데이터통신

- 실습 13주차. Stop and Wait ARQ -

충남대학교 데이터네트워크 연구실

공지사항

- 현재 시험 설문 제출률 (26/45)
- 화요일 오후 3시 / 화요일 오후 7시
- 설문 금요일까지!
- 밀린 채점 다음주 예정
- 다음주 실습시간에 1시간 이론수업, 1시간 실습 수업 진행

Feedback

- 과제 제출률 75.5% (34/45)
- Wireshark 설정법 사이버캠퍼스에 어제 업로드 완료
- 이번주 과제 패킷캡쳐 내용 제외
- 금요일날 서버 정답코드 공유 예정

패킷캡쳐방법이 너무 어려웠습니다.

파이썬을 VM에서 돌리는 것에 익숙치 않았을 뿐더러, VM에 HOST PC의 파일을 복사하는 방법을 찾기 어려웠음.

wireshark 설정법 알려주셨으면 좋겠어요

클라이언트 코드는 실습 코드와 같아 어렵지 않았으나,
서버 코드가 예시 코드가 없어 VM에서 파일을 찾아 소켓으로 보내는 방법에 대한 길잡이가 없어서 아쉬웠음.

Our file transfer protocol using UDP: DCFT Protocol

DCFT2: Data communication file transfer protocol version 2

- Protocol: 서버 - 클라이언트간의 약속
- UDP 기반 파일 전송 약속

파일 정보 요청

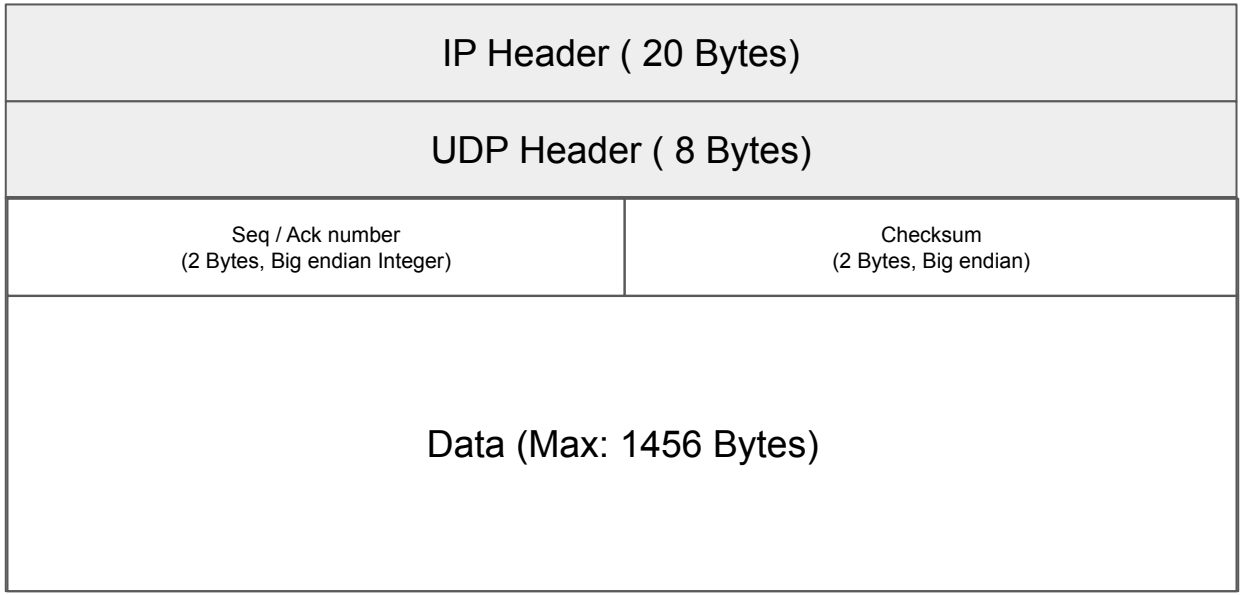
- ⇒ INFO FILENAME
- ⇐ UTF-8 Encoded Text (File size in bytes)
    - FILENAME이 없을 경우 UTF-8 Encoded '404 Not Found'

파일 다운로드 요청

- ⇒ DOWNLOAD FILENAME
- ⇐ File Binary

DCFT2: Data communication file transfer protocol version 2

- UDP Packet 위에 첫 2 Bytes 공간을 Seq / Ack number 로 사용
- 이후 2 Bytes 는 Checksum 으로 활용



