컴퓨터 네트워크

- HW09 : PacketCature_ProtocolHeader -

제 출 일	2017년 11월 19일
분 반	00반
담당교수	김기일
학 과	컴퓨터공학과
학 번	201302423
이 름	신종욱

1. 코드 및 분석

일단 sudo find 명령을 이용하여서 헤더들의 구조체를 파악하였다.

ethernet 헤더

u201302423@u201302423: /usr/include/net

```
31   systems.  */
32 struct ether_addr
33 {
34   u_int8_t ether_addr_octet[ETH_ALEN];
35 } __attribute__ ((__packed__));
36
37 /* 10Mb/s ethernet header */
38 struct ether_header
39 {
40   u_int8_t  ether_dhost[ETH_ALEN];  /* destination eth addr */
41   u_int8_t  ether_shost[ETH_ALEN];  /* source ether addr */
42   u_int16_t ether_type;             /* packet type ID field  */
43 } __attribute__ ((__packed__));
44
```

IP헤더

```
struct iphdr
  {
#if __BYTE_ORDER == __LITTLE_ENDIAN
    unsigned int ihl:4;
    unsigned int version:4;
#elif __BYTE_ORDER == __BIG_ENDIAN
    unsigned int version:4;
    unsigned int ihl:4;
#else
# error "Please fix <bits/endian.h>"
#endif
    u_int8_t tos;
    u_int16_t tot_len;
    u_int16_t id;
    u_int16_t frag_off;
    u_int8_t ttl;
    u_int8_t protocol;
    u_int16_t check;
    u_int32_t saddr;
    u_int32_t daddr;
    /*The options start here. */
  };
                                                    64,1          15%
```

ICMP헤더 구조체

```
struct icmphdr
{
  u_int8_t type;                /* message type */
  u_int8_t code;                /* type sub-code */
  u_int16_t checksum;
  union
  {
    struct
    {
      u_int16_t id;
      u_int16_t sequence;
    } echo;                     /* echo datagram */
    u_int32_t   gateway;        /* gateway address */
    struct
    {
      u_int16_t __glibc_reserved;
      u_int16_t mtu;
    } frag;                     /* path mtu discovery */
  } un;
};
                                                        23,0-1        8%
```

TCP헤더 구조체

root@u201302423: /usr/include/netinet

```
/*
 * TCP header.
 * Per RFC 793, September, 1981.
 */
struct tcphdr
  {
    __extension__ union
    {
      struct
      {
        u_int16_t th_sport;             /* source port */
        u_int16_t th_dport;             /* destination port */
        tcp_seq th_seq;         /* sequence number */
        tcp_seq th_ack;         /* acknowledgement number */
# if __BYTE_ORDER == __LITTLE_ENDIAN
        u_int8_t th_x2:4;               /* (unused) */
        u_int8_t th_off:4;              /* data offset */
# endif
# if __BYTE_ORDER == __BIG_ENDIAN
        u_int8_t th_off:4;              /* data offset */
        u_int8_t th_x2:4;               /* (unused) */
# endif
        u_int8_t th_flags;
                                                        93,1-8        26%
```

아래에 더 많은 데이터들이 있다.

UDP헤더 구조체

root@u201302423: /usr/include/netinet

```
/* UDP header as specified by RFC 768, August 1980. */

struct udphdr
{
  __extension__ union
  {
    struct
    {
      u_int16_t uh_sport;               /* source port */
      u_int16_t uh_dport;               /* destination port */
      u_int16_t uh_ulen;                /* udp length */
      u_int16_t uh_sum;         /* udp checksum */
    };
    struct
    {
      u_int16_t source;
      u_int16_t dest;
      u_int16_t len;
      u_int16_t check;
    };
  };
};
```

