

Assignment 4

2016311893 서진용

0. 기본 기능만 구현했습니다. Extension goal을 구현하지 못했습니다.

1. Development Environment

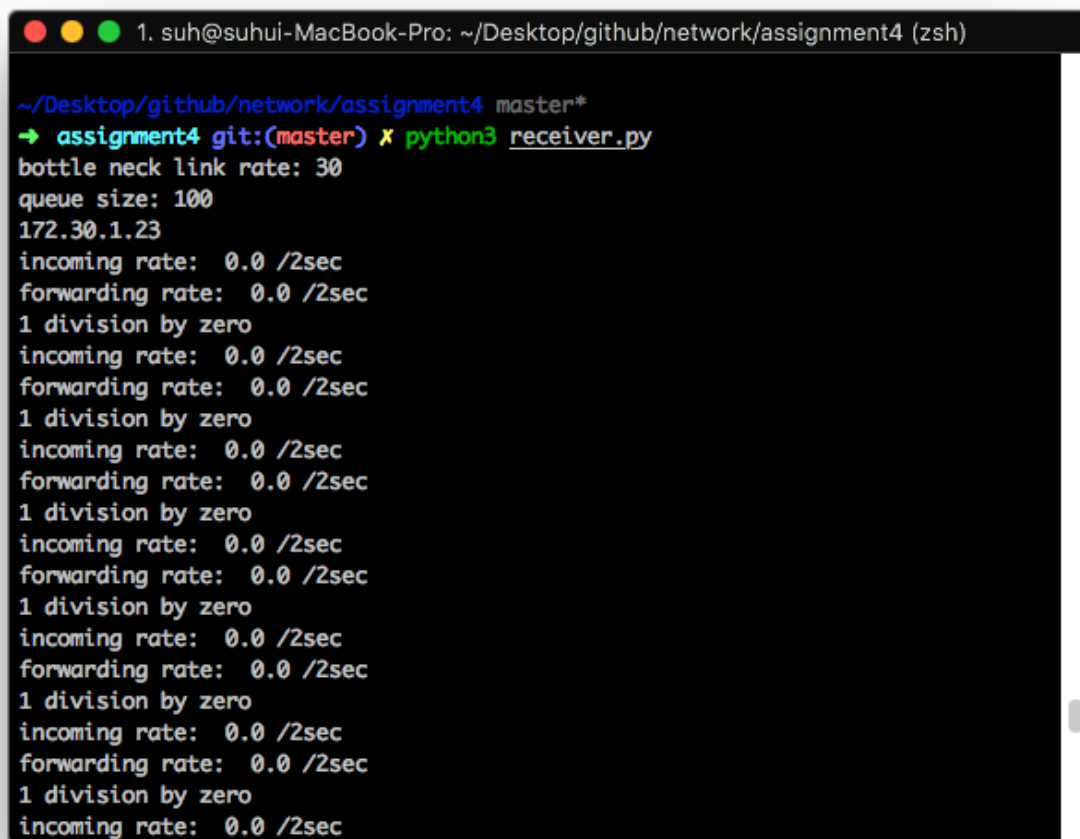
OS: Mac

Language : Python 3.6.2

IDE: Pycharm

2. How to run

(1) 먼저 receiver.py를 실행합니다.



```
1. suh@suhui-MacBook-Pro: ~/Desktop/github/network/assignment4 (zsh)

~/Desktop/github/network/assignment4 master*
→ assignment4 git:(master) x python3 receiver.py
bottle neck link rate: 30
queue size: 100
172.30.1.23
incoming rate: 0.0 /2sec
forwarding rate: 0.0 /2sec
1 division by zero
incoming rate: 0.0 /2sec
forwarding rate: 0.0 /2sec
1 division by zero
incoming rate: 0.0 /2sec
forwarding rate: 0.0 /2sec
1 division by zero
incoming rate: 0.0 /2sec
forwarding rate: 0.0 /2sec
1 division by zero
incoming rate: 0.0 /2sec
forwarding rate: 0.0 /2sec
1 division by zero
incoming rate: 0.0 /2sec
forwarding rate: 0.0 /2sec
1 division by zero
incoming rate: 0.0 /2sec
```

receiver.py만 실행했을 때 아직 sender로 부터 packet이 전송되지 않으므로, 모든 값이 0으로 출력됩니다.

(2) sender.py, sender2.py, sender3.py를 실행합니다.

```
2. suh@suhui-MacBook-Pro: ~/Desktop/github/network/assignment4 (git-remote-ht...
→ network git:(master) x cd assignment4

~/Desktop/github/network/assignment4 master*
→ assignment4 git:(master) x ls
receiver.py sender.py sender2.py test.py

~/Desktop/github/network/assignment4 master*
→ assignment4 git:(master) x python3 sender.py
enter receiver ip: 172.30.1.23
enter initial sending rate: 30
sending rate: 29.5 /2sec
goodput: 29.0 /2sec
goodput ratio: 0.9830508474576272
sending rate: 30.5 /2sec
goodput: 30.5 /2sec
goodput ratio: 1.0
sending rate: 31.5 /2sec
goodput: 31.0 /2sec
goodput ratio: 0.9841269841269841
sending rate: 31.0 /2sec
goodput: 31.0 /2sec
goodput ratio: 1.0
sending rate: 31.0 /2sec
goodput: 31.0 /2sec
goodput ratio: 1.0
```

값이 출력되는 것을 확인할 수 있습니다.

