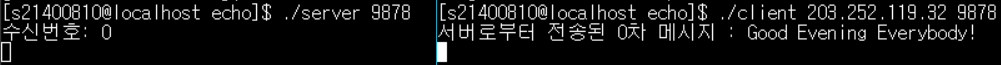
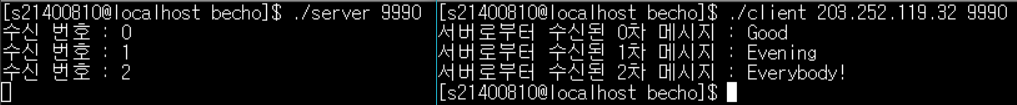
(a) 수업시간에 다룬 becho\_server.c/becho\_client.c와 같은 동작 형태를 지닌 TCP와 UDP 프로그램을 작성하여 동작을 시키고, TCP와 UDP의 차이점을 설명하십시오.

**TCP**



server는 accept 후, message를 수신하기 전 5초간 정지한다. 이 5초동안 client는 3개의 message를 byte 단위로 server에게 전송한다. 이 예제에서 client가 server에게 보낸 message size의 총 합은 buffer의 maximum size 보다 작으므로, 저장된 모든 data는 recv 함수로 한 번에 읽어진다. server는 읽은 data를 client에게 send 함수를 사용해 돌려 보내는데, 여기서 client의 maximum buffer size 또한 전체 전송된 message보다 크므로 message 전체를 수신한다.

**UDP**



server가 recvfrom 함수를 호출하기 전 5초동안, client가 sendto 함수호출로 3개의 message를 message 단위로 server에게 전송한다. 이 예제는 UDP소켓을 사용하기 때문에 message의 boundary를 구분할 수 있어서 data는 server에 message 단위로 저장이 되고, server는 recvfrom 함수로 message를 하나씩 수신한다(출력 결과를 보면 server의 수신 번호가 총 3개인 것을 볼 수 있다). server는 받은 message를 다시 client에게 sendto 함수를 통해 돌려준다. server의 sendto 함수를 통해 전송된 message는 client에 message단위로 저장되고, client는 그 message를 하나씩 받아 출력한다. client의 출력 결과를 보면 message단위로 출력한 것을 볼 수 있다.

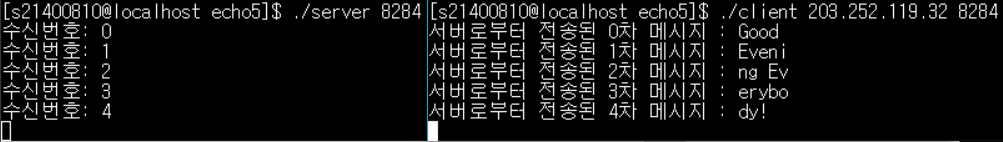
**차이점**

소켓 버퍼의 사이즈가 충분히 크고 data를 받기 전 충분한 시간이 주어지면,   
TCP는 byte stream이므로 message를 받는 쪽은 주어진 시간 동안에 byte 단위로 data를 저장한다. 이 때 저장된 각 message 간의 boundary가 없기 때문에, data를 읽을 때에는 버퍼 사이즈만큼 한번에 읽는다. 이 예제에서는 버퍼의 사이즈가 충분하므로 한 번에 전체 message를 수신한다. 하지만 UDP는 버퍼의 사이즈가 충분함에도 불구하고, 각 data는 boundary를 갖고 있어 message 단위로 저장이 되고, data를 읽을 때에는 한 번에 하나의 message를 읽는다.

또한, TCP에서 client의 send 함수는 총 3회 호출되었고, server의 recv 함수는 총 1회 호출된 것을 보면 함수의 호출 횟수는 큰 의미를 지니지 않는다는 것을 알 수 있다. UDP에서는 client의 sendto 함수가 총 3회 호출되었고 server의 recvfrom 함수는 총 3회 호출된 것으로 보아, UDP에서는 함수 호출 횟수가 일치한다는 것을 알 수 있다.

