2. Packet Capture

김수현
shkim950921@cs-cnu.org
데이터네트워크 연구실(633호)

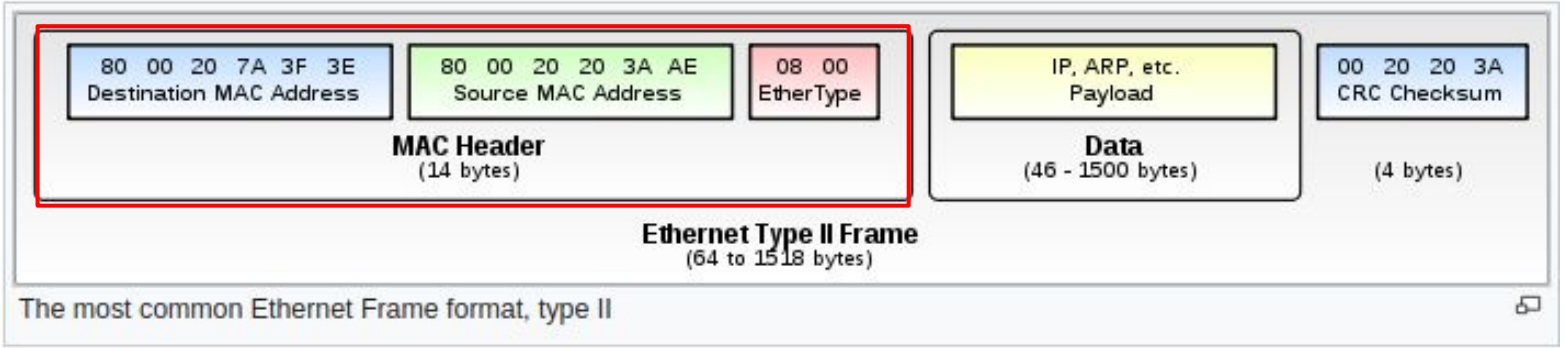
0. 지난과제리뷰 - 1

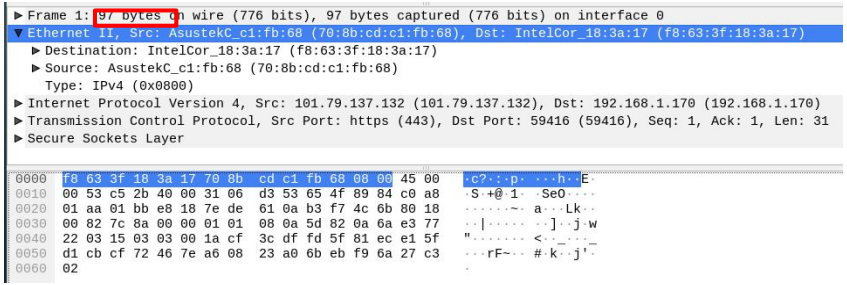
- 제출률 : 47/50
- 채점 시트 : https://docs.google.com/spreadsheets/d/1WR5nEUEwVyvrUgBgt2OyREWdJbp7RbgOFMW45JGoVvc/edit?usp=sharing
- 채점 기준 :
  - 패킷을 보고 어떤 구조로 되어있는지 혼자 스스로 공부해봤다는것이 지난실습의 목표
    - ⇒ 틀려도 점수는 깎지 않았음
  - 감점 요인
    - 설명이나 사진없이 답만 올려져 있는경우
    - 5개모두 패킷 분석을 하지 않은경우
    - 문제에 대한 답이 없는경우
  - 채점안함(ppt에 형식안맞추면 채점안한다 명시)
    - .zip파일 아닌경우
    - .pdf파일 아닌경우
  - 기타: .pcap파일로 저장할것

0. 지난과제리뷰 - 2

- Is the frame an outgoing or an incoming frame?
- What is the source IP address of the network-layer header in the frame?
- What is the destination IP address of the network-layer header in the frame?
- What is the total number of bytes in the whole frame?
- What is the number of bytes in the Ethernet (data-link layer) header?
- What is the number of bytes in the IP header?
- What is the number of bytes in the TCP header?
- What is the total bytes in the message (at the application layer)?

