네트워크 프로그래밍
- 소켓 프로그래밍 (UDP) -

순천향대학교 사물인터넷학과

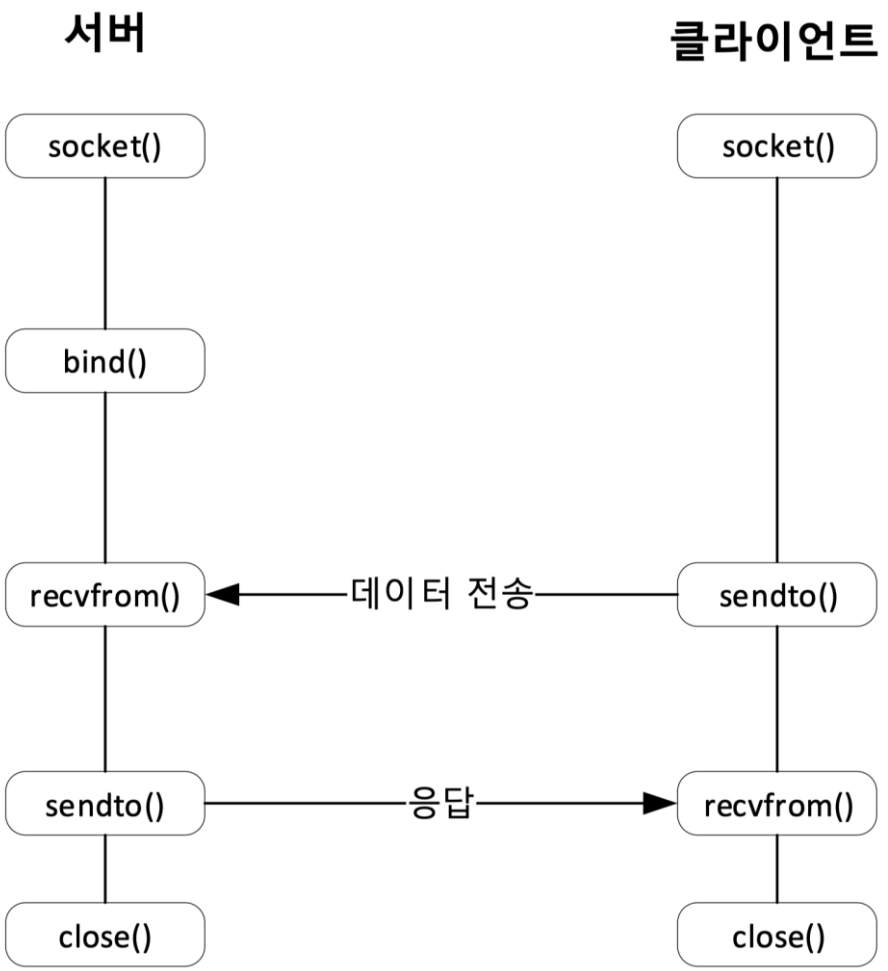
UDP 소켓 프로그래밍

UDP 소켓 프로그래밍 순서

- 연결 설정 과정이 없음
  - ✓ listen(), accept(), connect()
- UDP 소켓 유형: SOCK_DGRAM
- 서버
  - ✓ 소켓 생성(socket()) 후 바인드(bind())
- 클라이언트
  - ✓ 소켓 생성(socket()) 후 송신

- 송신: sendto(data, addr)
- 수신: recvfrom(buffer_size)

- 편지를 이용한 통신과 유사
  - ✓ 주소를 적어 보내면 전달됨
  - ✓ 하지만, 중간에 분실될 수 있음



UDP 에코 서버/클라이언트 프로그램

UDP 에코 서버

- 클라이언트로부터 수신한 데이터를 출력하고, 상대방에게 다시 전송
- 연결 설정이 없으므로, 다수의 사용자와 송수신 가능

udp_echo_server.py

```
import socket

port = 2500
BUFFSIZE = 1024

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind(('', port))

while True:
    msg, addr = sock.recvfrom(BUFFSIZE)
    print('Received: ', msg.decode())

    sock.sendto(msg, addr)
```

UDP 에코 서버/클라이언트 프로그램

UDP 에코 클라이언트

- 서버로 메시지를 전송하고, 수신 메시지를 출력
- UDP 프로토콜은 connect()와 같은 연결이 없음

udp_echo_client.py


```python
import socket

port = 2500
BUFFSIZE = 1024

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

while True:
    msg = input('Enter a message: ')
    if msg == 'q':
        break
    sock.sendto(msg.encode(), ('localhost', port))
    data, addr = sock.recvfrom(BUFFSIZE)
    print('Server says: ', data.decode())

sock.close()
```

```
(20241234) PS D:\library\src> python .\udp_echo_server.py
Received:  Hello, UDP
Received:  UDP is easy!!!
```

```
(20241234) PS D:\library\src> python .\udp_echo_client.py
Enter a message: Hello, UDP
Server says:  Hello, UDP
Enter a message: UDP is easy!!!
Server says:  UDP is easy!!!
Enter a message: q
(20241234) PS D:\library\src>
```

1:1 UDP 채팅 프로그램

- 1:1 채팅을 할 수 있는 서버 프로그램
  - 수신 메시지는 화면에 출력하고, 키보드 입력 메시지를 상대방으로 송신하는 서버 프로그램
  - 지속적인 채팅이 가능하도록 무한루프로 실행

udp_chat_server.py

```
from socket import *

port = 3333
BUFFSIZE = 1024

sock = socket(AF_INET, SOCK_DGRAM)
sock.bind(('', port))

while True:
    data, addr = sock.recvfrom(BUFFSIZE)
    print('<- ', data.decode())
    resp = input('-> ')
    sock.sendto(resp.encode(), addr)
```

1:1 UDP 채팅 프로그램

- 1:1 채팅을 할 수 있는 클라이언트 프로그램
  - 키보드 입력 메시지를 상대방으로 송신하고 수신 메시지는 화면에 출력하는 클라이언트 프로그램
  - 지속적인 채팅이 가하도록 무한루프로 실행

udp_chat_client.py

```
from socket import *

port = 3333
BUFFSIZE = 1024

sock = socket(AF_INET, SOCK_DGRAM)

while True:
    msg = input('-> ')
    sock.sendto(msg.encode(), ('localhost', port))
    data, addr = sock.recvfrom(BUFFSIZE)
    print('<- ', data.decode())
```