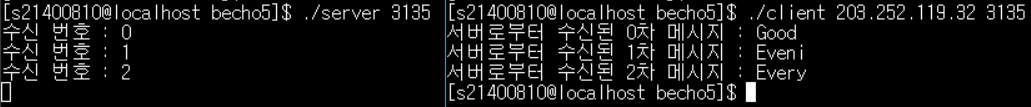
(b) (a)의 program에서 server의 buffer 크기를 5 (#define BUFSIZE 5)로 하여 TCP와 UDP를 동작시키고, TCP와 UDP의 차이점을 설명하십시오.

**TCP**



server가 sleep하는 5초동안 모든 data는 이미 server에게 전송이 된 상태가 된다. TCP는 byte stream이므로 client가3개의 message를 server에 1byte씩 나눠 보내고 저장한다. client는 server에게 3개의 message를 보내지만 TCP는 메시지의 경계를 구분하지 않고 저장하므로, server가 data를 읽을 때 앞에서부터 5byte씩 수신하고 그 수신한 data를 client에게 전송한다. 이와 같은 과정이 server가 버퍼에 있는 모든 message를 보내기까지 반복된다.

**UDP**



UDP는 message 단위로 전송해서 하나의 message에는 boundary가 정해져 있기 때문에, 한 번에 하나의 message를 보내고 읽는다. 위 예제에서는 먼저, 5초동안 client가 server로 3개의 message를 보낸다. 하지만 server의 maximum buffer size는 5byte이기 때문에 client가 보낸 각각의 message에 대해 상위 5개의 character만 서버의 buffer에 저장된다. server는 상위 5byte만 저장된 message를 sendto 함수를 통해 client에게 되돌려주고, client는 recvfrom 함수를 호출해 server가 보낸 data를 수신한다. client에서 수신한 data는 상위 5byte만 있는 것을 볼 수 있다.

**차이점**

소켓 버퍼의 사이즈가 충분하지 않고 data를 받기 전 충분한 시간이 주어지면,   
TCP는 message의 boundary를 고려하지 않고 한 byte씩 전송된다. 하지만 버퍼에 있는 data를 읽을 때는 buffer의 size만큼 읽기 때문에, 그 size만큼 나눠서 읽는다. UDP에서는 버퍼의 사이즈가 data 보다 작게 되면, data loss가 발생한다. 그 이유는 UDP는 data의 boundary를 갖고 있어 각 message를 따로 buffer에 저장하는데, buffer의 사이즈가 충분하지 않으면 버퍼의 사이즈만큼만 수신하기 때문이다.

또한, 이 예제에서도 TCP에서 함수의 호출 횟수는 큰 의미를 지니지 않는다는 것을 알 수 있다(send 3회, recv 5회). UDP에서는 함수 호출 횟수가 일치한다는 것을 보아 함수 호출 횟수가 완전히 일치해야 송신된 data를 전부 수신할 수 있다는 것을 알 수 있다(sendto 3회, revcfrom 3회).

위 (a), (b) 문제에서 알 수 있는 것 이외에도 TCP와 UDP는 connection setup이라는 점에서 차이를 갖는다. TCP는 처음에 server는 socket을 열고, 소켓과 주소값을 bind시킨 후, listen이라는 단계에서 client의 연결을 받아들일 수 있다. 만약 listen중에 client가 연결을 요청했다면, server는 연결요청을 허용, 즉 accept를 한다. 이후엔 server와 client는 서로 send와 recv 함수를 통해 서로 데이터를 송수신한다.

UDP는 server에서 socket을 열고, socket과 주소값을 bind 시킨다. UDP에서는 listen과 accept 단계가 없기 때문에 별다른 연결 과정 없이 recvfrom을 통해 바로 message를 받고 sendto를 통해 message를 보낸다. TCP의 경우에는 connect 함수호출 때 IP와 PORT#가 할당되는데, UDP에서는 처음 sendto 함수를 호출했을 때 IP와 PORT#가 자동으로 할당된다.