

# 2019년 데이터통신

- HW 11 -

이름	노효근
학번	201502049
분반	02

## \* 과제 목표

- GoBackN을 파이썬에서 구현해본다.
- Stop-and-Wait와의 성능을 비교해본다.

## \* sender.py

이전 과제와 마찬가지로 기본 상태를 세팅해준다. socket 라이브러리를 import한다. 이후, socket객체를 생성하여 통신에 필요한 배경을 만들어준다.

이전 stop-and-wait와 같은 배경에서 보내는 방법과 Ack 받는 형식만 다르기 때문에 sendto 하는 부분만 수정하도록 하였다.

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

# checksum 계산
def calc_checksum(string):
    sum = 0
    for i in range(len(string)):
        sum = sum + string[i]
    temp = sum >> 16
    chs = sum + temp
    if chs >= 65536 :
        chs -= 65536
    if chs >= 131072 :
        chs -= 131072
    chs = chs ^ 0xffff
    return chs

# 퍼센트 계산
def calc_percent(current_length, file_size):
    percent = current_length/file_size*100
    if(current_length >= file_size):
        current_length = file_size
        percent = 100
    print("current_size / total_size : " , current_length, "/" , file_size , "=" , round(percent), "%\n")
    print("File send end")
    end = t.time()
    print(end-start)
    sys.exit()
    print("current_size / total_size : " , current_length, "/" , file_size , "=" , round(percent), "%\n")

# checksum을 계산하여 payload를 만듦
def make_payload(current_length, seq_num):
    file_type = "d"
    current_length += 1024
    b_current_length = current_length.to_bytes(20, byteorder = "big")
    b_seq_num = str(seq_num).encode().ljust(15)
    payload = file.read(1024)
    b_payload = file_type.encode()+b_current_length+b_seq_num+payload
    payload_checksum = calc_checksum(b_payload).to_bytes(20, byteorder = "big")
    payload_frame = file_type.encode()+payload_checksum+b_current_length+b_seq_num+payload
    return payload_frame
```

- def calc\_checksum을 통해 내가 보낼 header와 payload에 대한 checksum을 구하는 함수를 구현하였다.

- def calc\_percent을 통해 내가 보낸 파일에 대한 %를 출력하고, 100%가 되었을시 전체 프로그램이 종료되도록 함수 구현하였다.

- def make\_payload를 통해 보낼 파일의 1025byte를 읽고 보낼 packet을 생성하도록 함수를 구현하였다.

```

## GoBackN ##
s=socket,socket(socket,AF_INET,socket,SOCK_DGRAM)
s.settimeout(4)

host = '192.168.230.128'
port = 6000

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

file_type = "s"
file_name = input("Input File name : ")
file_size = os.path.getsize(file_name)

b_file_size = file_size.to_bytes(20,byteorder = "big")
b_file_name = file_name.encode().ljust(15)
current_length = 0
seq_num = 0
window = []
window_size = 4

with open(file_name, 'rb') as file:
    payload = file.read(1024)
    header_frame = file_type.encode()+b_file_size+b_file_name+payload
    header_checksum = calc_checksum(header_frame).to_bytes(20,byteorder = "big")
    header_frame = file_type.encode()+header_checksum+b_file_size+b_file_name+payload
    start = t,time()
    s.sendto(header_frame,(host,port))
    print("Send File Info(file_Type, Checksum, file Name, file Size, Payload ) to Server,...")
    while(1) :
        try :
            ACK, _ = s.recvfrom(1000)
            break

