

2019년 데이터통신

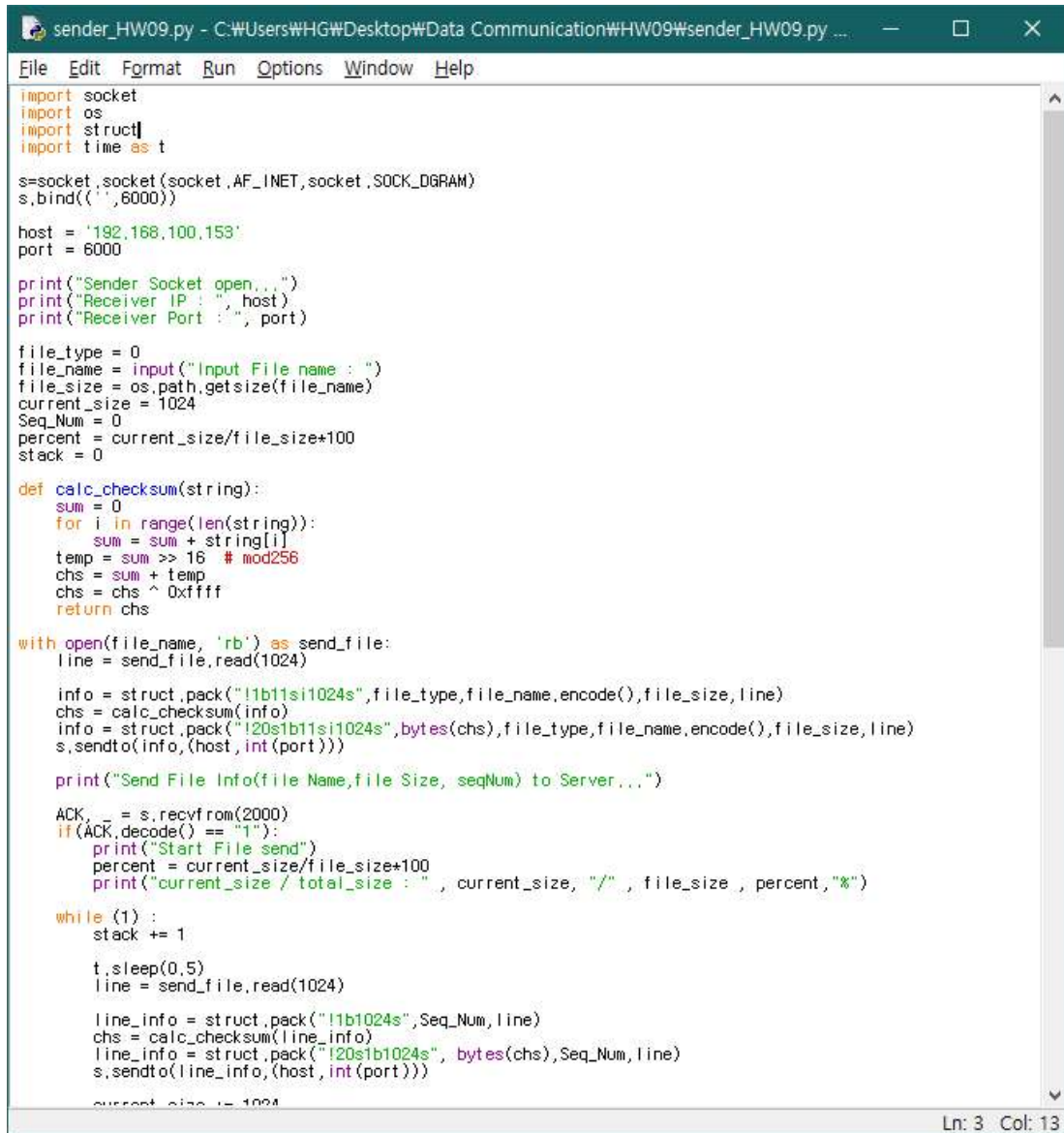
- HW 08 -

이름	노효근
학번	201502049
분반	02

* 과제 목표

- Stop-and-Wait 개념에 대해서 이해한다.
- Stop-and-Wait을 파이썬에서 구현해본다.