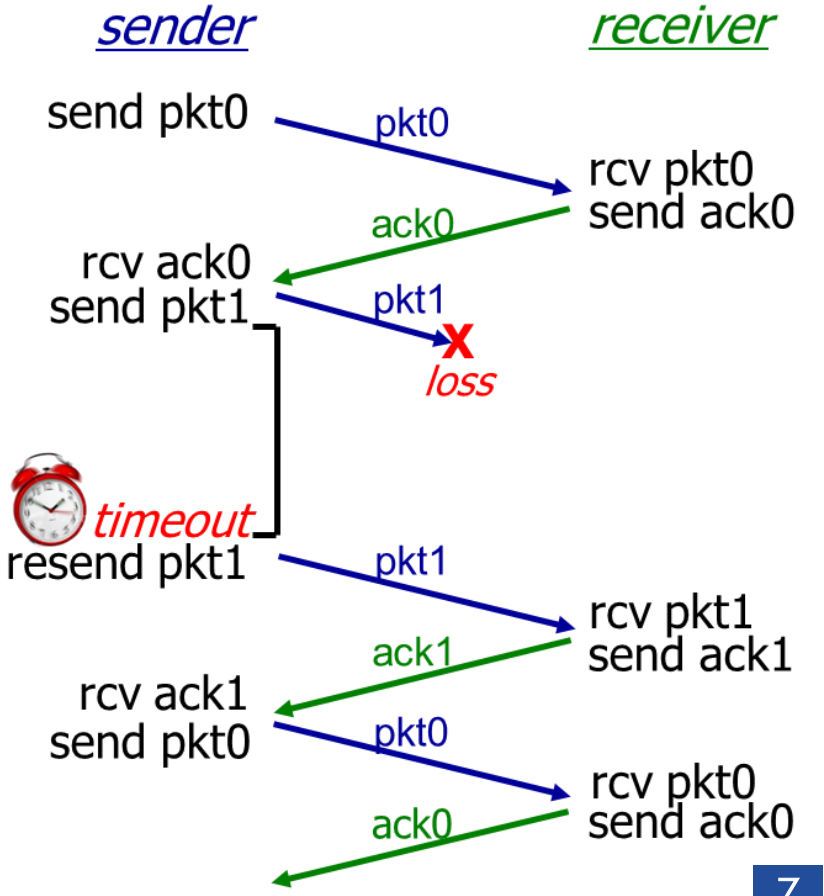
```
(20241234) PS D:\library\src> python .\udp_chat_server.py
<-  Hello
-> Hi, IoT
<-  This is my first chatting program.
-> Me, too. Good luck.
```

```
(20241234) PS D:\library\src> python .\udp_chat_client.py
-> Hello
<-  Hi, IoT
-> This is my first chatting program.
<-  Me, too. Good luck.
->
```

전송 중 데이터 손실 고려하기

- 전송 중, 데이터 손실이 발생할 수 있음
  - TCP를 이용하는 경우: TCP가 데이터 손실을 복구해 줌
  - UDP를 이용하는 경우: 애플리케이션이 데이터 손실을 복구해야 함

- 손실 복구 메커니즘
  - Stop-and-Wait 방식
  - 수신 측은 데이터를 전송 후, 데이터를 정상적으로 수신하면 ack 응답을 전송하고, 데이터를 수신하지 못하거나 비정상이면 응답을 보내지 않음
  - 송신 측은 ack 응답이 없으면 데이터를 재전송
  - 타임아웃 시간을 2배 씩 증가시키고, 최대 응답 시간이 되면 재전송을 중단



UDP 손실 복구 프로그램 - 서버

- 전송 손실을 가정한 UDP 서버 프로그램
  - 30%의 데이터 손실이 발생한 것으로 가정하여 수신 데이터 중에서 70%에 대해서만 응답을 전송

udp_loss_server.py

```
from socket import *
import random

BUFF_SIZE = 1024
port = 5555

s_sock = socket(AF_INET, SOCK_DGRAM)
s_sock.bind(('', port))
print('Listening...')

while True:
    data, addr = s_sock.recvfrom(BUFF_SIZE)
    if random.randint(1, 10) <= 3:  # 30% 데이터 손실
        print('Packet from {} lost!'.format(addr))
        continue
    print('Packet is {} from {}'.format(data.decode(), addr))

    s_sock.sendto('ack'.encode(), addr)
```

UDP 손실 복구 프로그램 - 클라이언트

- 손실 복구 기능을 가진 UDP 클라이언트 프로그램
  - 타임아웃이 발생하면, 재전송하고 타임아웃 시간을 2배 씩 증가
    - ✓ 타임아웃 시간이 2.0보다 커지면 재전송 중단
  - socket의 settimeout()함수 이용
    - ✓ 블로킹 소켓 연산에 시간제한을 설정

udp_loss_client.py

```
from socket import *

BUFF_SIZE = 1024
port = 5555

c_sock = socket(AF_INET, SOCK_DGRAM)
c_sock.connect(('localhost', port))

for i in range(10):      # 10번 전송
    time = 0.1           # 0.1초
    data = 'Hello, IoT'
    while True:
        c_sock.send(data.encode())
        print('Packet({}): Waiting up to {} secs for ack'.format(i, time))
        c_sock.settimeout(time)
        try:
            data = c_sock.recv(BUFF_SIZE)
        except timeout:
            time *= 2         # 대기시간 2배 증가
            if time > 2.0:    # 최대 대기시간 초과
                break
        else:
            print('Response', data.decode())
            break
```

UDP 손실 복구 프로그램

```
(20241234) PS D:\library\src> python .\udp_loss_server.py
Listening...
Packet is Hello, IoT from ('127.0.0.1', 56926)
Packet from ('127.0.0.1', 56926) lost!
Packet is Hello, IoT from ('127.0.0.1', 56926)
Packet is Hello, IoT from ('127.0.0.1', 56926)
Packet from ('127.0.0.1', 56926) lost!
Packet is Hello, IoT from ('127.0.0.1', 56926)
Packet is Hello, IoT from ('127.0.0.1', 56926)
Packet is Hello, IoT from ('127.0.0.1', 56926)
Packet is Hello, IoT from ('127.0.0.1', 56926)
Packet is Hello, IoT from ('127.0.0.1', 56926)
Packet is Hello, IoT from ('127.0.0.1', 56926)
Packet from ('127.0.0.1', 56926) lost!
Packet is Hello, IoT from ('127.0.0.1', 56926)
```

```
(20241234) PS D:\library\src> python .\udp_loss_client.py
Packet(0): Waiting up to 0.1 secs for ack
Response ack
Packet(1): Waiting up to 0.1 secs for ack
Packet(1): Waiting up to 0.2 secs for ack
Response ack
Packet(2): Waiting up to 0.1 secs for ack
Response ack
Packet(3): Waiting up to 0.1 secs for ack
Packet(3): Waiting up to 0.2 secs for ack
Response ack
Packet(4): Waiting up to 0.1 secs for ack
Response ack
Packet(5): Waiting up to 0.1 secs for ack
Response ack
Packet(6): Waiting up to 0.1 secs for ack
Response ack
Packet(7): Waiting up to 0.1 secs for ack
Response ack
Packet(8): Waiting up to 0.1 secs for ack
Response ack
Packet(9): Waiting up to 0.1 secs for ack
Packet(9): Waiting up to 0.2 secs for ack
Response ack
(20241234) PS D:\library\src>
```

