2016311739 김연재

Report

1. Development Environment

Operating System: window 10

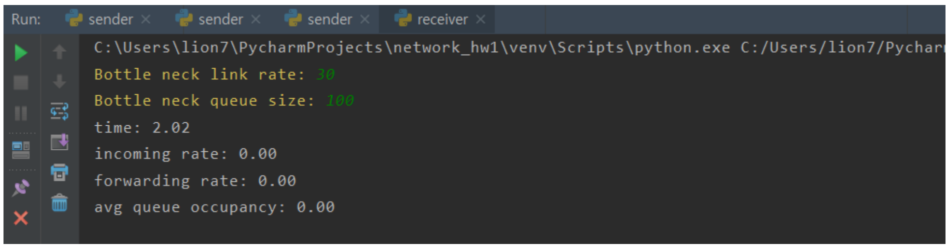
Language: python

Compilers: python 3.6

2. Run sender and receiver program including screen shot

2-1. run receiver.py in pycharm with python 3.6 interpreter

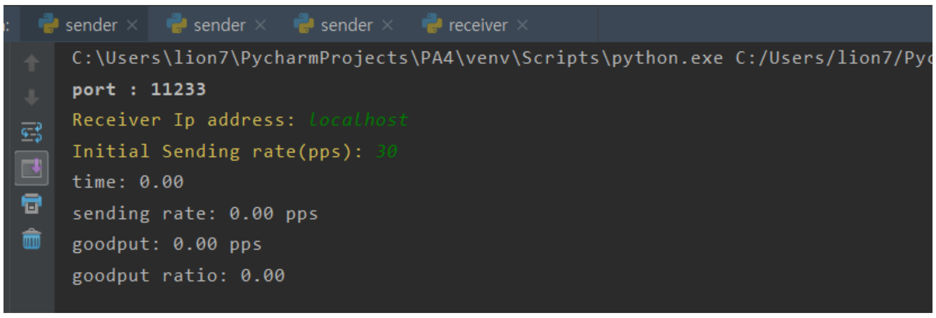
2-2. enter 'bottle neck link rate' and 'bottle neck queue size'



```
C:\Users\lion7\PycharmProjects\network_hw1\venv\Scripts\python.exe C:/Users/lion7/Pychar
Bottle neck link rate: 30
Bottle neck queue size: 100
time: 2.02
incoming rate: 0.00
forwarding rate: 0.00
avg queue occupancy: 0.00
```

2-3. run sender.py in pycharm with python 3.6 interpreter

2-4. enter 'receiver ip address' and 'initial sending rate'



```
C:\Users\lion7\PycharmProjects\PA4\venv\Scripts\python.exe C:/Users/lion7/Py
port : 11233
Receiver Ip address: localhost
Initial Sending rate(pps): 30
time: 0.00
sending rate: 0.00 pps
goodput: 0.00 pps
goodput ratio: 0.00
```

2-5. see what sender or receiver does

2-6. after execution, csv file would be created. It would contains goodput and goodput ratio tendency(or forwarding rate and queue occupancy tendency) as times go on.

3. how to design

3-1. Sender

- main thread: send packet every {1 / sending rate} seconds
- time stamp thread: print status every 2 seconds
- receive ack thread: receive ack, and adjust sending rate

3-2. Receiver

- main thread(NE): receive packet and store it into bottleneck queue. If queue is full, send nak.
- time stamp thread: print status every 2 seconds
- ack generate thread(AG): pop packet, and send ack to certain sender

3-3. Sending rate control

- receive ack:

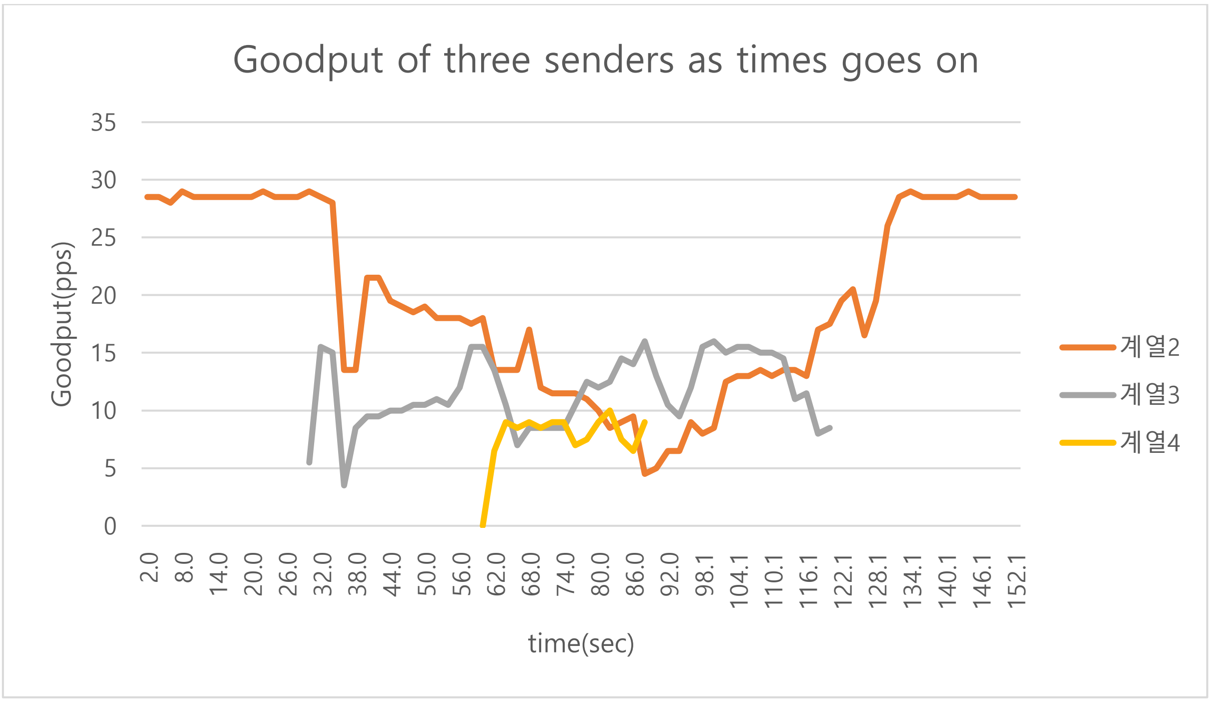
  sending_rate = sending_rate + 1 / sending_rate * 1 / 2

- receive nak:

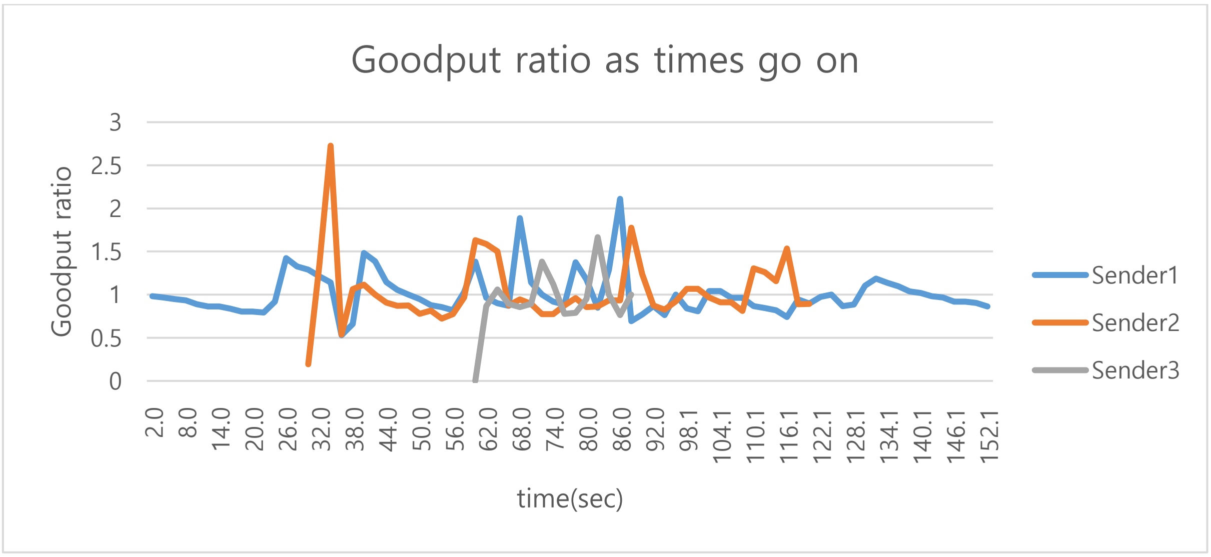
  sending_rate = (sending_rate - 3 / sending_rate) * 1 / 2

