데이터통신

제출일	2019. 03. 17
과제번호	02
분반	02
학과	컴퓨터공학과
학번	201502087
이름	이세형

코드 및 설명

ip header

```
def parsing_ip_header(data):
    ip_header = struct.unpack("!1c1c2s2s2s1c1c2s4c4c", data)
    ip_version = int(ip_header[0].hex()[0],16)
    ip_Length = int(ip_header[0].hex()[1],16)
    differentiated_service_codepoint = ip_header[1].hex()[0]
    explicit_congestion_notification = ip_header[1].hex()[1]
    total_length = int(ip_header[2].hex(),16)
    identification = int(ip_header[3].hex(),16)
    flags = "0x" + ip_header[4].hex()
    reserved_bit =  ip_header[4].hex()[0]
    not_fragments = ip_header[4].hex()[1]
    fragments = ip_header[4].hex()[2]
    fragments_offset = ip_header[4].hex()[3]
    Timetolive = int(ip_header[5].hex(),16)
    protocol = int(ip_header[6].hex(),16)
    header_checksum = "0x" + ip_header[7].hex()
    source_ip_address = convert_ip_address(ip_header[8:12])
    dest_ip_address = convert_ip_address(ip_header[12:16])

    print("======ip header======")
    print("ip_version:", ip_version)
    print("ip_Length:", ip_Length)
    print("differentiated_service_codepoint:", differentiated_service_codepoint)
    print("explicit_congestion_notification:", explicit_congestion_notification)
    print("total_length:", total_length)
    print("identification:", identification)
    print("flags:", flags)
    print(">>>reserved_bit:", reserved_bit)
    print(">>>not_fragments:", not_fragments)
    print(">>>fragments:", fragments)
    print(">>>fragments_offset:", fragments_offset)
    print("Time to live:", Timetolive)
    print("protocol:",protocol)
    print("header checksum:",header_checksum)
    print("source_ip_address:", source_ip_address)
    print("dest_ip_address:", dest_ip_address)

    return protocol;

def convert_ip_address(data):
    ip_addr = list()
    for i in data:
        ip_addr.append(str(int(i.hex(),16)))
    ip_addr = ".".join(ip_addr)
    return ip_addr
```

wireshark를 통해서 IP header를 분석한 결과는 아래와 같고 아래와 같습니다.

IP header
  ip_header = 총 14바이트.
  ip_version = 4비트
  ip_Length = 4비트
  differentiated_service_codepoint = 4비트
  explicit_congestion_notification = 4비트
  total_length = 2바이트
  identification = 2바이트
  flags = 2바이트고 밑에 4개 포함 -hex
  reserved_bit = 4비트
  not_fragments = 4비트
  fragments = 4비트
  fragments_offset = 4비트
  Time to live = 1바이트
  protocol = 1바이트
  header checksum = 2바이트 -hex
  source_ip_address = 4바이트
  dest_ip_address = 4바이트

ethenet 의 예제를 참고해서 위의 분석결과에 맞게 파싱하여서 int로 변환하거나 hex로 변환해서 결과를 도출했습니다. 또한 ip header parsing 이후에 tcp , udp header parsing을 할지 결정해야하므로 이에대한 protocol값을 리턴을 하게끔 했습니다.