* sender.py



```
import socket
import os
import struct
import time as t

s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind(('',6000))

host = '192.168.100.153'
port = 6000

print("Sender Socket open...")
print("Receiver IP : ", host)
print("Receiver Port : ", port)

file_type = 0
file_name = input("Input File name : ")
file_size = os.path.getsize(file_name)
current_size = 1024
Seq_Num = 0
percent = current_size/file_size*100
stack = 0

def calc_checksum(string):
    sum = 0
    for i in range(len(string)):
        sum = sum + string[i]
    temp = sum >> 16 # mod256
    chs = sum + temp
    chs = chs ^ 0xffff
    return chs

with open(file_name, 'rb') as send_file:
    line = send_file.read(1024)

    info = struct.pack("11b11s1024s", file_type, file_name.encode(), file_size, line)
    chs = calc_checksum(info)
    info = struct.pack("120s1b11s1024s", bytes(chs), file_type, file_name.encode(), file_size, line)
    s.sendto(info, (host, int(port)))

    print("Send File Info(file Name, file Size, seqNum) to Server...")

    ACK, _ = s.recvfrom(2000)
    if ACK.decode() == "1":
        print("Start File send")
        percent = current_size/file_size*100
        print("current_size / total_size : ", current_size, "/", file_size, percent, "%")

        while (1):
            stack += 1

            t.sleep(0.5)
            line = send_file.read(1024)

            line_info = struct.pack("11b1024s", Seq_Num, line)
            chs = calc_checksum(line_info)
            line_info = struct.pack("120s1b1024s", bytes(chs), Seq_Num, line)
            s.sendto(line_info, (host, int(port)))

            current_size += 1024
```

이전 과제와 마찬가지로 기본 상태를 세팅해준다. socket 라이브러리를 import한다. 이후, socket객체를 생성하여 통신에 필요한 배경을 만들어준다. 내가 보내줄 파일을 입력할 창을 만들고, 내가 보낼 파일의 총 size를 os.path.getsize() 메소드를 이용하여 구한다. 이후 with open 메소드를 이용하여, 파일 읽기 전용으로 파일을 바이트로 연다.

파일의 이름과 총 size와 1024bytes 읽은 데이터를 pack 라이브러리를 통해 1040bytes로 묶는다. 묶은 info를 통해 calc_checksum 함수를 이용하여 checksum를 구하고 구한 checksum과 info를 다시 pack 하여, 내가 보낼 장치의 ip주소와 port 번호를 이용하여 전송한다. 이후 위 방법과 마찬가지로, seq_num와 1024bytes 읽은 데이터(line_info)를 이용하여 checksum을 구하고, 구한 checksum과 line_info을 보낸다.

```
sender_HW09.py - C:\Users\HG\Desktop\Data Communication\HW09\sender_HW09.py ...
File Edit Format Run Options Window Help

percent = current_size/file_size*100
print("current_size / total_size : " , current_size, "/" , file_size , percent,"%")

while (1) :
    stack += 1

    t.sleep(0.5)
    line = send_file.read(1024)
    line_info = struct.pack("1b1024s",Seq_Num,line)
    chs = calc_checksum(line_info)
    line_info = struct.pack("20s1b1024s", bytes(chs),Seq_Num,line)
    s.sendto(line_info,(host,int(port)))

    current_size += 1024
    percent = current_size/file_size*100
    if(percent > 100):
        current_size = file_size
        percent = 100.0
    print("current_size / total_size : " , current_size, "/" , file_size , percent,"%")

    if(percent == 100.0):
        print("File send end")
        break
    ACK, _ = s.recvfrom(2000)

    if(ACK.decode() == "wait"):
        print("-----Time_Out!-----")
        print("Retransmit")
        current_size -= 1024
        continue

    if(ACK.decode() == "2"):
        print("Retransmit")
        current_size -=1024
        if(Seq_Num == 1):
            Seq_Num = 0
            continue
        if(Seq_Num == 0):
            Seq_Num = 1
            continue

    elif(ACK.decode() == "NAK"):
        print("Retransmit")
        current_size -= 1024
        continue

    if(ACK.decode() == "1"):
        if(stack == 7):
            continue
        if(Seq_Num == 1):
            Seq_Num = 0
            continue
        if(Seq_Num == 0):
            Seq_Num = 1
            continue
```

Ln: 54 Col: 8

이때 reciver에게 ACK를 잘 받았을 경우 (ACK = 1), seq_num을 0에서1 바꾸고 다음 data를 읽어 보내도록 한다. 이때, 보낸 byte와 총 파일의 size를 이용하여, 보낸 %를 계산하여 출력하도록 한다.

