컴퓨터 네트워크 FINAL EXAM

소프트웨어학부 20171656 유성현

O 각code block의 구조 및 기능 설명

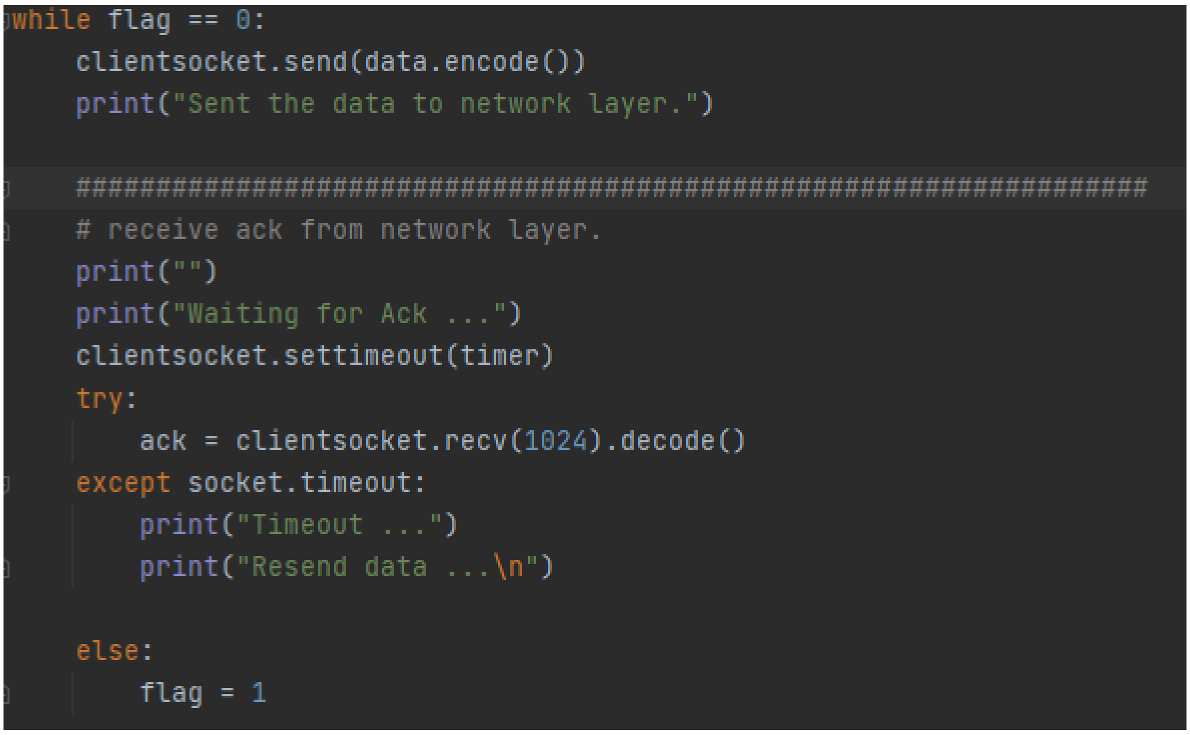
1. Application_layer_sender

- Python 표준 입출력을 통해 data를 input 하고 socket을 이용하여 transport layer로 전송함.

- transport layer로부터 ack를 수신하고 수신완료 메시지를 출력함.

2. Transport_layer_sender

- application layer로부터 받은 data를 stop_and_wait potocol을 사용하여 network layer로 전송하고 ack 메시지를 기다림. Timer 내에 ack 메시지를 받지 못하면 재전송함.



```
while flag == 0:
    clientsocket.send(data.encode())
    print("Sent the data to network layer.")

    ####################################################################
    # receive ack from network layer.
    print("")
    print("Waiting for Ack ...")
    clientsocket.settimeout(timer)
    try:
        ack = clientsocket.recv(1024).decode()
    except socket.timeout:
        print("Timeout ...")
        print("Resend data ...\n")

    else:
        flag = 1
```

- network layer로부터 ack를 수신하고 applicaition layer에 ack를 한번 더 보내 수신완료 했다는 신호를 보내줌.

