國立臺灣大學

Assignment 3

Retransmission & Congestion Control

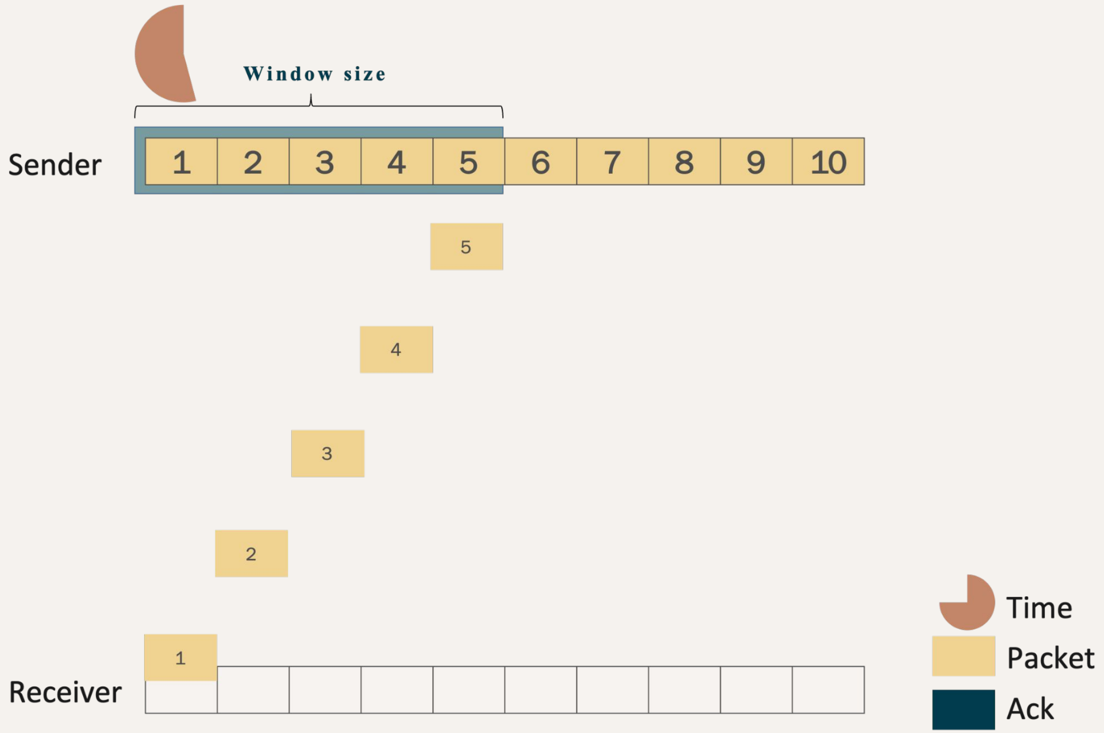
Prof. Ai-Chun Pang
TA / Chun-Yu Lee, Wan-Chu Hsu

Goals

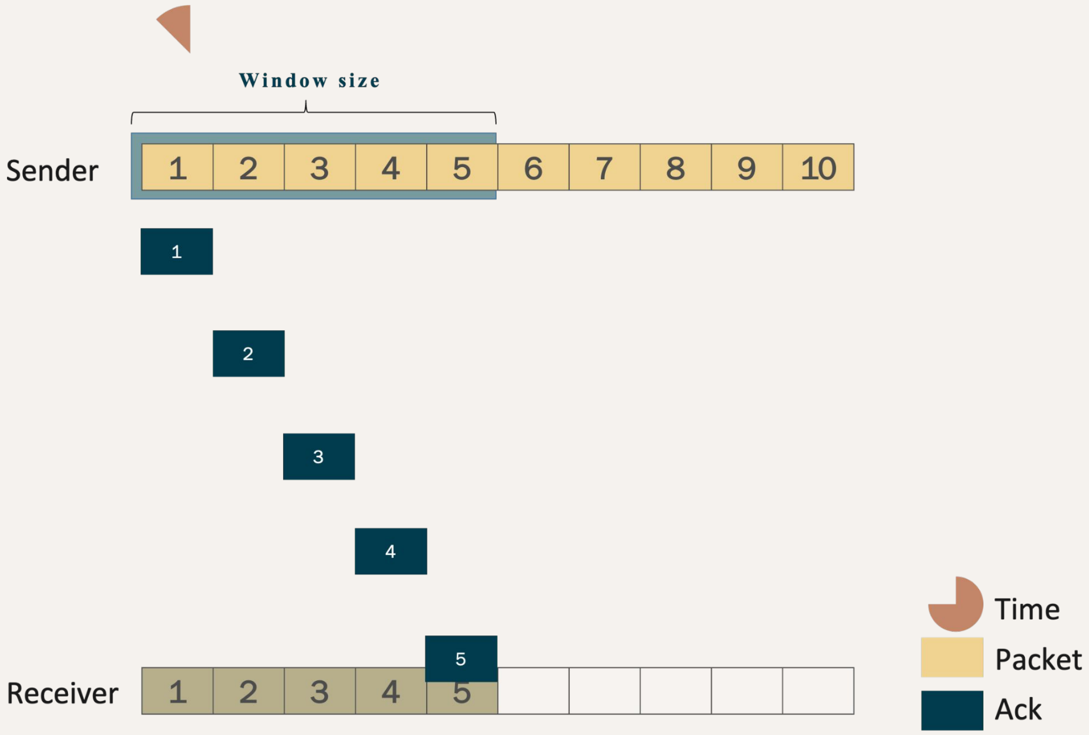
- UDP socket (multimedia)
- Reliable data transfering (Go-Back-N)
- Congestion control

What is Go-Back-N?

