**CL PROGRAMMING assignment #2**

학번 : 201720768 성명 : 김수영

1. 목적

UDP 프로그래밍을 확장해서 학습, 이해를 하고 UDP의 Connectionless 의 장점을 극대화 하면서, 이 특징이 가지는 단점을 극복하기 위해서, ‘Stop-and-Wait ARQ’ 프로토콜을 학습해서, 기존 UDP 프로그램에 접목시킨다. 구체적으로 ‘Stop-and-Wait ARQ’ 에서 packet loss 되는 과정과 retransmission 중심으로 프로그램을 설계한다.

1. 프로그램 설계

**Syntax**:

(1) client 에서 sendto 하는 syntax : 보내려는 메시지와, sequence number 를 가지고 있는 구조체 배열의 특정 구조체의 포인터를 문자열 포인터로 변환, 이 문자열을 int 변수 4byte 와 보내려는 메시지의 길이를 합한 길이 만큼 sendto()를 이용해서 server 에 전송하는 syntax다. Timeout 이 걸리면 재전송하는 syntax 와 동일하다.

(2) 확률적으로 packet loss가 발생하지 않았을 때를 가정하여, server가 client 에게 보내는 echo message syntax : client 에게 받은 메시지와, client 에게 받은 메세지의 길이를 받은 sequence number 와 합친 새로운 sequence number 를 가지고 있는 구조체를 문자열 포인터로 변환하고 이를 client 에게 받은 syntax 길이만큼 전송하는 syntax다.

**Semantics:**

1. *retval = sendto(sock, (char\*)&sendMessage[inputNum], randLength+4, 0, (SOCKADDR\*)&serveraddr, sizeof(serveraddr))*

이전에 설정한 서버의 소켓 주소의 포인터를 목적지로 메시지와 sequence number를 전송한다. 반환되는 retval 의 값은 전송이 성공했을 때 보낸 메시지의 byte 크기, 전송 실패 시 error -1이 출력된다.

다른 위치에 같은 Semantics를 작성해 timeout 시 client 에서 재전송을 하는 기능을 도와준다.

1. *retval = recvfrom(sock, buf, BUFSIZE, 0, (SOCKADDR\*)&clientaddr, &addrlen)*

client 가 sendto()를 통해 메시지를 보내면, 메시지를 보낸 client 의 주소에서부터 server 의 socket 를 통해 지정된 BUFSIZE(1024)만큼 ‘buf’라는 버퍼에 문자열을 받아온다. retval 이 -1를 받아오면 error 받아오는 것을 성공하면 보내온 문자열의 길이를 retval 에 반영된다.

1. *retval = sendto(sock, (char\*)&sendMessage, retval, 0, (SOCKADDR\*)&clientaddr, sizeof(clientaddr))*

server 가 이후에 확률적으로 discard 할 필요없을때 (이론적으로 client 에서 packet loss 가 발생하지 않았을 때) 소켓을 통해서 이전에 recvfrom() 한 client 의 주소로 메시지와 새로운 sequence number를 전송한다. 이때 retval 은 **Semantics-(1**)과 마찬가지다.

1. *retval = recvfrom(sock, buffer, BUFFERSIZE, 0, (SOCKADDR\*)&peeraddr, &addrlen)*

Cient 는 echo syntax를 전송한 server의 주소를 peeraddr라는 이름의 소켓 주소 포인터에서 socket 를 통해서 buffer 안에 특정 BUFFERSIZE 만큼을 받아온다. retval 은 **Semantics-(2)**와 동일하지만, error가 발생하면 이 에러가 timeout 에 해당하는지 판단하고, timeout 에 해당하면 지정한 일정 시간 이후에 한번 더 client 가 server 로 ‘*retval = sendto(sock, (char\*)&sendMessage[inputNum], randLength+4, 0, (SOCKADDR\*)&serveraddr, sizeof(serveraddr))* semantics’ 를 이용해서 다시 재전송을 한다. Recvfrom() 이 timeout 면에서 성공할때까지 재전송 기능을 하는 **Semantics-(1), Semantics-(2), Semantics-(3), Semantics-(4)**를반복한다.

**Timing**:

1. Client 에서 보내는 sequence number 는 초기의 sequence number 0으로 전송 🡪 2)Client 에서 보낸 message 길이만큼 + Client 에서 보낸 sequence number = new Sequence number로 Server 에서 Client 에 메시지 전송 🡪 3)Client 는 Server 에서 받은 메시지의 sequence number 를 server에 메시지를 보낼 때 사용 🡪 2)3)반복한다.

