國立臺灣大學

Assignment 3

Retransmission & Congestion Control

Prof. Ai-Chun Pang
TA / Yu-Yu Chen, Shao-Cheng Fan, Kuang-Hui Huang

Before we start

- Let's talk about plagiarism. 🐒
- Can we discuss each other?
  - Sure. We encourage discussions.
  - To prevent plagiarism, you should avoid sharing, seeing others' codes.
    - I call it "code-level discussion."
- I just found some useful and amazing(really?) codes from the Internet, can I use it?
  - You should cite and use them in a reasonable way.
    - Cite the sources either in your report or in your source codes.
  - Try to understand what those codes mean and rewrite them by yourself.
  - Gentle reminder: TAs have all of the copies of the homework in previous years.

Case study

- Do you think this is plagiarism?

```c
void cmd_not_fd(){
    //printf("Command not found.\n");
}


int judge_mpg(char filenm[]){
//If the video file is not a “.mpg” file while playing a video file, please print out “The ‘<videofile>’ 
    int i;
    for(i = 0; i < strlen(filenm); i++)
        if(filenm[i] == '.')
            break;
    if(filenm[i+1] == 'm' && filenm[i+2] == 'p' && filenm[i+3] == 'g' && strlen(filenm) == (i+4))
        return 1;
    else
        return 0;
}
```

```c
int is_mpg(char filename[]){
    int i;
    for ( i = 0; i < strlen(filename); i++)
    {
        if (filename[i]=='.')
        {
            break;
        }
    }
    if (filename[i+1]=='m'&&filename[i+2]=='p'&&filename[i+3]=='g'&&strlen(filename)==i+4)
    {
        return 1;
    }
    else return 0;
}
int main(int argc , char *argv[]){
```

Case study

- Do you think this is plagiarism?

```
void init_req(Request* req) {
    req->cmd = CMD_NONE;
    bzero(req->filename, sizeof(char) * FILE_NAME_MAX);
    bzero(req->buf, sizeof(char) * BUFF_SIZE);
    req->fp = NULL;
    req->ls_num = 0;
    req->send_bit = 0;
    req->last_bit_transferred = 0;


    req->video_width = 0;
    req->video_height = 0;
    req->img_size = 0;
    req->frame_cnt = 0;
    req->total_frame = 0;
}
```

```
void clear_client(Client* client) {
    client -> ins = INS_CLEAR;
    bzero(client -> filename, sizeof(char) * FILE_NAME_MAX);
    client -> fp = NULL;
    client -> last_bit_transferred = 0;


    client -> vid_wid = 0;
    client -> vid_hei = 0;
    client -> img_size = 0;
    client -> frame_cnt = 0;
    client -> total_frame = 0;
}

void ins_ls(int socket, Client* client) {
```

Case study

- Do you think this is plagiarism?

```
}
int count = 0;
while(1){
    //get a frame from the video to the container on server.
    cap >> imgServer;
    // get the size of a frame in bytes
    int imgSize = imgServer.total() * imgServer.elemSize();

    //傳 imgSize
    std::string imgSize_str = std::to_string(imgSize);
    strcpy(buffer, imgSize_str.c_str());
    sent = send(remoteSocket, buffer, BUFF_SIZE, 0);
    printf("sent imgSize = %d to client\n", imgSize);
    bzero(buffer, sizeof(char)*BUFF_SIZE);

    uchar *buffer_u = (uchar*)malloc( imgSize * sizeof(uchar));

    // copy a frame to the buffer
    memcpy(buffer_u, imgServer.data, imgSize);
    sent = write(remoteSocket, buffer_u, imgSize);
```

```
}
int count = 0;

while(1){
    //get a frame from the video to the container on server.
    cap >> imgServer;
    // get the size of a frame in bytes
    int imgSize = imgServer.total() * imgServer.elemSize();

    //傳 imgSize
    std::string imgSize_str = std::to_string(imgSize);
    strcpy(buffer, imgSize_str.c_str());
    sent = write(remoteSocket, buffer, BUFF_SIZE);
    //printf("sent imgSize = %s to client\n", buffer);
    bzero(buffer, sizeof(char)*BUFF_SIZE);

    uchar *buffer_u = (uchar*)malloc( imgSize * sizeof(uchar));

    // copy a frame to the buffer
    memcpy(buffer_u, imgServer.data, imgSize);
```