과제-UDP1: UDP message 송수신 프로그램

UDP message 송수신 프로그램

- 서버
  - ✓ 클라이언트로부터 "send [mboxID] message" 메시지 수신 시, 해당 [mboxID]에 message를 저장하고, "OK"를 클라이언트로 전송
    - [mboxID]는 숫자, 문자 모두 가능
    - message는 여러 단어로 구성될 수 있음
  - ✓ 클라이언트로부터 "receive [mboxID]" 메시지 수신 시, 해당 [mboxID]의 제일 앞에 있는 메시지를 클라이언트로 전송한 후 삭제
    - 존재하지 않는 [mboxID]나 메시지가 없는 [mboxID]에 대해 "receive [mboxID]"를 수신하는 경우 "No messages"를 클라이언트로 전송
  - ✓ 클라이언트로부터 "quit" 메시지 수신 시, 프로그램 종료

- 클라이언트
  - ✓ 사용자로부터 "send [mboxID] message" 또는 "receive [mboxID]" 또는 "quit" 문자열을 입력받음
  - ✓ 입력받은 문자열을 서버로 전송
    - "quit" 문자열 전송 후, 프로그램 종료
  - ✓ 응답으로 받은 문자열을 화면에 출력

과제-UDP1: UDP message 송수신 프로그램

- 클라이언트 예제

```
> python udp_message_client.py
Enter the message("send mboxId message" or "receive mboxId"):send 1 Hello, IoT
OK
Enter the message("send mboxId message" or "receive mboxId"):send 2 How are you?
OK
Enter the message("send mboxId message" or "receive mboxId"):send 1 This is UDP.
OK
Enter the message("send mboxId message" or "receive mboxId"):send 3 See you later!
OK
Enter the message("send mboxId message" or "receive mboxId"):receive 1
Hello, IoT
Enter the message("send mboxId message" or "receive mboxId"):receive 1
This is UDP.
Enter the message("send mboxId message" or "receive mboxId"):receive 1
No messages
Enter the message("send mboxId message" or "receive mboxId"):receive 2
How are you?
Enter the message("send mboxId message" or "receive mboxId"):receive 2
No messages
Enter the message("send mboxId message" or "receive mboxId"):receive 3
See you later!
Enter the message("send mboxId message" or "receive mboxId"):receive 3
No messages
Enter the message("send mboxId message" or "receive mboxId"):receive 4
No messages
Enter the message("send mboxId message" or "receive mboxId"):quit
```

과제-UDP1: UDP message 송수신 프로그램

■ 과제-UDP1

- 서버, 클라이언트 각각 1개의 소스 코드(.py)로 저장
- 실행화면 캡처 파일 1개 (클라이언트)

■ 소스 코드

- GitHub에 hw-udp1 폴더 생성 후, 소스 코드 업로드
- GitHub 화면 캡처
  - ✓ 폴더 이름과 파일 이름이 보이도록 캡처할 것

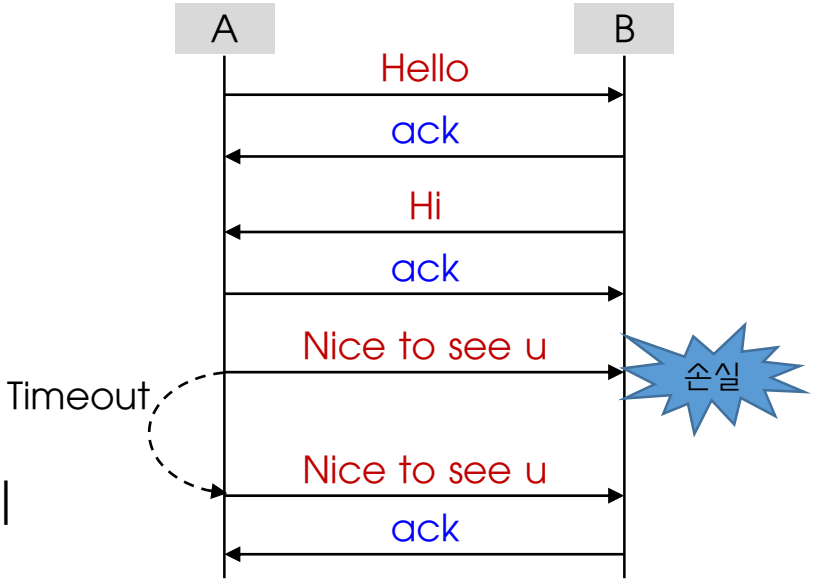
■ 제출

- 과제 공지 후 1주일, eClass 제출
- 제출물
  - ✓ 실행화면 캡처 파일 (1개)
  - ✓ GitHub 화면 캡처 파일 (1개)

과제-UDP2: 데이터 손실을 고려한 채팅 프로그램

■ UDP 채팅 프로그램 기반 (슬라이드 5, 6)

- 고려사항
  - ✓ 채팅 메시지가 전송 도중에 손실될 수 있음
  - ✓ 채팅 메시지가 잘 전송되었는지 알기 위해, 수신 측에서는 'ack' 메시지를 전송



- 데이터 손실 처리
  - ✓ 송신 측
    - 타임아웃: 2초 (2초 동안 'ack'을 받지 못하면, 채팅 메시지가 손실된 것으로 판단, 재전송)
    - 최대 5회 재전송 (최초 메시지 포함 4번 전송)
    - 채팅 메시지 전송 시, 재전송횟수(0, 1, 2, 3, 4, 5)를 포함하여 상대방으로 전송
  - ✓ 수신 측
    - 50%의 확률로 'ack'을 보내지 않음 (고의로 50%의 데이터 손실을 만듬)
    - 'ack' 전송 후, 채팅 메시지를 화면에 출력