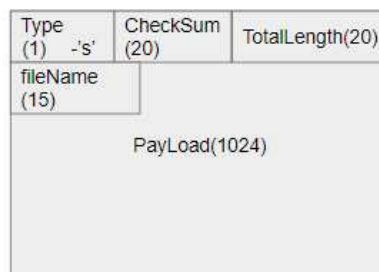
        except socket.timeout: # timeout으로 인한 재전송 요청
            print("* TimeOut!! ***")
            print("Retransmission")
            s.sendto(header_frame,(host,port))
            s.settimeout(4)
            continue

    if (ACK.decode() == "head"):
        print("Start File send")
        current_length +=1024
        calc_percent(current_length,file_size)

```

내가 보내줄 파일을 입력할 창을 만들고, 내가 보낼 파일의 총 size를 os.path.getsize() 메소드를 이용하여 구한다.

이후 with open을 통해 파일을 열고 처음으로 보낼 head에 관한 내용을 가져온다. header에 필요한 내용을 가져와, 헤더에 대한 전송프레임에 맞게 to\_bytes를 이용하여 아래 그림과 같은 format으로 Frame을 생성하고 전송한다. 마찬가지로 header에서도 checksum을 계산하고 timeout에 관한 예외처리를 하고 payload를 보내도록 준비한다.



```

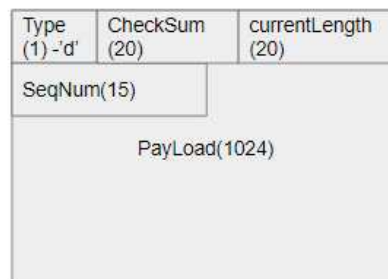
for i in range(window_size):
    current_length += 1024
    payload_frame = make_payload(current_length, seq_num)
    s.sendto(payload_frame, (host, port))
    if (len(window) > 3):
        window.pop(0)
        window.append(seq_num)
        print("**** Window Change! ****")
        print("Present Window : ", window)
    else:
        window.append(seq_num)
    seq_num += 1
    seq_num = seq_num % 8
    calc_percent(current_length, file_size)

while(1):
    current_length += 1024
    payload_frame = make_payload(current_length, seq_num)
    s.sendto(payload_frame, (host, port))
    if (len(window) > 3):
        window.pop(0)
        window.append(seq_num)
        print("**** Window Change! ****")
        print("Present Window : ", window)
    else:
        window.append(seq_num)
    seq_num += 1
    seq_num = seq_num % 8
    calc_percent(current_length, file_size)
    n = 1