4. show graph(given scenario)

4-1. goodput of the three senders as times go on

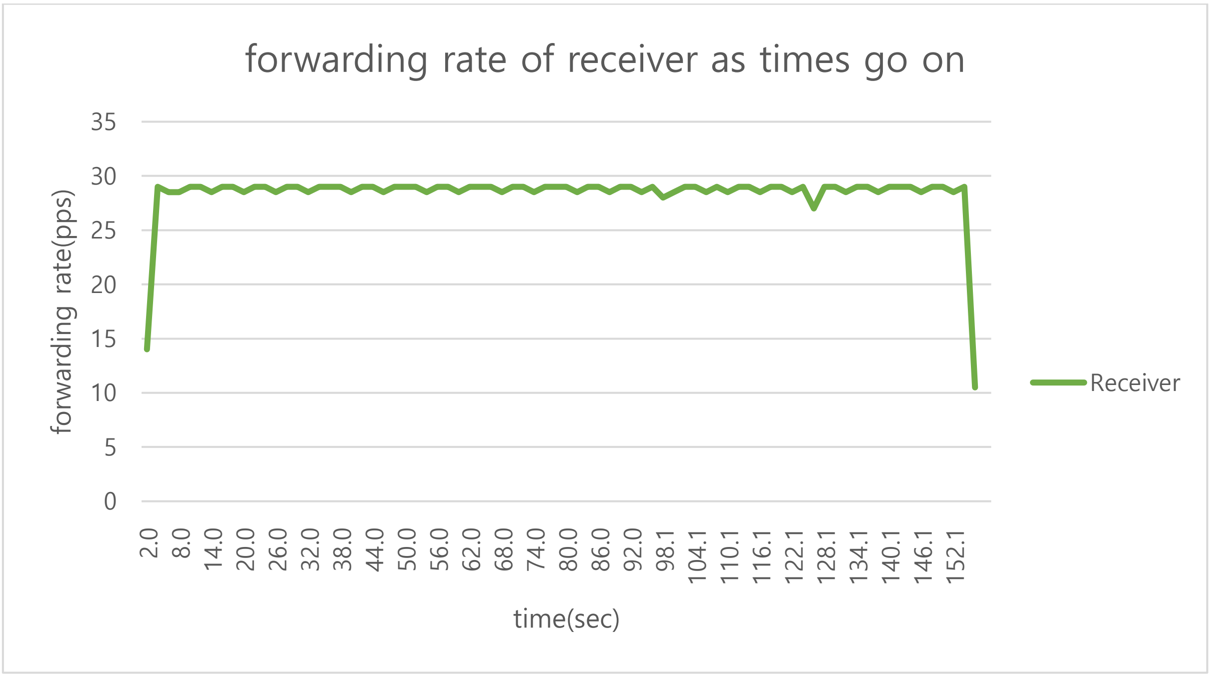


4-2. goodput ratio of the three senders as times go on

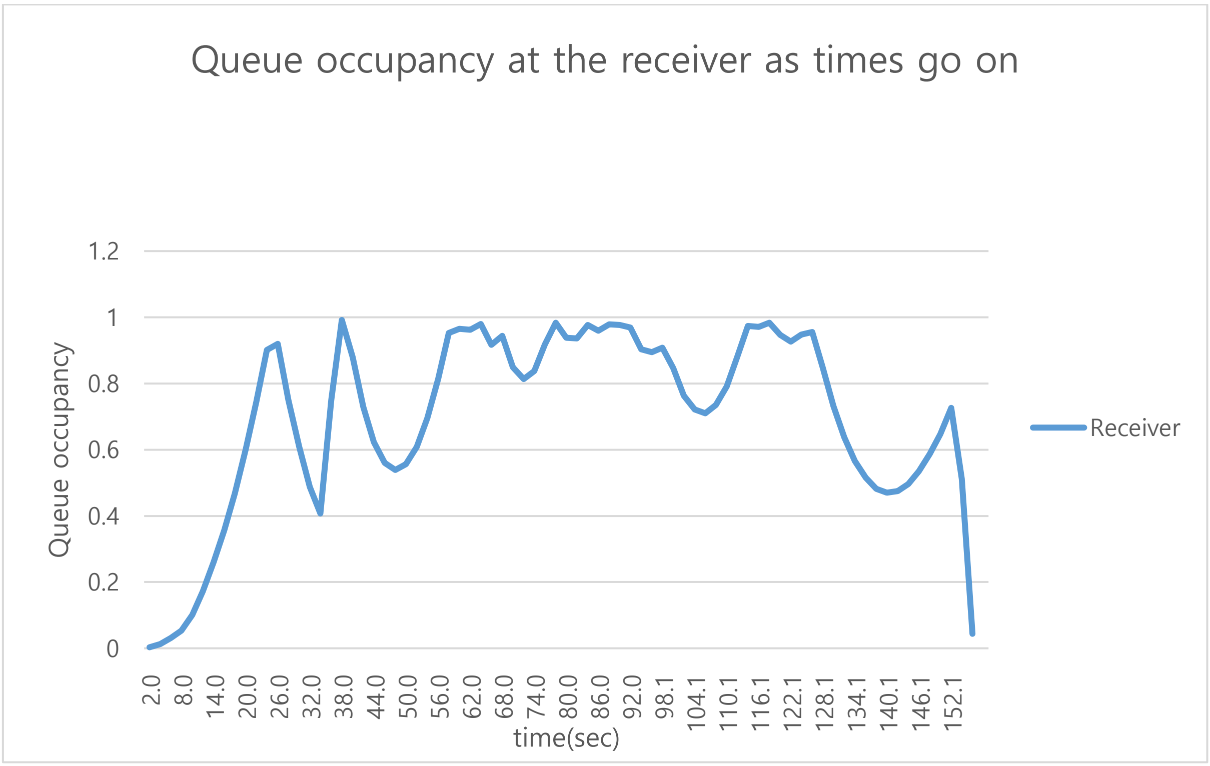


2016311739 김연재

4-3. forwarding rate at the receiver as times go on



4-4. queue occupancy at the receiver as times go on



2016311739 김연재

5. Extension goal

- goodput ratio (goodput / sending rate) gets close to 1 (5 pts)

As you can see in 4-2, all of goodput ratio gets close to 1.

- queue occupancy gets as low as possible (5 pts)

As you can see in 4-3, it tries to lower queue occupancy as possible.

- multiple senders have fair bandwidth (5 pts)

As you can see in 4-1, all of senders have similar goodput.

- while maximizing the utilization of the bottleneck link (5 pts)

As you can see in 4-4, utilization ratio of the bottleneck link is close to 1.

- If you design the extension goals in a stateless fashion at NE, you will get additional 10 points. (10 pts)

When sender controls sending rate, it uses just sending rate to adjust itself. It does not store any information about connection with receiver. I thought I designed in a stateless fashion.