과제-UDP2: 데이터 손실을 고려한 채팅 프로그램

■ 서버 프로그램 (메시지 수신 처리 부분)

- 클라이언트로부터 메시지 수신 후 50% 확률로 응답 하지 않음 (메시지 손실)
- 손실되지 않은 경우, 'ack' 응답을 보내고, 화면에 채팅 메시지 출력

udp_loss_chat_server.py

```
from socket import *
import random
import time

port = 3333
BUFF_SIZE = 1024

sock = socket(AF_INET, SOCK_DGRAM)
sock.bind(('', port))

while True:
    sock.settimeout(None)          # 소켓의 블로킹 모드 timeout 설정
    while True:                    # None인 경우, 무한정 블로킹됨
        data, addr = sock.recvfrom(BUFF_SIZE)
        if random.random() <= 0.5:
            continue
        else:
            sock.sendto(b'ack', addr)
            print('<-', data.decode())
            break
```

과제-UDP2: 데이터 손실을 고려한 채팅 프로그램

■ 서버 프로그램 (메시지 송신 처리 부분)

- 사용자로부터 채팅 메시지를 입력 받음
- 재전송 횟수가 5번 이하인 경우, 메시지 전송
- 타임아웃 시간을 2초로 설정하고, 'ack' 응답을 기다림
- 타임아웃이 발생할 경우, 재전송 횟수를 1 증가시키고, 재전송

```
msg = input('-> ')
reTx = 0
while reTx <= 5:
    resp = str(reTx) + ' ' + msg
    sock.sendto(resp.encode(), addr)
    sock.settimeout(2)          # 소켓의 timeout 설정. 해당 timeout 내 메시지
                                # 수신을 못하면 timeout 예외 발생
    try:
        data, addr = sock.recvfrom(BUFF_SIZE)
    except timeout:
        reTx += 1
        continue
    else:
        break
```

과제-UDP2: 데이터 손실을 고려한 채팅 프로그램

■ 실행화면

```
> python udp_loss_chat_server.py
<- 0 Hello
-> Hi
<- 2 How are you?
-> Fine. Thank you. And you?
<- 0 Me, too.
-> Who are you?
<- 0 I am your father.
-> I am sorry.
<- 0 This is your homework.
-> What's the deadline?
<- 1 Within 1 week.
-> OK.
```

```
> python udp_loss_chat_client.py
-> Hello
<- 0 Hi
-> How are you?
<- 0 Fine. Thank you. And you?
-> Me, too.
<- 3 Who are you?
-> I am your father.
<- 1 I am sorry.
-> This is your homework.
<- 0 What's the deadline?
-> Within 1 week.
<- 2 OK.
->
```

과제-UDP2: 데이터 손실을 고려한 채팅 프로그램

■ 과제-UDP2

- 서버, 클라이언트 각각 1개의 소스 코드(.py)로 저장
- 실행화면 캡처 파일 (1개 또는 2개)

■ 소스 코드

- GitHub에 hw-udp2 폴더 생성 후, 소스 코드 업로드
- GitHub 화면 캡처
  - ✓ 폴더 이름과 파일 이름이 보이도록 캡처할 것

■ 제출

- 과제 공지 후 2주일, eClass 제출
- 제출물
  - ✓ 실행화면 캡처 파일(1개 또는 2개)
  - ✓ GitHub 화면 캡처 파일 (1개)

서버-클라이언트 통합 UDP 프로그램

■ 서버-클라이언트 통합 UDP 프로그램

- 실행 인수에 따라 서버와 클라이언트 기능을 모두 수행할 수 있는 프로그램
- 프로그램을 실행할 때 'c' 또는 's' 인수를 지정하면 각각 Client 또는 Server 함수가 실행됨
  - ✓ 예) 클라이언트 실행: `python udp_total.py c`, 서버: `python udp_total.py s`

■ 기능

- 서버는 클라이언트로부터 받은 메시지의 바이트 수를 계산하여 응답
  - ✓ 서버는 일정 비율(`prob`)에 따라 메시지에 대한 응답을 보내지 않을 수 있음
- 클라이언트는 현재 시각을 `count` 개수만큼 서버로 전송
  - ✓ 서버의 응답이 없을 경우 재전송
    - 재전송 타임아웃 시간(`time`)은 0.1로 시작하여 손실 발생 시 2배로 증가(최대 2.0)

■ 명령행 인자

- `-s:` 서버 주소
- `-p:` 포트 번호
- `-prob:` 서버가 응답을 보내지 않을 확률
- `-count:` 클라이언트에서 보내는 패킷 개수

서버-클라이언트 통합 UDP 프로그램

udp_total.py

```
import argparse, socket, random
from datetime import datetime

BUFF_SIZE = 1024

def Server(ipaddr, port):    # 서버 함수
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((ipaddr, port))
    print('Waiting in {}...'.format(sock.getsockname()))
    while True:
        data, addr = sock.recvfrom(BUFF_SIZE)
        if random.random() < prob:
            print('Message from {} is lost.'.format(addr))
            continue
        print('{} client message {!r}'.format(addr, data.decode()))
        text = 'The length is {} bytes.'.format(len(data))
        sock.sendto(text.encode(), addr)

def Client(hostname, port):  # 클라이언트 함수
    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

    index = 1   # 보낸 메시지 번호
    time = 0.1  # seconds
    while True:
        data = str(datetime.now())
        sock.sendto(data.encode(), (hostname, port))
        print('({}) Waiting for {} sec'.format(index, time))
        sock.settimeout(time)
```

서버-클라이언트 통합 UDP 프로그램

```
        try:
            data, addr = sock.recvfrom(BUFF_SIZE)
        except socket.timeout:
            time *= 2
            if time > 2.0:
                print('{}th packet is lost'.format(index))
                if index >= sending_counts:
                    break
                index += 1
                time = 0.1
        else:
            print('Server reply: {!r}'.format(data.decode()))
            if index >= sending_counts:
                break
            index += 1
            time = 0.1

if __name__ == '__main__':
    mode = {'c': Client, 's': Server}
    parser = argparse.ArgumentParser(description='Send and receive UDP packets with setting drop probabili
ty')
    parser.add_argument('role', choices=mode, help='which role to take between server and client')
    parser.add_argument('-s', default='localhost', help='server that client sends to')
    parser.add_argument('-p', type=int, default=2500, help='UDP port (default:2500)')
    parser.add_argument('-prob', type=float, default=0, help='dropping probability (0~1)')
    parser.add_argument('-count', type=int, default=10, help='number of sending packets')
    args = parser.parse_args()
    prob = args.prob
    sending_counts = args.count
    if args.role == 'c':
        mode[args.role](args.s, args.p)
    else:
        mode[args.role]('', args.p)
```