try:
    ACK, _ = s.recvfrom(1000)
    ACK = int.from_bytes(ACK, byteorder = "big")
    if (ACK == (window[0]+n)%8):
        n+=1
        if (n>4):
            break
        continue
except socket.timeout: # timeout으로 인한 재전송 요청
    print("**** TimeOut!! ****")
    file.seek(-1024*window_size, 1)
    current_length = current_length-1024*window_size
    for i in range(window_size):
        current_length += 1024
        payload_frame = make_payload(current_length, window[0+i])
        s.sendto(payload_frame, (host, port))
        calc_percent(current_length, file_size)
    continue

```

GoBackN 방식대로 window배열을 만들어 내가 앞으로 보낼 payload의 seq\_num을 저장하여 timeout이 발생하면 저장된 seq\_num의 payload를 보낼 수도록 한다. make\_payload를 이용하여 아래와 같은 format으로 Frame을 생성한다.



또한, payload를 계속해서 보낼 때마다, 순차적으로 seq\_num을 window배열에 저장하도록 한다. 이때 바뀐 window 배열과 percent를 출력한다.

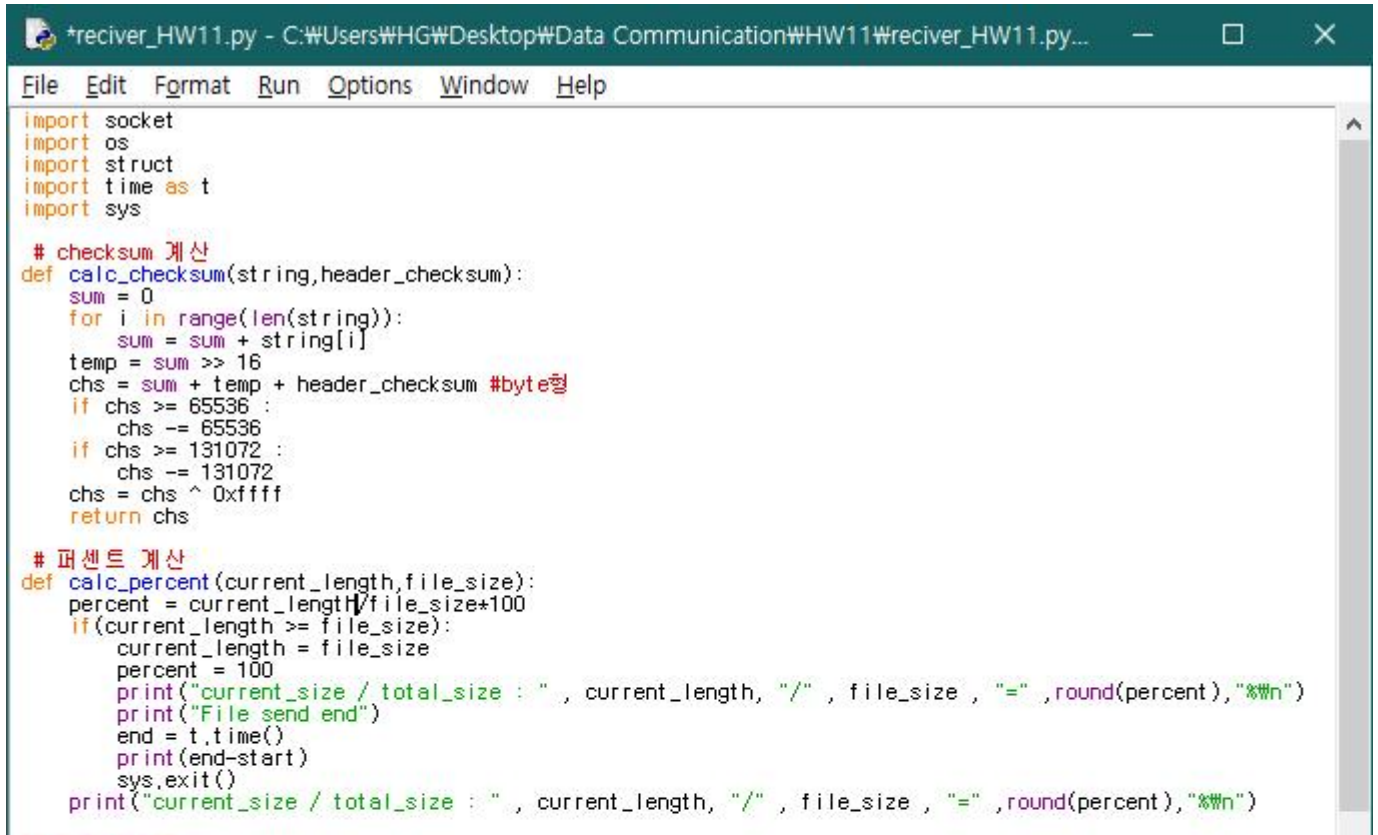
이후 reciver에게 받은 Ack를 확인하여, window 배열에서 pop을 통해 제거 하도록 한다.

이후 timeout이 발생할 경우 file.seek 함수를 통해 buffer 이전에 파일을 읽었던 곳으로 돌리고, 현재 window배열을 검사하여, 필요한 payload를 다시 읽어 재전송하고 percent를 출력한다. 재전송 하도록 한다.

## \* reciver.py

이전 과제와 마찬가지로 기본 상태를 세팅해준다. socket 라이브러리를 import한다. 이후, socket객체를 생성하여 통신에 필요한 배경을 만들어준다.

이전 stop-and-wait와 같은 배경에서 보내는 방법과 Ack 받는 형식만 다르기 때문에 recvfrom하는 부분만 수정하도록 하였다.