Case study

- Do you think this is plagiarism?

```
while(1){
    printf("Waiting for connections...\n");
    printf("Server Port: %d\n", port);
    //select_2_start
    FD_ZERO(&readfds);
    FD_SET(localSocket, &readfds);
    max_sd = localSocket;
    //add child sockets to set
    for(int i = 0;  i< max_clients; i++){

        sd = client_socket[i];
        if(sd > 0)
            FD_SET(sd, &readfds);
        if(sd > max_sd)
            max_sd = sd;
    }
    activity = select(max_sd+1,&readfds, NULL, NULL, NULL);
    if ((activity < 0) && (errno != EINTR)) {
        printf("select error\n");
    }
    //If something happened on the master socket ,
    //then its an incoming connection
    if (FD_ISSET(localSocket, &readfds)){
        //remoteSocket = accept(localSocket, (struct sockaddr *)&remoteAddr, (socklen_t*)&addrLen);
        if ((remoteSocket = accept(localSocket, (struct sockaddr *)&remoteAddr, (socklen_t*)&addrLen)
            perror("Accept error");
            exit(1);
        }
```

```
while(1){
    FD_ZERO(&readfds);
    FD_SET(server_sockfd, &readfds);
    max_sd = server_sockfd;
    for(int i = 0; i < client_socket.size(); i++){

        sd = client_socket[i];
        if(sd > 0){
            FD_SET(sd, &readfds);
        }
        if(sd > max_sd){
            max_sd = sd;
        }
    }
    activity = select( max_sd + 1 , &readfds , NULL , NULL , NULL);
    if ((activity < 0) && (errno!=EINTR)){
        ERR_EXIT("select failed\n");
    }
    // If something happened on the master socket, then its an incoming connection.
    if(FD_ISSET(server_sockfd, &readfds)){
        if((client_sockfd = accept(server_sockfd, (struct sockaddr*)&client_addr, (socklen_t*)&client_
            ERR_EXIT("accept failed\n");
        }
        // Read the client's name
        memset(buffer, '\0', sizeof(buffer));
        if ((read_byte = read(client_sockfd, buffer, sizeof(buffer) - 1)) < 0){
            ERR_EXIT("read client's name failed\n");
        }
```

Case study

- Do you think this is plagiarism?

```c
else if (instruction[0] == 'p' && instruction[1] == 'u' && instruction[2] == 't' && strlen(instruc
    //printf("Into put\n");
    scanf("%s", filenm);
    int write_val = write(localSocket, instruction, BUFF_SIZE);
    ////printf("write inst: %d bytes\n", write_val);
    if( write_val < 0){
        perror("Write to server error, inst: put.");
        exit(1);
    }
    write_val = write(localSocket, filenm, BUFF_SIZE);
    ////printf("write filenm: %d bytes\n", write_val);

    if(write_val < 0){
        perror("Write to server error, filenm.");
        exit(1);
    }
    //printf("write filenm: %s, write_val = %d\n", filenm, write_val );
    //先確認檔名存在
    char exist[2];
    //確認檔案存在
    bzero(exist, sizeof(char)* 2);
    //printf("Finding file...\n");
    find_file(filenm, exist);
    int sent;
    sent = write(localSocket, exist, 2);
    //printf("sent = %d, exist[0] = %c\n", sent, exist[0]);
    //存在後再傳檔案
    if(exist[0] == '1'){
        //printf("Sending file start...\n");
        send_file(filenm, localSocket);//這裡很怪，localSocket跟remote socket搞不懂@@
        //printf("Sending file successful.\n");
    }
}
```

```c
else if (instruction[0] == 'p' && instruction[1] == 'u' && instruction[2] == 't' && strlen(instruc
    //printf("Into put\n");
    scanf("%s", filenm);
    int write_val = write(sockfd, instruction, BUFF_SIZE);
    ////printf("write inst: %d bytes\n", write_val);
    if( write_val < 0){
        perror("Write to server error, inst: put.");
        exit(1);
    }
    write_val = write(sockfd, filenm, BUFF_SIZE);
    ////printf("write filenm: %d bytes\n", write_val);

    if(write_val < 0){
        perror("Write to server error, filenm.");
        exit(1);
    }
    //printf("write filenm: %s, write_val = %d\n", filenm, write_val );
    //先確認檔名存在
    char exist[2];
    //確認檔案存在
    bzero(exist, sizeof(char)* 2);
    //printf("Finding file...\n");
    find_file(filenm, exist);
    int sent;
    sent = write(sockfd, exist, 2);
    //printf("sent = %d, exist[0] = %c\n", sent, exist[0]);
    //存在後再傳檔案
    if(exist[0] == '1'){
        //printf("Sending file start...\n");
        send_file(filenm, sockfd, argv[1]);//這裡很怪，sockfd跟remote socket搞不懂@@
        //printf("Sending file successful.\n");
    }
}
```

Summary

- There's no use doing the follows (We can easily detect them)
  - Renaming variables
  - Removing comments
  - Rearranging functions
  - Loop unrolling
- Which cases will less likely be treated as plagiarism?
  - A classical method. (e.g., bubble sort)
  - A predefined function or object (e.g., `VideoCapture cap("./tmp.mpg")`)
  - Doing same thing with a different implementation. (e.g., `for` vs `foreach`)
- Some of you will receive an email about plagiarism in these days, so your grade is subjected to change.