실습1) Echo Character Seq/Ack Server / Client 실행

- Seq/Ack Short (2Bytes) 만 구현
  - 흐름 제어를 위한 코드만 구현
- 흐름 제어 자체가 구현되지는 않음!
- 네트워크 환경에 따라서 패킷이 뒤바뀌어 올 수 있음

```
Data: Data Communications 2024
Received D from ('34.168.194.7', 3034) with 0
Received a from ('34.168.194.7', 3034) with 1
Received t from ('34.168.194.7', 3034) with 0
Received i from ('34.168.194.7', 3034) with 1
Received a from ('34.168.194.7', 3034) with 1
Received o from ('34.168.194.7', 3034) with 0
Received   from ('34.168.194.7', 3034) with 1
Received n from ('34.168.194.7', 3034) with 1
Received s from ('34.168.194.7', 3034) with 0
Received 0 from ('34.168.194.7', 3034) with 1
Received 2 from ('34.168.194.7', 3034) with 0
Received 2 from ('34.168.194.7', 3034) with 0
Received 4 from ('34.168.194.7', 3034) with 1
Received C from ('34.168.194.7', 3034) with 1
Received o from ('34.168.194.7', 3034) with 0
Received   from ('34.168.194.7', 3034) with 0
Received u from ('34.168.194.7', 3034) with 1
Received m from ('34.168.194.7', 3034) with 1
Received m from ('34.168.194.7', 3034) with 0
Received n from ('34.168.194.7', 3034) with 0
Received i from ('34.168.194.7', 3034) with 1
Received c from ('34.168.194.7', 3034) with 0
Received a from ('34.168.194.7', 3034) with 1
Received t from ('34.168.194.7', 3034) with 0
```

실습2) Echo Character Seq/Ack Server 실행

- 실습1 Client와 통신
- Seq/Ack Short (2Bytes) 만 구현
  - 흐름 제어를 위한 코드만 구현
- 흐름 제어 자체가 구현되지는 않음!
- 네트워크 환경에 따라서 클라이언트에 패킷이 뒤바뀌어 갈 수 있음
  - localhost (127.0.0.1) 에서는 대부분 안 바뀜

```
Received Data Communications 2024 from ('127.0.0.1', 39038)
Sending D... 0
Sending a... 1
Sending t... 0
Sending a... 1
Sending  ... 0
Sending C... 1
Sending o... 0
Sending m... 1
Sending m... 0
Sending u... 1
Sending n... 0
Sending i... 1
Sending c... 0
Sending a... 1
Sending t... 0
Sending i... 1
Sending o... 0
Sending n... 1
Sending s... 0
Sending  ... 1
Sending 2... 0
Sending 0... 1
Sending 2... 0
Sending 4... 1
```

실습1) UDP Echo Character SeqAck Client

```
import socket
import struct

FLAGS = _ = None
DEBUG = False


def main():
    if DEBUG:
        print(f'Parsed arguments {FLAGS}')
        print(f'Unparsed arguments {_}')

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    print(f'Ready to send using {sock}')

    while True:
        try:
            data = input('Data: ').strip()
            request = f'{data}'
            size = len(request)
            sock.sendto(request.encode('utf-8'), (FLAGS.address, FLAGS.port))
            for _ in range(size):
                chunk, server = sock.recvfrom(FLAGS.chunk_maxsize)
                seq = chunk[:2]
                seq = struct.unpack('>H', seq)[0]
                data = chunk[2:]
                data = data.decode('utf-8')
                print(f'Received {data} from {server} with {seq}')
        except KeyboardInterrupt:
            print(f'Shutting down... {sock}')
            break
```

```
if __name__ == '__main__':
    import argparse

    parser = argparse.ArgumentParser()
    parser.add_argument('--debug', action='store_true',
                        help='The present debug message')
    parser.add_argument('--address', type=str, default='localhost',
                        help='The address to send data')
    parser.add_argument('--port', type=int, default=3034,
                        help='The port to send data')
    parser.add_argument('--chunk_maxsize', type=int, default=2**16,
                        help='The recvfrom chunk max size')

    FLAGS, _ = parser.parse_known_args()
    DEBUG = FLAGS.debug

    main()
```