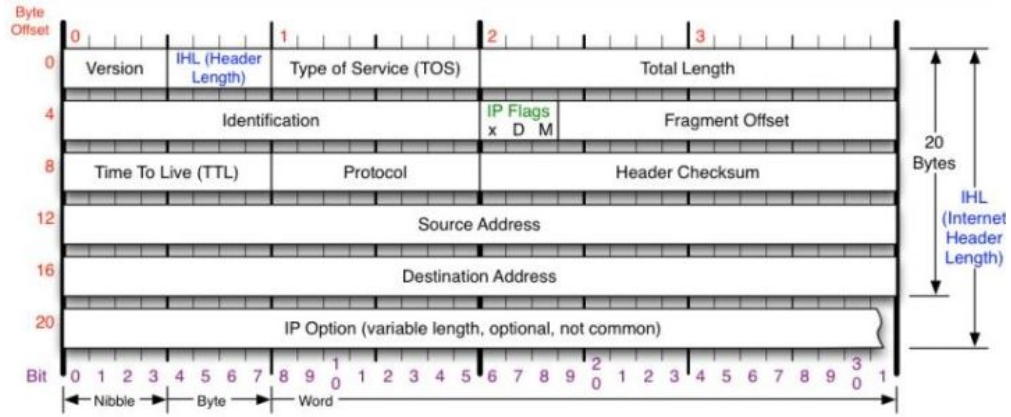
0. 지난 과제 리뷰 - 이더넷 헤더





- What is the total number of bytes in the whole frame?
- What is the number of bytes in the Ethernet (data-link layer) header?
- 각컴퓨터(NIC)들이 가진 고유의 MAC주소
- 네트워크 카드들은 이더넷 헤더를보고 자신에게 온것인지 확인한다.

0. 지난 과제 리뷰 - IP헤더



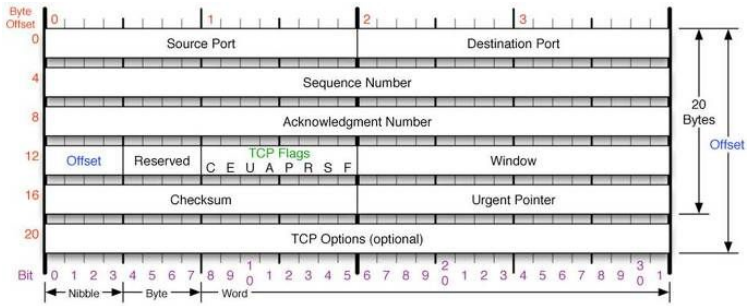

```
▼ Internet Protocol Version 4, Src: 172.217.26.42 (172.217.26.42), Dst: 192.168.1.170 (192.168.1.170
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 86
    Identification: 0xbd40 (48448)
  ▶ Flags: 0x0000
    Time to live: 117
    Protocol: TCP (6)
    Header checksum: 0xff0b [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.217.26.42 (172.217.26.42)
    <Source or Destination Address: 172.217.26.42 (172.217.26.42)>
    <[Source Host: 172.217.26.42]>
    <[Source or Destination Host: 172.217.26.42]>
    Destination: 192.168.1.170 (192.168.1.170)
    <Source or Destination Address: 192.168.1.170 (192.168.1.170)>
    <[Destination Host: 192.168.1.170]>
    <[Source or Destination Host: 192.168.1.170]>
```

- Is the frame an outgoing or an incoming frame?
- What is the source IP address of the network-layer header in the frame?
- What is the destination IP address of the network-layer header in the frame?

주 역할:

- 주소를 지정해주며, 길찾는 역할을 해줌
- 통신 노드 간의 IP패킷을 전송하는 기능과 라우팅 기능을 담당한다.

