2016314827 Kyuyeon Kim

Assignment 4 – Code Document

1. Development Environment

· Operating System: **Windows 10 Pro**

· Language: **Python 3.6.5**

· Editor: Pycharm

2. How to run the rode

· This program is recommended to be executed in **Windows 10, with Pycharm IDE**.

· Please execute and ready the **receiver** program first, and then the **sender** program.

3. Code explanation

· There are **3 threads** in **receiver program**: status printing thread, ACK sending thread, and the main thread which receives packets from the multiple senders.

· If receiver gets the packet but the bottleneck queue is full, it immediately sends **queue full message** to the sender. Otherwise, if the bottleneck queue is not full yet, it enqueues the sender's address into the queue.

· In ACK sending thread, it constantly dequeues the element from the bottleneck queue regarding the **bottleneck link rate**. With the dequeued element, which is the address of the sender, receiver simply sends the ACK message to the sender.

· Status printing thread calculates **average queue occupancy** for each 0.1 second. After calculating 20 times, when 2 seconds passed, it shows the status of receiver on the console.

· There are **3 threads** in **sender program**: status printing thread, ACK receiving thread, and the main thread which sends packets to the receiver.

· The sender constantly sends 1,000-byte packet to the receiver according to its sending rate.

· In ACK receiving thread, it gets the message from the receiver. The message could be either **queue full message** that receiver feedbacks, or **ACK message** that receiver successfully gets the packet.

· In ACK receiving thread, if the message is **queue full feedback**, sender shrink the sending rate to the **60%.**

· Else if the message is **ACK**, sender updates its sending rate by:

$$sending\_rate = sending\_rate + \frac{1}{sending\_rate^{3/2}}$$

· The implementation is **stateless**: it does not sends or receives messages about flow.

4. Screenshots of the test

A. Initial sending rate: 30, Bottleneck link rate: 30, Queue size: 100 (1 sender, 1 receiver)

```
Receiver IP address: localhost
Initial sending rate (pps): 30

Sending rate: 28.50 pps
Goodput: 28.00 aps
Goodput ratio: 0.98

Sending rate: 29.50 pps
Goodput: 28.50 aps
Goodput ratio: 0.97

Sending rate: 29.50 pps
Goodput: 28.50 aps
Goodput ratio: 0.97

Sending rate: 27.50 pps
Goodput: 27.00 aps
Goodput ratio: 0.98
```

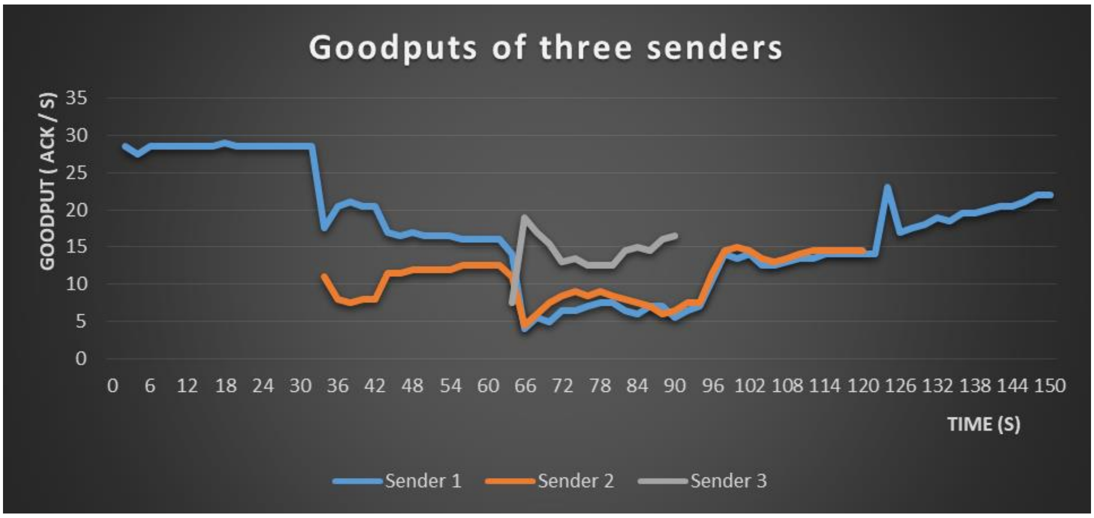
```
Incoming rate: 29.50 pps
Forwarding rate: 28.50 pps
Average queue occupancy: 2.85 %

Incoming rate: 28.50 pps
Forwarding rate: 28.00 pps
Average queue occupancy: 4.60 %

Incoming rate: 30.00 pps
Forwarding rate: 28.50 pps
Average queue occupancy: 6.40 %

Incoming rate: 30.00 pps
Forwarding rate: 28.50 pps
Average queue occupancy: 9.10 %

Incoming rate: 30.00 pps
Forwarding rate: 28.50 pps
Average queue occupancy: 12.50 %
```

