# Introduction to Computer Networks

## Assignment 4: Pipelined Reliable Data Transfer over UDP

### 1. Goal

- Develop a pipelined reliable data transfer protocol using UDP socket programming.
- Generate packet losses at a receiver program

### 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. Functionalities to implement

- Sender
  - When starting a sender program, enter the IP address of a receiver, timeout, and window size. (the port number of a receiver is 10080.)
  - After running the program, allow a user to enter the name of a file to send to the receiver.
    - The files must be in the same directory where the sender program is located.
    - Send the file name and the file itself to the receiver.
    - Divide the file into packets that are 1400 bytes or less in size.
    - Transmit packets as much as the given window size in a TCP fashion
    - During file transfer, the user can enter a new file name to send the file.
  - By receiving ACKs, detect and recover dropped packets.
    - Use a single timer to check "timeout" of packets.
    - Use "3 duplicated ACKs" for Fast Retransmission.
  - Per-file transfer, write a log file
    - E.g. "fileAAA_sending_log.txt"
    - Write packet & ACK event information with the logging time as shown in Figure 2-1.

        - If the file transfer is finished, display the throughput (goodput) in the log file :
          packets / sec ( the number of unique packets / the total transfer time ).

- Receiver
    - Run a receiver program and a sender program on different computers.
    - When starting a receiver, ask a user "packet loss probability". ( e.g. 0 ~ 1.0 )
    - Bind a socket with the 10080 port number.
    - Display "socket buffer size". If the size is less than 10 MB, set it to 10 MB (or as much as your system allows)
    - If the packet loss probability is greater than zero, drop incoming packets according to the probability. ex) if the packet loss probability is 0.1, drop packets with a 10% probability.
    - If a packet is successfully received, send a cumulative ACK.
        - Store received in-order packets in a file.
        - Store received out-of-order packets in temporary buffer.
    - Per-file transfer, write a log file
        - E.g. "fileAAA_receiving_log.txt"
        - Write packet & ACK event information with the logging time as shown in Figure 1-1.
        - If the file transfer is finished, display the throughput (goodput) in the log file :
          packets / sec ( the number of unique packets / the total transfer time ).
    - Allow concurrent file transfers.

- Miscellaneous
    - Time information starts from 0 seconds for each file transfer.
    - The sequence number is started from 0 for each file transfer, and increases one per packet.
    - Set the ACK number equal to the packet sequence number.
    - In the 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 a sender program, you may use a thread for receiving ACKs. If the receiving thread is started before the sending thread is started, you can see a socket related error message. You have to call bind( ) with the port number of zero (to request any unused source port number)
    - To allow concurrent file transfers, per-flow management is required.

## 4. Experimentation

- For **one-to-one** file transfer, make following experimentations.
- 1) Draw a goodput graph with different probabilities of packet loss.  
  - the window size is fixed to 8 and timeout is 0.05 seconds  
  - the loss probability is changed to 0.02, 0.04, 0.08, 0.16.
- 2) Draw a goodput graph with different window sizes.  
  - the loss probability is fixed to 0.02 and timeout is 0.05 seconds  
  - Change the window size to 2, 4, 8, 16
- Testing file size should be large enough ( **more than 10 Mbytes** )

## 4. Submission

- The deadline is **11.20 (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: 2017001_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 two graphs for the experimentation results.

## 5. 채점 기준

- Total 100 points
  - 20 points: the sender transmits packets as much as the window size, and transmits the next packet whenever receiving ACK. (Figure 1 & Figure 2)
    - assume no packet loss
    - display packet & ACK event information in log files
  - 10 points: Calculate goodput after the file transfer is finished.
    - display goodput in log files at the sender and the receiver.
  - 10 points: Provide packet drops according the given probability.
    - display packet drop events in the log file at the receiver.

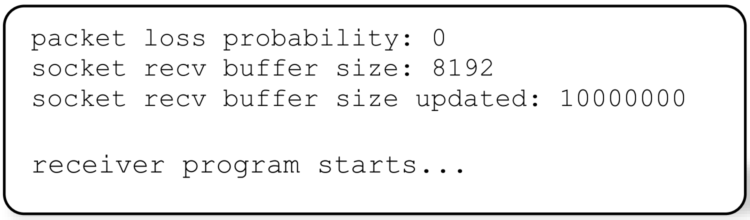
      example)
      ```
      0.005 pkt: 3  | received
      0.005 pkt: 3  | dropped
      ```
  - 20 points: Detect a timeout of a packet and retransmit the packet.
    - display timeout events in the log file at the sender.

      example)
      ```
      1.063 pkt: 8  |  timeout since 1.010
      1.063 pkt: 8  |  retransmitted
      ```
  - 10 points: Detect 3 duplicated ACKs and packet retransmissions at a sender.
    - display the events in the log file at the sender.

      ```
      1.063 ack: 5  |  received
      1.064 ack: 5  |  received
      1.065 ack: 5  |  received
      1.066 ack: 5  |  received
      1.066 pkt: 5  |  3 duplicated ACKs
      1.066 pkt: 6  |  sent
      ```
  - 10 points: Allow concurrent file transfers.
    - Whenever entering a new file name at the sender program, start sending the new file while processing existing file transfers.
  - 20 points: Report
    - 10 points for the basic documentation.
    - 10 points for the graphs of two experiments.