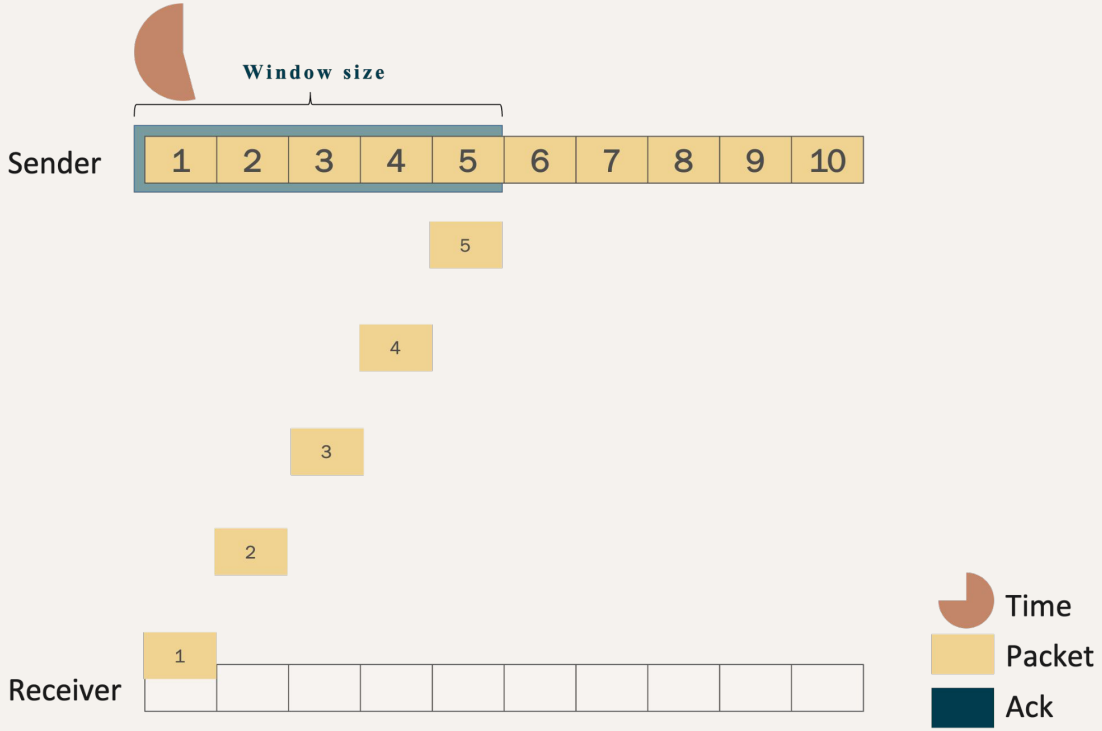
Lines Matched
852
411
545
380
210
207
168
149
84
65
84
68
56
78
72
71

Goals

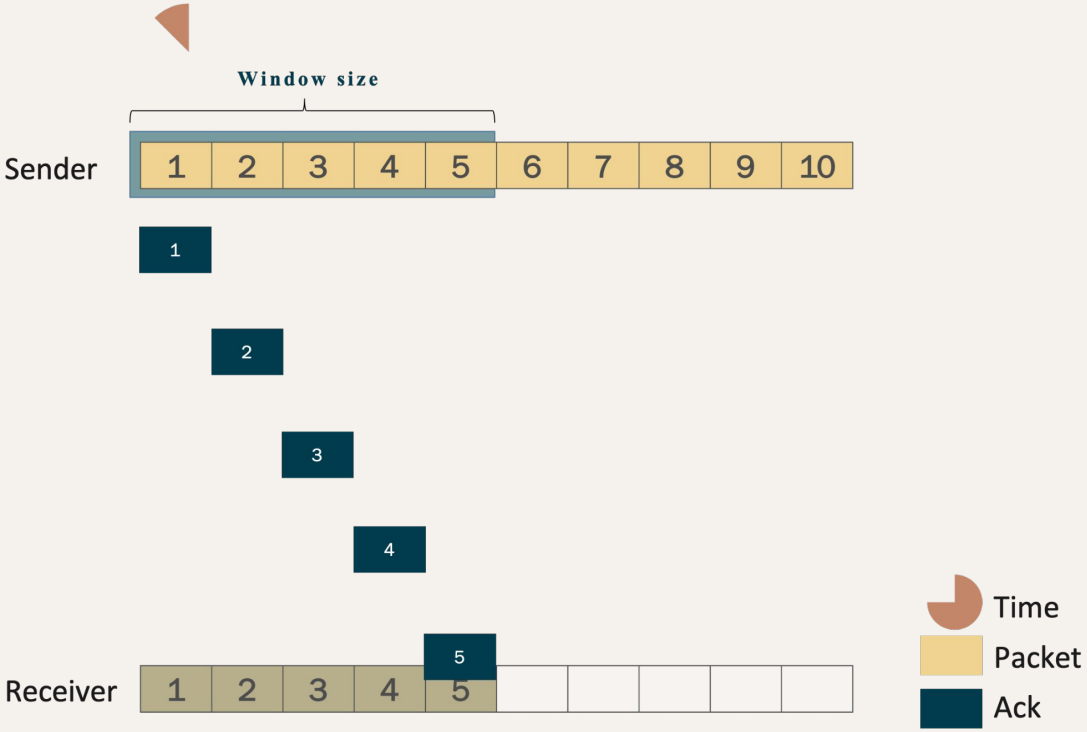
- **UDP socket (multimedia)**
- **Reliable data transfering (Go-Back-N)**
- **Congestion control**

What is Go-Back-N?