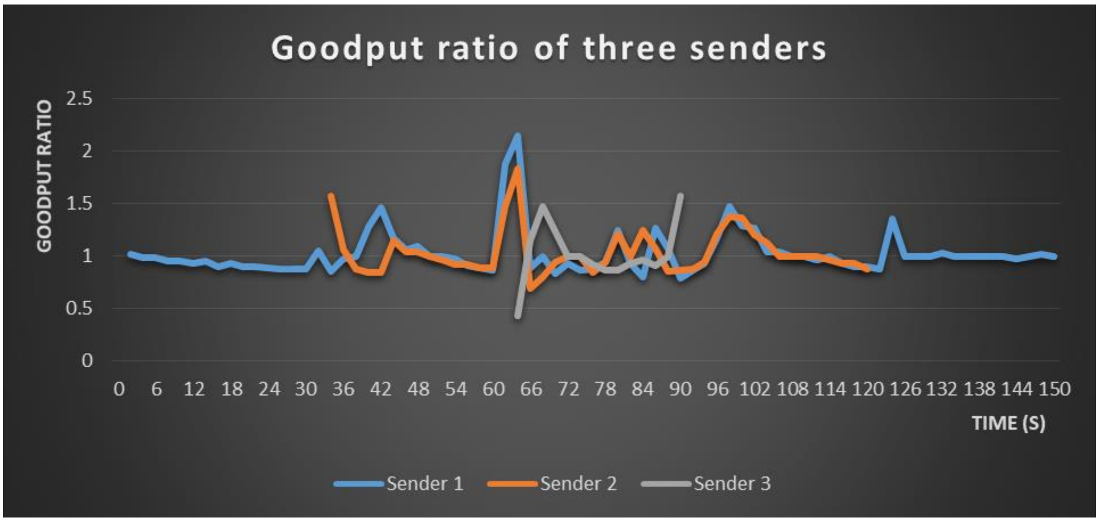
B. Initial sending rate: 30, Bottleneck link rate: 30, Queue size: 100 (3 senders, 1 receiver)

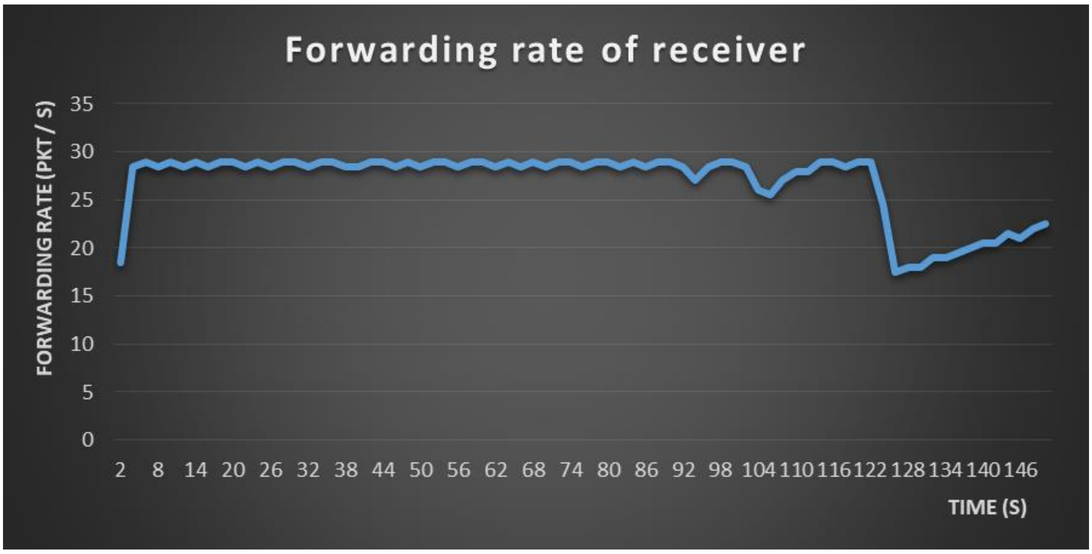
```
Sending rate: 11.50 pps
Goodput: 11.00 aps
Goodput ratio: 0.96

Sending rate: 12.00 pps
Goodput: 11.50 aps
Goodput ratio: 0.96

Sending rate: 9.00 pps
Goodput: 11.50 aps
Goodput ratio: 1.28

Sending rate: 7.50 pps
Goodput: 6.00 aps
Goodput ratio: 0.80
```