0. 지난과제 리뷰 - TCP헤더 구조

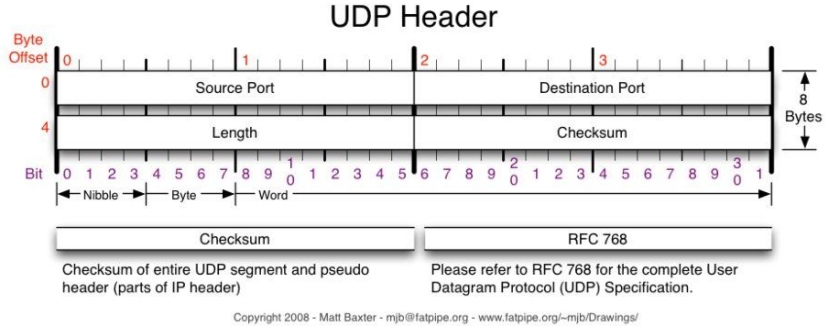



```
▼ Transmission Control Protocol, Src Port: https (443), Dst Port: 41626 (41626), Seq: 1475594076, Ack: 2580430059, Len: 34
    Source Port: https (443)
    Destination Port: 41626 (41626)
    <Source or Destination Port: https (443)>
    <Source or Destination Port: 41626 (41626)>
    [Stream index: 0]
    [TCP Segment Len: 34]
    Sequence number: 1475594076
    [Next sequence number: 1475594110]
    Acknowledgment number: 2580430059
    1000 .... = Header Length: 32 bytes (8)
  ▶ Flags: 0x018 (PSH, ACK)
    Window size value: 244
    [Calculated window size: 62464]
    [Window size scaling factor: 256]
    Checksum: 0x846f [correct]
    [Checksum Status: Good]
    [Calculated Checksum: 0x846f]
    Urgent pointer: 0
  ▶ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▶ [SEQ/ACK analysis]
  ▶ [Timestamps]
    TCP payload (34 bytes)
```

- What is the number of bytes in the TCP header?
- Transport Layer
- 송신자와 수신자를 연결하는 통신서비스 제공
- (신뢰성, 흐름제어, 오류검출등등)

- TCP 최소 헤더길이 :20 bytes
- Option이 추가되면 최대 40바이트까지 늘어날수있음

0. 지난과제 리뷰 - UDP헤더

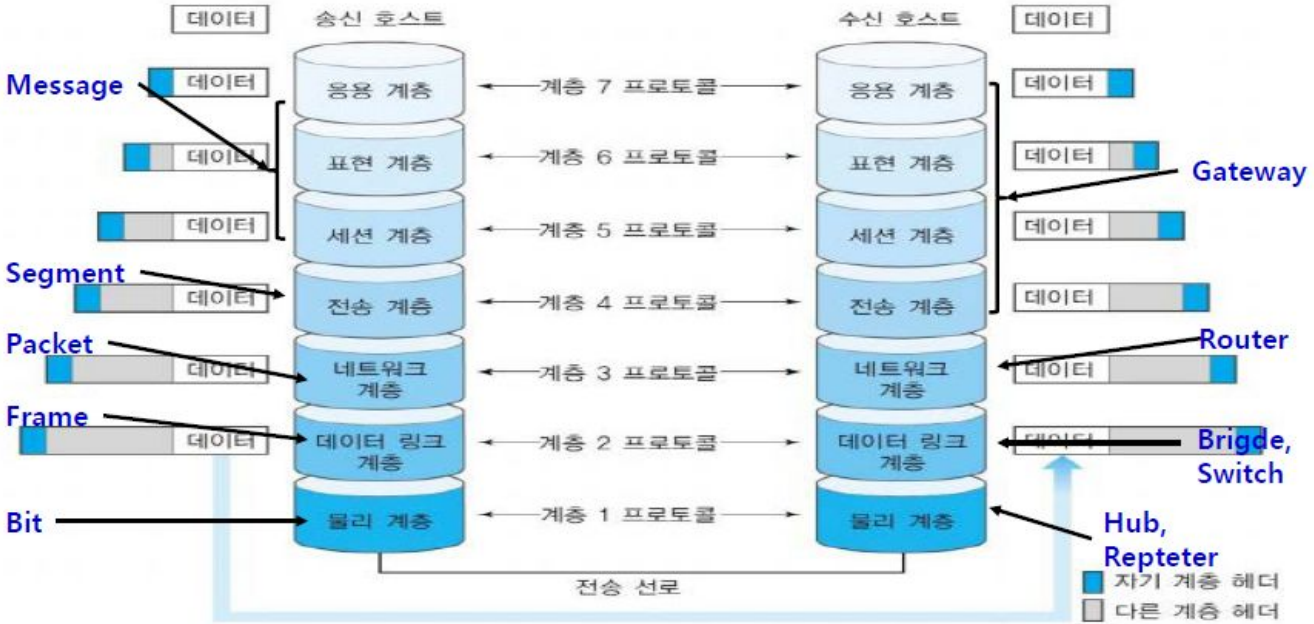



```
▼ User Datagram Protocol, Src Port: https (443), Dst Port: 54831 (54831)
    Source Port: https (443)
    Destination Port: 54831 (54831)
    <Source or Destination Port: https (443)>
    <Source or Destination Port: 54831 (54831)>
    Length: 33
    Checksum: 0xc30a [unverified]
    [Checksum Status: Unverified]
    [Stream index: 2]
```

- Transport Layer
- 송신자와 수신자를 연결하는 통신서비스 제공
- TCP와 유사하나 비신뢰성과 비연결성임

0. 지난과제 리뷰 (PDU)

- What is the total bytes in the message (at the application layer)?