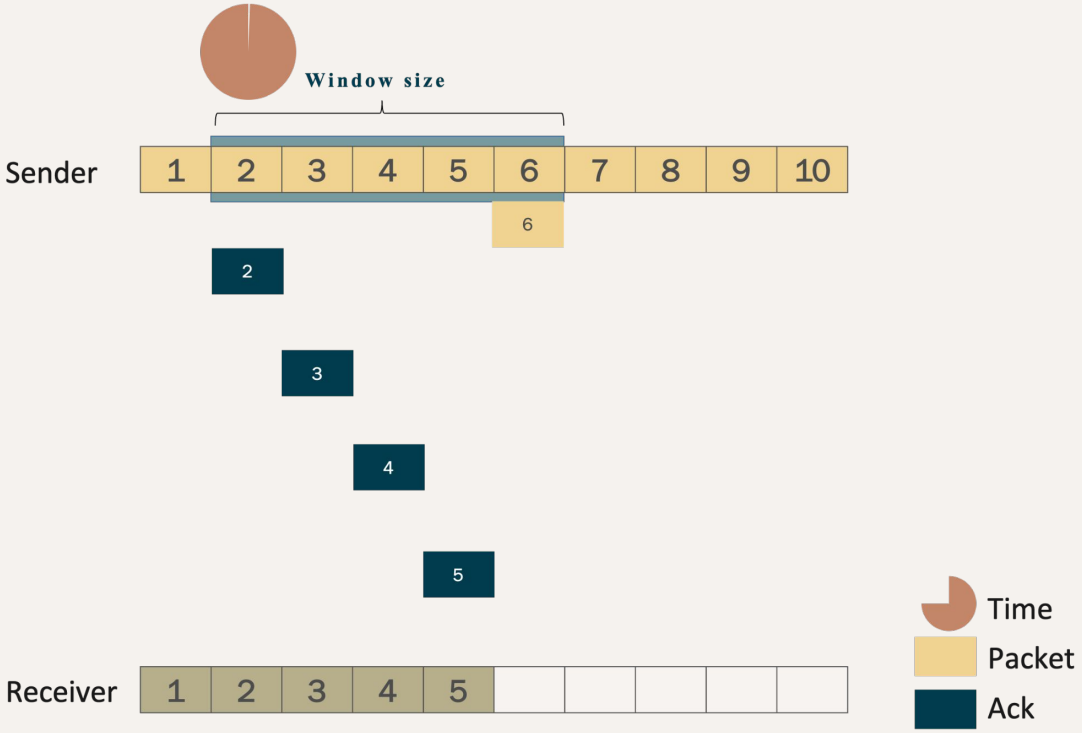
Go-Back-N case 1 (working normally)



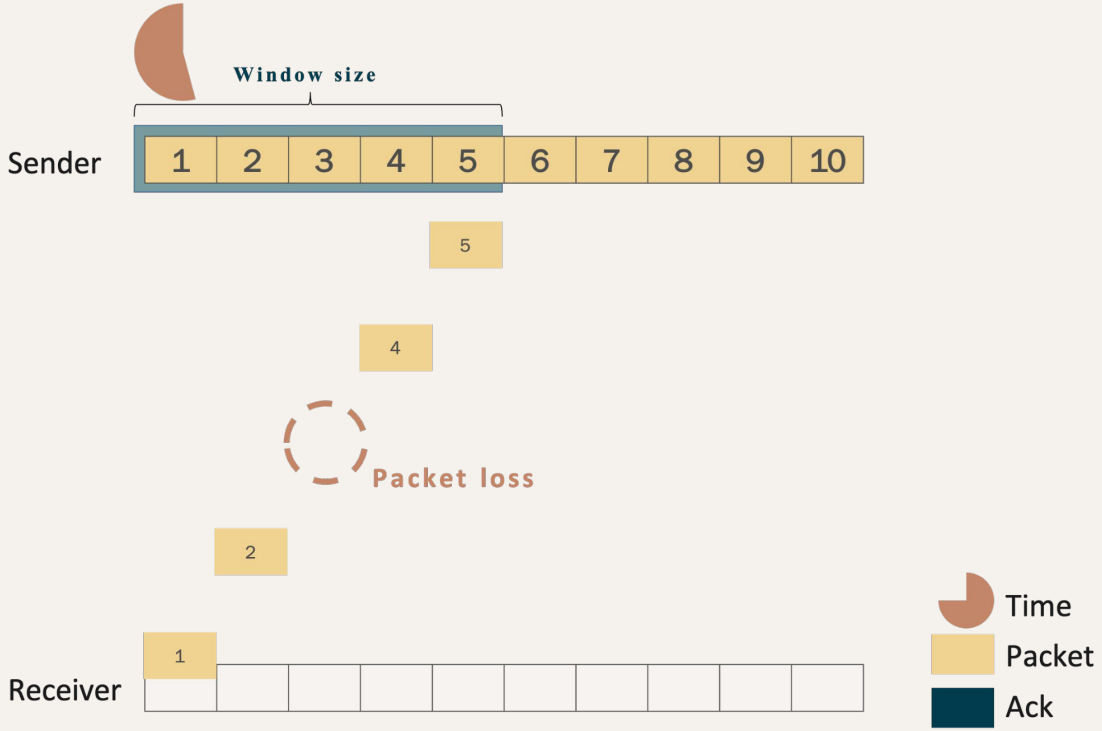
Go-Back-N case 1 (working normally)