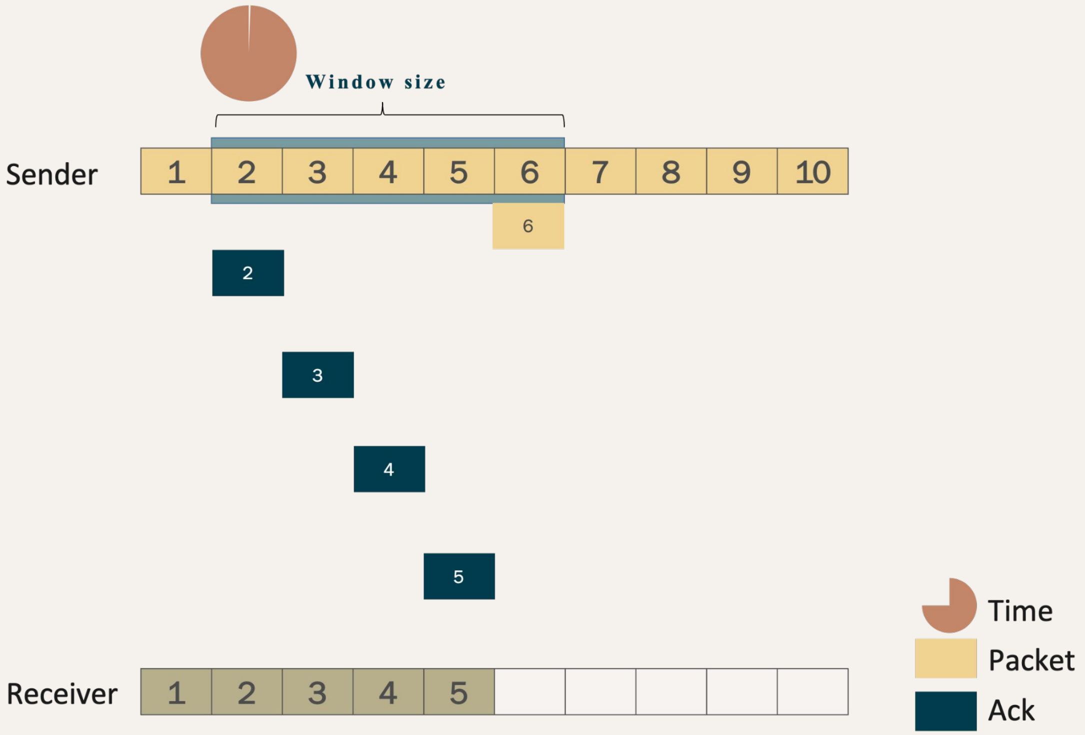
Go-Back-N case 1 (working normally)



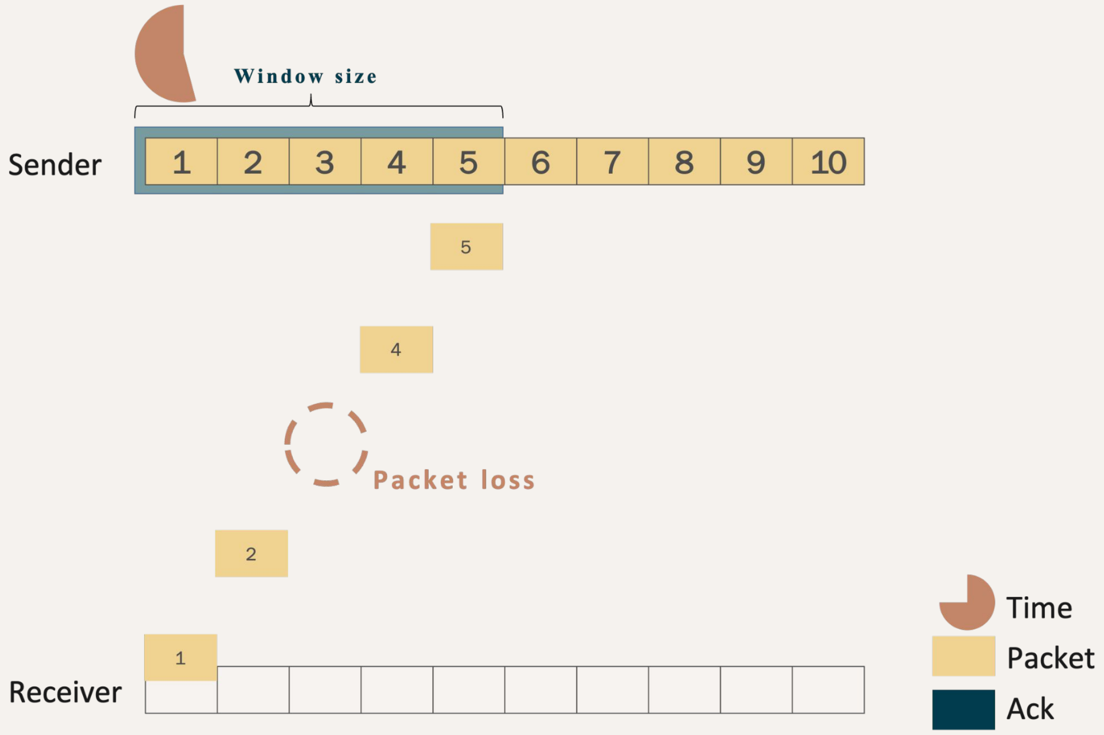
Go-Back-N case 1 (working normally)



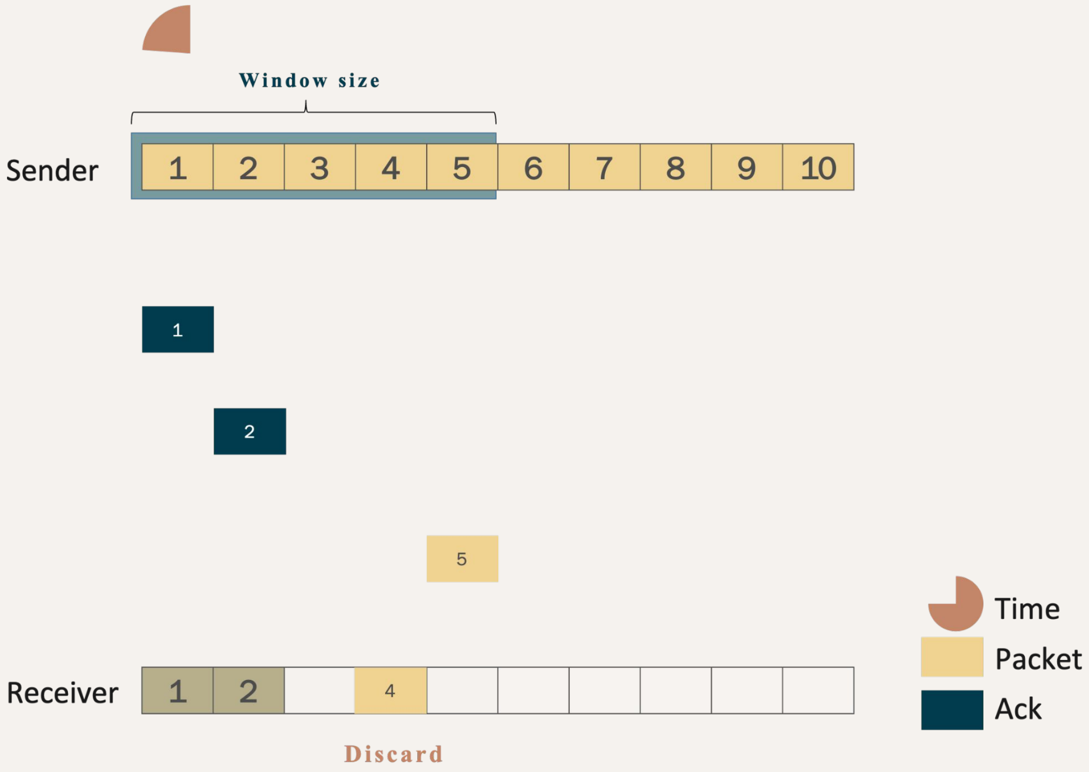
Go-Back-N case 1 (working normally)