1. ACK = wait를 받는 경우는 Time_out이 발생하여 재전송하는 경우
 2. ACK = 2를 받는 경우는 중간에 seq_num이 겹쳐서 올 때(1->1)를 가정하여 재전송하는 경우
 3. ACK = NAK를 받는 경우는 중간에 checksum이 달라질 때 재전송하는 경우
- 이렇게 3가지 경우를 만들어서 stack이란 변수를 이용하여, stop-and-wait 매커니즘이 잘 작동하는지 확인한다.

* reciver.py

```
reciver_HW09.py - C:\Users\HG\Desktop\reciver_HW09.py (3.7.3)
File Edit Format Run Options Window Help

import socket
import os
import struct
import time as t

s=socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
s.bind(('',6000))

host = '192.168.230.128'
port = 6000

print("Sender Socket open...")
print("Receiver IP : 192.168.100.153")
print("Receiver Port : ",port)

info,_ = s.recvfrom(1060)
u_info = struct.unpack("120s1b11s1i1024s",info)
file_name = u_info[2].decode().strip('\0')
file_size = u_info[3]
print("File Name = ", file_name)
print("File Size = ", file_size)

ACK = 1
current_size = 0
check = 1
percent = 0
stack = 0

def calc_checksum(string):
    sum = 0
    for i in range(len(string)):
        sum = sum + string[i]
    temp = sum >> 16 # mod256
    chs = sum + temp
    chs = chs ^ 0xffff
    return chs

with open(file_name, 'wb') as receive_file :
    receive_file.write(u_info[4])
    current_size += 1024
    percent = current_size/file_size*100
    if(percent > 100):
        current_size = file_size
        percent = 100.0
    print("current_size / total_size : ", current_size, "/" , file_size , percent,"%")
    s.sendto(str(ACK).encode(),(host,int(port)))

    while (1) :
        if(percent == 100.0):
            print("file receive end")
            break
        stack += 1

        if(stack == 3):
            print("Wait for 5...")
            ACK = "wait"
            t.sleep(5)
            s.sendto(str(ACK).encode(),(host,int(port)))
            stack = 0

Ln: 25 Col: 0
```

이전 과제와 마찬가지로 기본 상태를 세팅해준다. socket 라이브러리를 import한다. 이후, socket객체를 생성하여 통신에 필요한 배경을 만들어준다. sender에서 보낸 파일이름과 ip주소, 파일의 총 size를 revfrom 메소드를 통해 받는다. 이후 받은 파일에 대한 정보를 출력한다. with open 메소드를 통해, 파일을 쓰기 전용으로 파일을 바이트로 만든다. revfrom으로 1060byte 파일을 받아와 만든 파일에 쓰도록한다. 잘 받았으면 ACK =1를 전송하여, sender에게 보내주어 다음 파일을 받을 준비를 한다. stack변수를 이용하여, 5일 경우 ACK를 받지않고 Time_out되는 상황을 만들어본다.

이후 1045bytes 씩 받는 파일에서 checksum을 계산하여, 맞으면 파일을 받고 틀리면, "*Received NAK - Retransmit!" 에러를 출력하고, ACK =NAK를 보내 checksum이 틀려 재전송을 요청한다.

```
reciver_HW09.py - C:\Users\HG\Desktop\reciver_HW09.py (3.7.3)
File Edit Format Run Options Window Help
print(current_size / total_size , current_size , "/" , file_size , percent , "%")
s.sendto(str(ACK).encode(),(host,int(port)))

while (1) :
    if(percent == 100,0):
        print("file receive end")
        break
    stack += 1

    if(stack == 3):
        print("Wait for 5...")
        ACK = "wait"
        t.sleep(5)
        s.sendto(str(ACK).encode(),(host,int(port)))
        continue

    ACK = 1
    data,_ = s.recvfrom(1045)
    u_data = struct.unpack("!20s1b1024s",data)

    chs=u_data[0]
    seq=u_data[1]
    line=u_data[2]

    r_chs = struct.pack("1b1024s",seq,line)
    r_chs = calc_checksum(r_chs)
    r_chs = struct.pack("20s",bytes(r_chs))

    if(stack == 5):
        r_chs = r_chs+bytes(2)

    if(chs == r_chs):
        if(u_data[1] == check):
            print("* Packet corrupted!! *** - Send To Sender NAK(2)")
            ACK = "2"
            s.sendto(str(ACK).encode(),(host,int(port)))
            continue
        else:
            recive_file.write(line)
            current_size += 1024
            percent = current_size/file_size*100
            if(percent > 100):
                current_size = file_size
                percent = 100,0
            print("current_size / total_size : " , current_size , "/" , file_size , percent,"%")
            check = u_data[1]

            # ACK 보내기
            s.sendto(str(ACK).encode(),(host,int(port)))
            if(current_size > file_size):
                print("file receive end")
                break
        else:
            print("*Received NAK - Retransmit!")
            ACK = "NAK"
            s.sendto(str(ACK).encode(),(host,int(port)))
            continue

Ln: 87 Col: 39
```

