TCP over UDP

(HW#04)

1. 과제의 목표 (내용 및 도출해야 할 결과)

- 소켓 프로그래밍으로 UDP를 이용하여 TCP 구현하기
  - TCP의 많은 기능 중 일부를 구현함
    - sliding window
    - Timeout Retransmission
  - 두 개의 쓰레드를 생성하여 하나의 쓰레드가 송신을 맡고 나머지 쓰레드가 수신을 담당함
  - 라즈베리파이에는 서버를, 실습실 PC는 클라이언트를 실행하여 파일 전송으로 구현한 프로그램을 테스트함

2. 과제의 해결 방법

A. 주요 모듈 설명

1) 소스 파일 구성

* tcp.h : tcp 헤더구조체 정의, 윈도우구조체 정의 등.

* serv.c : 서버프로그램

* clie.c : 클라이언트 프로그램

2) TCP 헤더 구조체

```
typedef struct _TCP_HEADER {
                    unsigned short   tcp_sport;        /* source port 2*/
                    unsigned short   tcp_dport;        /* destination port 2*/
                    unsigned int     tcp_seq;          /* sequence number 4*/
                    unsigned int     tcp_ack;          /* acknowledge sequence4 */
                    unsigned char    tcp_offset;       /* NO USE 1*/
                    unsigned char    tcp_flag;         /* control flag 1*/
                    unsigned short   tcp_window;       /* NO USE 4*/
                    unsigned short   tcp_cksum;        /* check sum 4*/
                    unsigned short tcp_len; //4
                    unsigned short   tcp_urgptr;       /* NO USE4 */
                    unsigned char    Padding[4]; //4
                    unsigned char tcp_data[TCP_MAX_DATA];    /* 1300data part */
          } TCP_HEADER, *PTCP_HEADER;
```

위 구조체는 TCP헤더이다.

3) window 구조체

```
typedef struct _WINDOW{
                TCP_HEADER TCPseg;
                int winSeq;
                int ackOK;
                //struct sgiaction act;
}WINDOW, *PWINDOW;
```

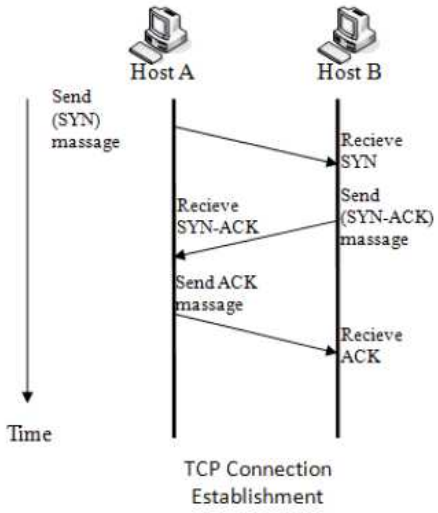
다음은 슬라이딩 윈도우에 사용될 구조체이다. 윈도우는 TCP세그먼트, 윈도우 시퀀스넘버, ack를 받았나 안받았나 확인 할 수 있는 필드로 구성되어있다. 이는 파일을 다 담을 수 있을 만큼 배열로 동적 할당된다.

4) 소켓 초기화

```
BOOL initUDPSock(int* psock,struct sockaddr_in* pserv_addr,unsigned int ipaddr, int port)//소켓 초기화
{
     *psock = socket(PF_INET, SOCK_DGRAM, 0);

     if(*psock<0){
         printf("ㅁㄴㅇㄹㄴㅇㄹㅇㄴㅁㄹ");
         return FALSE;
     }

    // printf("UDP socket desc : %d₩n", sock);
     memset(pserv_addr, 0, sizeof(*pserv_addr));
     pserv_addr->sin_family = AF_INET;
     pserv_addr->sin_addr.s_addr = ipaddr;//inet_addr(ipaddr);
     pserv_addr->sin_port = htons(port);

     return TRUE;
}
```

다음 함수는 UDP소캣을 초기화하여 포인터를 반환한다.

◎ 3way handshaking 구현



3way handshaking의 경우 순차적으로 그림과 같이 이루어지게 된다. 특별한 처리 없이 서버가 먼저 연결을 기다리고, 클라이언트 측에서 SYN패킷을 보내면 그 때부터 handshaking이 이루어지게 된다. 이렇게 handshaking하는 과정에 서버에서 SYN-ACK를 보낼 때 TCP헤더의 data필드에 파일명과 파일사이즈를 클라이언트에게 보내준다.