3. Network_layer_sender

- transport layer로부터 data를 받고 datalink layer로 전송함.

- datalink로부터 ack를 받고 transport layer로 전송함.

4. Datalink_lyaer_sender

- network layer로부터 받은 data를 bit stuffing 한 후 CSMA/CD를 사용하여 physical layer로 전송함 (collision 발생확률을 0 으로 설정함).

```
#bit stuffing

stuffedData = bit_stuff(data)
print(stuffedData)
```

```
while Channelbusy == 0:
    print("The channel is busy ... ")  #1-Persistent Methods
    Channelbusy = random.randint(0,1)
    time.sleep(1)

print("The channel is idle, Station can transmit ... ")
time.sleep(1)
if collisionChance > random.randint(0, 10): #collision 이 발생할 확률
    statistics.append("Failed")
    print("A conflict has occurred ... ")
else:
    clientsocket.send(stuffedData.encode())
    print("Sent the data to physical layer.")
    success = True
    statistics.append("Successful")

if success == False:
    print("Send a jamming signal ...")
    time.sleep(1)
    k += 1
    if k < limit:
        Tbtime = Tb * 0.1 * chooseR(k)  # wait Tb time (maxtimum
propagation time * R)
        print("Waiting the TB timer to expire and to start a new
attemp ... : ",Tbtime)
        time.sleep(1 + Tbtime)

    else:
        print("The whole process was aborted. We need to try another
time.")
        abort = True
```

- physical layer로 부터받은 ack를 bit unstuffing 한 후 network layer로 보냄.

```
#bit unstuffing

stuffedAck = bit_unstuff(ack)

#send ack to datalink layer.

client_socket.send(ack.encode())
```

5. Physical_layer_sender

- datalink 로부터 받은 data를 MLT_3 schema로 multi transition 한 후 사용하여 receiver의 physical layer로 전송함

```
#Multi transition MLT 3 scheme

data = multi_transition(data)


#send data to physical layer of receiver.

clientsocket.send(data.encode())
```

- receiver의 physical layer로부터 받은 ack를 Reverse MLT3 scheme 하여 datalink layer로 보냄.

```
#Reverse MLT 3 scheme

ack = r_multi_transition(rack)

#send the ack to physical layer.

client_socket.send(ack.encode())
```

6. Physical_layer_receiver

- sender의 physical layer로부터 받은 data를 Reverse MLT3 scheme 하여 datalink layer로 보냄.

```
#Reverse MLT 3 scheme

data = r_multi_transition(data)
print(data)

#send data to datalink layer.

clientsocket.send(data.encode())
```

- datalink 로부터 받은 ack를 MLT_3 schema로 multi transition 한 후 사용하여 sender의 physical layer로 전송함

```
#Multi transition MLT 3 scheme

ack = multi_transition(ack)

#send the ack to physical layer.

client_socket.send(ack.encode())
```

7. Datalink_layer_receiver

- physical layer로 부터받은 data를 bit unstuffing 한 후 network layer로 보냄.

```
#bit unstuffing

stuffedData = bit_unstuff(data)

#send data to physical layer of receiver.

clientsocket.send(stuffedData.encode())
```