Ethernet,IP 헤더출력 코드

```
void ethernet_header(unsigned char* Buffer, int Size)
{
    struct ethhdr *eth = (struct ethhdr *)Buffer;

    printf( "\n");
    printf( "Ethernet Header\n");
    printf( "   |-Destination Address : %.2X-%.2X-%.2X-%.2X-%.2X-%.2X \n", eth->h_dest[0] , eth->h_dest[1] , eth->h_
    printf( "   |-Source Address      : %.2X-%.2X-%.2X-%.2X-%.2X-%.2X \n", eth->h_source[0] , eth->h_source[1] , eth-
    printf( "   |-Protocol            : %u \n",(unsigned short)eth->h_proto);
}//이더넷 헤더에는 도착주소 출발주소 그리고 프로토콜이 저장되어있다.

void ip_header(unsigned char* Buffer, int Size)
{
    ethernet_header(Buffer , Size);

    unsigned short iphdrlen;

    struct iphdr *iph = (struct iphdr *)(Buffer  + sizeof(struct ethhdr) );
    iphdrlen =iph->ihl*4;

    memset(&source, 0, sizeof(source));
    source.sin_addr.s_addr = iph->saddr;

    memset(&dest, 0, sizeof(dest));
    dest.sin_addr.s_addr = iph->daddr;

    printf( "\n");
    printf( "IP Header\n");
    printf( "   |-IP Version        : %d\n",(unsigned int)iph->version);
    printf( "   |-IP Header Length  : %d DWORDS or %d Bytes\n",(unsigned int)iph->ihl,((unsigned int)(iph->ihl))*4);
    printf( "   |-Type Of Service   : %d\n",(unsigned int)iph->tos);
    printf( "   |-IP Total Length   : %d  Bytes(Size of Packet)\n",ntohs(iph->tot_len));
    printf( "   |-Identification    : %d\n",ntohs(iph->id));
    printf( "   |-TTL      : %d\n",(unsigned int)iph->ttl);
    printf( "   |-Protocol : %d\n",(unsigned int)iph->protocol);
    printf( "   |-Checksum : %d\n",ntohs(iph->check));
    printf( "   |-Source IP        : %s\n",inet_ntoa(source.sin_addr));
    printf( "   |-Destination IP   : %s\n",inet_ntoa(dest.sin_addr));
}//IP헤더에는 버전,헤더길이,IP길이,TTL,프로토콜,체크섬등이 있다.
```

일단 모든출력에 들어가는 이더넷 헤더와 IP헤더 출력문이다.
버퍼를 받아 해당 헤더에 맞게 형변환을 해준후 해당 구조체에 맞게 원하는 정보를 출력하도록 하였다.

ICMP 출력 코드

```
void icmp_packet(unsigned char* Buffer , int Size)
{//Icmp는 대표적으로 Ping보내 서로 연결이 잘되어있는지 확인할때 사용되는 프로토콜이다.
 //간단한 기능인 만큼 헤더의 크기는 작은편이다.
    unsigned short iphdrlen;

    struct iphdr *iph = (struct iphdr *)(Buffer  + sizeof(struct ethhdr));
    iphdrlen = iph->ihl * 4;

    struct icmphdr *icmph = (struct icmphdr *)(Buffer + iphdrlen  + sizeof(struct ethhdr));


    printf( "\n\n--------------------ICMP Packet--------------------\n");

    ip_header(Buffer , Size);

    printf( "\n");

    printf( "ICMP Header\n");
    printf( "   |-Type : %d",(unsigned int)(icmph->type));
    printf( "   |-Code : %d\n",(unsigned int)(icmph->code));
    printf( "   |-Checksum : %d\n",ntohs(icmph->checksum));

    printf( "\n");
    printf( "\n------------------------------------------------------------");
}
```

핑을 보낼 때 주로 실행되는 icmp이다 헤더에는 타입과 코드 그리고 체크섬 등이있다.