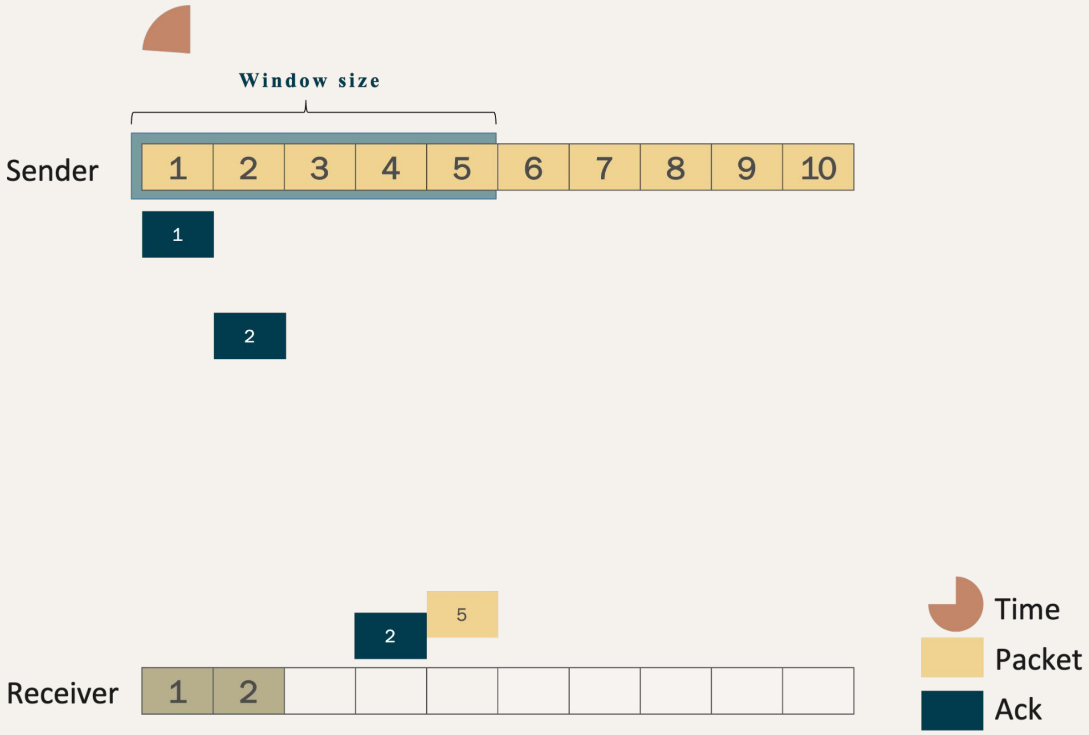
Go-Back-N case 2 (packet loss)



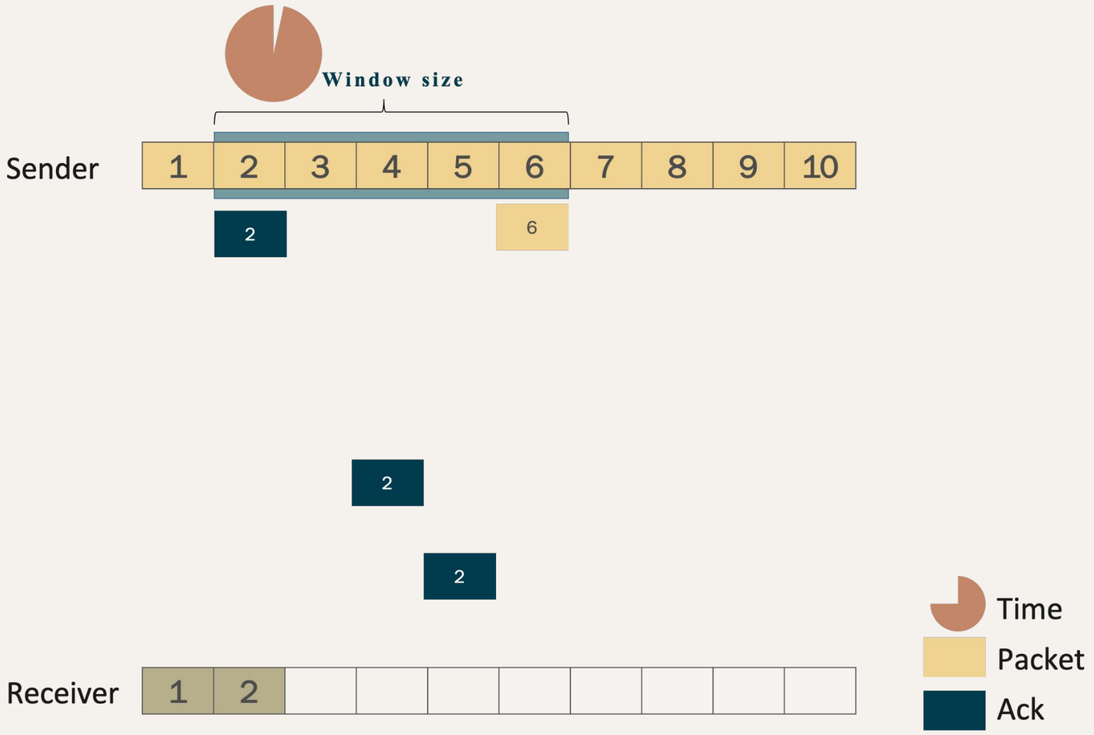
Go-Back-N case 2 (packet loss)



