프로토콜 분석모듈 설계에 의한 TCP 패킷 분석

Analysis of TCP packet by Protocol Analysis module Design

엄금용
(Gum Yong Eom)

Abstract - Transmission control protocol(TCP) is protocol used in internet. TCP is seldom transmission error and is protocol based on wire environment. TCP uses 3 way handshake ways, data transmission control through windows size, data transmission control through reception confirmation, sliding window for packet delivery. In this study, designed TCP packet abstraction module for analyze the TCP segments & correct information about TCP. TCP capture in internet using designed TCP module and analysed TCP segments composition. Through this, could analyze the correct information of protocol in network.

Key Words : TCP, Transmission protocol, 3 way handshake, TCP segments, Packet

1. 서론

전송계층은 데이터 전송을 위해 흐름제어 및 오류제어 기능을 담당하여 전이중방식(Full-duplex)을 통한 신뢰성 있는 데이터흐름 프로토콜을 구현하게 된다. 이 계층의 프로토콜로 TCP와 UDP(User Datagram Protocol)가 있다. TCP는 한 컴퓨터 시스템에서 실행되는 응용프로그램과 또 다른 컴퓨터 시스템에서 실행되는 다른 응용프로그램 사이의 동작간 신뢰할 수 있는 데이터흐름 프로토콜을 구현[1]한다. 즉 TCP는 응용프로그램으로부터 큰 블럭정보를 받아서 여러개의 세그먼트로 분리하여 다시배열하고 각 세그먼트에 번호를 부여한 후 순서대로 정렬, 전송하는 기능을 수행[2]한다. 수신측에서는 하나의 응용프로그램에 속하는 단위들이 모두 도착할 때까지 기다리고 오류여부를 확인 한 후 오류가 없도록 만들며 데이터스트림으로서 수신응용프로그램에 전달하는 임무를 수행하여 신뢰성 있는 전송기능[3]을 수행한다.

본 연구는 이러한 TCP의 신뢰성 있는 전송기능을 수행하는데 필요한 세그먼트 형식을 분석하고 TCP 패킷 추출모듈을 설계[4][5] 하였다. 설계된 TCP 패킷추출모듈을 이용하여 실제 네트워크상에서 전송되는 TCP를 캡처하여 분석[6]하고 신뢰성 있는 TCP 전송프로토콜의 기능을 확인 하였다.

2. 관련이론

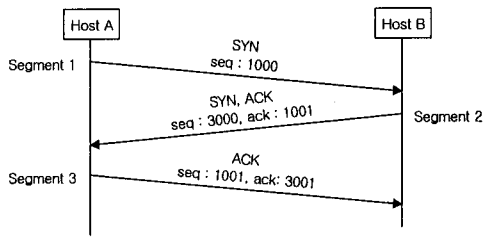
2.1 TCP의 연결설정 방법

TCP를 이용한 두 시스템 사이의 정보전달은 세그먼트(Segments) 단위로 이루어지며 다음과 같은 3가지 핸드세이크(3 Way Handshake) 통신 절차에 의하여 이루어진다.

① 동기(SYN) 세그먼트 1 전송
- 호스트 A --> 호스트 B에게 동기세그먼트1을 전송
  . 초기순서번호(1000)

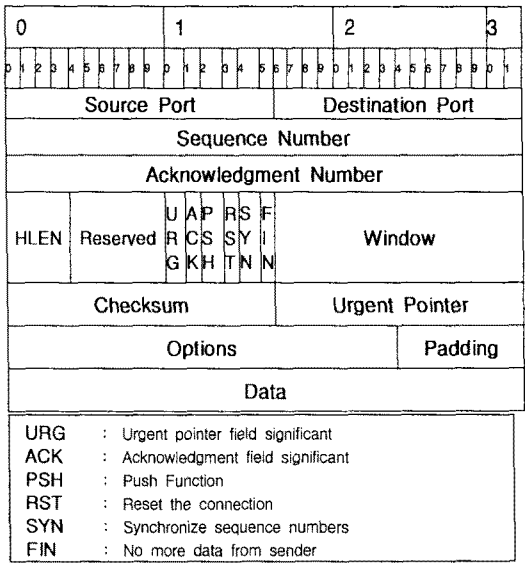
② 동기(SYN) + 승인(ACK) 세그먼트 2 전송
- 호스트 B --> 호스트 A에게 동기(SYN) + 승인(ACK)을 전송
  . 호스트 B의 초기순서번호(3000) & 초기세그먼트1의 수신확인(1001)