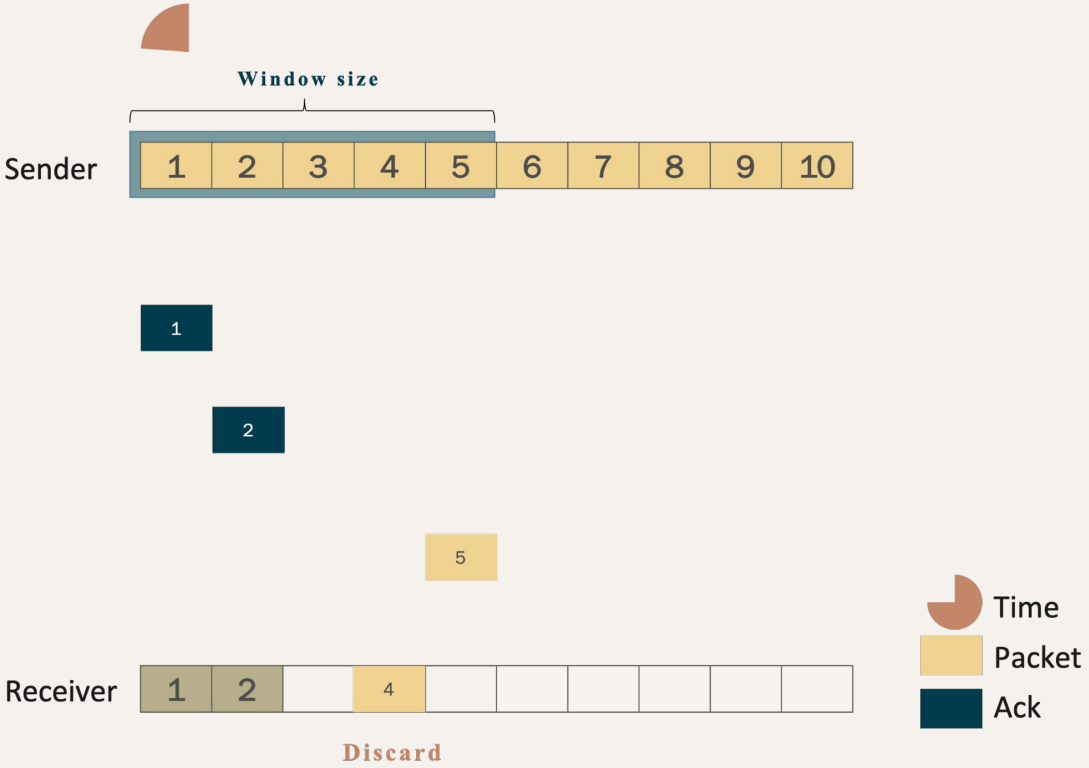
Go-Back-N case 1 (working normally)