목표

1. Ethernet, IPv4, TCP, UDP의 헤더 구조를 파악한다.
2. python struct모듈을 이용하여 패킷 구조 분석한다.
3. python으로 패킷 캡처 기능을 구현한다.

1. 이번주 과제

1. wireshark를 사용하지 않고 파이썬으로 패킷 캡처를 구현할것

2. 패킷을 파싱하여 ethernet_header, tcp_header, udp_header의 정보를 출력할것

1. 이번주 과제

- 파이썬
  a. 스크립트언어
  b. 인터프리터식, 객체지향적, 동적 타이핑 대화형 언어이다.
  c. 문법참고자료 : https://wikidocs.net/book/1

1. 이번주 과제 - 구현 화면

```
<<<<<<Packet Capture Start>>>>>>
========ethernet_header=======
src_mac_address: f8:63:3f:18:3a:17
dest_mac_address: 70:8b:cd:c1:fb:68
ip_version: 0x0800
=========ip_header==========
ip_version: 4
ip_Length: 5
differentiated_service_codepoint: 0
explicit_congestion_notification: 0
total_length: 40
identification 0
flags: 0x4000
>>>reserved_bit: 0
>>>not_fragments: 1
>>>fragments: 0
>>>fragments_offset: 0
Time to live: 50
protocol: 06
header checksum: 0x21e9
source_ip_addrses: 125.209.230.195
dest_ip_address: 192.168.1.170
=========tcp_header==========
src_port: 443
dec_port: 55276
seq_num: 3604219216
ack_num: 1504381297
header_len: 5
flags: 16
>>>reserved: 0
>>>nonce: 0
>>>cwr: 0
>>>urgent: 0
>>>ack: 1
>>>push: 0
>>>reset: 0
>>>syn: 0
>>>fin: 0
window_size_value: 137
checksum: 28539
urgent_pointer: 0
```

tcp일 경우

```
<<<<<<Packet Capture Start>>>>>>
========ethernet_header=======
src_mac_address: f8:63:3f:18:3a:17
dest_mac_address: 70:8b:cd:c1:fb:68
ip_version: 0x0800
=========ip_header==========
ip_version: 4
ip_Length: 5
differentiated_service_codepoint: 0
explicit_congestion_notification: 0
total_length: 1378
identification 0
flags: 0x4000
>>>reserved_bit: 0
>>>not_fragments: 1
>>>fragments: 0
>>>fragments_offset: 0
Time to live: 51
protocol: 11
header checksum: 0xe24f
source_ip_addrses: 216.58.197.174
dest_ip_address: 192.168.1.170
=========udp_header==========
src_port: 443
dst_port: 45122
leng: 1358
header checksum: 0xf983
```

udp일경우

2. 과제를 구현하기위해 알아야할것

1. 각 패킷의 종류 및 헤더 구조
2. python socket programming
3. python struct 모듈

2. 과제를 구현하기위해 알아야할것

1. 각 패킷의 종류 및 헤더 구조 ⇒ 지난과제리뷰때 설명
2. python socket programming
3. python struct 모듈

2. python socket programming(Linux 계열)

- NIC(네트워크 카드)에서 패킷을 송수신함
- 송수신한 패킷은 파일로 저장이됨(Linux 계열)
- 인터넷이 되기까지 복잡한 과정을 거침(인터럽트처리, 레이어처리등등)
- 이러한 복잡한 과정을 추상화한것이 socket
- java, C, Python등도 라이브러리 및 모듈로서 사용됨
- 복잡한 작업을 거치지 않고도 socket 모듈을 사용하면 간단하게 상대 컴퓨터와 통신가능함

2. python socket programming

- python에서 socket programming을 위한 기본 라이브러리
- https://docs.python.org/3/library/socket.html

```
socket.socket(family=AF_INET, type=SOCK_STREAM, proto=0, fileno=None)
```

- 소캣 생성
  - socket Type
    - SOCK_STREAM: TCP용 소캣 생성
    - SOCK_DGRAM : UDP용 소캣 생성
    - SOCK_RAW: SOCKET 바이너리를 볼 수 있다.