◎ SLIDE WINDOW 구현

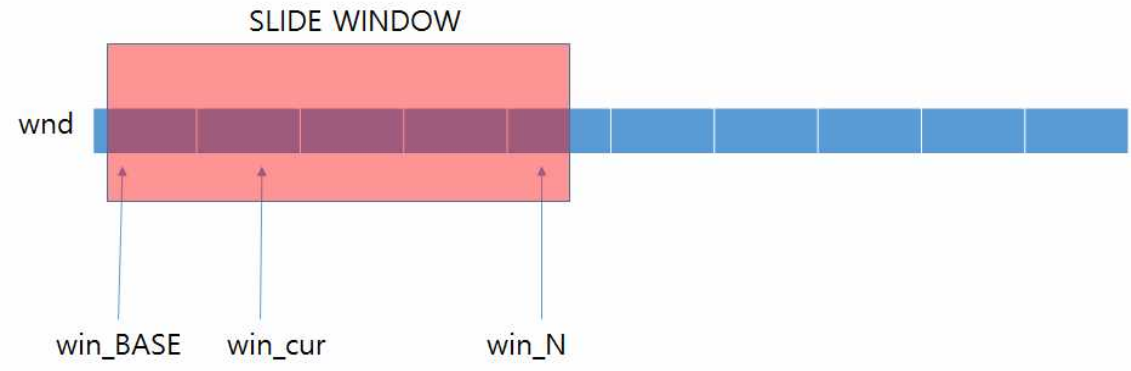
핸드쉐이킹 과정 이후 연결이 수립되게 되면 서버에서는 파일사이즈를 TCP_MAX_DATA만큼 나눈 후 +1 만큼의 윈도우 배열을 생성하게된다.

segn=(getFileSize(f)/TCP_MAX_DATA)+1

+1을 하는 이유는 TCP_MAX_DATA로 나눈후 나머지 부분까지 윈도우에 담기 위함이다.

윈도우는 다음과 같이 초기화한다.

```
void windowInit()
{ //window initialize
    wnd=(PWINDOW)malloc(sizeof(WINDOW)*segn);
    win_N=SLIDE_WINDOW_SIZE;
    win_BASE=0;
    win_cur=0;
}
```



위 그림과 같이 사용되게 된다.

win_BASE는 슬라이드윈도우의 시작점, win_cur는 현재 읽고있는 지점, win_N은 슬라이드 윈도우의 끝점을 나타낸다.

윈도우를 초기화 하고 나면 파일을 읽어 TCP_MAX_DATA만큼 쪼개 TCP헤더를 만들어 윈도우 구조체에 넣고, 시퀀스넘버를 하나하나 붙여준다.

thread_send에서 파일의 조각을 보낼 때 다음과 같은 조건으로 슬라이드 윈도우는 움직이게 된다.

① if(win_BASE+win_N>win_cur)

이 조건은 win_cur가 증가되는 조건으로 이 때 파일의 조각들이 순차적으로 전송된다.

② if(win_BASE+win_N==win_cur)

이 조건은 슬라이드 윈도우안의 파일의 조각이 다 전송되고 ACK를 기다리는 조건이다.

이때 슬라이드윈도우가 움직일 수 있는지 없는지를 검사하고 얼마나 움직일지를 검사하게 된다.

```
if(win_BASE+win_N==win_cur)
{

    int c;// moveable window size

    if((c=slidewindowmoveable())>0)
    {
        pthread_mutex_lock(&mutex);
        win_BASE+=c; //move slide window
        //printf("move slide window %d₩n",c);
        pthread_mutex_unlock(&mutex);
        alarm(1);//timer set!!
    }
    //printf("wait₩n");
}
```

얼마나 움직일까를 정하는 것은 슬라이드 윈도우 내의 송신된 패킷들 중 제일 먼저 송신한 패킷의 ACK가 왔나 안왔나를 검사하는 것으로 결정된다. 그에 관한 처리는 slidewindowmoveable() 함수에 정의 되었다.

```
int slidewindowmoveable()
{//how much move slide window????????

    int i;

    for(i=win_BASE; i<win_BASE+win_N-1;i++)
    {
        if(wnd[i].ackOK==0)
        {

            return i-win_BASE-1;
        }
    }
    //printf("moveable!!!!₩n");
    return win_N-1;

}
```

슬라이드 윈도우 내의 패킷들을 다음과 같이 ackOK필드를 검사하게 된다. 그리고 얼마나 움직일 수 있는지를 계산하여 리턴한다. 그 리턴값을 win_BASE에 더하게 됨으로써 슬라이드 윈도우를 움직이게 된다.