③ 승인(ACK) 세그먼트 3 전송
- 호스트 A --> 호스트 B에게 승인 세그먼트 전송
  . 세그먼트2의 수신확인(3001) & 세그먼트2의 순서번호(1001)



2.2 TCP 세그먼트 형식

TCP는 신뢰성 있는 전송계층 역할 수행을 위해 일정한 정보를 나타내는 헤더를 가지게 되며 두 장치 사이에 전달되는 데이터단위, 즉 세그먼트의 형식은 다음과 같다.



3. TCP의 모듈설계

TCP 패킷을 추출하여 신뢰성 있는 전송기능을 확보[7]하기 위하여 TCP 헤더모듈을 다음과 같이 설계 하였다.

```
//TCP Analysis module design

#include "stdafx.h"
#include "LANEdu.h"
#include "LANEduDoc.h"
#include "LANEduView.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


#define ETHERLEN                    14
#define IEEE802_3LEN        17
unsigned short tcp_chk(unsigned short *ptcp, unsigned short *pip, int len)
{
        unsigned chksum = 0;
        unsigned short finalchk = 0;
        unsigned short ntcp_len = len;
        unsigned short nip_len = 8;
        unsigned short length;
        int i;
        length = ntcp_len;
        nip_len >>= 1;
        ntcp_len >>= 1;
                        for(i = 0; i < nip_len; i++)
        {
           chksum += (*(pip + 6 + i));
        }
                        //length += 17;
        length += 6;
                        chksum += htons(length);
        for(i = 0; i < ntcp_len; i++)
        {
          chksum += (*ptcp++);
        }
                        if (len%2 == 1)
                        {
                          *(BYTE *) (&finalchk) = *(BYTE *) ptcp;
                          chksum += finalchk;
                        }
        chksum = (chksum >> 16) + (chksum & 0xffff);
        chksum += (chksum >> 16);
        finalchk = (~chksum & 0xffff);
        return finalchk;
}
struct DATA_IP
{
                BYTE            ip_verlen;      // IP version & header length (in longs)
                BYTE            ip_tos;                 // type of service
                WORD            ip_len;                 // total packet length (in octets)
                WORD            ip_id;                  // datagram id
                WORD            ip_fragoff;      // fragment offset (in 8-octet's)
                BYTE            ip_ttl;         // time to live, in gateway hops
                BYTE            ip_proto;       // IP protocol (see IPT_* above)
                WORD            ip_cksum;               // header checksum
                BYTE            ip_src[4];      // IP address of source
                BYTE            ip_dst[4];      // IP address of destination
                BYTE            ip_data[1];      // 뒤의 데이터의 포인터


if (usType==0x0800) //IP Protocol analysis
                        {
WriteText("\r\n==================== IP Protocol ====================");
sprintf(csText, "\r\nVersion : %X ", pPacket[14]>>4);
                WriteText(csText);
sprintf(csText, "\r\nInternet Header Length : %d ", (pPacket[14]&0x0f)*4));
                WriteText(csText);
sprintf(csText, "\r\nType of Service : %X ", pPacket[15]);
                WriteText(csText);
sprintf(csText, "\r\nTotal Length : %d", (pPacket[16]<<8)+pPacket[17]);
                WriteText(csText);
sprintf(csText, "\r\nIdentifier: %d", (pPacket[18]<<8)+pPacket[19]);
                WriteText(csText);
sprintf(csText, "\r\nFlag: %02X\r\nFragment Offset: %02X", pPacket[20],pPacket[21]);
                WriteText(csText);
sprintf(csText, "\r\nTime to Live: %d\r\nProtocol: %02X", pPacket[22], pPacket[23]);
                WriteText(csText);
sprintf(csText, "\r\nIP Header Checksum: %02X%02X", pPacket[24], pPacket[25]);
                WriteText(csText);
sprintf(csText, "\r\nSource IP address : %d.%d.%d.%d", pPacket[26], pPacket[27], pPacket[28],
                 pPacket[29]);
                WriteText(csText);
                                }
   if (pPacket[26] <128) {
sprintf(csText,"\r\n\tClass A  : Net ID : %d.0.0.0, Host ID : %d.%d.%d.%d "
      , pPacket[26], pPacket[26],pPacket[27],pPacket[28],pPacket[29]);
                WriteText(csText);
                                }
    else if (pPacket[26]<192){
sprintf(csText,"\r\n\tClass B  : Net ID : %d.%d.0.0, Host ID : %d.%d.%d.%d"
, pPacket[26],pPacket[27],pPacket[26],pPacket[27], pPacket[28],pPacket[29]);
                WriteText(csText);
                                }
          else {
sprintf(csText,"\r\n\tClass C  : Net ID : %d.%d.%d.0, Host ID : %d.%d.%d.%d"
, pPacket[26], pPacket[27],     pPacket[28],pPacket[26],pPacket[27],pPacket[28], pPacket[29]);
                WriteText(csText);
                }
 sprintf(csText, "\r\nDestination IP address : %d.%d.%d.%d"
 , pPacket[30], pPacket[31], pPacket[32], pPacket[33]);
                WriteText(csText);
       if (pPacket[30] <128) {
 sprintf(csText,"\r\n\tClass A  : Net ID : %d.0.0.0, Host ID : %d.%d.%d.%d "
 , pPacket[30], pPacket[30],pPacket[31],pPacket[32],pPacket[33]);
                WriteText(csText);
                                }
      else if (pPacket[30]<192){
 sprintf(csText,"\r\n\tClass B  : Net ID : %d.%d.0.0, Host ID : %d.%d.%d.%d"
 , pPacket[30],pPacket[31],pPacket[30],pPacket[31], pPacket[32],pPacket[33]);
                WriteText(csText);
                                }
```