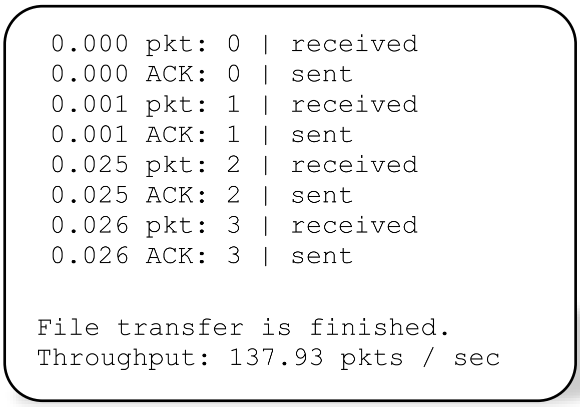
## 6. Q&A

- Leave your questions on the google sheet



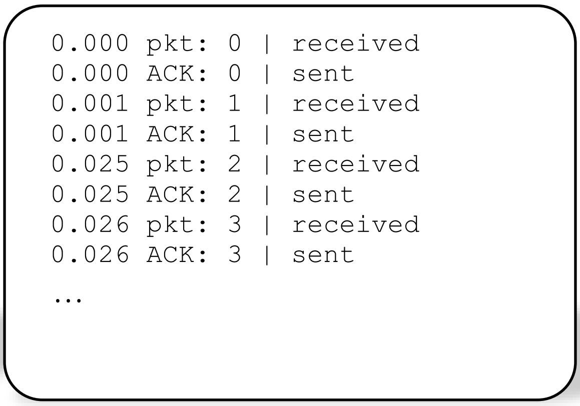
```
packet loss probability: 0
socket recv buffer size: 8192
socket recv buffer size updated: 10000000

receiver program starts...
```

**Figure 1. Basic receiver operation**



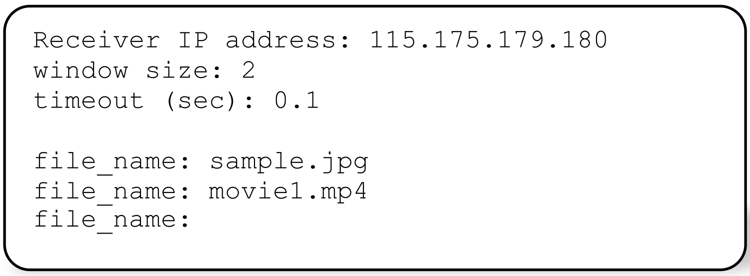
```
 0.000 pkt: 0 | received
 0.000 ACK: 0 | sent
 0.001 pkt: 1 | received
 0.001 ACK: 1 | sent
 0.025 pkt: 2 | received
 0.025 ACK: 2 | sent
 0.026 pkt: 3 | received
 0.026 ACK: 3 | sent


File transfer is finished.
Throughput: 137.93 pkts / sec
```

**Figure 1-1. "sample.jpg_receiving_log.txt"**



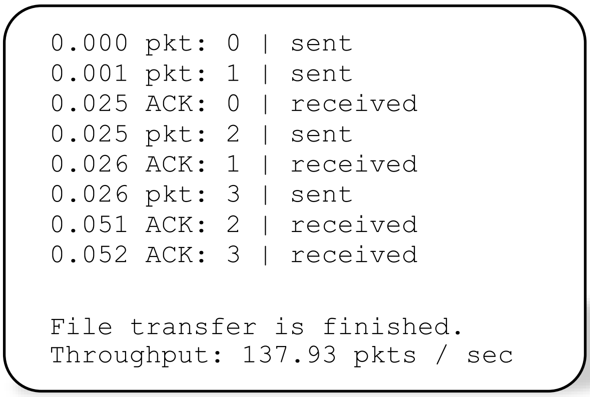
```
0.000 pkt: 0 | received
0.000 ACK: 0 | sent
0.001 pkt: 1 | received
0.001 ACK: 1 | sent
0.025 pkt: 2 | received
0.025 ACK: 2 | sent
0.026 pkt: 3 | received
0.026 ACK: 3 | sent
...
```

**Figure 1-1. "sample.jpg_receiving_log.txt"**



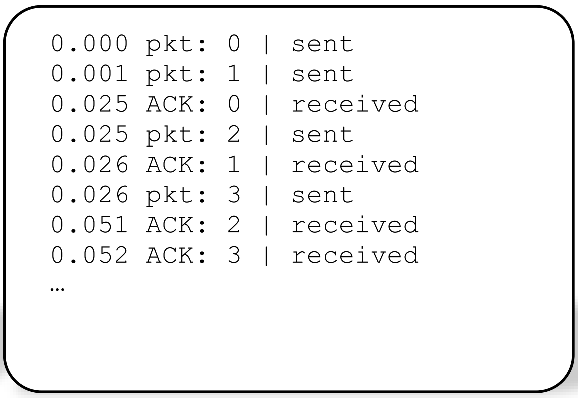
```
Receiver IP address: 115.175.179.180
window size: 2
timeout (sec): 0.1

file_name: sample.jpg
file_name: movie1.mp4
file_name:
```

**Figure 2. Basic sender operation**



```
0.000 pkt: 0 | sent
0.001 pkt: 1 | sent
0.025 ACK: 0 | received
0.025 pkt: 2 | sent
0.026 ACK: 1 | received
0.026 pkt: 3 | sent
0.051 ACK: 2 | received
0.052 ACK: 3 | received


File transfer is finished.
Throughput: 137.93 pkts / sec
```

**Figure 2-1. "sample.jpg_sending_log.txt"**



```
0.000 pkt: 0 | sent
0.001 pkt: 1 | sent
0.025 ACK: 0 | received
0.025 pkt: 2 | sent
0.026 ACK: 1 | received
0.026 pkt: 3 | sent
0.051 ACK: 2 | received
0.052 ACK: 3 | received
…
```

**Figure 2-1. "movie1.mp4_sending_log.txt"**