실습2) UDP Echo Character SeqAck Server

```
import socket
import struct

FLAGS = _ = None
DEBUG = False
MAXSEQ = 2


def main():
    if DEBUG:
        print(f'Parsed arguments {FLAGS}')
        print(f'Unparsed arguments {_}')

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.bind((FLAGS.address, FLAGS.port))
    print(f'Listening on {sock}')

    while True:
        try:
            data, client = sock.recvfrom(FLAGS.mtu)
            data = data.decode('utf-8')
            print(f'Received {data} from {client}')

            data = data
            seq = 0
            sock.settimeout(FLAGS.timeout)
            for d in data:
                chunk = struct.pack('>H', seq) + d.encode('utf-8')
                sock.sendto(chunk, client)
                print(f'Sending {d}... {seq}')
                seq = (seq+1)%MAXSEQ
            print(f'Send {data} to {client}')
            sock.settimeout(None)
        except KeyboardInterrupt:
            print(f'Shutting down... {sock}')
            break
```

```
if __name__ == '__main__':
    import argparse

    parser = argparse.ArgumentParser()
    parser.add_argument('--debug', action='store_true',
                        help='The present debug message')
    parser.add_argument('--address', type=str, default='0.0.0.0',
                        help='The address to serve service')
    parser.add_argument('--port', type=int, default=3034,
                        help='The port to serve service')
    parser.add_argument('--mtu', type=int, default=1500,
                        help='The maximum transmission unit')
    parser.add_argument('--timeout', type=int, default=3,
                        help='The timeout seconds')

    FLAGS, _ = parser.parse_known_args()
    DEBUG = FLAGS.debug

    main()
```