◎ 타임아웃에 의한 재전송

타임아웃에 의한 재전송을 구현하기위해 alarm()으로 타이머를 정의하였다. 타이머가 리셋되는 시점은 언제인가? ***전송을 시작할 때, 슬라이드 윈도우가 움직일 때, 재전송을 하였을 때*** 타이머를 리셋하게 된다.

앞서 SLIDE WINDOW 구현에서 설명했던 슬라이드윈도우가 움직이는 조건 중 **if(win_BASE+win_N==win_cur)** 조건에서 움직일 수 있는가 없는가를 판별하면서 움직일 수 없다면 while문을 돌면서 기다리게 된다. 기다리다가 앞서 타이머가 다 되면 바로 재전송이 이루어지게된다.

```
void timer(int sig)
{
    TCP_HEADER sendTCP;
    printf("timeout!!₩n");
    int i;
    for(i=win_BASE; i<win_BASE+win_N-1; i++)
    {
        if(wnd[i].ackOK==0) //retransmission
        {
            memcpy(&sendTCP,&wnd[i].TCPseg,sizeof(TCP_HEADER));
            printf("resend   sendTCP sq %d₩n",sendTCP.tcp_seq);
            sendto(udp_sock, &sendTCP, sizeof(TCP_HEADER), 0, (struct sockaddr*)&client_addr, client_addr_size);
        }
    }
    alarm(1);
}
```

타임아웃 시 호출되는 함수는 위와 같다.

슬라이드 윈도우 내의 세그먼트의 ackOK를 확인하여 ack가 도착하지 않은 세그먼트를 다시 한번 전송하게 된다. 다 전송하고 나면 타이머를 다시 리셋 한다.

◎ 클라이언트가 수신 시 하는 일

클라이언트에 구현된 기능은 의외로 간단하다. 핸드쉐이킹시 SYN_ACK패킷에 담겨있던 파일명과 파일사이즈를 가지고 수신윈도우를 할당한다. 수신 윈도우에는 각각 시퀀스 넘버가 담겨지게 된

```
printf("server: TCP connection Success!!!!!!!!₩n").
{
            ////////////////
    segn=(filesize/TCP_MAX_DATA)+1;
    wnd=(PWINDOW)malloc(sizeof(WINDOW)*segn);
    for(i=0; i<segn; i++)
    {
        wnd[i].winSeq=i;
    }

    pthread_create(&trcv,NULL,thread_rcv,0);
    pthread_join(trcv,&ee);
            ////////////////
}
```

다.

이렇게 하게 되면 수신 받은 패킷의 시퀀스 넘버와 수신윈도우의 시퀀스넘버를 같게 할 수 있어 수신 받은 패킷을 수신윈도우로 넣기 쉬워지고, 파일출력 시 순차적으로 처리만 하면 되기 때문에 편리하다.

```
while(1)
{

    recvData=(PTCP_HEADER) malloc(sizeof(TCP_HEADER));
    recvfrom(sock, recvData,sizeof(TCP_HEADER), 0, (struct sockaddr*)&from_serv, &addr_size);
    //printf("recv %d₩n",recvData->tcp_seq);
    /////////////

    memcpy(&wnd[recvData->tcp_seq].TCPseg,recvData,sizeof(TCP_HEADER));

    //printf("%d %d₩n",wnd[recvData->tcp_seq].TCPseg.tcp_seq,wnd[recvData->tcp_seq].TCPseg.tcp_len);
    //////////////
    m_sHeader.tcp_flag=ACK_FLAG;
    m_sHeader.tcp_seq=recvData->tcp_seq;

    sendto(sock, &m_sHeader, sizeof(TCP_HEADER), 0, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
    printf("sendto ACK %d₩r",recvData->tcp_seq);

    free(recvData);
    if(recvData->tcp_seq==segn-1)
        break;
}
```