```
*reciver_HW11.py - C:\Users\HG\Desktop\Data Communication\HW11\reciver_HW11.py...
File Edit Format Run Options Window Help

import socket
import os
import struct
import time as t
import sys

# checksum 계산
def calc_checksum(string,header_checksum):
    sum = 0
    for i in range(len(string)):
        sum = sum + string[i]
    temp = sum >> 16
    chs = sum + temp + header_checksum #byte형
    if chs >= 65536 :
        chs -= 65536
    if chs >= 131072 :
        chs -= 131072
    chs = chs ^ 0xffff
    return chs

# 퍼센트 계산
def calc_percent(current_length,file_size):
    percent = current_length/file_size*100
    if(current_length >= file_size):
        current_length = file_size
        percent = 100
    print("current_size / total_size : " , current_length, "/" , file_size , "=" ,round(percent),"%\n")
    print("File send end")
    end = t.time()
    print(end-start)
    sys.exit()
    print("current_size / total_size : " , current_length, "/" , file_size , "=" ,round(percent),"%\n")
```

- def calc\_checksum을 통해 내가 받은 header와 payload에 대한 checksum을 구하는 함수를 구현하였다.
- def calc\_percent을 통해 내가 보낸 파일에 대한 %를 출력하고, 100%가 되었을시 전체 프로그램이 종료되도록 함수 구현하였다.

```

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

header_frame, addr = s.recvfrom(1080)
start = t.time()

print("Receiver Socket open...")
print("Sender IP :", addr[0])
print("Sender Port :", addr[1])

file_type = chr(header_frame[0])
header_checksum = int,from_bytes(header_frame[1:21],byteorder = "big")
file_size = int,from_bytes(header_frame[21:41],byteorder = "big")
file_name = header_frame[41:56]
file_name = file_name[0:file_name.find(32)].decode()
header_payload = header_frame[56:]
print("File Name =", file_name)
print("File Size =", file_size)
check_frame = file_type.encode()+file_size.to_bytes(20,byteorder = "big")+file_name.encode(),ljust(15)+header
check_sum = calc_checksum(check_frame,header_checksum)
current_length = 0
ACK = "head"
pre_payload_seqnum = 0
while (1):
    if(check_sum == 0):
        with open(file_name, 'wb') as file:
            file.write(header_payload)
            current_length += 1024
            calc_percent(current_length,file_size)
            s.sendto(ACK.encode(),addr)
        while (1):
            payload_frame,addr = s.recvfrom(1080)
            payload_type = chr(payload_frame[0])
            payload_checksum = int,from_bytes(payload_frame[1:21],byteorder = "big")
            payload_current_length = int,from_bytes(payload_frame[21:41],byteorder = "big")
            payload_seqnum = payload_frame[41:56]
            payload_seqnum = payload_seqnum[0:payload_seqnum.find(32)].decode()
            payload = payload_frame[56:]
            check_frame = payload_type.encode()+payload_current_length.to_bytes(20,byteorder = "big")+pay
            check_sum = calc_checksum(check_frame,payload_checksum)

            if(check_sum == 0):
                file.write(payload)
                current_length += 1024
                ACK = (int(payload_seqnum)+1) % 8
                ACK = ACK.to_bytes(4,byteorder = "big")
                calc_percent(current_length,file_size)
                t.sleep(1)
                s.sendto(ACK,addr)
                #print("Send to ACK ",int,from_bytes(ACK,byteorder="big"))

```

sender에서 보낸 header의 내용을 recvfrom을 통해 파일의 정보를 받아 출력해준다.

checksum을 계산하여 잘 받았으면,

with open을 통해 파일을 열고 쓴다. 이후 sender에서 보내는 payload에 대한, 내용을 읽어, checksum을 계산하여 데이터를 계속 쓰도록 한다.

이때 받은 seq\_num을 이용하여 다음 seq\_num에 대한 ACK를 전송하도록 구현하였다.



## \* 실행 결과

: 파일 전송을 하는 sender - 개인 노트북