Stop and Wait ARQ 구현

Stop and Wait ARQ

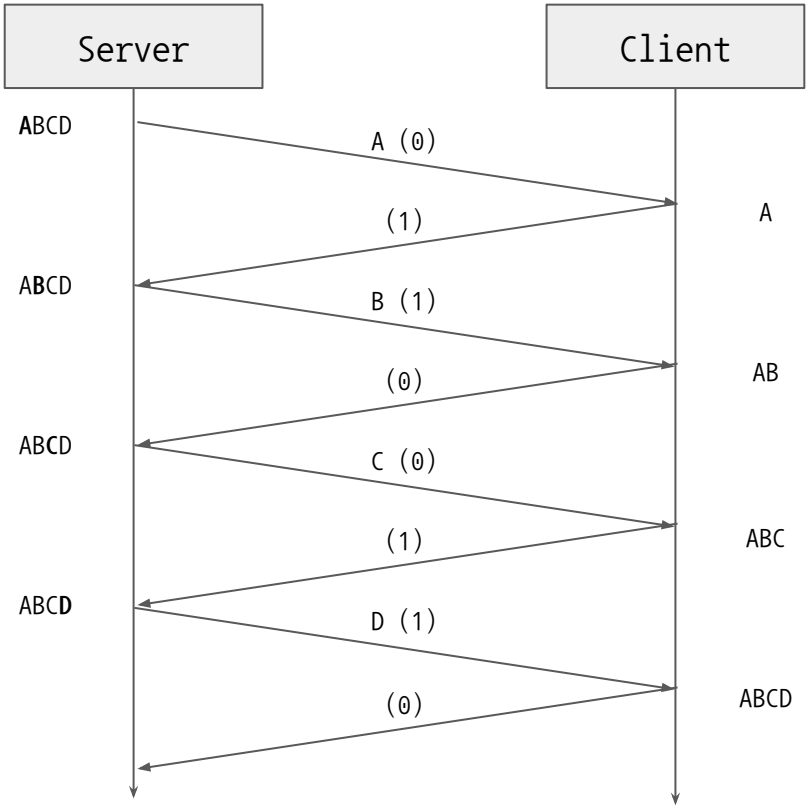
- 0-1 로 다음 데이터를 요청함

0-1-0-1 순서

Seq 보내는 데이터(패킷)의 고유 번호

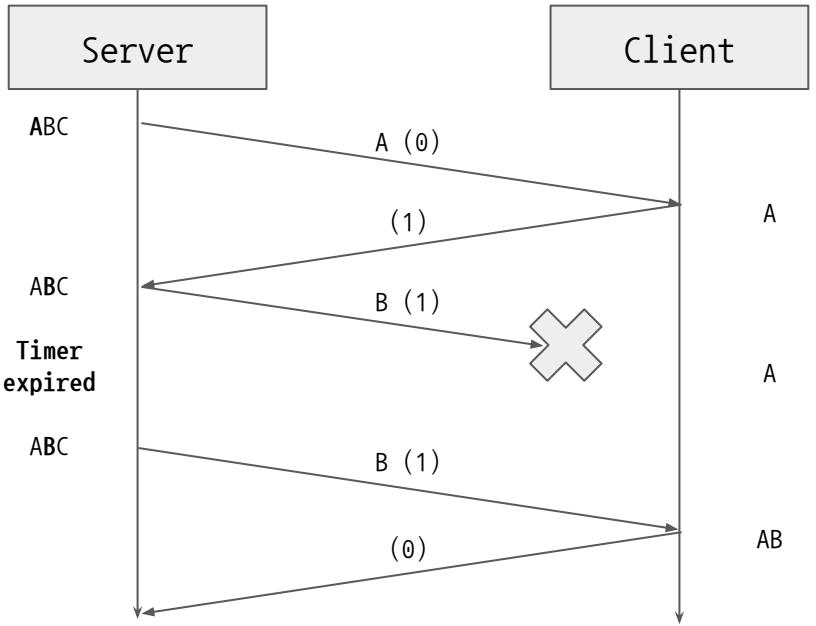
Ack : 직전 전송을 잘받았음을 표시 + 다음에 받아야할 Seq 번호

- ABCD Text Example!