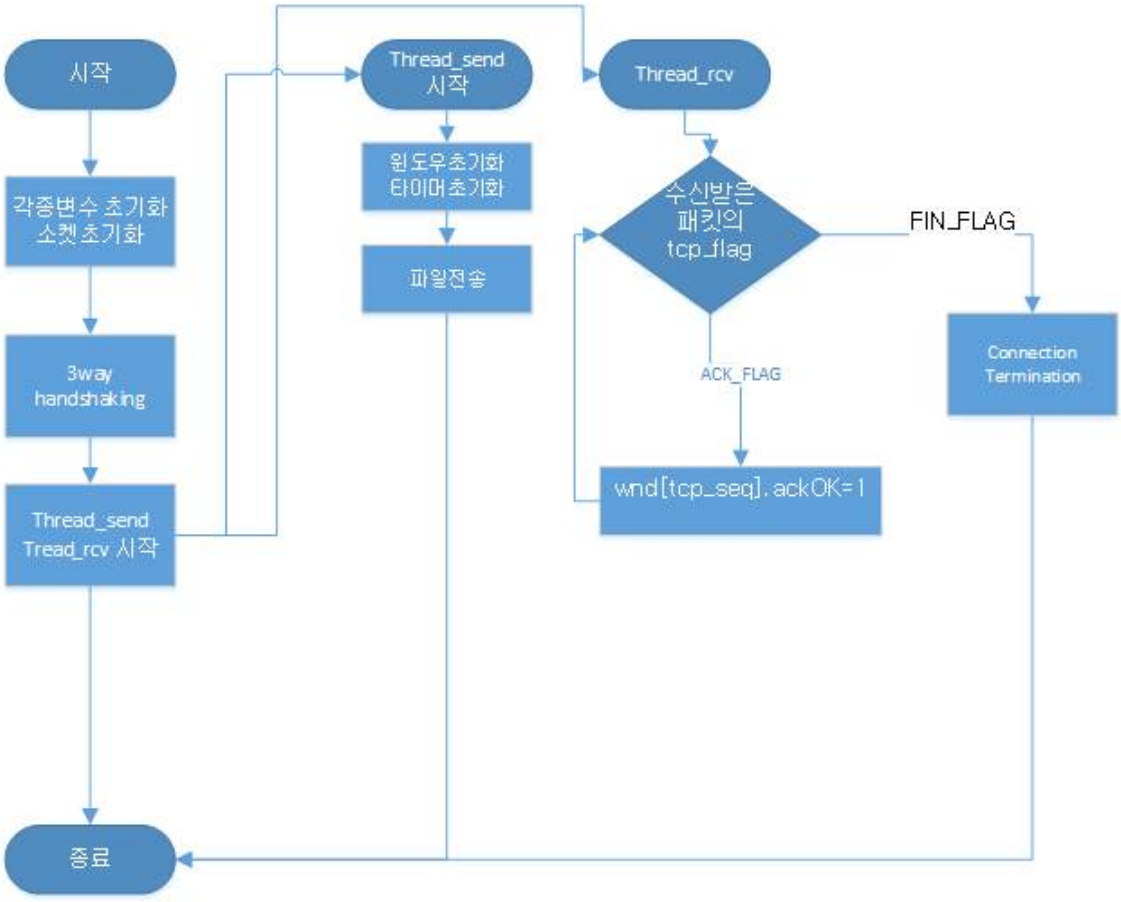
클라이언트 측에서는 그저 수신한 패킷에 대해 ACK만 제대로 보내주면 된다.

```
while(i<segn)
{
    //printf("[%d %d %d ]₩r",i,wnd[i].TCPseg.tcp_len,segn);
    //fprintf(f,"%s",wnd[i].TCPseg.tcp_data);
    fwrite(wnd[i].TCPseg.tcp_data,sizeof(unsigned char),wnd[i].TCPseg.tcp_len,f);
    i++;
}
fclose(f);
```

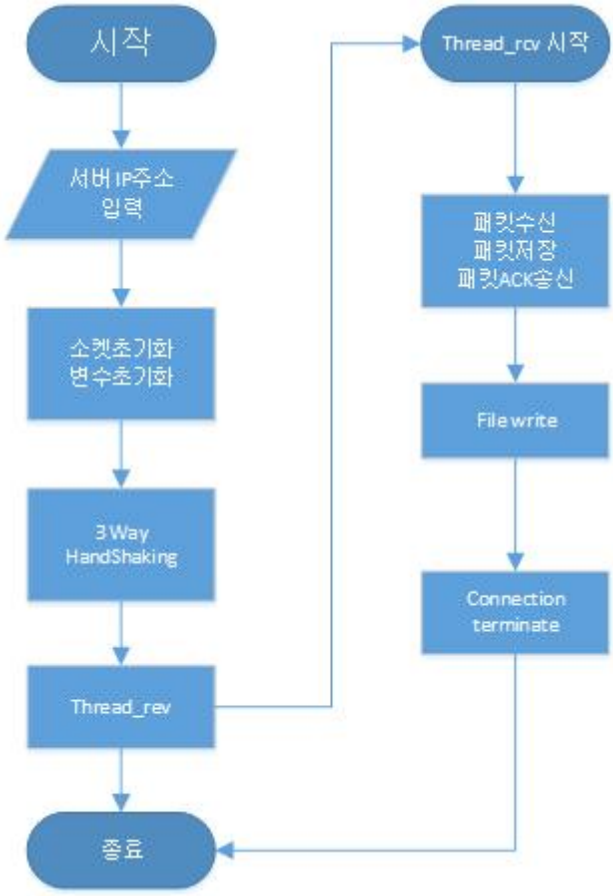
파일을 다 수신 하고나면 수신윈도우의 data를 file로 출력한다.