- network layer로부터 받은 ack를 bit stuffing 한 후 CSMA/CD를 사용하여 physical layer로 전송함 (collision 발생확률을 0 으로 설정함).

```
#bit stuffing

stuffedAck = bit_stuff(ack)
```

```
while Channelbusy == 0:
    print("The channel is busy … ")  #1-Persistent Methods
    Channelbusy = random.randint(0,1)
    time.sleep(1)

print("The channel is idle, Station can transmit … ")
time.sleep(1)

if collisionChance > random.randint(0, 10): #collision 이 발생할 확률
    statistics.append("Failed")
    print("A conflict has occurred … ")
else:
    client_socket.send(stuffedAck.encode())
    print("Current ack Status :", ack)
    print("Sent the ack to Physical layer.")
    success = True
    statistics.append("Successful")

if success == False:
    print("Send a jamming signal …")
    time.sleep(1)
    k += 1
    if k < limit:
        Tbtime = Tb * 0.1 * chooseR(k)  # wait Tb time (maxtimum
propagation time * R)
        print("Waiting the TB timer to expire and to start a new attemp
… : ",Tbtime)
        time.sleep(1 + Tbtime)

    else:
        print("The whole process was aborted. We need to try another
time.")
        abort = True
```

8. Network_layer_receiver

- datalink로부터 data를 받고 transport layer로 전송함.

- transport layer로부터 ack를 받고 datalink layer로 전송함.

9. Transport_layer_receiver

- network layer로부터 data를 수신하고 applicaition layer에 data전송함.

- ACK를 생성하여 Sender에게 전송하기 위해 network layer로 전송함

```
 ack = '010000010101001001001011' #ASCII code

random = randrange(0, 10)
if random < 10: #100% 확률로 ack를 보냄
    client_socket.send(ack.encode()) # send ack
    print("Current ack Status :", ack)
    print("Send a ack to Network layer.")
else:
    print("Failed to send ack ...")
```

10. Application_layer_receiver

- transport layer로부터 ack를 받고 송신메시지에 대한 수신확인 메시지를 출력함

```
data = client_socket.recv(1024).decode()

print("Received data from Transport layer.")
print("Received Data is :", data)
```

O How to run

1_application_layer_sender.py

2_transport_layer_sender.py

3_network_layer_sender.py

4_datalink_layer_sender.py

5_physical_layer_sender.py

6_physical_layer_receiver.py

7_datalink_layer_receiver.py

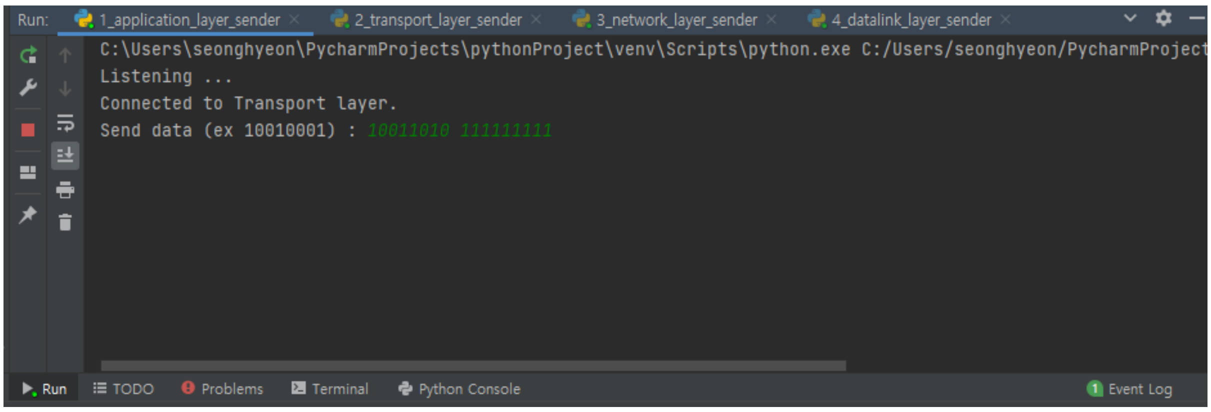
8_network_layer_receiver.py

9_transport_layer_receiver.py

10_application_layer_receiver.py

파일을 순서대로 실행하여 프로세스를 서로 connect 한 후

1_application_layer_sender.py 에서 data를 입력한다.



```
C:\Users\seonghyeon\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/seonghyeon/PycharmProject
Listening ...
Connected to Transport layer.
Send data (ex 10010001) : 10011010 111111111
```