Introduction to Computer Networks

Assignment 5: Network Emulation & Congestion Control

1. Goal

- Provide an network emulation (bandwidth & latency) at a receiver program using UDP
- Develop a congestion control protocol for a single / multiple sender(s) using UDP.

2. Development environments

- You can use C/C++ (Visual Studio 2015 or above version) or Python (3.x) language on Windows or latest GCC versions on Linux systems
- You have to describe your development environment information in detail in the report.

3. Assignment description

- Terminology
  NEM : Network Emulator Module at a receiver program
  RM : Receiver Module at a receiver program
  BLR : Bottleneck Link Rate of NEM
  FR : Forwarding Rate of NEM
  SR : Sending Rate of a sender program
  G : Goodput of a sender program
- Sender
  - When starting a sender program, enter the IP address of a receiver.
    (the port number of a receiver is 10080.)
  - Allow a user to enter "start initial_window_size "
    e.g. command>> **start 1** or command>> **start 10**
    - Start a Congestion Avoidance mode without Slow Start
    - Can use any congestion control algorithm
      Additive Increase and Multiplicative Decrease (AIMD), CUBIC, BBR etc.
    - Each packet size is 1400 bytes which consists of any data.
      (NOT necessary to transfer a real file data)
    - Calculate an avg RTT and timeout, and detect 3 duplicated ACKs.
    - Reduce a window size only once for multiple drops within the same congestion event.

(e.g. multiple packet drops within a single RTT are considered as one congestion event)

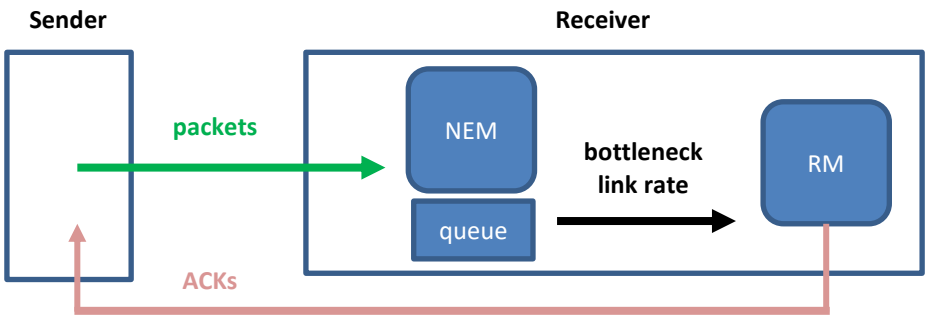
- Every two seconds, store the following information in *PortNumber*_log.txt
  (*PortNumber* can be obtained by `getsockaname()`)

  e.g. time | avg_RTT | SR | G

  - SR (Sending rate) is calculated by the number of packets sent / 2 seconds
  - G (Goodput) is calculated by the number of ACKs received / 2 seconds

- While the sender is on the sending operation, allow a user to enter "stop"
  e.g. command>> **stop**
  - Stop sending packets (if you need, notify the receiver).
  - Exit the sender program.
- TAs can run multiple sender programs concurrently for testing on the same machine.

- Receiver
  - Run a receiver program and sender programs on different computers.
  - Bind a socket with the 10080 port number.
  - Design a receiver program which consists of two modules
    - One is a Network Emulator Module, **NEM**
    - Another is a Receiver Module, **RM**, which is the almost same to the Assignment #4.



- When starting a receiver, enter **BLR** and **queue_size**
  e.g. configure>> **100** **10**
  - 100 of **BLR** means that forwarding 100 packets per second from **NEM** to **RM**.
  - 10 of **queue size** means that up to 10 packets can be stored in **NEM** before forwarding to **RM**.