```
Sender Socket open...
Receiver IP : 172.30.1.26
Receiver Port : 6000
Input File name : poem.txt
Send File Info(file_Type, Checksum, file Name, file Size, Payload )
to Server...
Start File send
current_size / total_size : 1024 / 27186 = 4 %

current_size / total_size : 2048 / 27186 = 8 %

current_size / total_size : 3072 / 27186 = 11 %

current_size / total_size : 4096 / 27186 = 15 %

current_size / total_size : 5120 / 27186 = 19 %

**** Window Change! ****
Present Window : [1, 2, 3, 4]
current_size / total_size : 6144 / 27186 = 23 %

**** Window Change! ****
Present Window : [2, 3, 4, 5]
current_size / total_size : 7168 / 27186 = 26 %

**** Window Change! ****
Present Window : [0, 1, 2, 3]
current_size / total_size : 21504 / 27186 = 79 %

**** Window Change! ****
Present Window : [1, 2, 3, 4]
current_size / total_size : 22528 / 27186 = 83 %

**** Window Change! ****
Present Window : [2, 3, 4, 5]
current_size / total_size : 23552 / 27186 = 87 %

**** Window Change! ****
Present Window : [3, 4, 5, 6]
current_size / total_size : 24576 / 27186 = 90 %

**** Window Change! ****
Present Window : [4, 5, 6, 7]
current_size / total_size : 25600 / 27186 = 94 %

**** Window Change! ****
Present Window : [5, 6, 7, 0]
current_size / total_size : 26624 / 27186 = 98 %

**** Window Change! ****
Present Window : [6, 7, 0, 1]
current_size / total_size : 27186 / 27186 = 100 %

File send end
21.05427861213684
```

: 파일을 받는 reciver - 개인 데스크탑

```
Reciver Socket open...
Sender IP : 192.168.230.1
Sender Port : 59863
File Name = poem.txt
File Size = 27186
current_size / total_size : 1024 / 27186 = 4 %