서버-클라이언트 통합 UDP 프로그램

- 옵션 확인하기
  - python udp_total.py -h

```
(20241234) PS D:\library\src> python udp_total.py -h
usage: udp_total.py [-h] [-s S] [-p P] [-prob PROB] [-count COUNT] {c,s}

Send and receive UDP packets with setting drop probability

positional arguments:
  {c,s}          which role to take between server and client

options:
  -h, --help     show this help message and exit
  -s S           server that client sends to
  -p P           UDP port (default:2500)
  -prob PROB     dropping probability (0~1)
  -count COUNT   number of sending packets
```

서버-클라이언트 통합 UDP 프로그램

서버

```
python udp_total.py s -prob 0.5
```

```
(20241234) PS D:\library\src> python udp_total.py s -prob 0.5
Waiting in ('0.0.0.0', 2500)...
Message from ('127.0.0.1', 59694) is lost.
Message from ('127.0.0.1', 59694) is lost.
('127.0.0.1', 59694) client message '2024-05-04 11:18:15.888743'
('127.0.0.1', 59694) client message '2024-05-04 11:18:15.891700'
('127.0.0.1', 59694) client message '2024-05-04 11:18:15.893685'
Message from ('127.0.0.1', 59694) is lost.
('127.0.0.1', 59694) client message '2024-05-04 11:18:16.011311'
Message from ('127.0.0.1', 59694) is lost.
Message from ('127.0.0.1', 59694) is lost.
Message from ('127.0.0.1', 59694) is lost.
('127.0.0.1', 59694) client message '2024-05-04 11:18:16.731100'
```

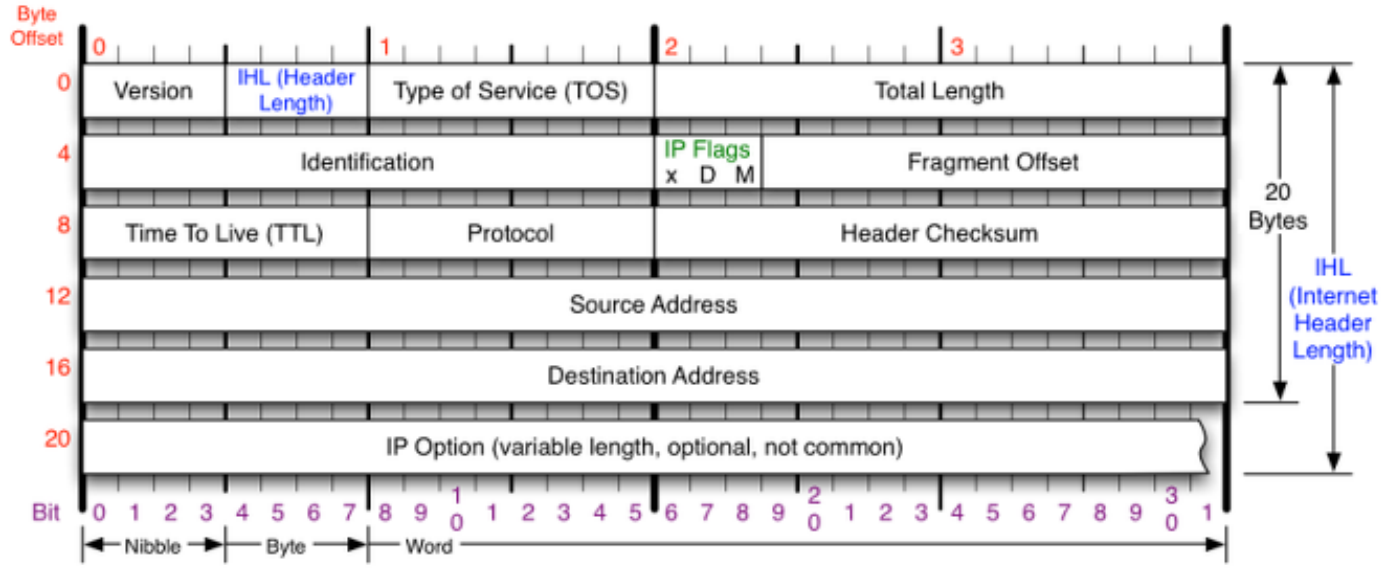
클라이언트

```
python udp_total.py c -count 5
```

```
(20241234) PS D:\library\src> python udp_total.py c -count
(1) Waiting for 0.1 sec
(1) Waiting for 0.2 sec
(1) Waiting for 0.4 sec
Server reply: 'The length is 26 bytes.'
(2) Waiting for 0.1 sec
Server reply: 'The length is 26 bytes.'
(3) Waiting for 0.1 sec
Server reply: 'The length is 26 bytes.'
(4) Waiting for 0.1 sec
(4) Waiting for 0.2 sec
Server reply: 'The length is 26 bytes.'
(5) Waiting for 0.1 sec
(5) Waiting for 0.2 sec
(5) Waiting for 0.4 sec
(5) Waiting for 0.8 sec
Server reply: 'The length is 26 bytes.'
```

IPv4 헤더 표현하기

- IPv4 헤더
  - 옵션 제외 시, 20바이트



Version

Version of IP Protocol. 4 and 6 are valid. This diagram represents version 4 structure only.

Header Length

Number of 32-bit words in TCP header, minimum value of 5. Multiply by 4 to get byte count.

Protocol

IP Protocol ID. Including (but not limited to):

1 ICMP	17 UDP	57 SKIP
2 IGMP	47 GRE	88 EIGRP
6 TCP	50 ESP	89 OSPF
9 IGRP	51 AH	115 L2TP

Total Length

Total length of IP datagram, or IP fragment if fragmented. Measured in Bytes.

Fragment Offset

Fragment offset from start of IP datagram. Measured in 8 byte (2 words, 64 bits) increments. If IP datagram is fragmented, fragment size (Total Length) must be a multiple of 8 bytes.

Header Checksum

Checksum of entire IP header

IP Flags

x D M

x 0x80 reserved (evil bit)
D 0x40 Do Not Fragment
M 0x20 More Fragments follow

RFC 791

Please refer to RFC 791 for the complete Internet Protocol (IP) Specification.