TCP header

```
def parsing_tcp_header(data):
    tcp_header = struct.unpack("!2s2s4s4s2s2s2s2s", data)
    src_port = int(tcp_header[0].hex(),16)
    dec_port = int(tcp_header[1].hex(),16)
    seq_num = int(tcp_header[2].hex(),16)
    ack_num = int(tcp_header[3].hex(),16)
    header_len = int(tcp_header[4].hex()[0],16)
    flags = "0x" + tcp_header[4].hex()[1:4]
    flag_bi = bin(int(flags,16))[2:].zfill(12)
    reserved = int(flag_bi[0:3])
    nonce = flag_bi[4]
    cwr = flag_bi[5]
    urgent = flag_bi[6]
    ack = flag_bi[7]
    push = flag_bi[8]
    reset = flag_bi[9]
    syn = flag_bi[10]
    fin = flag_bi[11]
    window_size_value = int(tcp_header[5].hex(),16)
    checksum = "0x"+ tcp_header[6].hex()
    urgent_pointer = int(tcp_header[7].hex(),16)


    print("======tcp_header======")
    print("src_port:", src_port)
    print("dec_port:", dec_port)
    print("seq_num:",seq_num)
    print("ack_num:",ack_num)
    print("flags:", flags)
    print(">>>reserved:",reserved)
    print(">>>nonce:", nonce)
    print(">>>cwr:", cwr)
    print(">>>urgent:",urgent)
    print(">>>ack:",ack)
    print(">>>push:",push)
    print(">>>reset:",reset)
    print(">>>syn:",syn)
    print(">>>fin:",fin)
    print("window_size_value:",window_size_value)
    print("checksum:",checksum)
    print("urgent_pointer:", urgent_pointer)
```

wireshark를 통해서 Tcp header를 분석한 결과는 아래와 같고 아래와 같습니다.

TCP

Src port = 2byte
des port = 2byte
seq_num = 4byte
ack_num = 4byte
header_len, flags (hex)= 2byte (앞의 4비트 len , 뒤의 12비트중 3bit , 1bit x 9 해서 flags 10개)
windows_size = 2byte
checksum_size = 2byte hex
urgent = 2byte

ethenet 의 예제와 직접구현한 ip header 함수를 참고해서 위의 분석결과에 맞게 파싱하여서 int로 변환하거나 hex로 변환해서 결과를 도출했습니다.

UDP header

```
def parsing_udp_header(data):
    udp_header = struct.unpack("!2s2s2s2s", data)
    src_port = int(udp_header[0].hex(),16)
    dec_port = int(udp_header[1].hex(),16)
    leng = int(udp_header[2].hex(),16)
    header_checksum = "0x"+ udp_header[3].hex()


    print("======udp_header======")
    print("src_port:", src_port)
    print("dec_port:", dec_port)
    print("leng:",leng)
    print("header_checksum:",header_checksum)
```

wireshark를 통해서 Udp header를 분석한 결과는 아래와 같고 아래와 같습니다.

UDP

총 8**byte**
Src port = 2byte
des port = 2byte
leng = 2byte
header_checksum = 2byte - hex

ethenet 의 예제와 직접구현한 ip header parsing , tcp header parsing 를 참고해서 위의

분석결과에 맞게 파싱하여서 int로 변환하거나 hex로 변환해서 결과를 도출했습니다.

Main 부분

```
recv_socket = socket.socket(socket.AF_PACKET,socket.SOCK_RAW, socket.ntohs(0x800))

while True:
    data = recv_socket.recvfrom(20000)
    parsing_ethernet_header(data[0][0:14])
    protocol = parsing_ip_header(data[0][14:34])
    if(protocol==6):
        parsing_tcp_header(data[0][34:54])
    if(protocol==17):
        parsing_udp_header(data[0][34:42])
```