3.프로그램 구현

1. Client 구현

서버로 보낼 랜덤 메시지와 sequence number 를 가지는 구조체 배열, 서버에서 echo syntax를 받을 구조체를 생성한다. *SOCKET sock = socket(AF\_INET,SOCK\_DGRAM, 0)*으로 소켓과 소켓 주소 구조체를 생성, 초기화 하고 서버와의 데이터 통신을 하는 while문 안에서, buf 라는 이름의 문자열에 랜덤한 문자열을 넣는다. 생성한 랜덤 문자열과 초기값 혹은 이 전에 전송한(재전송을 제외한) 문자열의 길이들의 합인 sequence number를 이 전에 생성한 구조체에 데이터 값을 넣는다. *retval = sendto(sock, (char\*)&sendMessage[inputNum], randLength+4, 0, (SOCKADDR\*)&serveraddr, sizeof(serveraddr));*에서 server에 구조체를 전달, *setsockopt(sock, SOL\_SOCKET, SO\_RCVTIMEO, (char\*)&tout, sizeof(tout));*를 이용해서 sendto() 하자마자 tout만큼 timer 를 작동시킨다. *retval = recvfrom(sock, buffer, BUFFERSIZE, 0, (SOCKADDR\*)&peeraddr, &addrlen);*에서 server로부터 echo message를 받아오고, 반환되는 retval 에 따라 timeout 발생시 tout만큼 기다린후 sento()를 이용해서 메시지를 server에 재전송한다. While 문 안에 특정 변수 sendAmount가 100이 될 때 while문을 중단하고, socket를 닫는다.

1. Server 구현

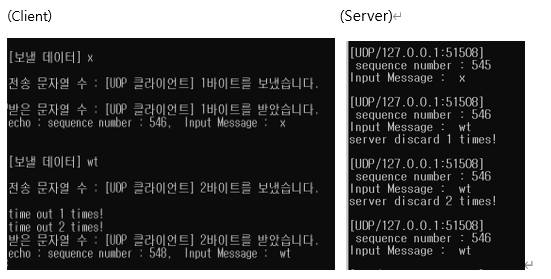
소켓 생성 이후에, *retval = bind(sock, (SOCKADDR\*)&serveraddr, sizeof(serveraddr));*통해서 bind 시키고, client로부터 받을 랜덤 메시지와 sequence number를 가지는 구조체 배열, client 로 보낼 echo message 구조체를 생성한다. Client 와 마찬가지로, while문안에 *retval = recvfrom(sock, buf, BUFSIZE, 0, (SOCKADDR\*)&clientaddr, &addrlen)*를 통해서 client 로부터 syntax를 받아온다. 0.1~0.9까지의 random한 값을 생성해서, p값이하 즉 0.7이하의 값을 가지면 sequence number를 client로부터 받은 문자열의 길이만큼 합해서 갱신하여, 메시지와 함께 sendto()동작을 한다. 이 경우에만 배열위치를 +1 해주고, p값 초과일 경우, 처음 recvfrom()부터 다시 동작 시킨다. p값이 0.7이하일 때 특정 변수 sendAmount 를 1씩 증가시켜 client가 재전송 메세지가 아닌 초기 100개의 메시지를 보낸 이후 while문을 빠져나가 server socket 를 종료 시킨다.

4.동작 결과

1. 동작환경

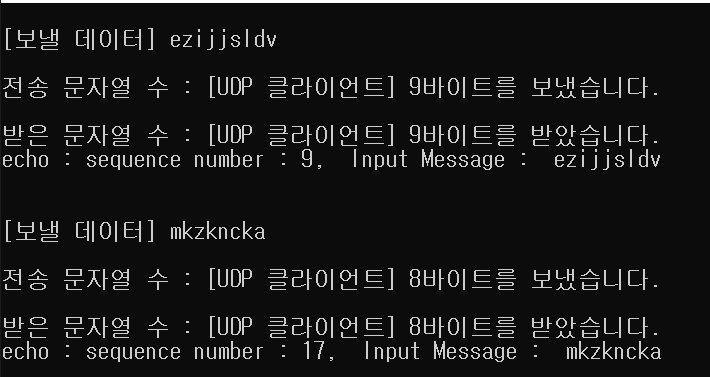
* OS환경 : windows 10
* 사용한 IDE : Visual studio 2019

1. 동작결과
2. Server 컴파일 -> Client 컴파일 (빠른 출력 결과를 보기 위해서, time out 기준 시간을 1초로 지정했으므로, server 먼저 컴파일한다.)
3. Client 컴파일 이후에 timeout이 걸린 경우