current_size / total_size : 2048 / 27186 = 8 %

current_size / total_size : 3072 / 27186 = 11 %

current_size / total_size : 4096 / 27186 = 15 %

current_size / total_size : 5120 / 27186 = 19 %

current_size / total_size : 6144 / 27186 = 23 %

current_size / total_size : 7168 / 27186 = 26 %

current_size / total_size : 8192 / 27186 = 30 %

current_size / total_size : 9216 / 27186 = 34 %

current_size / total_size : 10240 / 27186 = 38 %

current_size / total_size : 11264 / 27186 = 41 %

current_size / total_size : 12288 / 27186 = 45 %

current_size / total_size : 13312 / 27186 = 49 %

current_size / total_size : 16384 / 27186 = 60 %

current_size / total_size : 17408 / 27186 = 64 %

current_size / total_size : 18432 / 27186 = 68 %

current_size / total_size : 19456 / 27186 = 72 %

current_size / total_size : 20480 / 27186 = 75 %

current_size / total_size : 21504 / 27186 = 79 %

current_size / total_size : 22528 / 27186 = 83 %

current_size / total_size : 23552 / 27186 = 87 %

current_size / total_size : 24576 / 27186 = 90 %

current_size / total_size : 25600 / 27186 = 94 %

current_size / total_size : 26624 / 27186 = 98 %

current_size / total_size : 27186 / 27186 = 100 %

File send end
```

## \* StopAndWait & GoBackN Netems 실험

Delay 0		
Byte	GoBackN	StopAndWait
100k	current_size / total_size : 102400 / 102400 = 100 % File send end 11.1058	current_size / total_size : 102400 / 102400 = 100 % File send end 17.456
1Mb	current_size / total_size : 1048576 / 1048576 = 100 % File send end 52.124	current_size / total_size : 1048576 / 1048576 = 100 % File send end 78.65
10Mb	current_size / total_size : 10485760 / 10485760 = 100 % File send end 532.5004	current_size / total_size : 10485760 / 10485760 = 100 % File send end 709.56