실습에서 했던방법대로 recv_socket를 통해 패킷정보들을 받아와서

처음 14 바이트에 해당하는 정보를 ethernet parsing 에 넣어주고 분석을하고

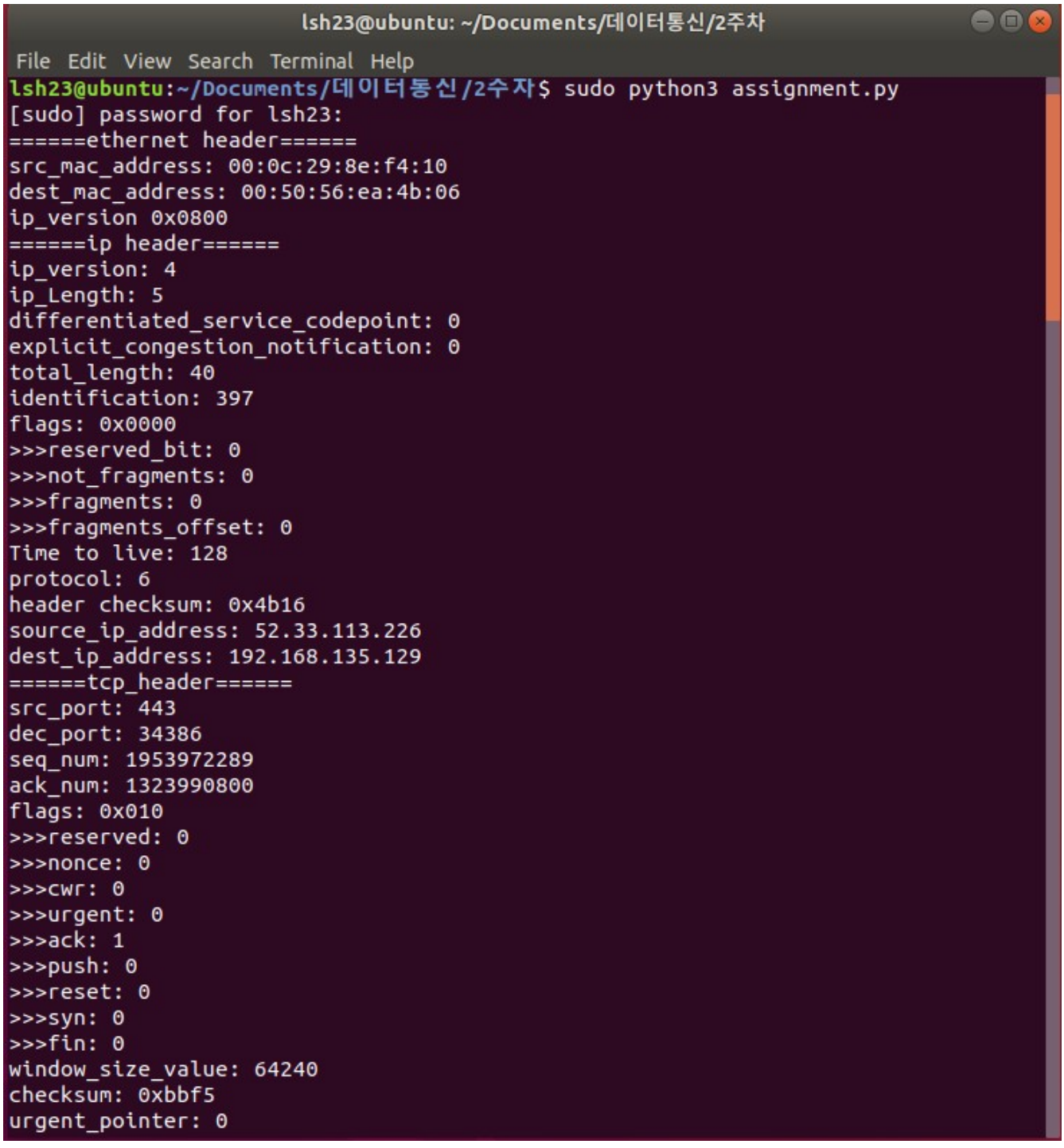
그다음 20바이트는 ip_header_parsing 함수에 넣어서 분석을 한뒤 protocol 값을 반환받아서