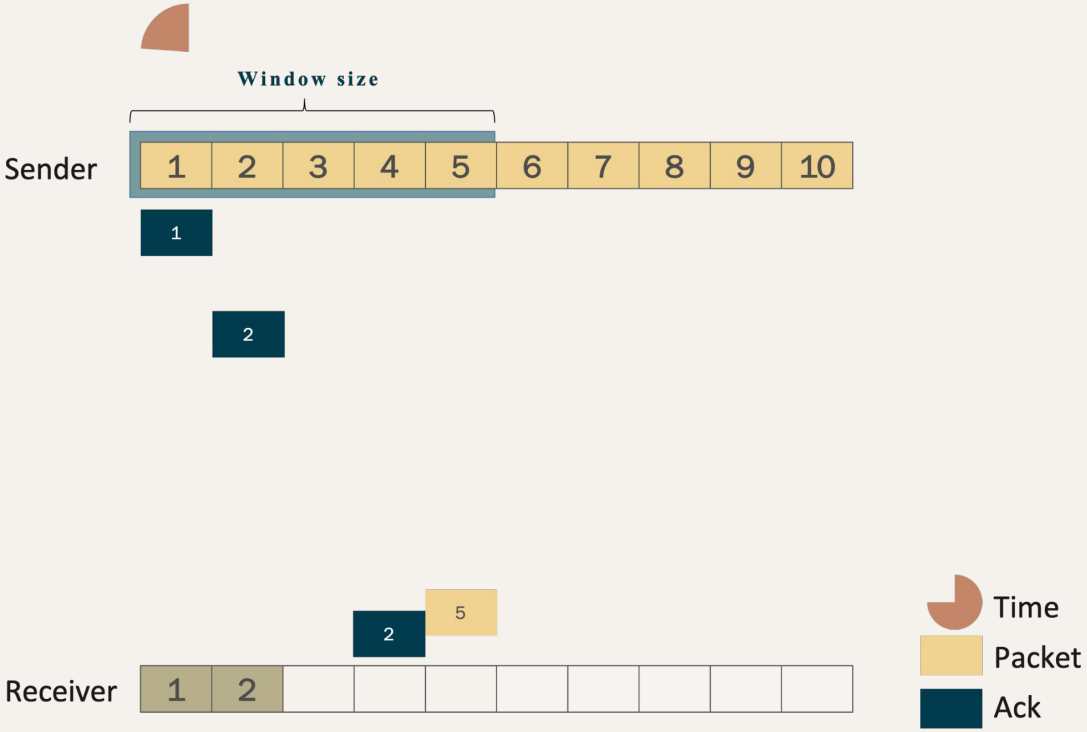
Go-Back-N case 2 (packet loss)



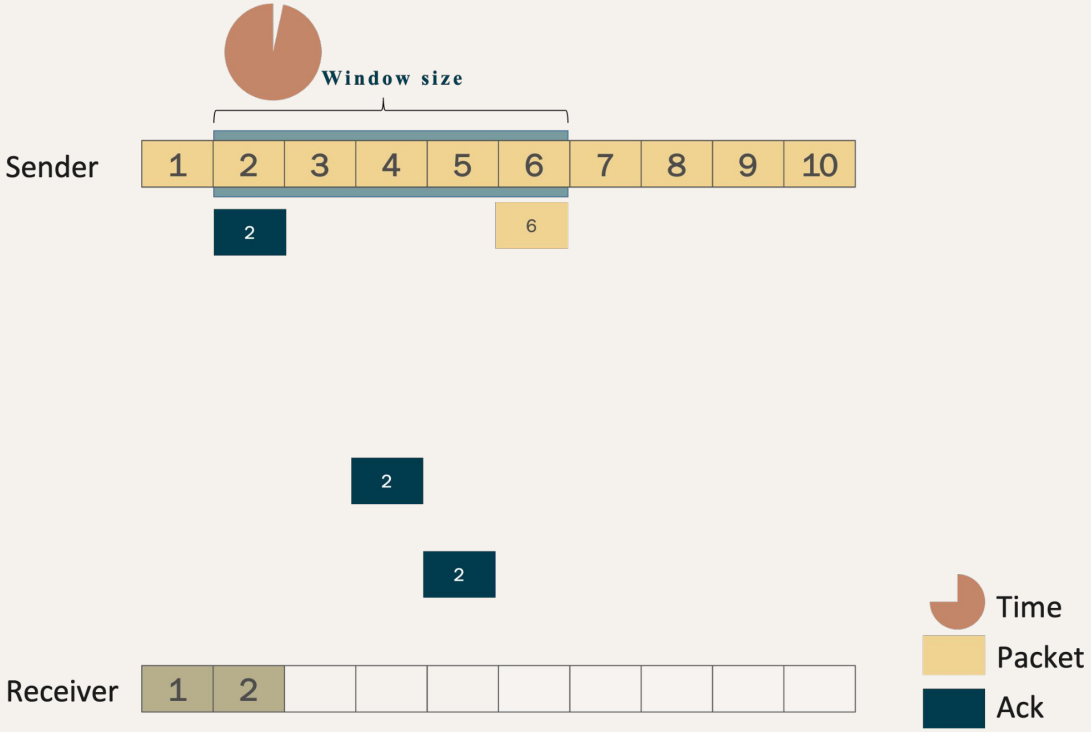
Go-Back-N case 2 (packet loss)