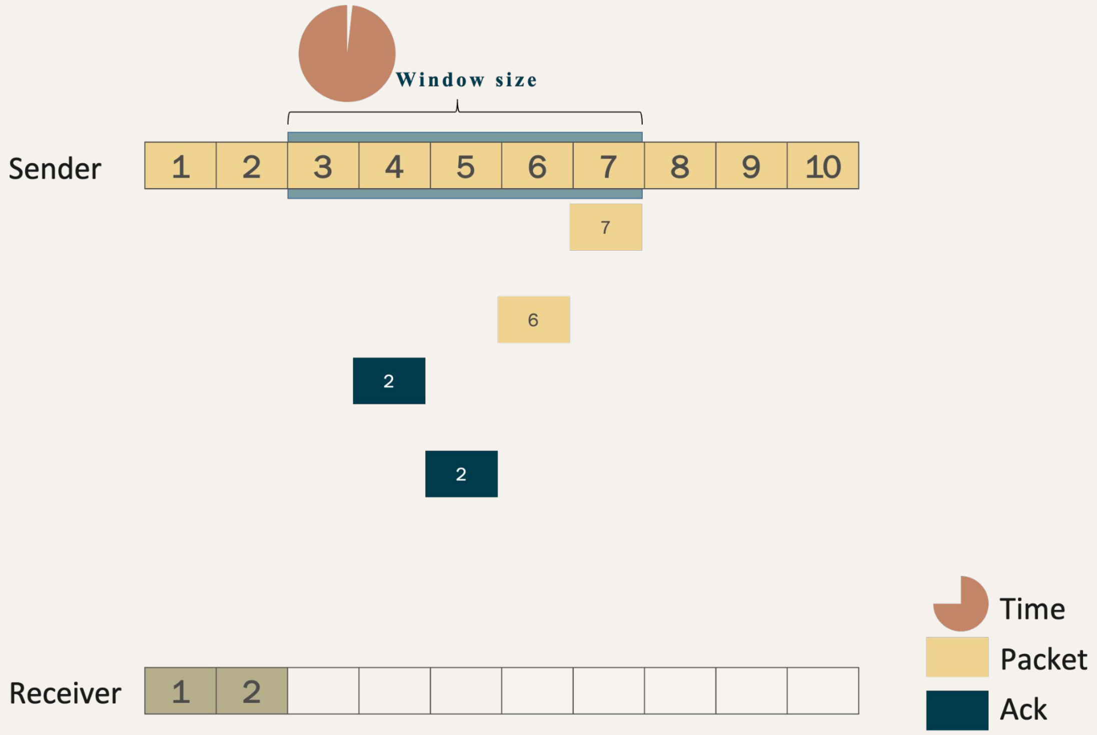
Go-Back-N case 2 (packet loss)



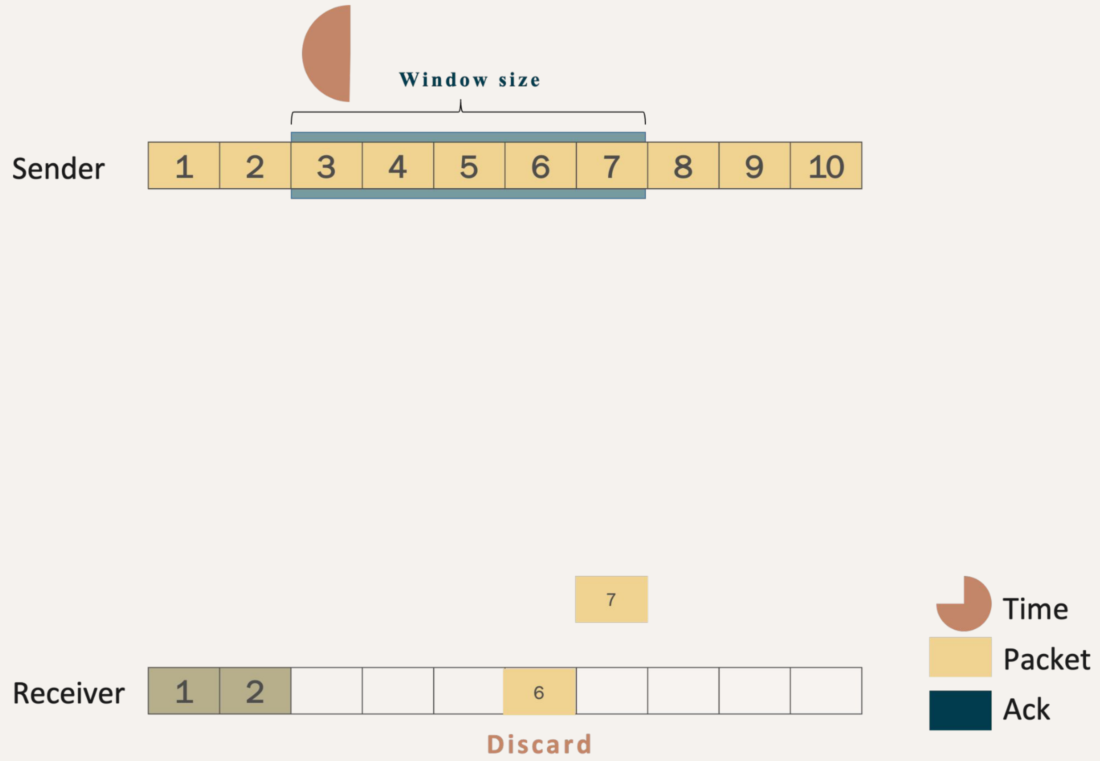
Go-Back-N case 2 (packet loss)



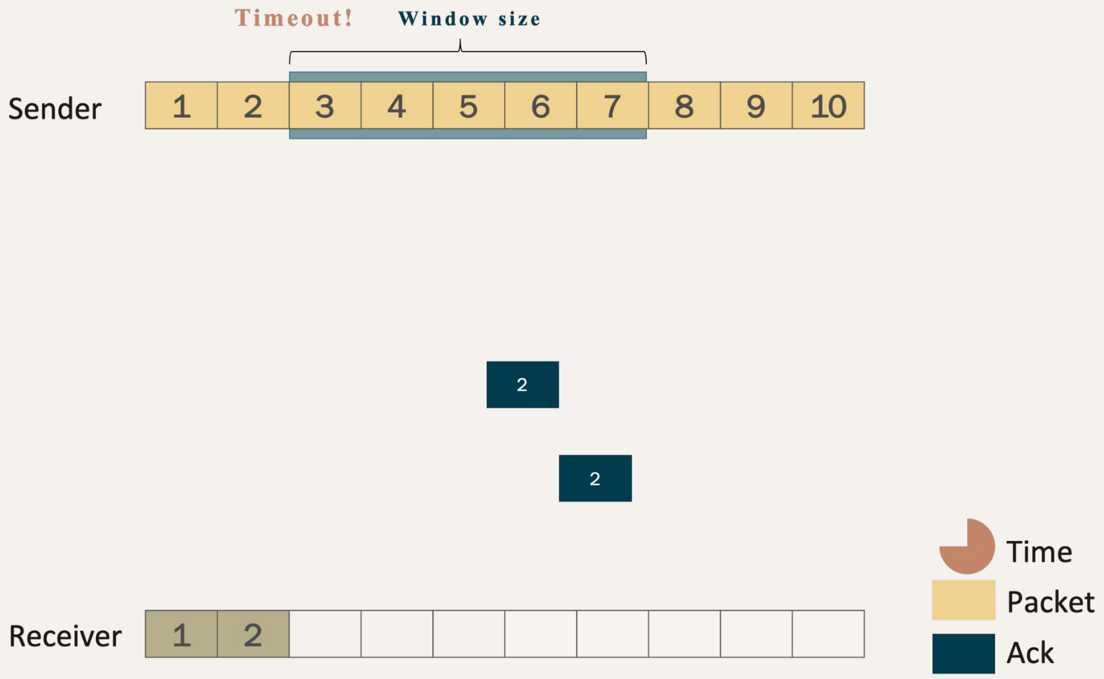
Go-Back-N case 2 (packet loss)



