컴퓨터 네트워크(01)

Socket Program HW1 19.10.25

21300691 정원식

실험 (a) 결과

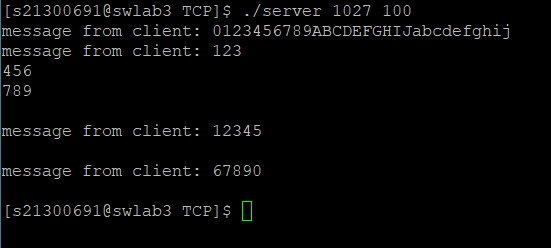


그림1 실험(a) server

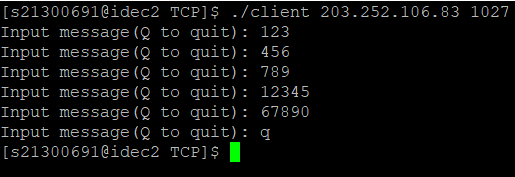


그림2 실험(a) client

실험 (b) 결과

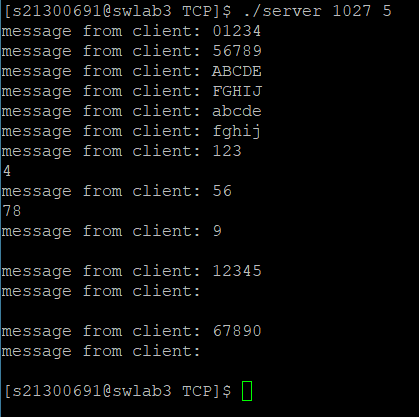


그림3 실험(b) server

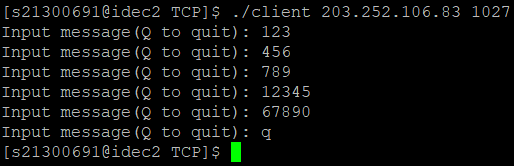


그림4 실험(b) client

실험 (c) 결과

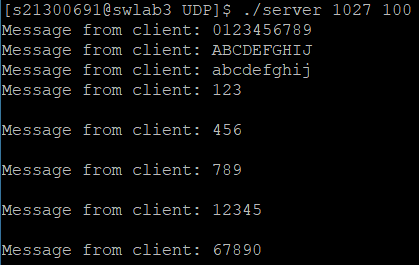


그림5 실험(c) server

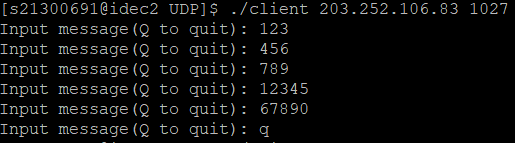


그림6 실험(c) client

실험 (d) 결과

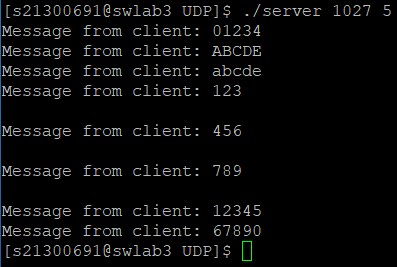


그림7 실험(d) server

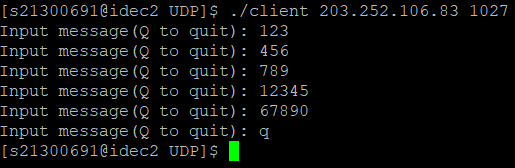


그림8 실험(d) client

1. 실험 a와 b를 통해 TCP는 메시지 경계가 존재하지 않는다는 것을 확인할 수 있었다. 우선 a의 경우 서버의 버퍼 사이즈를 100으로 함으로서 클라이언트가 처음 보낸 3개의 메시지를 한번에 받은 것을 확인할 수 있다. 이후에 보낸 “123”, “456”, “789” 메시지들 또한 서버가 sleep 하는 동안 수신하여 한번에 세개의 메시지를 받았다. (그림 1,2 참조)

또한 실험 b의 경우 서버의 버퍼 사이즈를 5로 함으로 인해 클라이언트가 처음 보낸 3개의 메시지를 6번에 걸쳐 받은 것을 확인할 수 있다. 이후에 보낸 “123”, “456”, “789” 메시지들은 서버가 sleep 하는 동안 모두 수신하였으나 버퍼 사이즈의 한계로 인해 세 개의 메시지로 나누어 받았다. (그림 3,4 참조)

이를 통해 TCP는 메시지 경계가 존재하지 않는다는 것을 확인할 수 있었다.

또한 클라이언트가 종료될 때 서버를 종료시키려면 클라이언트가 보내는 메시지를 확인하면 된다. 정상적인 경우, 클라이언트가 close 함수를 호출하면 서버의 recv 함수는 0을 받게 된다. 그럼 이때 서버도 종료하면 된다.

1. 실험 c와 d를 통해 UDP는 메시지의 경계가 존재한다는 것을 확인할 수 있었다. 우선 c의 경우 서버의 버퍼 사이즈를 100으로 하였으나 클라이언트가 보낸 메시지 개수만큼만 수신하였다. 이후에 보낸 “123”, “456”, “789” 메시지들은 서버가 sleep 하는 동안 모두 수신하였으나 세개의 메시지로 받았다. (그림 5,6 참조)

또한 실험 d의 경우 서버의 버퍼 사이즈를 5로 함으로 인해 처음 클라이언트가 보낸 메시지를 일부인 사이즈 5 밖에 못 받은 것을 확인할 수 있다. (그림 7,8 참조)

이를 통해 UDP는 메시지 경계가 존재함을 확인할 수 있었다.

또한 클라이언트가 종료될 때 서버를 종료시키려면 클라이언트가 보내는 메시지를 확인하면 된다. 단 TCP와는 달리 UDP는 비연결 지향 프로토콜이므로 클라이언트가 프로세스를 종료할 예정임을 알려주는 메시지를 따로 보내야 한다. 이때 패킷 손실이 발생 할 수 있으므로 TCP에서 close 시 사용하는 4-way handshake을 변형하여 사용하도록 한다. 그림 9는 구현한 close 과정을 표현한 것이다. 클라이언트는 사용자가 “q\n” 입력시 서버에게 “q”를 전송한다. (이때 패킷 손실이 일어날 수 있으므로 자식 프로세스를 이용하여 5초 간격으로 재전송해준다.) 서버는 클라이언트가 보낸 “q”를 확인하고 “ack”을 전송한다. (이때도 마찬가지로 자식 프로세스를 이용하여 3초 간격으로 재전송해준다.) 클라이언트는 서버로부터 “ack”을 받은 뒤, 자식 노드를 종료하고 다시 ack을 10번 전송 후 종료한다. 서버는 클라이언트로부터 “ack”을 확인하고 종료한다.

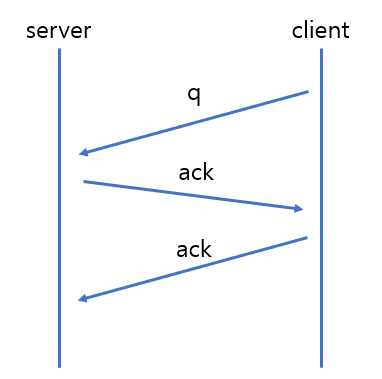


그림9 UDP close 과정