3.Implement

receiver.py

(1) 기본 변수 설정

```
#bottlenec rate와 queue size를 입력받습니다.
botRate = int(input("bottle neck link rate: "))
qSize = int(input("queue size: "))

#socket을 생성합니다.
receiver = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
receiver.bind((socket.gethostbyname(socket.gethostname()), 10080))
```

(2) network emulator 모듈과 forwarding하는 함수

```
#network emulator 모듈에서 bottleneck 실행 후 Ack generator 모듈로 forwarding하는 함수 입니다.
def ne2ag(sock, addr, botRate):
    global run, forPck, go
    #sender에 해당하는 queue를 받아옵니다.
    neQueue = qList.get(addr[1])
    _go = go.get(addr[1])
    #forwarding 후 sender에게 ack을 보냅니다.
    if _go:
        neQueue.get()
        forPck += 1
        sock.sendto('ack'.encode('utf-8'), addr)

    timer = threading.Timer(1/botRate, ne2ag, args=[sock, addr, botRate])
    if run.get(addr[1]):
        timer.start()
```

(3) packet을 queue에 추가하는 함수

```
#network emulator에서 packet을 queue에 추가하는 함수입니다.
def ntwrkEmul(qSize, data, addr):

    neQueue = qList.get(addr[1])
    #queue가 다 찼으면 drop 시킵니다.
    if neQueue.qsize() < qSize:
        neQueue.put(data.decode())
    else:
        pass
```

(4) 2초마다 출력하는 메시지 함수

```
#2초마다 출력하는 메시지 함수입니다.
def twoSecMsg():
    global run, qSize
    global inPck, forPck, occList

    try:
        print("incoming rate: ", inPck/2, "/2sec")
        print("forwarding rate: ", forPck/2, "/2sec")
        print("avg queue occupancy: ", (getAvgOcc()/len(addrList)/20)/qSize)
        inPck = 0
        forPck = 0
    except Exception as e:
        print(e)

    timer = threading.Timer(2.1, twoSecMsg)
    if 1:
        timer.start()
```

(5) occupancy를 0.1초마다 추가

```
#각 sender마다 queue에서의 occupancy의 값을 0.1초마다 추가합니다.  
def avgOccupancy(addr):  
    global occList, qList, run  
    occ = qList.get(addr[1]).qsize()  
    occList[addr[1]] = occList.get(addr[1]) + occ  
    t = threading.Timer(0.1, avgOccupancy, args=[addr])  
    if run.get(addr[1]):  
        t.start()
```

(6) 평균 occupancy를 구하는 함수

```
#queue의 평균 occupancy값을 구하는 함수입니다.  
def getAvgOcc():  
    global occList, addrList  
    i = 0  
    sum = 0  
    try:  
        while addrList[i]:  
            temp = occList.get(addrList[i])  
            sum = sum + temp  
            occList[addrList[i]] = 0  
            i += 1  
    except Exception as e:  
        pass  
    return sum
```

(7) packet을 받을 때의 함수

```
#packet을 받을 때의 함수입니다.
def rcvMsg(sock):
    global inPck, qSize, index, qList, go, run, addrList
    while True:
        try:
            data, addr = sock.recvfrom(1024)

            #sender가 처음 접속시 enter 메시지를 받고, sender를 리스트에 추가합니다.
            #sender마다 queue를 생성합니다.
            if data.decode() == 'enter':
                addrList.append(addr[1])
                run[addr[1]] = 1
                occList[addr[1]] = 0
                go[addr[1]] = 0
                neQueue = Queue()
                qList[addr[1]] = neQueue
                ne2ag(sock, addr, botRate)
                avgOccupancy(addr)
                go[addr[1]] = 1

            #처음 접속이후 packet을 받을 때마다 incoming packet을 증가시키고 ntwrkEmul함수를 실행합니다.
            elif data:
                inPck += 1
                ntwrkEmul(qSize, data, addr)
        except:
            continue
```

sender.py

(1) 기본 변수 설정

```
#receiver의 ip와 init send rate를 입력받습니다.
ip = input("enter receiver ip: ")
initSendRate = int(input("enter initial sending rate: "))

#socket을 생성합니다.
sender = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sender.bind((socket.gethostbyname(socket.gethostname()), 0))
```

(2) ackcount를 계산하는 함수

```
#1000byte 짜리 text를 생성합니다.
text = '11111'
while j<199:
    text = '11111' + text
    j += 1

#receiver로 부터 ack을 받을 때마다 ackCount를 증가시킵니다.
def rcvMsg(sock):
    global ackCount
    while True:
        data, addr = sock.recvfrom(1024)
        if data:
            ackCount += 1
```

(3) 2초마다 출력되는 메시지 함수

```
#2초마다 출력되는 메시지 함수입니다.
def twoSecMsg():
    global once
    if once:
        time.sleep(2)
        once = 0
    global pckCount, ackCount

    print("sending rate: ", pckCount/2, "/2sec")
    print("goodput: ", ackCount/2, "/2sec")
    print("goodput ratio: ", ackCount/pckCount)
    pckCount = 0
    ackCount = 0
    timer = threading.Timer(2.1, twoSecMsg)
    if 1:
        timer.start()
```

(4) packet을 전송하는 함수

```
#receiver에게 packet을 전송하는 함수입니다.  
def sendMsg(sock):  
    global allPck, pckCount, text  
    sender.sendto(text.encode('utf-8'), (ip, 10080))  
    #packet을 전송할 때마다 pckCount를 증가시킵니다.  
    pckCount += 1  
    timer = threading.Timer(1/initSendRate, sendMsg, args=[sock])  
    if 1:  
        timer.start()
```