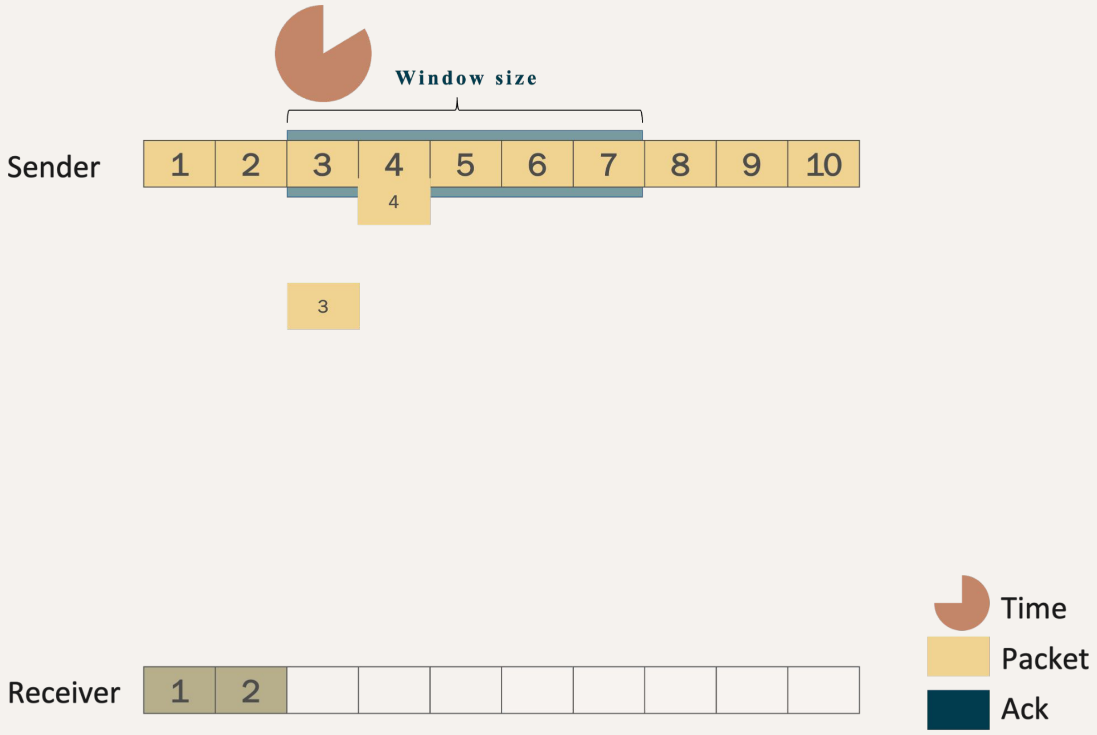
Go-Back-N case 2 (packet loss)



Go-Back-N case 2 (packet loss)



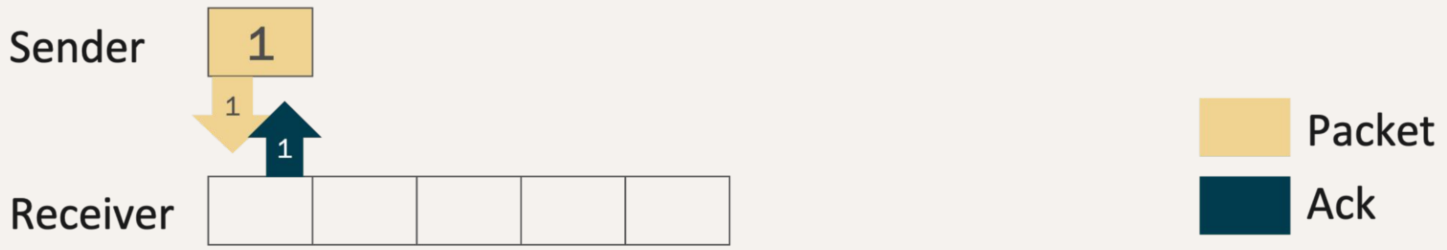
Go-Back-N case 2 (packet loss)



Go-Back-N with Congestion Control

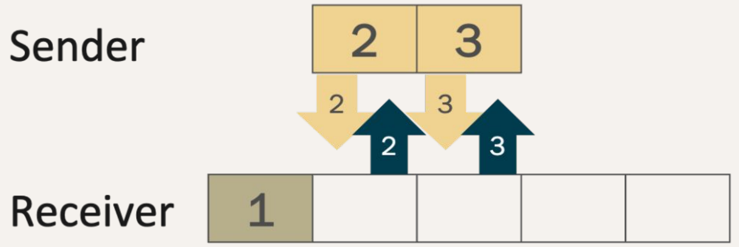
Go-Back-N + Congestion Control

- Sender sends Data 1
- Congestion window = 1. Threshold = 2
- Receiver sends ACK 1