UDP헤더 출력 코드

```
void udp_packet(unsigned char *Buffer , int Size)
{
    //프로토콜이 17번일때 UDP이다 UDP는 단순하고 기능이 적은대신 빠른 통신 방법이기때문에 헤더의 크기는 작다.
    unsigned short iphdrlen;

    struct iphdr *iph = (struct iphdr *)(Buffer +  sizeof(struct ethhdr));
    iphdrlen = iph->ihl*4;

    struct udphdr *udph = (struct udphdr*)(Buffer + iphdrlen  + sizeof(struct ethhdr));

    printf( "\n\n--------------------UDP Packet--------------------\n");

    ip_header(Buffer,Size);

    printf( "\nUDP Header\n");
    printf( "   |-Source Port      : %d\n" , ntohs(udph->source));
    printf( "   |-Destination Port : %d\n" , ntohs(udph->dest));
    printf( "   |-UDP Length       : %d\n" , ntohs(udph->len));
    printf( "   |-UDP Checksum     : %d\n" , ntohs(udph->check));

    printf( "\n");
    printf( "\n-----------------------------------------------------------");
}
```

데이터를 주고받으며 통신을 할 때 가볍고 빠르게 하기위한 Udp통신의 헤더이다
빠른만큼 기능도 적고 헤더의 크기도 적다 도착,출발포트, 길이 체크섬을 출력하였다.

TCP헤더 출력코드

```
void tcp_packet(unsigned char* Buffer, int Size)
{//프로토콜이 6번일때 TCP이다 TCP는 다양한 기능들이 있다. 그만큼 헤더의 양도 많다.
    unsigned short iphdrlen;

    struct iphdr *iph = (struct iphdr *)( Buffer  + sizeof(struct ethhdr) );
    iphdrlen = iph->ihl*4;

    struct tcphdr *tcph=(struct tcphdr*)(Buffer + iphdrlen + sizeof(struct ethhdr));

    printf( "\n\n------------------TCP Packet--------------------\n");

    ip_header(Buffer,Size);

    printf( "\n");
    printf( "TCP Header\n");
    printf( "   |-Source Port      : %u\n",ntohs(tcph->source));
    printf( "   |-Destination Port : %u\n",ntohs(tcph->dest));
    printf( "   |-Sequence Number    : %u\n",ntohl(tcph->seq));
    printf( "   |-Acknowledge Number : %u\n",ntohl(tcph->ack_seq));
    printf( "   |-Header Length      : %d DWORDS or %d BYTES\n" ,(unsigned int)tcph->do
    printf( "   |-Urgent Flag          : %d\n",(unsigned int)tcph->urg);
    printf( "   |-Acknowledgement Flag : %d\n",(unsigned int)tcph->ack);
    printf( "   |-Push Flag            : %d\n",(unsigned int)tcph->psh);
    printf( "   |-Reset Flag           : %d\n",(unsigned int)tcph->rst);
    printf( "   |-Synchronise Flag     : %d\n",(unsigned int)tcph->syn);
    printf( "   |-Finish Flag          : %d\n",(unsigned int)tcph->fin);
    printf( "   |-Window         : %d\n",ntohs(tcph->window));
    printf( "   |-Checksum       : %d\n",ntohs(tcph->check));
    printf( "   |-Urgent Pointer : %d\n",tcph->urg_ptr);
    printf( "\n");
    printf( "\n");
    printf( "\n-----------------------------------------------------------------");
}
```

TCP는 UDP와 달리 다양한 기능을 가지고 있지만 그만큼 헤더의 크기가 커져있다 리눅스의 헤더파일도 그만큼 많은 양을 가지고있었다.

