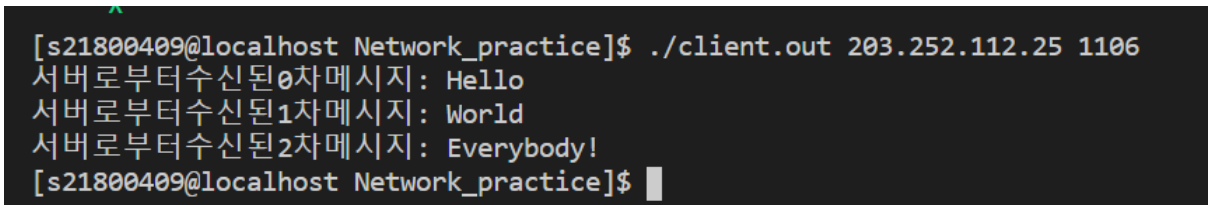
```
[s21800409@localhost Network_practice]$ ./client.out 203.252.112.25 1106
서버로부터수신된0차메시지: I love
서버로부터수신된1차메시지: Network
서버로부터수신된2차메시지: S2 S2 S2
[s21800409@localhost Network_practice]$
```

TCP_client.c

2. becho_server.c 에서 BUFSIZE를 5로 (#define BUFSIZE 5)로 변경한 후, server와 client를 실행하시오. 수정하지 않았을 경우와 수정하였을 때에 어떤 차이가 있는 가를 비교하고, 그 이유를 설명하시오.

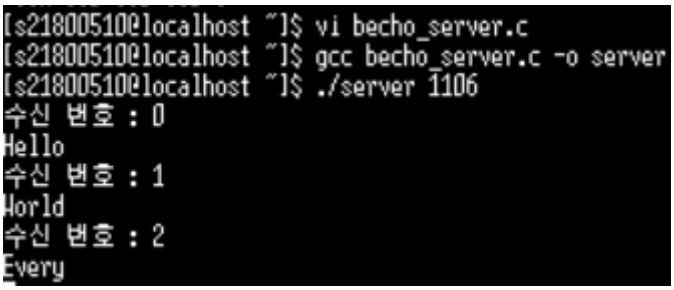
- 기존 UDP program 실행 결과 (BUFSIZE=30)

UDP_server.c

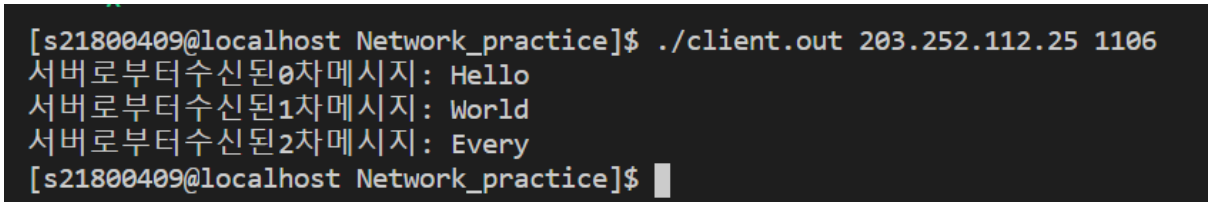


UDP_client.c

- 변경된 UDP program 실행 결과 (BUFSIZE=5)



UDP_server.c



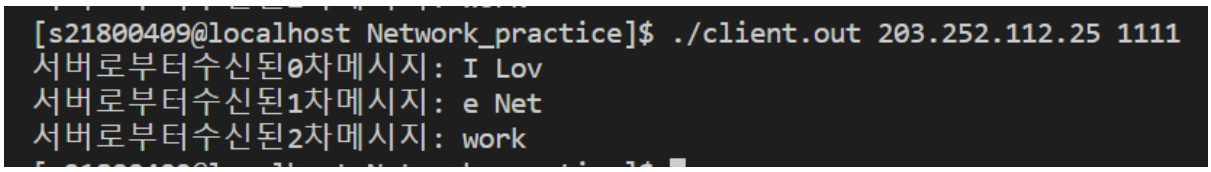
UDP_client.c

차이: BUFFERSIZE를 30으로 했을 경우 3번째 문구인 everybody!가 모두 출력이 정상적으로 되지만 반면에 BUFFERSIZE를 5로 했을 경우 every까지만 출력이 됩니다.