IPv4 헤더 표현하기

- 리눅스 커널 내 IPv4 헤더
  - C 구조체로 표현
  - 파이썬에서는?

```
struct iphdr {
        __u8    version:4,
                ihl:4;
        __u8    tos;
        __be16  tot_len;
        __be16  id;
        __be16  frag_off;
        __u8    ttl;
        __u8    protocol;
        __sum16 check;
        __be32  saddr;
        __be32  daddr;
        /*The options start here.  */
};
```

IPv4 헤더 표현하기

- 파이썬
  - 클래스 사용

```
(20241234) PS D:\library\src> python iphdr.py
Traceback (most recent call last):
  File "D:\library\src\iphdr.py", line 16, in <module>
    sock.sendto(bytes(test), ('localhost', 2500))
                ^^^^^^^^^^^^
TypeError: cannot convert 'Iphdr' object to bytes
```

iphdr.py

```python
import socket

class Iphdr:
    def __init__(self, tot_len, protocol, saddr, daddr):
        self.ver_len = 0x45
        self.tos = 0
        self.tot_len = tot_len
        self.id = 0
        self.frag_off = 0
        self.ttl = 127
        self.protocol = protocol   #TCP(6)/UDP(17)
        self.check = 0
        self.saddr = socket.inet_aton(saddr)
        self.daddr = socket.inet_aton(daddr)

test = Iphdr(1000, 6, '10.0.0.1', '11.0.0.1')
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.sendto(bytes(test), ('localhost', 2500))
```

struct 모듈

■ struct 모듈

- 이진 데이터를 bytes로 만들거나(pack), bytes를 이진 데이터로 해석(unpack)하기 위한 모듈

- 예) IP header의 tot_len 필드의 값이 10이라고 가정할 때, 이것을 bytes로 변환한 값은 얼마인가?
  - ✓ tot_len: `b'\x0a'`, `b'\x00\x0a'`, `b'\x0a\x00'`, `b'\x00\x00\x00\x0a'`, `b'\x0a\x00\x00\x00'`?

- How? `struct.pack('!h', 10)`

```
>>> binascii.b2a_hex(struct.pack('!h', 10))
b'000a'
```

- `struct.pack(format, v1, v2, ...)`
  - ✓ v1, v2, ...값을 포함하고 포맷 문자열 format에 따라 패킹된 bytes 객체를 반환
  - ✓ 인자는 format이 요구하는 값과 정확히 일치해야 함
- `struct.unpack(format, buffer)`
  - ✓ 포맷 문자열 format에 따라 버퍼 buffer를 언패킹
  - ✓ 결과는 튜플이며, 버퍼의 크기는 format이 요구하는 크기와 정확히 일치해야 함

struct 모듈

■ 바이트 순서, 크기 및 정렬

- 다음 표에 따라, 포맷 문자열의 첫 번째 문자를 사용하여 패킹된 데이터의 바이트 순서, 크기 및 정렬을 표시

문자	바이트 순서	크기	정렬
@	네이티브	네이티브	네이티브
=	네이티브	표준	none
<	리틀 엔디언	표준	none
>	빅 엔디언	표준	none
!	네트워크	표준	none

- 첫 번째 문자가 이들 중 하나가 아니면 '@'를 가정
  - ✓ C 컴파일러의 `sizeof` 표현식을 사용하여 결정
- 네이티브 바이트 순서는 호스트 시스템에 따라 달라짐
  - ✓ 빅 엔디언 또는 리틀 엔디언
  - ✓ `sys.byteorder`로 확인 가능
- 표준 크기를 사용하면, 포맷 문자에 따라 원하는 크기를 사용할 수 있음

struct 모듈

■ 포맷 문자

- 포맷 문자 앞에 정수 반복 횟수를 적을 수 있음. 예) '4h' == 'hhhh'
- 포맷 사이의 공백 문자는 무시됨

포맷	C 타입	Python 타입	표준 크기
c	char	길이가 1인 bytes	1
b	signed char	정수	1
B	unsigned char	정수	1
?	_Bool	bool	1
h	short	정수	2
H	unsigned short	정수	2
i	int	정수	4
I	unsigned int	정수	4
l	long	정수	4
L	unsigned long	정수	4
q	long long	정수	8
Q	unsigned long long	정수	8
f	float	float	4
d	double	float	8
s	char[]	bytes	가변

struct 모듈

■ 예제

```
>>> import struct
>>> struct.pack('hhI', 1, 2, 3)  # 네이티브
b'\x01\x00\x02\x00\x03\x00\x00\x00'
>>> struct.unpack('hhI', b'\x01\x00\x02\x00\x03\x00\x00\x00')
(1, 2, 3)

>>> struct.pack('!hhI', 1, 2, 3) # 네트워크
b'\x00\x01\x00\x02\x00\x00\x00\x03'
>>> struct.unpack('!hhI', b'\x00\x01\x00\x02\x00\x00\x00\x03')
(1, 2, 3)

>>> struct.pack('<hhI', 1, 2, 3) # 리틀엔디언
b'\x01\x00\x02\x00\x03\x00\x00\x00'
>>> struct.pack('>hhI', 1, 2, 3) # 빅엔디언
b'\x00\x01\x00\x02\x00\x00\x00\x03'
>>> struct.calcsize('hhI')
8
```

struct 모듈

```
(20241234) PS D:\library\src> python iphdr_1.py
b'450003e8000000007f0600000a0000010b000001'
```

- IP header packing

iphdr_1.py

```
import socket
import struct
import binascii

class Iphdr:
    def __init__(self, tot_len, protocol, saddr, daddr):
        self.ver_len = 0x45
        self.tos = 0
        self.tot_len = tot_len
        self.id = 0
        self.frag_off = 0
        self.ttl = 127
        self.protocol = protocol   #TCP(6)/UDP(17)
        self.check = 0
        self.saddr = socket.inet_aton(saddr)
        self.daddr = socket.inet_aton(daddr)

    def pack_Iphdr(self):
        packed = b''
        packed += struct.pack('!BBH', self.ver_len, self.tos, self.tot_len)
        packed += struct.pack('!HH', self.id, self.frag_off)
        packed += struct.pack('!BBH', self.ttl, self.protocol, self.check)
        packed += struct.pack('!4s', self.saddr)
        packed += struct.pack('!4s', self.daddr)
        return packed

ip = Iphdr(1000, 6, '10.0.0.1', '11.0.0.1')
packed_iphdr = ip.pack_Iphdr()
print(binascii.b2a_hex(packed_iphdr))
```

```
450003e8
00000000
7f060000
0a000001
0b000001
```

struct 모듈

```
(20241234) PS D:\library\src> python iphdr_2.py
b'450003e8000000007f0600000a0000010b000001'
(69, 0, 1000, 0, 0, 127, 6, 0, b'\n\x00\x00\x01', b'\x0b\x00\x00\x01')
Packet size:1000 Protocol:6 IP:('10.0.0.1', '11.0.0.1')
```