2. python socket programming -예제

```
import socket
recv_socket=socket.socket(socket.PF_PACKET, socket.SOCK_RAW, socket.ntohs(0x0800))
data = recv_socket.recvfrom(65565)
```

- socket.PF_PACKET: 프로토콜 패밀리(Ipv4, Ipv6등 다양한 프로토콜들을 사용하겠다)
- socket.SOCK_RAW : ip 밑에단도 보겠다.
  - SOCK.DGRAM이라면 ? UDP패킷만 캡처됨
  - SOCK.STREAM 이라면? TCP패킷만 캡처됨
- socket.ntohs(0x800)
  - Ipv4 wireshark에서 찍으면? 0x800 ⇒ ipv4만 보겠다.

- 생성한 소켓으로부터 조건을 만족하는 패킷들을 65565바이트만큼 읽어들이겠다.
- 65565는 버퍼크기 임의로 설정가능
- 너무 작으면 하나의 패킷이 안들어올 수 있음

2. python socket programming

```
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import socket
>>> recv_socket=socket.socket(socket.PF_PACKET, socket.SOCK_RAW, socket.ntohs(0x0800))
>>> data = recv_socket.recvfrom(65565)
>>> print(data)
(b'\xf8c?\x18:\x17\x08]\xddK\xd2#\x08\x00E\x00\x00b\xaf\x9e@\x00\xf3\x066@4U\xc0)\xac\x1e\x01\x1a\x01\xbb\xe8(\xd1\x003\xd8\xd
5\xf2/\x16\x80\x18\x00~\x05\xf3\x00\x00\x01\x01\x08\ng}i\xe3j\xee\x1c\xb3\x17\x03\x03\x00)\xf4\x91\xd0\xf5BT\xe82\x11\xa5\x9a\
x15\x08Y+9\x87[[t\x8c\tDt\xcfp\x1a\r\x18G\x8f\xc0\xdb\x91F\xdd\x95\xf5\xed\xa9\xda', ('wlp1s0', 2048, 0, 1, b'\x08]\xddK\xd2#'
))
```

- 그대로 읽어들이면
- wireshark 바이트코드 캡처한것처럼 보이게됨
- 다음을 파싱해서 원하는 데이터를 출력해야함

3. struct

- 데이터를 바이트화/바이트데이터를 복호화해주는 모듈
- 네트워크 데이터 파싱할때 사용
- https://docs.python.org/3/library/struct.html#format-strings
- C언어의 **FormatString**을 이용하여 바이트데이터 복호화

`struct.`**pack**(*format*, *v1*, *v2*, ...)

Return a bytes object containing the values *v1*, *v2*, ... packed according to the format string *format*. The arguments must match the values required by the format exactly.

`struct.`**unpack**(*format*, *buffer*)

Unpack from the buffer *buffer* (presumably packed by `pack(format, ...)`) according to the format string *format*. The result is a tuple even if it contains exactly one item. The buffer's size in bytes must match the size required by the format, as reflected by `calcsize()`.

Format	C Type	Python type	Standard size	Notes
x	pad byte	no value		
c	char	bytes of length 1	1	
b	signed char	integer	1	(1),(3)
B	unsigned char	integer	1	(3)
?	_Bool	bool	1	(1)
h	short	integer	2	(3)
H	unsigned short	integer	2	(3)
i	int	integer	4	(3)
I	unsigned int	integer	4	(3)
l	long	integer	4	(3)
L	unsigned long	integer	4	(3)
q	long long	integer	8	(2), (3)
Q	unsigned long long	integer	8	(2), (3)
n	ssize_t	integer		(4)
N	size_t	integer		(4)
e	(7)	float	2	(5)
f	float	float	4	(5)
d	double	float	8	(5)
s	char[]	bytes		
p	char[]	bytes		
P	void *	integer		(6)

3. struct - 예제

kimsoohyun@kimsoohyun-900X5N: ~/Desktop/network/dataCommunication_lecture/DataCommunic...