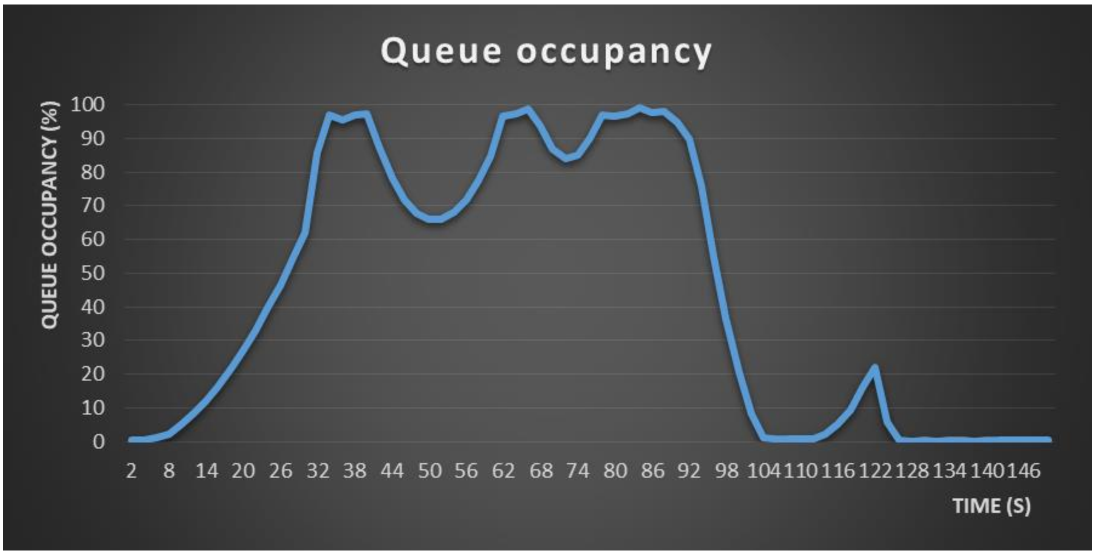
```
Incoming rate: 23.50 pps
Forwarding rate: 28.50 pps
Average queue occupancy: 87.50 %

Incoming rate: 25.50 pps
Forwarding rate: 29.00 pps
Average queue occupancy: 78.60 %

Incoming rate: 26.00 pps
Forwarding rate: 28.50 pps
Average queue occupancy: 71.85 %

Incoming rate: 27.50 pps
Forwarding rate: 29.00 pps
Average queue occupancy: 67.95 %
```

5. Graphs for extension goals









· You can see the data of graphs in each Excel file.

· On the first graph, all senders, especially sender 1 and sender 2 are having almost same bandwidth, which fits to the goal: multiple senders have fair bandwidth.

· On the second graph, the goodput ratio of three senders are always around 1.0, which fits to the goal: gootdput ratio gets close to 1.

· On the third graph, the forwarding rate almost gets close to 30 packets/s, which fits to the goal: maximize the utilization of the bottleneck link.

· On the last graph, the queue occupancy between 30~60 seconds time interval is about 60%~100%, and for 60~90 seconds time interval, is about 80%~100%. From 90 seconds point, the point when sender 2 leaves, the queue occupancy severely decreases to 0, and then goes around 0% ~ 25%.

· The emulator and sending rate calculation algorithm is implemented in stateless fashion: no such information is embedded in the packet between the sender and the receiver