- IP header unpacking

iphdr_2.py

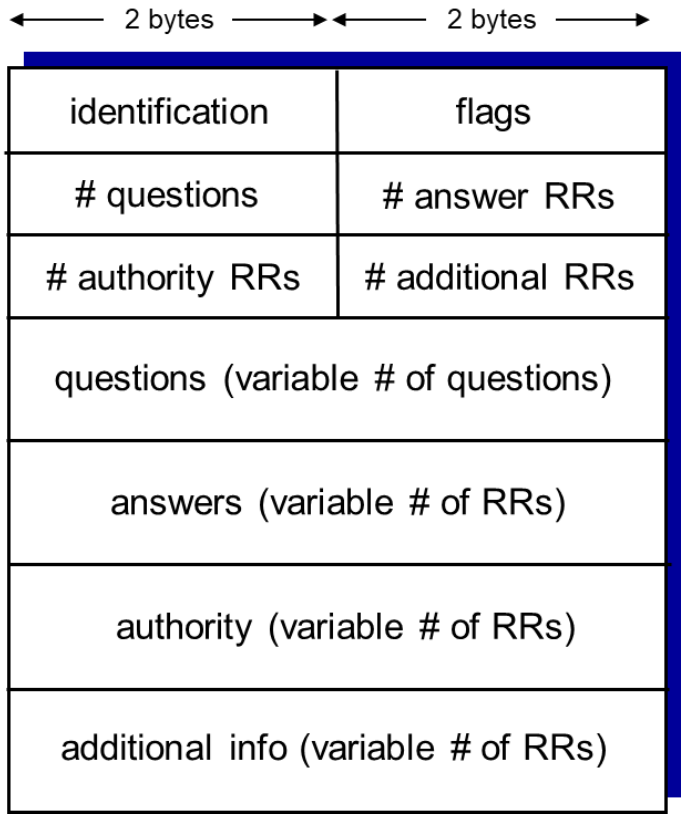
```
# 이전 슬라이드의 import문 및 Class Iphdr 필요

def unpack_Iphdr(buffer):
    unpacked = struct.unpack('!BBHHHBBH4s4s', buffer[:20])
    return unpacked

def getPacketSize(unpacked_ipheader):
    return unpacked_ipheader[2]

def getProtocolId(unpacked_ipheader):
    return unpacked_ipheader[6]

def getIP(unpacked_ipheader):
    src_ip = socket.inet_ntoa(unpacked_ipheader[8])
    dst_ip = socket.inet_ntoa(unpacked_ipheader[9])
    return (src_ip, dst_ip)


ip = Iphdr(1000, 6, '10.0.0.1', '11.0.0.1')
packed_iphdr = ip.pack_Iphdr()
print(binascii.b2a_hex(packed_iphdr))

unpacked_iphdr = unpack_Iphdr(packed_iphdr)
print(unpacked_iphdr)
print('Packet size:{} Protocol:{} IP:{}' \
    .format(getPacketSize(unpacked_iphdr), getProtocolId(unpacked_iphdr), getIP(unpacked_iphdr)))
```

```
450003e8
00000000
7f060000
0a000001
0b000001
```

DNS 클라이언트 만들기

- DNS
  - 도메인 네임을 IP 주소로 변환해주는 애플리케이션 계층 프로토콜
  - UDP, 53번 포트 사용

- DNS Query




```
Frame 2937: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
Ethernet II, Src: Giga-Byt_42:80:55 (e0:d5:5e:42:80:55), Dst: Alcatel-_82:f7:01 (2c:fa:a2:82:f7:01)
Internet Protocol Version 4, Src: 114.71.220.95, Dst: 220.69.193.130
User Datagram Protocol, Src Port: 49273, Dst Port: 53
Domain Name System (query)
    Transaction ID: 0xbe63
    Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
    Queries
        www.google.co.kr: type A, class IN
            Name: www.google.co.kr
            [Name Length: 16]
            [Label Count: 4]
            Type: A (Host Address) (1)
            Class: IN (0x0001)

0000  2c fa a2 82 f7 01 e0 d5  5e 42 80 55 08 00 45 00   ,······ ^B·U··E·
0010  00 3e 7c cd 00 00 80 11  00 00 72 47 dc 5f dc 45   ·>|····· ··rG·_·E
0020  c1 82 c0 79 00 35 00 2a  ec aa be 63 01 00 00 01   ···y·5·* ···c····
0030  00 00 00 00 00 00 03 77  77 77 06 67 6f 6f 67 6c   ·······w ww·googl
0040  65 02 63 6f 02 6b 72 00  00 01 00 01              e·co·kr· ····
```