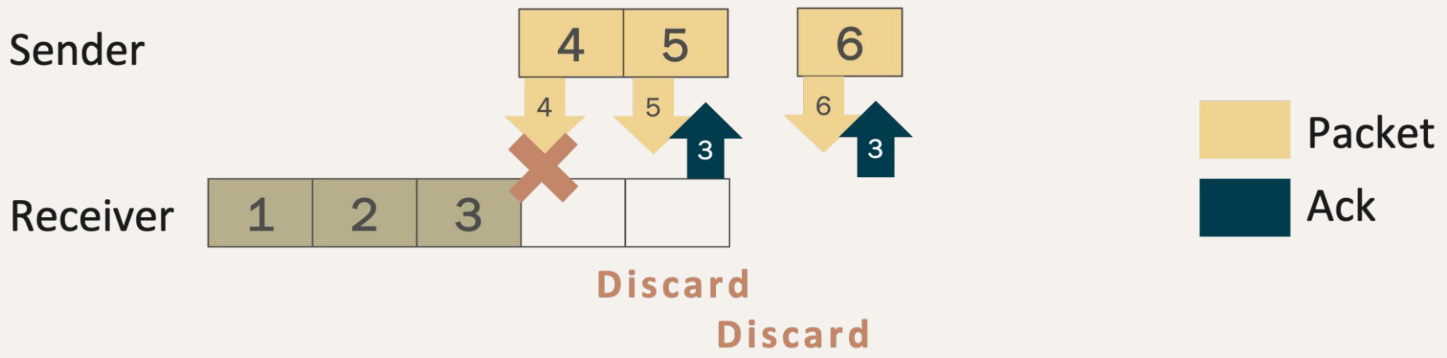
Go-Back-N + Congestion Control

- Sender sends Data 2,3
- Congestion window = 2. Threshold = 2
- Receiver sends ACK 2,3



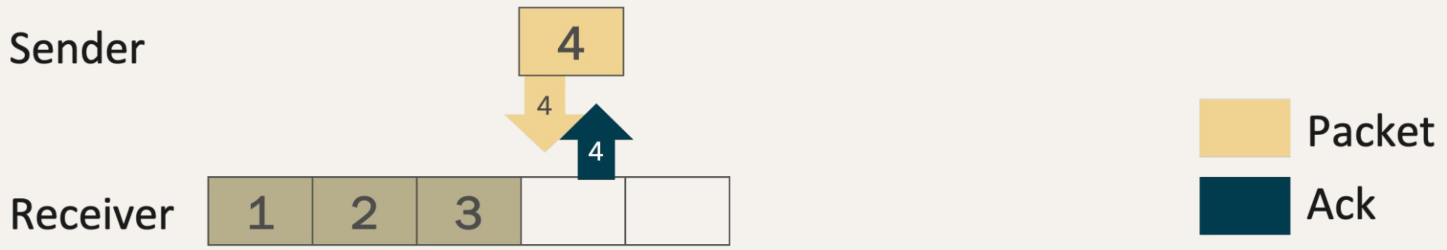
Go-Back-N + Congestion Control

- Sender sends Data 4, 5, 6
- Congestion window = 3. Threshold = 2
- Receiver drops Data 5, sends ACK 3, drops Data 6, sends ACK 3



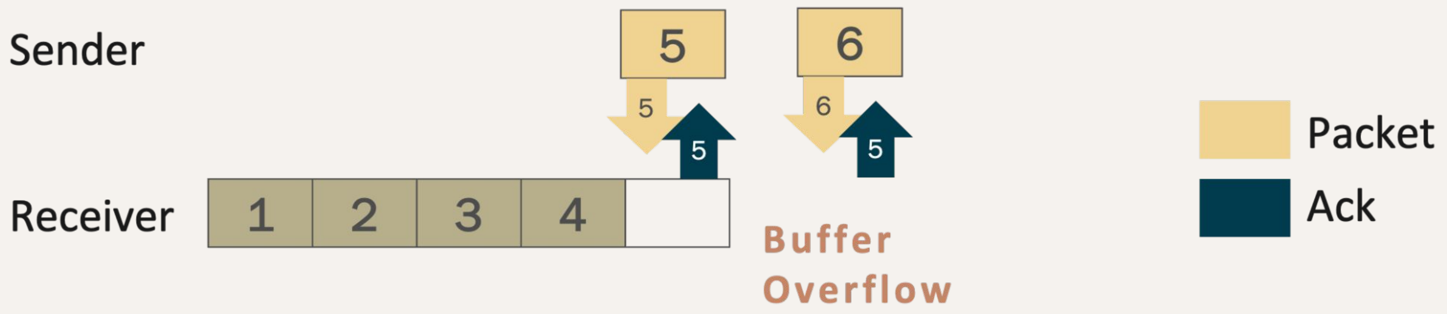
Go-Back-N + Congestion Control

- Sender sends Data 4
- Congestion window = 1. Threshold =1
- Receiver sends ACK 4



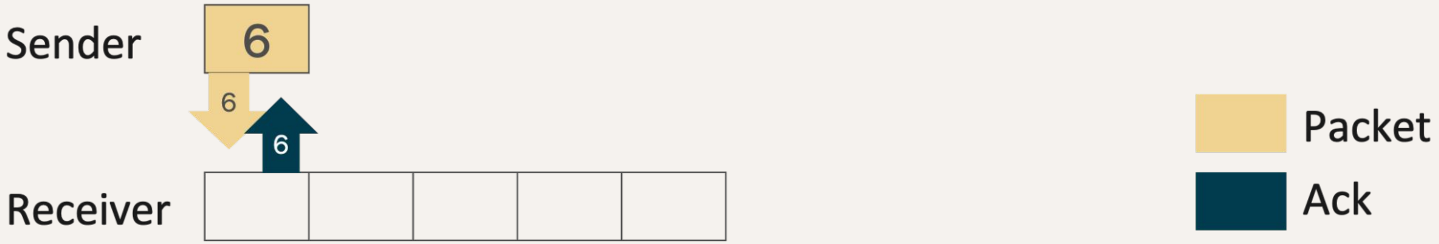
Go-Back-N + Congestion Control

- Sender sends Data 5,6
- Congestion window = 3. Threshold = 2
- Receiver sends ACK 5, drops Data 6, flush buffer()