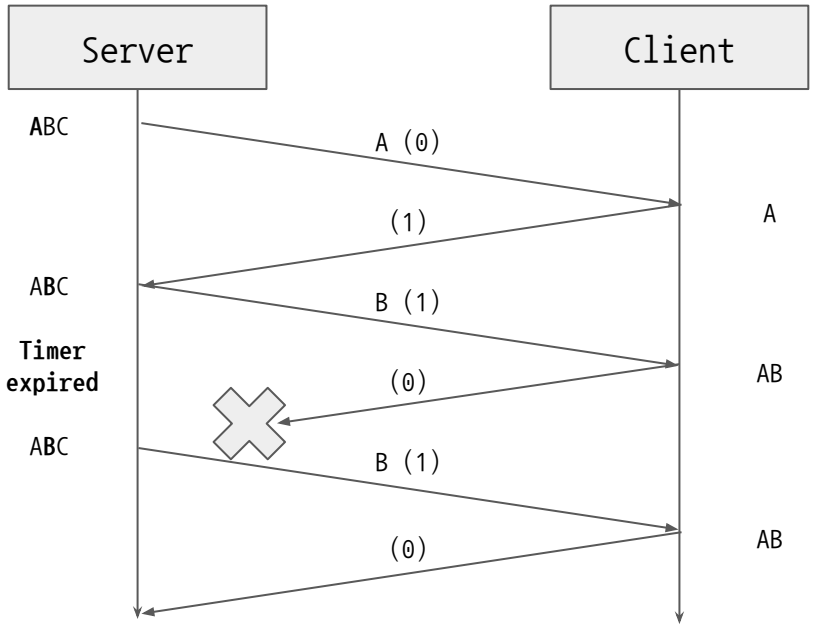
Stop and Wait ARQ: 에러 제어

- Case 1) Server → Client
  - Timer expired 되면 재전송



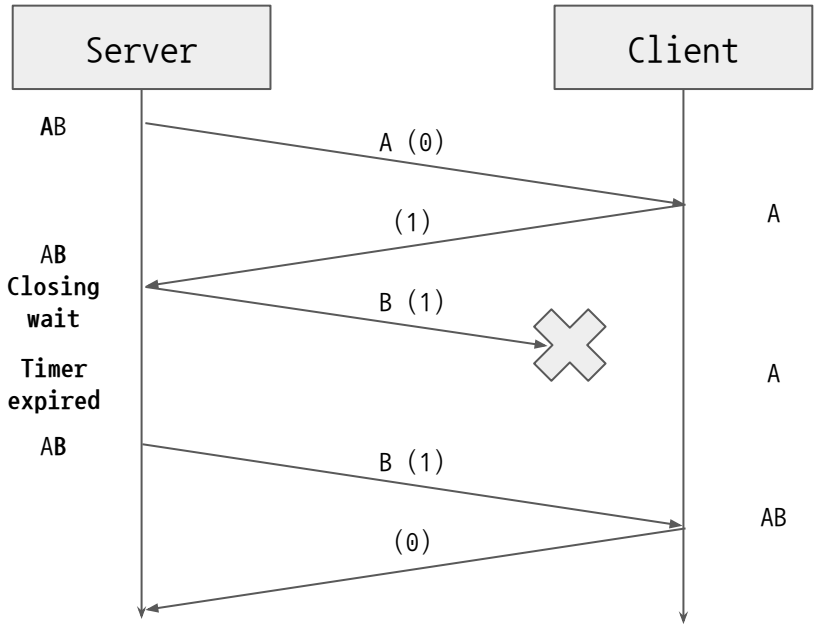
Stop and Wait ARQ: 에러 제어

- Case 2) Client → Server
  - Timer expired 되면 재전송
    - 새로 받은 데이터는 무시(이미 받았으므로)
    - ACK은 전송



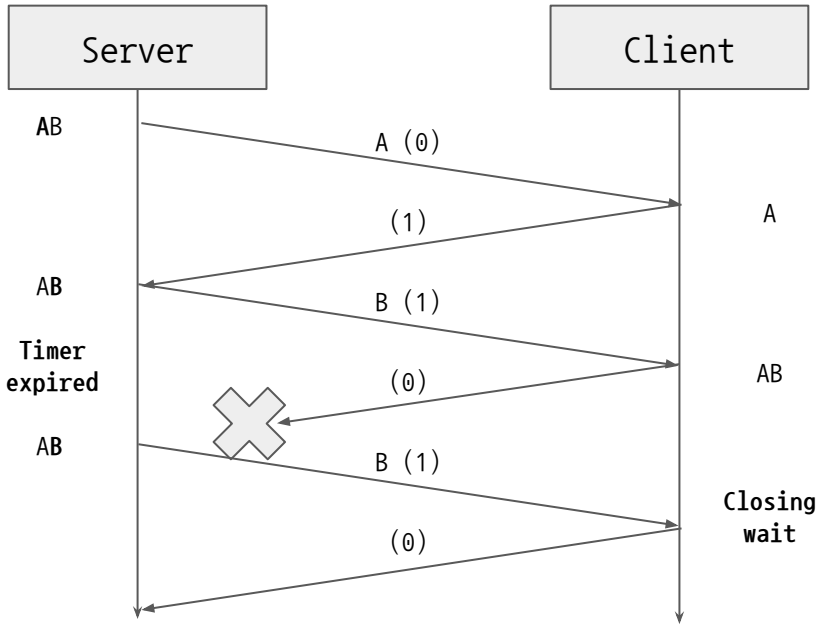
Stop and Wait ARQ: 에러 제어

- Case 3) Server → Client at the end
  - Timer expired 되면 재전송
  - 마지막 Data 을 보내고 나서도 대기
    - Timer 의 2배 대기
    - 왜?



Stop and Wait ARQ: 에러 제어

- Case 4) Client → Server at the end
  - Timer expired 되면 재전송
  - 마지막 Ack 을 보내고 나서도 대기
    - Timer 의 2배 대기
    - 왜?



Checksum 구현

- 2Bytes 1의 보수 합
- 빅 엔디안 형식으로 데이터를 읽어서 계산
  - 데이터만 계산!
  - Seq/Ack 은 제외하고 계산!
- checksum과 함께 계산시 0이 되는지 확인!

```
>>> def calculate_checksum(data):
... 구현 쭉!
>>>

>>> calculate_checksum(struct.pack('>H',
calculate_checksum(b'\x00\x00\x00\x01'))+b'\x00\x01')
0
```

```
Received checksum: 0
Receiving from ('127.0.0.1', 3034) with 1: 4865952/4865992
Received checksum: 0
Receiving from ('127.0.0.1', 3034) with 0: 4865992/4865992
File download success
```

과제: Download password image using Stop and Wait ARQ

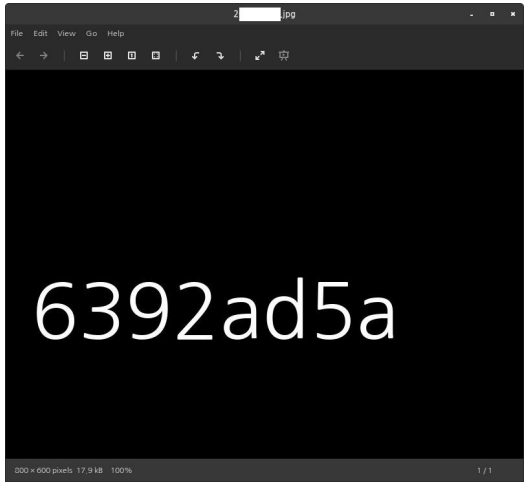
과제) 비밀번호 찾기!