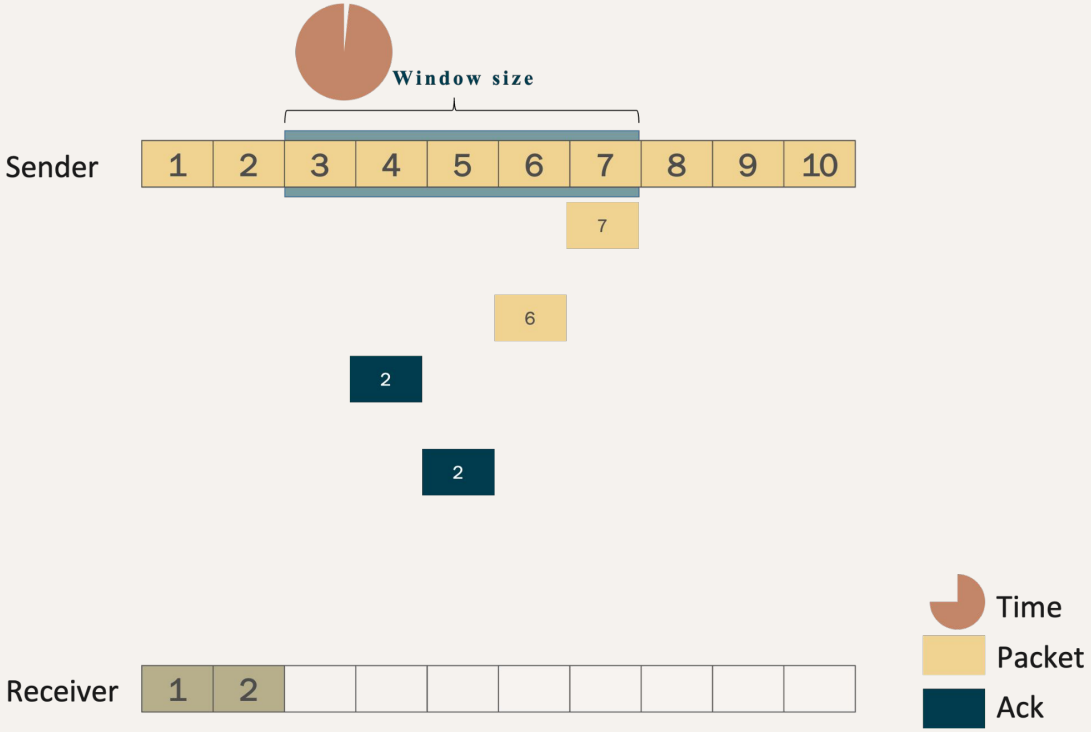
Go-Back-N case 2 (packet loss)



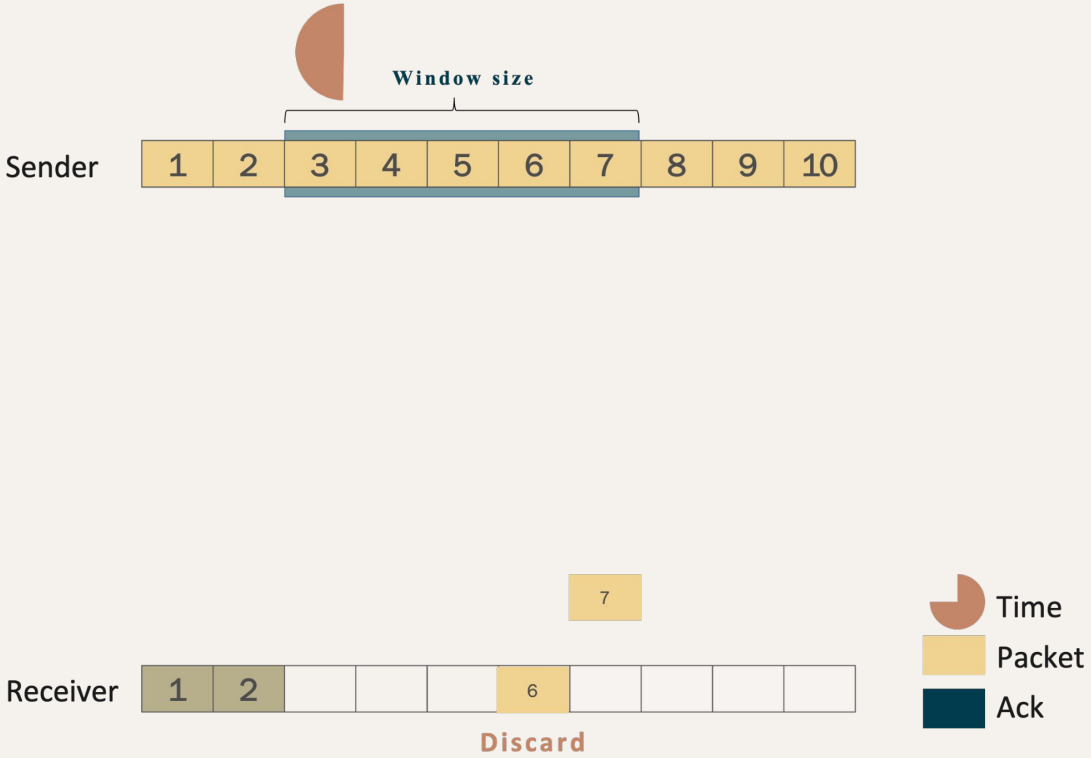
Go-Back-N case 2 (packet loss)