50Mb	current_size / total_size : 52428800 / 52428800 = 100 % File send end 1834.549198	current_size / total_size : 52428800 / 52428800 = 100 % File send end 2552.4132

Loss 1%		
Byte	GoBackN	StopAndWait
100k	current_size / total_size : 102400 / 102400 = 100 % File send end 37.5	current_size / total_size : 102400 / 102400 = 100 % File send end 42.00156
1Mb	current_size / total_size : 1048576 / 1048576 = 100 % File send end 328.201562	current_size / total_size : 1048576 / 1048576 = 100 % File send end 587.25046
10Mb	current_size / total_size : 10485760 / 10485760 = 100 % File send end 804.953144	current_size / total_size : 10485760 / 10485760 = 100 % File send end 904.702201
50Mb	current_size / total_size : 52428800 / 52428800 = 100 % File send end 2137.9800005	current_size / total_size : 52428800 / 52428800 = 100 % File send end 2913.508223

Loss 5%		
Byte	GoBackN	StopAndWait
100k	current_size / total_size : 102400 / 102400 = 100 % File send end 78.29651046	current_size / total_size : 102400 / 102400 = 100 % File send end 174.000003
1Mb	current_size / total_size : 1048576 / 1048576 = 100 % File send end 748.112457	current_size / total_size : 1048576 / 1048576 = 100 % File send end 974.58131101
10Mb	current_size / total_size : 10485760 / 10485760 = 100 % File send end 1511.111026	current_size / total_size : 10485760 / 10485760 = 100 % File send end 1326.861045
50Mb	current_size / total_size : 52428800 / 52428800 = 100 % File send end 3087.232978	current_size / total_size : 52428800 / 52428800 = 100 % File send end 3604.236

## \* Graph

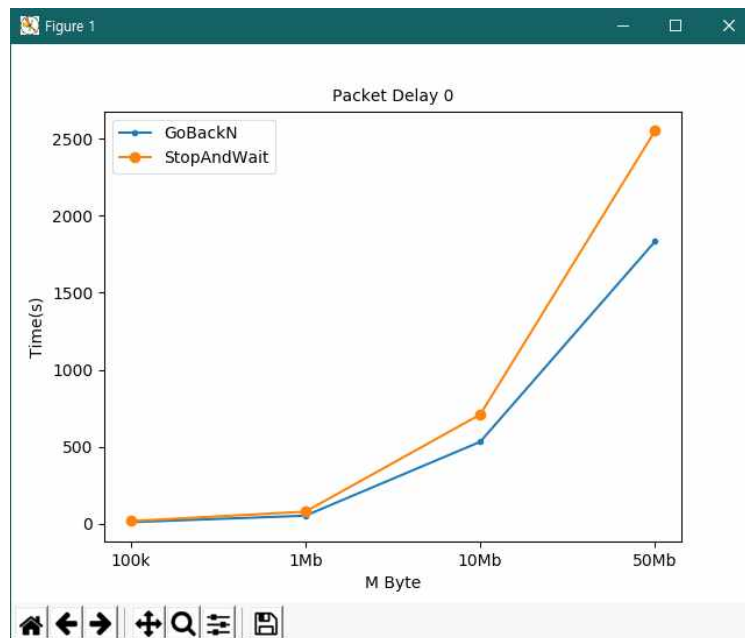


그림) Delay 0일 때 GoBackN과 StopAndWait 성능 비교  
그래프



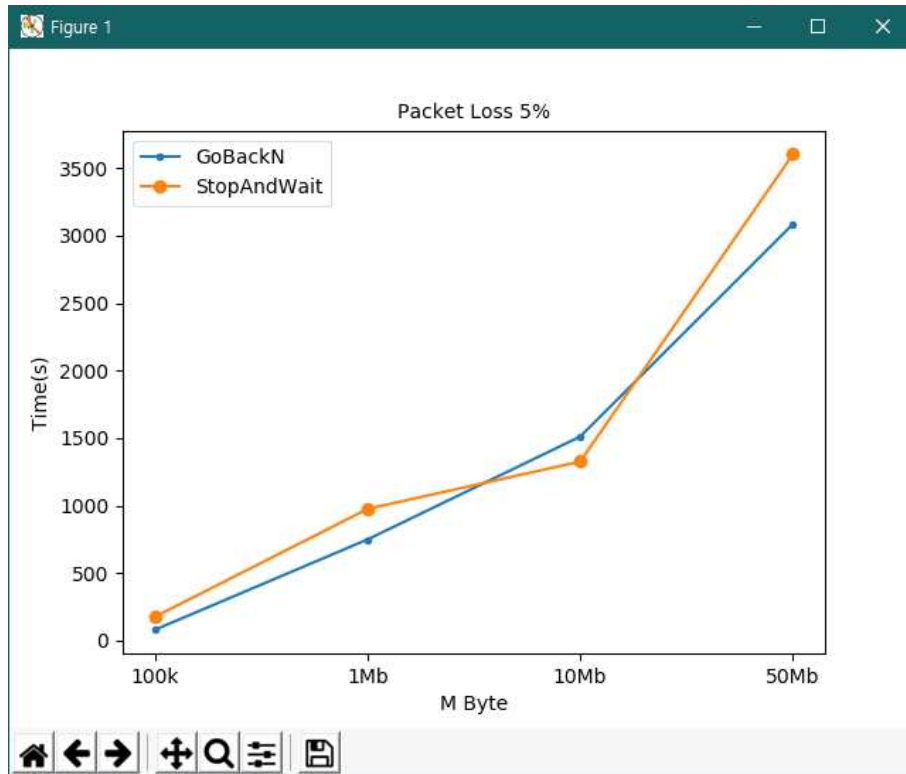


그림) Loss 1%일 때 GoBackN과 StopAndWait 성능 비교 그래프

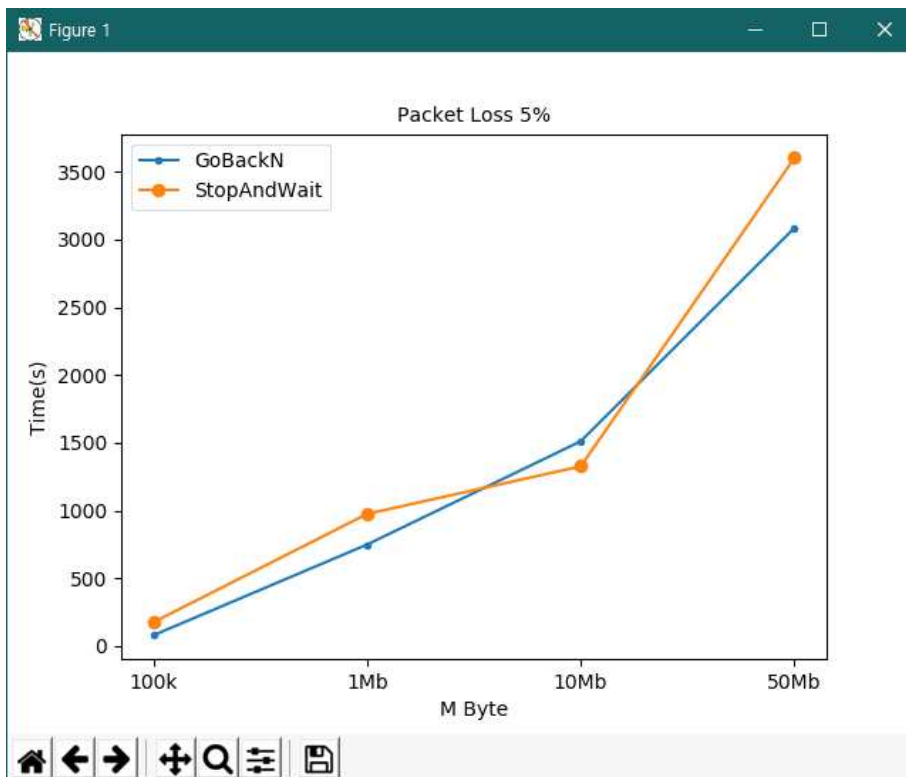


그림) Loss 5%일 때 GoBackN과 StopAndWait 성능 비교 그래프

\* GitHub ID : Nroot33