서버

- GCP에 실행해둔 공개 서버
  IP: 34.171.223.63
  PORT:3034

클라이언트

- 서버에 DCFT2 에 맞추어서 요청
  - **학번.JPG** (대문자)
- 서버가 반환한 데이터 **'저장'**
  - jpg 이미지
- jpg 이미지에 써 있는 글자 제출
  - 학생마다 글자가 다름!




```
Filename: 2        .jpg
Request 2        .jpg to (34.168.194.7, 3034)
Calculated checksum: 0 with 0
Receiving from ('34.168.194.7', 3034) with 0: 1456/16746
Calculated checksum: 0 with 1
Receiving from ('34.168.194.7', 3034) with 1: 2912/16746
Calculated checksum: 0 with 1
Incorrect sequence packet arrived
Calculated checksum: 0 with 0
Receiving from ('34.168.194.7', 3034) with 0: 4368/16746
Calculated checksum: 0 with 1
Receiving from ('34.168.194.7', 3034) with 1: 5824/16746
Calculated checksum: 0 with 1
Incorrect sequence packet arrived
Calculated checksum: 0 with 0
Receiving from ('34.168.194.7', 3034) with 0: 7280/16746
Calculated checksum: 0 with 1
Receiving from ('34.168.194.7', 3034) with 1: 8736/16746
Calculated checksum: 0 with 0
Receiving from ('34.168.194.7', 3034) with 0: 10192/16746
Calculated checksum: 0 with 1
Receiving from ('34.168.194.7', 3034) with 1: 11648/16746
Calculated checksum: 59905 with 0
Incorrect sequence packet arrived
Calculated checksum: 125 with 0
Incorrect sequence packet arrived
Calculated checksum: 0 with 0
Receiving from ('34.168.194.7', 3034) with 0: 13104/16746
Calculated checksum: 0 with 1
Receiving from ('34.168.194.7', 3034) with 1: 14560/16746
Calculated checksum: 0 with 1
Incorrect sequence packet arrived
Calculated checksum: 0 with 0
Receiving from ('34.168.194.7', 3034) with 0: 16016/16746
Calculated checksum: 34863 with 1
Incorrect sequence packet arrived
Calculated checksum: 0 with 1
Receiving from ('34.168.194.7', 3034) with 1: 16746/16746
File download success
Filename:
```

힌트) 과제 구현해야 하는 것! 과 추천 순서!

1. 12주차 DCFT에서 실제 파일 전송하는 부분에 덧붙여가며 VM또는 localhost 사이에서 송수신
   a. Checksum 함수구현 (Checksum 2byte)
   b. 서버쪽 Seq, 클라이언트쪽 Ack 기능 추가
2. 완성된 클라이언트로 GCP서버에서 본인 학번 .JPG (대문자) 다운로드

실습13 과제 정리 및 채점 배점

- 이미지 다운로드 후 비밀번호 해독: 20점
  - 비밀번호 보고서 포함 필수!

또는…

- 파일 전송 서버, 클라이언트: 18점
  - VMware: Server
  - Host: Client
  - Stop and Wait ARQ Case 1~4 모두 처리
  - 어떻게 처리한 것인지 그림 또는 자세한 설명으로 증명
  - 송수신이 잘 되는 스크린샷

- 기타 부정행위시 감점

- zip 파일 압축: DC02-학번-이름.zip
  - 보고서 (학번-이름.**pdf**)
    - 자신이 했음을 증명
      ⇒ 스크린샷 및 설명
  - 소스코드 (*.py)
  - 비밀번호! 또는 서버-클라이언트 스크린샷
- e-learning 사이버캠퍼스 제출
- 설문조사: [클릭]

- 제출 기한: 2025. 06. 04. 23:59

- 추가제출 1일 (30% 감점)

과제 서버 유의사항

잘못된 ACK 보냈을시, Timeout 10초 설정
해두었으니 서버가 정상작동하지않는것 같으면
(INFO TIMEOUT 발생시) 10초간 대기할 것!

저장된 파일은 .JPG (jpg 아님) 이므로 대소문자
구분할것