checksum이 맞으면 이전 seq_num을 저장하는 check 변수를 이용하여, seq_num이 제대로 번갈아가며 오는지 확인한다.

이때, 같은 seq_num이 온다면, "* Packet corrupted!! *** - Send To Sender NAK(2)" 에러를 출력하고 ACK = 2를 보내 seq를 바꾸어 다음파일을 받도록 요청한다.

만약 모든 과정을 다 통과하여 잘 받게 된다면, 받은 byte와 총 파일의 size를 이용하여, 받은 %를 계산하여 출력하도록 한다.

sender와 마찬가지로

1. ACK = wait를 받는 경우는 Time_out이 발생하여 재전송하는 경우
2. ACK = 2를 받는 경우는 중간에 seq_num이 겹쳐서 올 때(1->1)를 가정하여 재전송하는 경우
3. ACK = NAK를 받는 경우는 중간에 checksum이 달라질 때 재전송하는 경우

이렇게 3가지 경우를 만들어서 stack이란 변수를 이용하여, stop-and-wait 매커니즘이 잘 작동하는지 확인한다.

* 실행 결과

- ◆ 지급받은 라즈베리 파이가 에러가 발생하여 개인 데스크탑 가상머신과 개인 노트북 가상머신끼리 통신하였습니다.

: 파일 전송을 하는 sender - 개인 노트북

```
Sender Socket open...
Receiver IP : 192.168.100.153
Receiver Port : 6000
Input File name : poem.txt
Send File Info(file Name,file Size, seqNum) to Server...
Start File send
current_size / total_size : 1024 / 11264 9.090909090909092 %
current_size / total_size : 2048 / 11264 18.181818181818183 %
current_size / total_size : 3072 / 11264 27.27272727272727 %
current_size / total_size : 4096 / 11264 36.36363636363637 %
-----Time_Out!-----
Retransmit
current_size / total_size : 4096 / 11264 36.36363636363637 %
current_size / total_size : 5120 / 11264 45.45454545454545 %
Retransmit
current_size / total_size : 5120 / 11264 45.45454545454545 %
current_size / total_size : 6144 / 11264 54.54545454545454 %
Retransmit
current_size / total_size : 6144 / 11264 54.54545454545454 %
current_size / total_size : 7168 / 11264 63.63636363636363 %
current_size / total_size : 8192 / 11264 72.72727272727273 %
current_size / total_size : 9216 / 11264 81.81818181818183 %
current_size / total_size : 10240 / 11264 90.9090909090909 %
current_size / total_size : 11264 / 11264 100.0 %
File send end
```

: 파일을 받는 reciver - 개인 데스크탑

```
Sender Socket open...
Receiver IP : 192.168.100.153
Receiver Port : 6000
File Name = poem.txt
File Size = 11264
current_size / total_size : 1024 / 11264 9.090909090909092 %
current_size / total_size : 2048 / 11264 18.181818181818183 %
current_size / total_size : 3072 / 11264 27.27272727272727 %
Wait for 5...
current_size / total_size : 4096 / 11264 36.36363636363637 %
*Received NAK - Retransmit!
current_size / total_size : 5120 / 11264 45.45454545454545 %
* Packet corrupted!! *** - Send To Sender NAK(2)
current_size / total_size : 6144 / 11264 54.54545454545454 %
current_size / total_size : 7168 / 11264 63.63636363636363 %
current_size / total_size : 8192 / 11264 72.72727272727273 %
current_size / total_size : 9216 / 11264 81.81818181818183 %
current_size / total_size : 10240 / 11264 90.9090909090909 %
current_size / total_size : 11264 / 11264 100.0 %
file receive end
```

* GitHub ID : Nroot33