DNS 클라이언트 만들기

- DNS Response

```
> Frame 876: 350 bytes on wire (2800 bits), 350 bytes captured (2800 bits) on interface 0
> Ethernet II, Src: Alcatel-_82:f7:01 (2c:fa:a2:82:f7:01), Dst: Giga-Byt_42:80:55 (e0:d5:5e:42:80:55)
> Internet Protocol Version 4, Src: 220.69.193.130, Dst: 114.71.220.95
> User Datagram Protocol, Src Port: 53, Dst Port: 62477
v Domain Name System (response)
    Transaction ID: 0x0001
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 4
    Additional RRs: 8
  v Queries
    v www.google.co.kr: type A, class IN
        Name: www.google.co.kr
        [Name Length: 16]
        [Label Count: 4]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  v Answers
    v www.google.co.kr: type A, class IN, addr 172.217.174.99
        Name: www.google.co.kr
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 279
        Data length: 4
        Address: 172.217.174.99
```

```
0000  e0 d5 5e 42 80 55 2c fa  a2 82 f7 01 08 00 45 00   ··^B·U,· ·····E·
0010  01 50 ce ff 00 00 3d 11  c1 2e dc 45 c1 82 72 47   ·P····=· ···E··rG
0020  dc 5f 00 35 f4 0d 01 3c  94 53 00 01 81 80 00 01   ·_·5···< ·S·····
0030  00 01 00 04 00 08 03 77  77 77 06 67 6f 6f 67 6c   ·······w ww·googl
0040  65 02 63 6f 02 6b 72 00  00 01 00 01 c0 0c 00 01   e·co·kr· ········
0050  00 01 00 00 01 17 00 04  ac d9 ae 63 c0 10 00 02   ········ ···c····
0060  00 01 00 00 bc 20 00 10  03 6e 73 32 06 67 6f 6f   ····· ·· ·ns2·goo
0070  67 6c 65 03 63 6f 6d 00  c0 10 00 02 00 01 00 00   gle·com· ········
0080  bc 20 00 06 03 6e 73 34  c0 42 c0 10 00 02 00 01   · ···ns4 ·B·····
0090  00 00 bc 20 00 06 03 6e  73 31 c0 42 c0 10 00 02   ··· ···n s1·B····
00a0  00 01 00 00 bc 20 00 06  03 6e 73 33 c0 42 c0 5a   ····· ·· ·ns3·B·Z
00b0  00 01 00 01 00 03 62 5b  00 04 d8 ef 26 0a c0 7e   ·····b[ ····&··~
```

DNS 클라이언트 만들기

```
(20241234) PS D:\library\src> python dns_client.py www.sch.ac.kr
www.sch.ac.kr 8.104.111.109
(20241234) PS D:\library\src> python dns_client.py homepage.sch.ac.kr
homepage.sch.ac.kr 220.69.189.98
(20241234) PS D:\library\src> python dns_client.py sw.sch.ac.kr
sw.sch.ac.kr 3.110.115.49
```

dns_client.py

```python
import socket
import struct
import sys

class DnsClient:
    def __init__(self, domainName):
        self.domainName = domainName

        # DNS Query Header
        self.TransactionId = 1
        self.Flag = 0x0100
        self.Questions = 1
        self.AnswerRRs = 0
        self.AuthorityRRs = 0
        self.AdditionalRRs = 0

    def response(self, packet):     # processing dns response
        dnsHeader = packet[:12]
        dnsData = packet[12:].split(b'\x00', 1)

        ansRR = packet[12+len(dnsData[0])+5:12+len(dnsData[0])+21]
        rr_unpack = struct.unpack('!2sHHIH4s', ansRR)
        ip_addr = socket.inet_ntoa(rr_unpack[5])
        print(self.domainName, ip_addr)
```

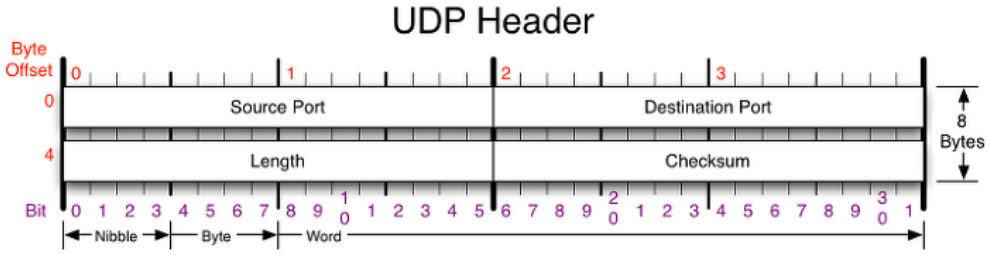
DNS 클라이언트 만들기

```
    def query(self):                    # create dns query
        # DNS header packing
        query = struct.pack('!HH', self.TransactionId, self.Flag)
        query += struct.pack('!HH', self.Questions, self.AnswerRRs)
        query += struct.pack('!HH', self.AuthorityRRs, self.AdditionalRRs)

        part = self.domainName.split('.')

        for i in range(len(part)):
            query = query + struct.pack('!B', len(part[i]))
            query = query + part[i].encode()

        query = query + b'\x00'

        query_type = 1  # Type: A
        query_class = 1 # Class: IN
        query = query + struct.pack('!HH', query_type, query_class)

        sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        addr = ('220.69.193.130', 53)  # 순천향대학교 DNS 서버 주소
        sock.sendto(query, addr)
        packet, address = sock.recvfrom(65535)
        self.response(packet)

if __name__ == '__main__':
    if len(sys.argv) > 1:
        client = DnsClient(sys.argv[1])
        client.query()
```

과제-UDP3: UDP 헤더 pack/unpack 해보기 슬라이드 31, 32 참조

■ UDP 헤더

- UDP 헤더는 아래 그림과 같이 2바이트 송신자 포트번호Source Port, 2바이트 수신자 포트번호Destination Port, 2바이트 UDP 패킷 길이Length, 2바이트 체크섬Checksum으로 구성됨 (총 8바이트)
- Udphdr 클래스 정의 및 pack_Udphdr() 함수 정의 (슬라이드 31 참조)
  - ✓ Udphdr 클래스를 이용해서 아래와 같이 UDP 헤더 생성
    - udp = Udphdr(5555, 80, 1000, 0xFFFF)
      - 송신자 포트번호: 5555, 수신자 포트번호: 80, 패킷 길이: 1000, 체크섬: 0xFFFF
- unpack 함수 정의 (슬라이드 32 참조)
  - ✓ unpack_Udphdr(): UDP 헤더를 unpack하는 함수
  - ✓ getSrcPort(), getDstPort(), getLength(), getChecksum() 함수 정의
    - unpack된 결과에서 해당하는 필드값을 가져오는 함수



UDP 헤더

```
> python udphdr.py
b'15b3005003e8ffff'
(5555, 80, 1000, 65535)
Source Port:5555 Destination Port:80 Length:1000 Checksum:65535
```

실행결과

과제-UDP3: UDP 헤더 pack/unpack 해보기

■ 과제-UDP3

- pack/unpack 수행하는 1개의 소스 코드(.py)로 저장
- 실행화면 캡처 파일 (1개)

■ 소스 코드

- GitHub에 hw-udp3 폴더 생성 후, 소스 코드 업로드
- GitHub 화면 캡처
  - ✓ 폴더 이름과 파일 이름이 보이도록 캡처할 것

■ 제출

- 과제 공지 후 1주일, eClass 제출
- 제출물
  - ✓ 실행화면 캡처 파일(1개)
  - ✓ GitHub 화면 캡처 파일 (1개)

Thank you

Questions?

Contact: dmk@sch.ac.kr