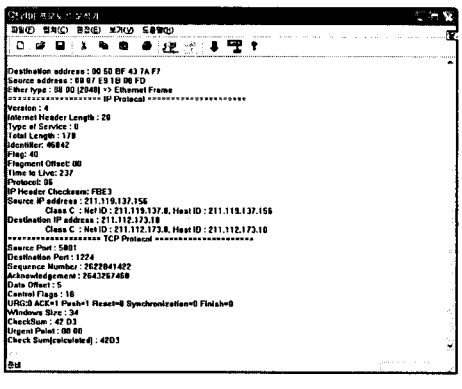
```
            else {
    sprintf(csText,"\r\n\tClass C  : Net ID : %d.%d.%d.0, Host ID : %d.%d.%d.%d"
    , pPacket[30], pPacket[31],    pPacket[32],pPacket[30],pPacket[31],pPacket[32], pPacket[33]);
            WriteText(csText);
                        }

    else if(pPacket[23]==0x06) { //TCP Protocol

WriteText("\r\n==================== TCP Protocol ====================");
    sprintf(csText,"\r\nSource Port : %d", (pPacket[34]<<8)+pPacket[35]);
            WriteText(csText);
    sprintf(csText,"\r\nDestination Port : %d", (pPacket[36]<<8)+pPacket[37]);
            WriteText(csText);
    sprintf(csText,"\r\nSequence Number : %u"
    //          (pPacket[38]<<24)+(pPacket[39]<<16)+(pPacket[40]<<8)+pPacket[41]);
                h2nl(pPacket+38));
            WriteText(csText);
    sprintf(csText,"\r\nAcknowledgement : %u"
    //          (pPacket[42]<<24)+(pPacket[43]<<16)+(pPacket[44]<<8)+pPacket[45]);
                h2nl(pPacket+42));
            WriteText(csText);
    sprintf(csText,"\r\nData Offset : %d",pPacket[46]>>4);
            WriteText(csText);
    sprintf(csText,"\r\nControl Flags : %02X", pPacket[47]);
            WriteText(csText);
    sprintf(csText,"\r\nURG:%d ACK=%d Push=%d Reset=%d Synchronization=%d Finish=%d"
    ,(pPacket[47]&0x20)>>5, (pPacket[47]&0x10)>>4, (pPacket[47]&0x08)>>3,(pPacket[47]&0x04)>>2
    ,(pPacket[47]&0x02)>>1, (pPacket[47]&0x01));
            WriteText(csText);
     sprintf(csText,"\r\nWindows Size : %d", pPacket[48]<<8+pPacket[49]);
            WriteText(csText);
     sprintf(csText,"\r\nCheckSum : %02X %02X", pPacket[50],pPacket[51]);
            WriteText(csText);
      sprintf(csText,"\r\nUrgent Point : %02X %02X", pPacket[52],pPacket[53]);
            WriteText(csText);
            DATA_IP* pIp = (DATA_IP*)(pPacket + 14);
            DATA_TCP* pTcp = (DATA_TCP*)pIp->ip_data;
            WORD chk, result, len;
            chk = pTcp->checksum;
            pTcp->checksum = 0;
            len = (pPacket[16]<<8)+pPacket[17] - 20;
            result = tcp_chk((WORD*)pTcp, (WORD*)pIp, len);
    sprintf(csText,"\r\nCheck Sum(calculated) : %02X", htons(result));
            WriteText(csText);
            } //end of TCP Protocol
```