B. 프로그램 순서도(흐름도 or 모듈 구성도)

1) serv.c



2) clie.c


시작
서버 IP주소
입력
소켓초기화
변수초기화
3 Way
HandShaking
Thread_rev
종료
Thread_rcv 시작
패킷수신
패킷저장
패킷ACK송신
File write
Connection
terminate

3. 과제의 결과

A. **컴파일&실행 방법**

* 서버 : 라즈베리파이

* 크라이언트 : vmware 우분투13.04

* 무선랜환경

*서버컴파일& 실행

root@raspberrypi:~/hw04# gcc -o serv serv.c -lpthread

./serv

*클라이언트 컴파일

gcc -o clie clie.c -lpthread

./clie

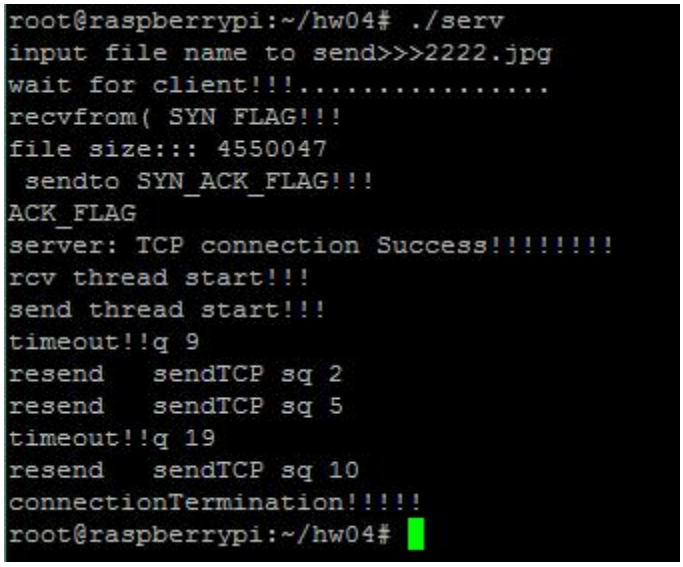
라즈베리파이로 ssh로 접속하여 테스트함

B. **결과에 대한 설명...**

우선 서버 먼저 실행되어 클라이언트를 기다리고, 클라이언트 측에서 접속을 시도하면 3way 핸드쉐이킹을 거쳐 파일 전송이 이루어지게 된다. 이때 실행코드에 2, 5, 10번을 전송하지않고 스킵하게 해두었다. 이에 대해 재전송이 재대로 이루어 졌고, 이외의 ACK가 제대로 도착하지 않은 패킷도 재전송이 제대로 이루어 진 것을 볼 수 있다.

C. 결과 화면 캡쳐

*** 서버**



*** 클라이언트**

```
1328root@user-virtual-machine:/home/user/Dropbox/linux/ComputerNetwork/hw04# ./clie
input server ip address >>>> 192.168.0.58
sendto SYN_FLAG
recvfromecvTCP->tcp_flag 3
filesize 4550047        2222.jpg
send ACK
server: TCP connection Success!!!!!!!!
rcv thread start!!!
sendto ACK 3500
file write!!

file write!! end
terminated!!!!!!connection Terminated!!!!!!!!
1328root@user-virtual-machine:/home/user/Dropbox/linux/ComputerNetwork/hw04#
```

* 전송받은 파일 확인

제대로 전송이 된 것을 볼 수 있다.

4. 과제 후기

A. 구현상 어려웠던 점

◎ 슬라이드윈도우

슬라이드 윈도우의 BASE와 윈도우 사이즈인 n으로 조건을 계산하는 것이 좀 햇갈리고 어려웠다.

또한 수업시간에 배운 개념적인 것을 실제 코드로 옮기는데 약간 어려웠다.

◎ 타이머

처음에 타이머를 모든 패킷에 붙여야 되는 줄 알고 고민했었지만, 그럴 필요가 없다는 것을 알게 되었다.

B. 이번 과제를 통해서 얻게 된 점(or 느낀 점)

중간고사 기간이 겹치고, 다른 과목의 과제들도 난이도가 있는 것들이라 구현하기 촉박하였다. 때문에 좀 더 프로그램을 모듈화 하지 못한 점이 아쉽다.

수업시간에 배운 이론을 실제 코드로 옮겨 그렇게 동작하는 것을 보니 내가 구현했어도 신기하다. 이번과제로 TCP의 이론을 좀 더 확실히 이해할 수 있었다.