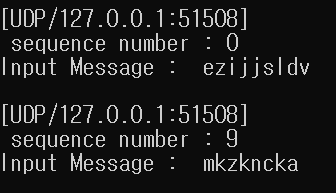


1. Client timeout 이 걸리지 않았을 때(random<=p)

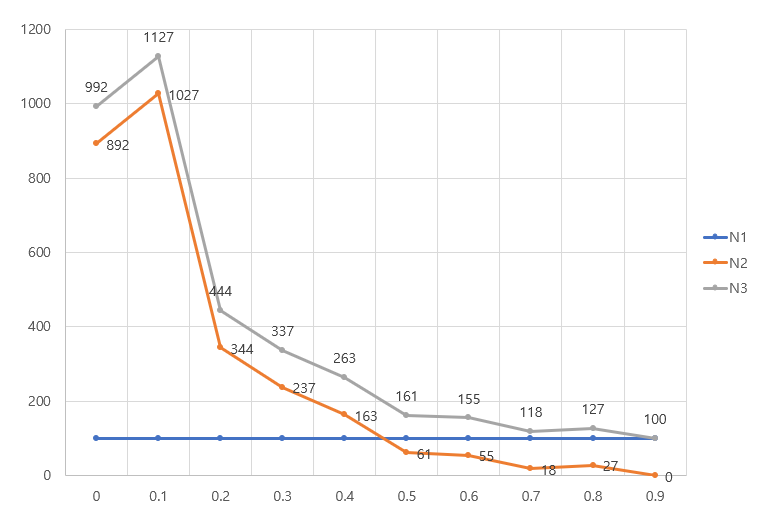
(Client)



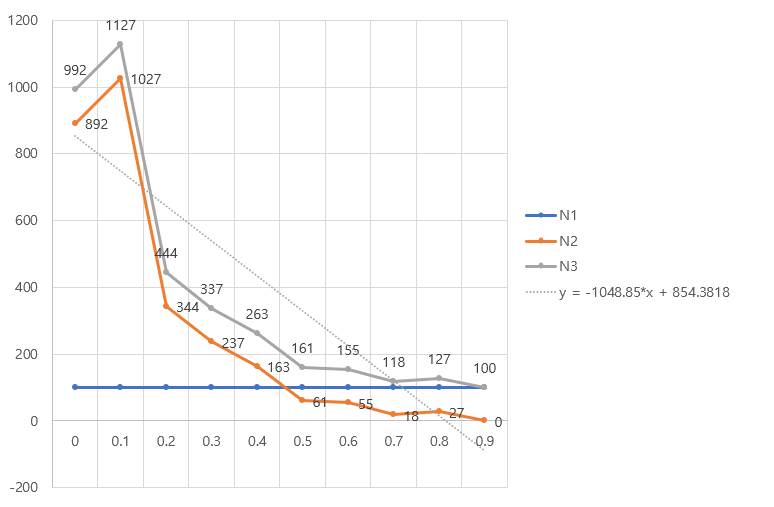
(Server)



1. 통계결과

먼저, X축의 p값에 따라서 N1의 값이 100으로 일정할 때, N2의 값을 나타내는 수식을 구한다. 이 수식을 위해서 회귀선을 이용했는데, <y= x\*b +a> 식으로,

b = {N\*(sum(x\*y)-sum(x)\*sum(y)) / {N\*sum(x^2)-(sum(x)^2)}를 이용해서 구할 수 있고 a = sum(y)/N – b\*sum(x)/N 를 통해서 구할 수 있다. 이때 x = p, y = N2, N = 10(0.1,0.2,,,0.9) 를 대입하면, 회귀선은 y = -1048.85\*x+754.3818 의 값으로 구할 수 있고, 다시 말하면 N2 = -1048.85\*p + 754.3818 이다. N3는 N2와 p값을 더한 값으로, 즉 N3 = -1048.85\*p + 854.3818 이다.

따라서 기존 그래프와 오차가 존재하면서 전체적으로 감소하는 수식을 가진다.

5. 프로그램 내역

R02\_201720768\_UDP\_assignment1 압축파일 안에

client 부분 소스파일 & exe파일

1. R02\_201720768\_UDP\_client.cpp
2. R02\_201720768\_UDP\_client.exe

server 부분 소스파일 & exe 파일

1. R02\_201720768\_UDP\_server.cpp
2. R02\_201720768\_UDP\_server.exe

* 소스파일과 exe파일만 압축
* 링커: ws2\_32.lib, 메니페스트 포함: 아니요, 특정 경고 사용 안함: 4996
* 빠른 출력 결과 확인을 위해서, Timeout 기준 시간 설정을 1초로 했기 때문에, udp라도 server 먼저 컴파일 해야 한다.