4. 결과분석

TCP의 신뢰성 있는 전송기능을 수행하는데 필요한 세그먼트 형식을 분석하고 TCP 패킷 추출모듈을 설계 하였다. 설계된 TCP의 패킷모듈을 실제 네트워크상에서 캡처하여 분석한 결과를 다음에 나타내었다.



TCP 패킷을 실제 캡처한 결과 헤더의 구성요소에서 송신지포트번호, 목적지 포트번호, 전체 TCP의 길이, TCP 체크섬, 전송된 데이터 등에 대한 분석을 할 수가 있었다. 이는 TCP 설계 알고리즘에 의하여 설계된 TCP가 실제 네트워크상에서 데이터를 전송하는 전송계층 역할 수행 시 설계된 패킷추출모듈을 전송하고 있음을 나타내는 결과로 사료된다. 설계된 TCP 패킷추출모듈을 통하여 신뢰성 있는 전송기능을 수행함을 확인 할 수 있었다.

4. 결론

네트워크에서 전이중방식을 통한 신뢰성 있는 데이터흐름 프로토콜이 TCP 이다. TCP는 응용프로그램으로부터 큰 블럭정보를 받아서 여러개의 세그먼트로 분리하여 다시 배열하고 각 세그먼트에 번호를 부여한 후 순서대로 정렬, 전송하는 기능을 수행한다.

본 연구에서는 이러한 TCP의 전송기능 수행에서 신뢰성 있는 전송기능을 수행하는데 필요한 패킷모듈을 설계하였다. 설계된 TCP 패킷추출모듈을 이용하여 실제 네트워크상에서 전송되는 TCP를 캡쳐한 결과 설계된 TCP 패킷추출모듈을 정확히 분석 및 확인 할 수 있었다. TCP 패킷추출모듈 분석을 통하여 신뢰성 있는 TCP 전송프로토콜의 기능을 확인 할 수 있었다.

참 고 문 헌

[1] 김범준, 김석규, 이재용, "상호연관성을 갖는 연속적인 패킷 손실에 대한 TCP 손실복구 성능분석", 한국통신학회 논문지, 제 27권, 7B호, 2004.

[2] 이보미, 정여진, 임혜숙, " TCP/IP Hardware Accelerator를 위한 TCP Engine 설계", 한국통신학회 논문지, 제 29권, 5B호, 2004.

[3] 박종훈, 배세인, 우명식, "버스트에러에 강인한 TCP 알고리즘", 한국통신학회 논문지, 제 29권, 6B호, 2004.

[4] 정길현, 이정규 , "멀티캐스트 트래픽 처리를 위한 WDM 포로토콜의 성능분석", 한국통신학회 논문지, 제 26권, 11호, 2001.

[5] 엄금용, 임정희, "UDP의 데이터그램을 이용한 체크섬과 프로토콜 분석", 한국통신학회 학술지, 제 21권, 9호, 2004.

[6] 서원, "TCP/IP 체크섬 기능이 내장된 고속 이더넷 MAC 제어기의 개발과 성능 분석기", 한국통신학회 논문지, 제 26권, 2호, 2001.

[7] 김동민, 김범준, 김석규, 이재용, "빠른 손실감지를 통한 TCP NewReno의 Fast Recovery 개선 알고리즘", 한국통신학회 논문지, 제 27권, 6B호, 2004.