이유: BUFFERSIZE는 packet을 받으면 그 packet을 저장하는 공간인데 BUFFERSIZE가 30이면 30byte만큼 받을 수 있지만 BUFFERSIZE가 5이면 5byte만큼 받을 수 있습니다. 그래서 everybody는 5byte가 넘기때문에 5byte만큼 크기인 every만 출력이 되는 것입니다.

3. becho_clinet.c 와 becho_server.c에서 송수신 방식과 동일하게 작성한 TCP program에서 server의 BUFSIZE를 5로 (#define BUFSIZE 5)로 변경한 후, server와 client를 실행하시오. 2번에서의 UDP의 경우와 어떤 차이가 있는가를 비교하고, 그 이유를 설명하시오.

TCP_server.c



TCP_client.c

차이: UDP의 경우 5byte를 넘은 나머지 문자열을 버리고 출력을 하지만 TCP의 경우 버리지 않고 따로 추가로 송신을 하여서 보냅니다. 그리고 나눠서 출력합니다.
이유: UDP의 경우 따로 저장을 하지 않고 패킷을 BUFFERSIZE만큼 보내지만 반대로 TCP의 경우 BUFFERSIZE만큼 보내는 대신 나머지 문자열을 buffer에 저장을 하고 추가로 보냅니다. 이를 통해 loss를 방지합니다.

4. 1번에서 구성한 TCP program에서 server 코드에서 sleep() 코드를 모두 제거하고 program을 실행하시오. 실행 결과를 sleep()이 있는 1번 TCP의 경우와 어떤 차이가 있는 가를 비교하고, 그 이유를 설명하시오.

```
[s21800409@localhost Network_practice]$ ./client.out 203.252.112.25 1111
서버로부터수신된0차메시지: I Love Network S2 S2 S2!
```

TCP_server.c

```
[s21800510@localhost ~]$ vi tcp_becho_server_rmsleep.c
[s21800510@localhost ~]$ gcc tcp_becho_server_rmsleep.c -o server
[s21800510@localhost ~]$ ./server 1111
nf: 4    ss: 3
수신 번호 : 0
I Love Network S2 S2 S2!
```

TCP_client.c

차이: 없음

이유: TCP는 받은 대로 보내기 때문에 순서가 유지가 되고 loss가 발생하지 않기 때문이다.

5. UDP에서 recevfrom() 함수에서 return 값이 0 이 될 수 있는가? 있다고 한다면 어떤 경우인가? 실험을 한 다음에 방법을 설명하시오. 위의 방법을 TCP에 적용시키면 어떤 결과가 나타나는가? UDP 경우와 어떤 차이가 있는가를 비교하고, 그 이유를 설명하시오.

```
[s21800409@localhost Network_practice]$ ./client.out 203.252.112.25 1111
서버로부터수신된0차메시지: Hello
```

UDP_server.c

```
[s21800510@localhost ~]$ vi becho_server.c
[s21800510@localhost ~]$ gcc becho_server.c -o server
[s21800510@localhost ~]$ ./server 1111
***recvfrom: 6
수신 번호 : 0
Hello
***recvfrom: 0
[s21800510@localhost ~]$
```

UDP_client.c

```
[s21800409@localhost Network_practice]$ ./client.out 203.252.112.25 1111
서버로부터수신된0차메시지: I Love Network S2 S2 S2!
```

TCP_server.c

```
nf: 4    ss: 3
read= 24수신 번호 : 0
I Love Network S2 S2 S2!
read= 0[s21800510@localhost ~]$
```

TCP_client.c

차이: UDP의 경우 0byte인 메시지를 수신받을 경우 종료되고 TCP의 경우에는 client 쪽에서 연결을 끊으면 즉시 중단이 됩니다.

이유: UDP의 경우 BUFFERSIZE만큼 패킷단위로 보내기 때문에 보낸 패킷이 0byte일 경우 중단이 되고 TCP의 경우 한꺼번에 확인을 하고 보내므로 연결이 끊기는지 그 여부를 통해 중단을 결정합니다.