- **NEM** acts as a bottleneck link.
  - Forward all incoming packets to **RM** but limits the rate to **BLR**.
  - If the incoming rate is higher than **BLR**, incoming packets should be store in the bottleneck queue.
  - **NEM** can perform queue management, such as a packet drop policy.
    - You can design any queue management such as a drop tail policy or random early notification etc.
  - Every two seconds, store the following information in NEM.log
    e.g. time | incoming_rate | forwarding_rate | avg_queue_utilization
    - incoming rate: (#packets arrived at **NEM** from **Sender**) / 2 seconds
    - forwarding rate: (#packets forwarded from **NEM** to **RM**) / 2 seconds
    - avg_queue_utilization: measure the queue utilization every 100ms, and get the average for 2 seconds.
- **RM** performs the basic functions of a receiver.
  - When packets arrive, RM responds ACKs by a TCP behavior.
  - Every two seconds, store the following information in RM.log
    time | jain_fairness_index
    sender_ip:port_number1 | receiving_rate
    sender_ip:port_number1 | receiving_rate
    …
    sender_ip:port_numberN | receiving_rate
    - Jain Fairness Index: https://en.wikipedia.org/wiki/Fairness_measure
    - receiving_rate: (#packets received from sender_IP:port_number) / 2 seconds
- **NEM** should be stateless (do NOT maintain all the status information of individual flows).
- **RM** is only allowed to copy the header information of data packets to the header of ACK packets.
- Allow running concurrent multiple senders but a single receiver.

- Miscellaneous
  - In log files, display sentences with proper alignment to improve readability
  - The socket receive buffer size should be large enough (10 Mbytes) otherwise the UDP packets can be dropped before the receiver program reads.
  - In this assignment, we will NOT evaluate the reliable packet retransmission.
  - You don't need to transfer actual files.

4. Experimentation

- Scenario1 (**BLR**: 500, queue_size: 50, and initial_window_size : 5)
  - start sender1 & sender2 at time 0 sec
  - start sender3 & sender4 at time 30 secs
  - start sender5 & sender6 at time 60 secs
  - stop sender5 & sender6 at time 90 secs
  - stop sender3 & sender4 at time 120 secs
  - stop sender1 & sender2 at time 150 secs
  - Show 5 graphs;
    - **G** of six senders as time goes on
    - **FR** as time goes on
    - queue utilization as time goes on
    - Jain_Fairness_Index for current sending flows as time goes on

5. Submission

- The deadline is **12.11 (Tue) 23:59**.
  - For delayed submissions, a penalty of -15 points applies every 24 hours. After 72 hours, you get zero points.
  - In the case of plagiarism, you will receive 0 points for the first time and ***F*** for the second.
- Submit a zip file including a **report** and two (sender and receiver) program sources to iCampus
  - The report file format should be PDF.
  - Name the Report file as follows ***StudentID_Name.pdf*** (ex: 2018001_홍길동.pdf)
  - The report have to include the following things;
    1) Describe your development environment information in detail
       (versions of operating systems, languages, compilers/interpreter versions, compile options)
    2) Present how to design your assignment such as data structures and algorithms.
    3) Explain how to run both sender and receiver programs including the screen capture.
    4) **Show 4 graphs** for the experimentation results.

5. Scoring (Total 100 points)

- Terminology
  NEM : Network Emulator Module at a receiver program
  RM : Receiver Module at a receiver program
  BLR : Bottleneck Link Rate of NEM
  FR : Forwarding Rate of NEM
  SR : Sending Rate of a sender program
  G : Goodput of a sender program

- In the case of a single sender:
  - 20 points: **NEM** limits **FR** as **BLR**
    Keep | 1 – **FR** / **BLR** | below 0.1 (10%)
  - 10 points: Keep | 1 – **G** / **BLR** | below 0.1 (10%)
  - 10 points: properly well-written log files.

- In the case of multiple senders:
  - 10 points: Allow multiple senders fully utilize **BLR**
    Keep | 1 – **FR** / **BLR** | below 0.1 (10%)
  - 20 points: Jain Fairness Index gets close to 1.0
    https://en.wikipedia.org/wiki/Fairness_measure

- With that **BLR** is utilized over 90%:
  - 10 points: Bottleneck queue utilization is kept low (less than 30%)

- 20 points: Report
  - 10 points for the basic documentation.
  - 10 points for the graphs of the experiment.

6. Q&A

- Leave your questions on the google sheet