2. 실행 결과

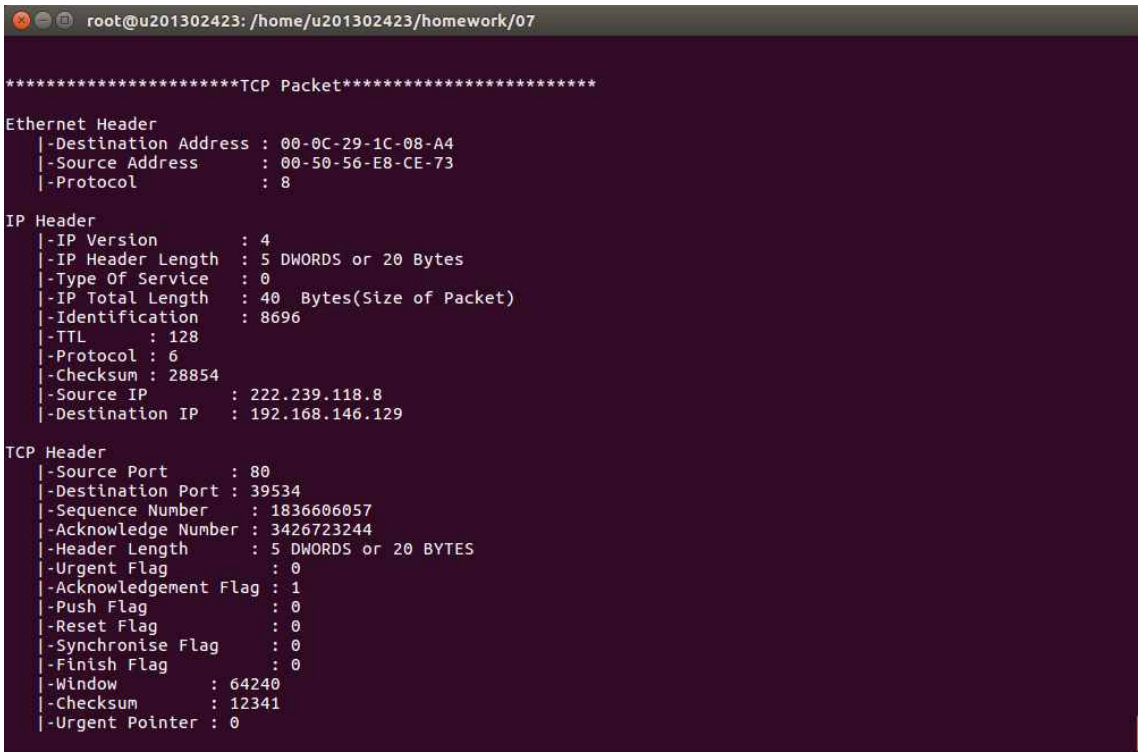
ICMP 출력

```
--------------------ICMP Packet--------------------

Ethernet Header
   |-Destination Address : 00-50-56-E8-CE-73
   |-Source Address      : 00-0C-29-1C-08-A4
   |-Protocol            : 8

IP Header
   |-IP Version        : 4
   |-IP Header Length  : 5 DWORDS or 20 Bytes
   |-Type Of Service   : 0
   |-IP Total Length   : 84  Bytes(Size of Packet)
   |-Identification    : 34717
   |-TTL      : 64
   |-Protocol : 1
   |-Checksum : 898
   |-Source IP        : 192.168.146.129
   |-Destination IP   : 125.209.222.142

ICMP Header
   |-Type : 8   |-Code : 0
   |-Checksum : 30451
```

UDP 출력

```
root@u201302423: /home/u201302423/homework/07
-----------------------------------------------------

--------------------UDP Packet--------------------

Ethernet Header
   |-Destination Address : 00-0C-29-1C-08-A4
   |-Source Address      : 00-50-56-E8-CE-73
   |-Protocol            : 8

IP Header
   |-IP Version        : 4
   |-IP Header Length  : 5 DWORDS or 20 Bytes
   |-Type Of Service   : 0
   |-IP Total Length   : 184  Bytes(Size of Packet)
   |-Identification    : 12328
   |-TTL      : 128
   |-Protocol : 17
   |-Checksum : 25656
   |-Source IP        : 192.168.146.2
   |-Destination IP   : 192.168.146.129

UDP Header
   |-Source Port      : 53
   |-Destination Port : 41180
   |-UDP Length       : 164
   |-UDP Checksum     : 52717
```

TCP 출력



3.느낀점

인터넷에 많은 자료들이 있어서 수정하여서 쉽게 만들었다. 단 패킷을 출력할때는 너무 TCP위주로 출력되어서 캡쳐하기위해 TCP출력을 끄고 테스트를 하였더니 잘되었다.