Go-Back-N + Congestion Control

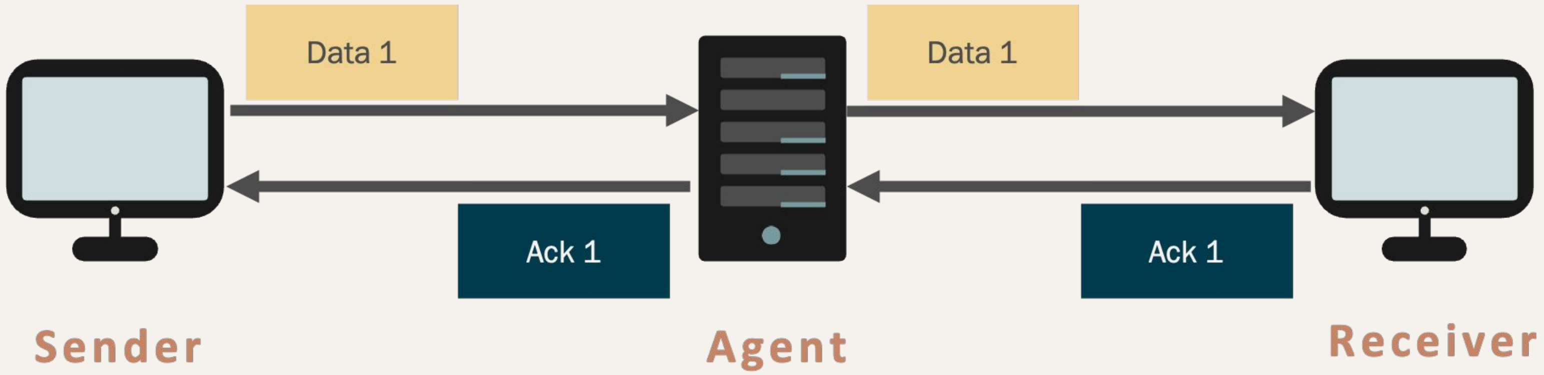
- Sender sends Data 1
- Congestion window = 1. Threshold = 1
- Receiver sends ACK 6



Assignment 3 Announcement

Specification (1/10)

- Implement three components: sender, receiver and agent.



Specification (2/10)

- **Programming language: C/C++**
- **Sender / Receiver**
  - Send / receive **video frame** by UDP
  - Provide reliable transmission
  - Congestion control
- **Agent**
  - Forward Data & ACK packets
  - **Randomly drop data packet, not ACK**
  - Compute loss rate

Specification (3/10)

- **Reliable Transmission**
  - Data & ACK
  - Time out & Retransmission (Go-Back-N)
  - Sequence number
- **Buffer handling** [receiver side]
  - Buffer Overflow: Drop the packet during out of buffer
  - Flush (write) to the file: Only when both buffer overflows and all packets in range are received.

Specification (4/10)

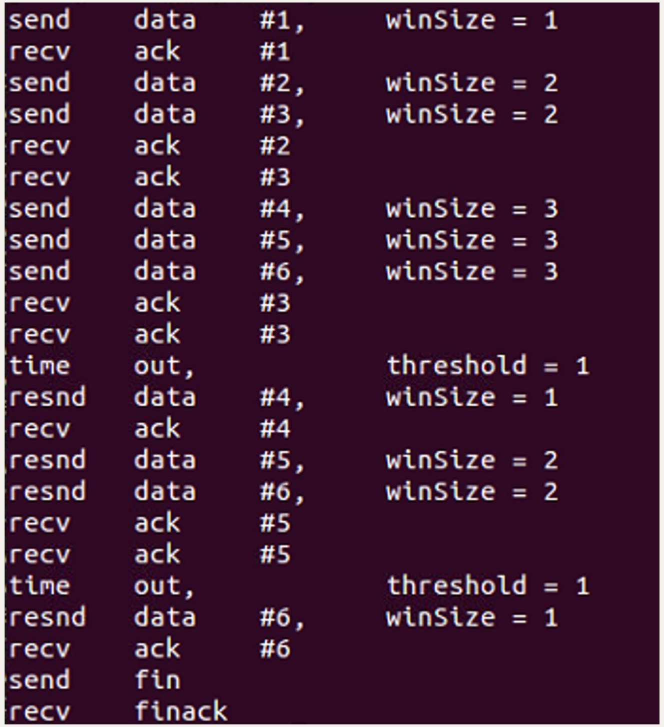
- **Congestion Control** [sender side]
  - Slow Start
    1. Send single packet in the beginning
    2. When window size is under the threshold, it increases exponentially until packet loses
    3. When window size is over the threshold, it increases linearly until packet loses
  - Packet loss / Time out
    1. Set threshold to max((window size)/2, 1)
    2. Set window size to 1
    3. Retransmit – from the first “unACKed packet”

Specification (5/10)

- **Show Message**
  - Sender:
    - send, recv, data, ack, fin, finack, sequence number, time out, resnd, winSize, threshold
  - Receiver:
    - send, recv, data, ack, fin, finack, sequence number, drop, flush
  - Agent:
    - get, fwd, data, ack, fin, finack, sequence number, drop, loss rate

Specification (6/10)

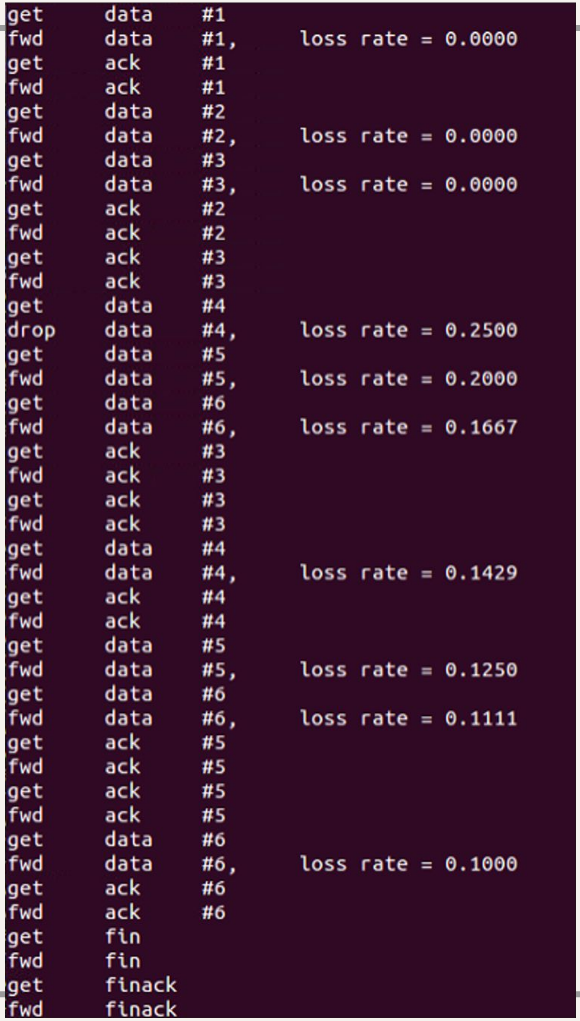
- **Show Message**
  - Sender:



```
send    data    #1,     winSize = 1
recv    ack     #1
send    data    #2,     winSize = 2
send    data    #3,     winSize = 2
recv    ack     #2
recv    ack     #3
send    data    #4,     winSize = 3
send    data    #5,     winSize = 3
send    data    #6,     winSize = 3
recv    ack     #3
recv    ack     #3
time    out,            threshold = 1
resnd   data    #4,     winSize = 1
recv    ack     #4
resnd   data    #5,     winSize = 2
resnd   data    #6,     winSize = 2
recv    ack     #5
recv    ack     #5
time    out,            threshold = 1
resnd   data    #6,     winSize = 1
recv    ack     #6
send    fin
recv    finack
```