반환값이 tcp에 해당하면 tcp headr parsing을 udp에 해당하면 udp 분석을 하도록 코드를

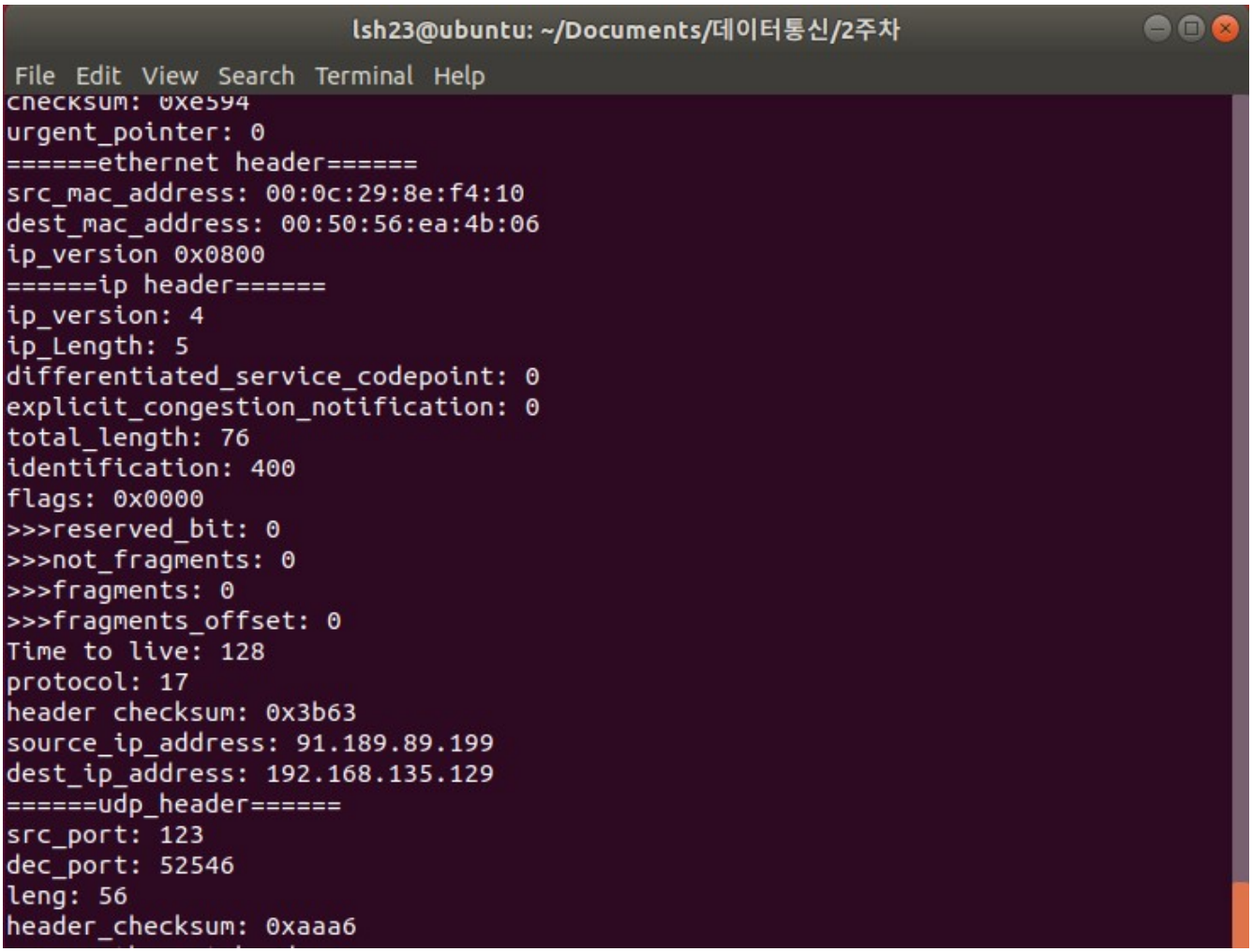
작성했습니다.

실행 결과

실행 결과 - tcp



실행 결과 – udp



느낀점

방학때 python에 대해서 조금 공부했었는데 그걸 해보지 않았다면 상당히 애를 먹었을 것 같다. 사실 python을 잘하는 것은 아니지만 그럼에도 데이터를 받아와서 hex 값을 bit단위로 쪼개거나 원하는 만큼 파싱해서 int값으로 바꾸거나하는 과정이 간단하지는 않았다. 그래도 이번과제를 하면서 세세하게 header 들을 분석해봄으로서 조금은 layer 계층에 대한 이해가 높아진 것 같아서 좋았다.