File Edit View Search Terminal Help

```python
import struct

number = 28
number_bytes = number.to_bytes(1, byteorder = "big")

decoded_number = struct.unpack("!1b", number_bytes)
print("Origin Data:", number_bytes)
print("Decoded Data:", decoded_number[0])



number = 36
number_bytes = number.to_bytes(4, byteorder = "big")

decoded_number = struct.unpack("!i", number_bytes)
print("Origin Data : ", number_bytes)
print("Decoded Data : ", decoded_number[0])
```

```
kimsoohyun@kimsoohyun-900X5N:~/Desktop/network/dataCommunicati
on_lecture/DataCommunicationLecture/2week$ python3 struct-exam
ple1.py
Origin Data: b'\x1c'
Decoded Data: 28
Origin Data : b'\x00\x00\x00$'
Decoded Data :  36
```

3. struct - 예제2

여러개 바이트도 복호화할 수 있음

kimsoohyun@kimsoohyun-900X5N: ~/Desktop/network/dataCommunication_lecture/DataCommunicationLecture/2week

File Edit View Search Terminal Help

```
import struct

number1 = 28
number2 = 36

number_bytes = number1.to_bytes(1,byteorder="big") + number2.to_bytes(4,byteorder = "big")
print("Origin Data :", number_bytes)

decoded_data = struct.unpack("!1bi", number_bytes)
print("Decoded Data[0]", decoded_data[0])
print("Decoded Data[1]", decoded_data[1])
```

```
kimsoohyun@kimsoohyun-900X5N:~/Desktop/network/data
Communication_lecture/DataCommunicationLecture/2wee
k$ python3 struct-example1.py
Origin Data : b'\x1c\x00\x00\x00$'
Decoded Data[0] 28
Decoded Data[1] 36
```

4. ethernet parsing

```python
import socket
import struct


def parsing_ethernet_header(data):
    ethernet_header = struct.unpack("!6c6c2s", data)
    ether_src = convert_ethernet_address(ethernet_header[0:6])
    ether_dest = convert_ethernet_address(ethernet_header[6:12])
    ip_header = "0x"+ethernet_header[12].hex()

    print("======ethernet header======")
    print("src_mac_address:", ether_src)
    print("dest_mac_address:", ether_dest)
    print("ip_version",ip_header)

def convert_ethernet_address(data):
    ethernet_addr = list()
    for i in data:
        ethernet_addr.append(i.hex())
    ethernet_addr = ":".join(ethernet_addr)
    return ethernet_addr


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])
```

```
======ethernet header======
src_mac_address: f8:63:3f:18:3a:17
dest_mac_address: 70:8b:cd:c1:fb:68
ip_version 0x0800
======ethernet header======
src_mac_address: f8:63:3f:18:3a:17
dest_mac_address: 70:8b:cd:c1:fb:68
ip_version 0x0800
======ethernet header======
src_mac_address: f8:63:3f:18:3a:17
dest_mac_address: 70:8b:cd:c1:fb:68
ip_version 0x0800
```

Homework

- 패킷캡처를 구현해서(이더넷,ip,tcp,udp) 보고서와 코드 제출
- tcp에서는 Option을 제외한 20바이트만 캡처
- 제출일 : 3월 20일 17시 59까지
- Github Classroom URL : **https://classroom.github.com/a/MMZQLtMf**
- 과제제출 방법 :https://docs.google.com/presentation/d/1cPaHMZirRn5xln7LgGQEzENN5rfC0i3DVWFiyyxp9JU/edit#slide=id.g51bb9611e5_0_64
- 보고서:
  - 코드는 github classroom에
  - 보고서는 이러닝사이트에

유의사항

- Github Classroom에 제출, 제출기한 반드시 확인할것
- 파일명 : DC02_02(과제번호)_학번_이름.py
- ex) DC02_01_20170000_김수현.py
  - 형식 지켜지지 않을시 채점안함
  - 보고서:PDF로 작성할것(HWP, DOC은 채점안함)
    - 과제 목표(도출해야할 결과)
    - 질문에 대한 답변 (사진 및 대답에 대한 이유 서술 필수)
    - 과제후기(느낀점 및 조교에게 하고싶은말, 선택사항)
    - 최소채점기준: ethernet까지구현한경우 2점

유의사항(Cont'd)

- 실습조교:김수현
- 메일:shkim950921@cs-cnu.org
- 연구실 633호(데이터 네트워크 연구실)
  - 방문전 사전 메일 필수
  - 가능하면 18시 이후 방문 요망, 18시 이전 방문은 못받을 수 있음
- 메일 보낼시 지켜야할 사항
  - 제목 : [데이터통신] 학번_이름
    - 지키지 않을시 질문 메일을 못볼 수 있음
  - OS환경 사전명시(예 - ubuntu 14.04.3 LTS 64bit)