Specification (7/10)

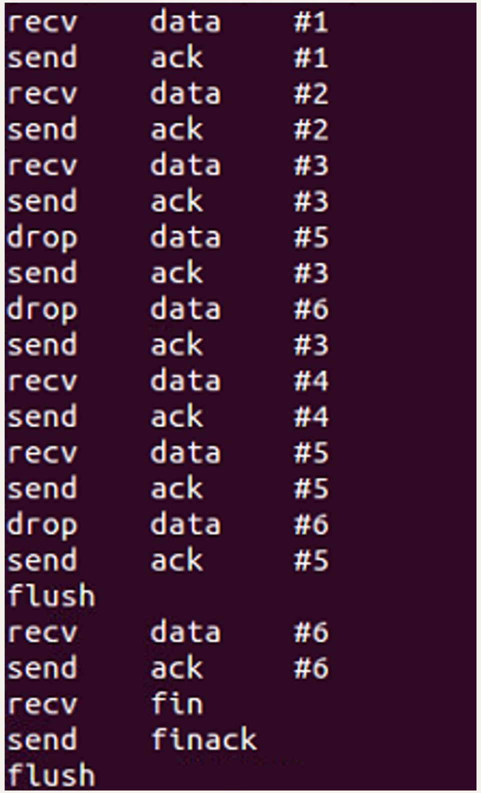
- **Show Message**
  - Agent:



```
get     data    #1
fwd     data    #1,     loss rate = 0.0000
get     ack     #1
fwd     ack     #1
get     data    #2
fwd     data    #2,     loss rate = 0.0000
get     data    #3
fwd     data    #3,     loss rate = 0.0000
get     ack     #2
fwd     ack     #2
get     ack     #3
fwd     ack     #3
get     data    #4
drop    data    #4,     loss rate = 0.2500
get     data    #5
fwd     data    #5,     loss rate = 0.2000
get     data    #6
fwd     data    #6,     loss rate = 0.1667
get     ack     #3
fwd     ack     #3
get     ack     #3
fwd     ack     #3
get     data    #4
fwd     data    #4,     loss rate = 0.1429
get     ack     #4
fwd     ack     #4
get     data    #5
fwd     data    #5,     loss rate = 0.1250
get     data    #6
fwd     data    #6,     loss rate = 0.1111
get     ack     #5
fwd     ack     #5
get     ack     #5
fwd     ack     #5
get     data    #6
fwd     data    #6,     loss rate = 0.1000
get     ack     #6
fwd     ack     #6
get     fin
fwd     fin
get     finack
fwd     finack
```

Specification (8/10)

- **Show Message**
  - Receiver:



```
recv    data    #1
send    ack     #1
recv    data    #2
send    ack     #2
recv    data    #3
send    ack     #3
drop    data    #5
send    ack     #3
drop    data    #6
send    ack     #3
recv    data    #4
send    ack     #4
recv    data    #5
send    ack     #5
drop    data    #6
send    ack     #5
flush
recv    data    #6
send    ack     #6
recv    fin
send    finack
flush
```

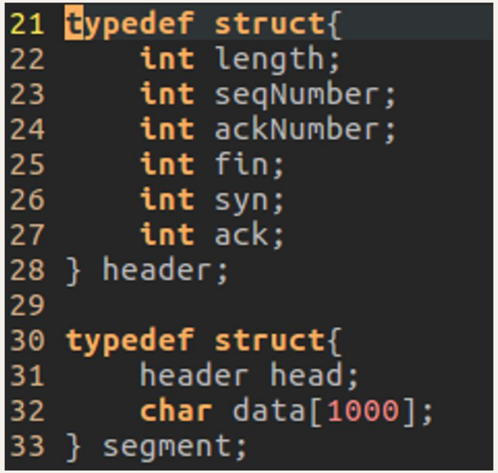
Specification (9/10)

- **Show Message**
  - The format used for transmission should be the same as follow:

fin: 0 or 1

syn: 0 or 1 (just make it 0)

ack: 0 or 1



```
typedef struct{
    int length;
    int seqNumber;
    int ackNumber;
    int fin;
    int syn;
    int ack;
} header;

typedef struct{
    header head;
    char data[1000];
} segment;
```

Specification (10/10)

- **Settings**
  - Sender
    - Arguments: IP, Port, path of source file, … etc.
    - Default threshold: 16
  - Receiver
    - Arguments: IP, port, … etc.
    - Default packet size (payload): 1000 bytes (recommended)
  - Agent
    - Arguments: IP, port, loss rate, … etc.
    - Data packet size (payload): 1000 bytes (recommended)
- Time out: Less than or equal to 1 sec (≤ 1 sec)

Grading Policy (1/2)

- This assignment accounts for 15% of the total score.
- **Video Streaming (20%)**
  - Correctly play the example video of HW2 (10%)
  - Correctly play a resolution-unknown video (10%)
- **Buffer handling (10%)**
- **Reliable transmission (20%)**
- **Congestion control (15%)**

Grading Policy (2/2)

- **Agent** **(10%)**
  - Randomly drop data packet (5%)
  - Compute loss rate (5%)
- **Show Message** **(10%)**
  - Show message correctly (10%)
- **Report** **(15%)**
  - How to compile and execute your program (3%)
  - Explain your program structure (3%*4)
    (including **3 flow charts** for sender, agent, and receiver)

Submission

- Requirements
  - Your report must be a **.pdf** file and named “**report.pdf**”.
  - Please put all the file into a folder named **<studentID>_hw3** and compress the folder as a **.zip** file. Submit your **.zip** file to **NTU COOL**.
    - e.g. B08902999_hw3.zip
  - The penalty for wrong format is 5 points.
  - If we cannot compile or execute your code, you will have a chance to demo your results in your own environment.
  - No plagiarism is allowed. A plagiarist will be graded zero.

Submission

- Deadline
    - Due Date : 23:59, December 28th, 2021
    - Penalty for late submission is **20 points per day**.

Sample Codes

- We will provide sample codes for your reference
  - *agent.c*
  - *video.mpg*

Contact us if you have any problem. •ω•)ฅ

TA Email: ntu.cnta@gmail.com