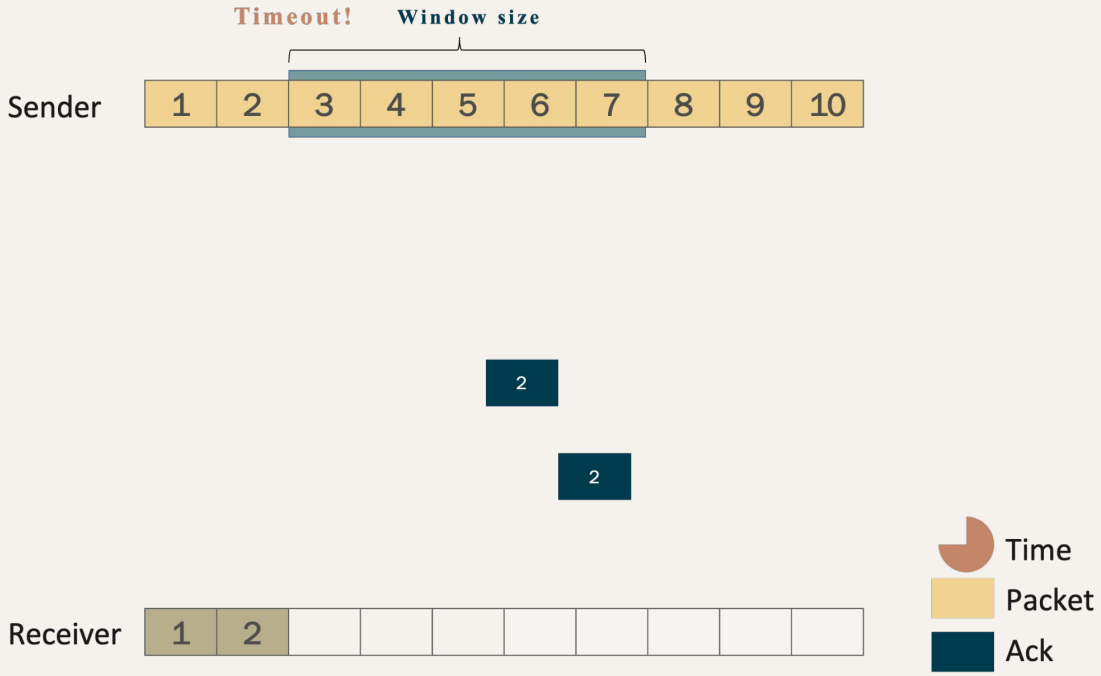
Go-Back-N case 2 (packet loss)



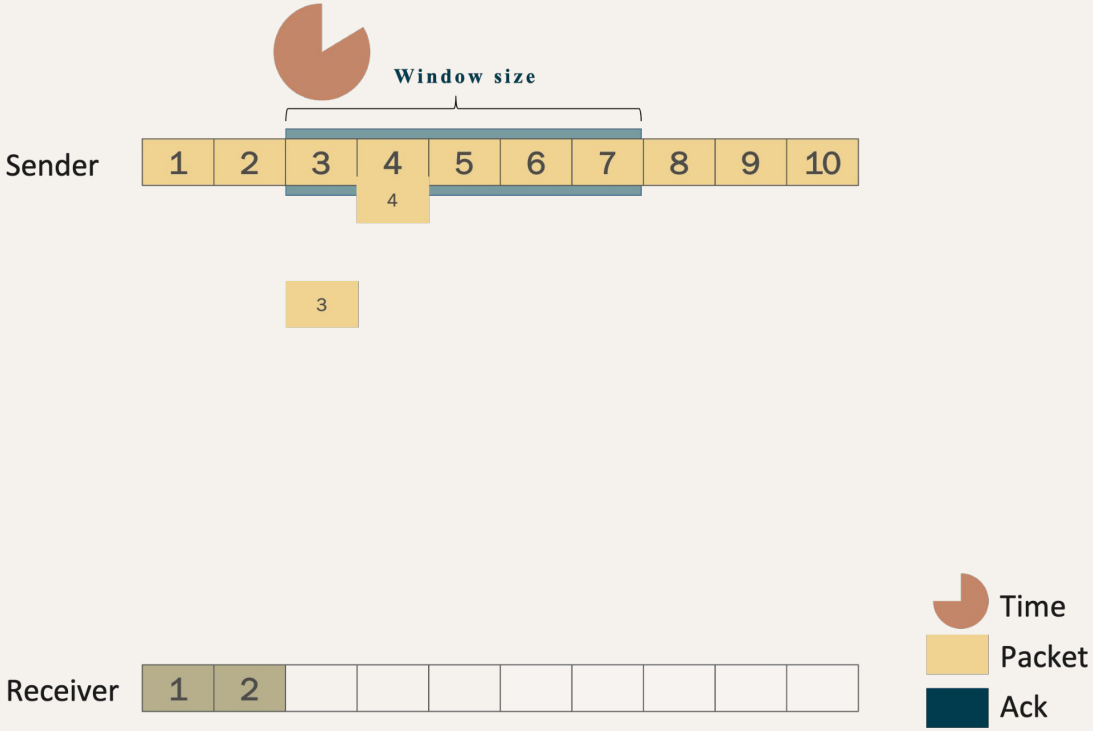
Go-Back-N case 2 (packet loss)



Go-Back-N case 2 (packet loss)



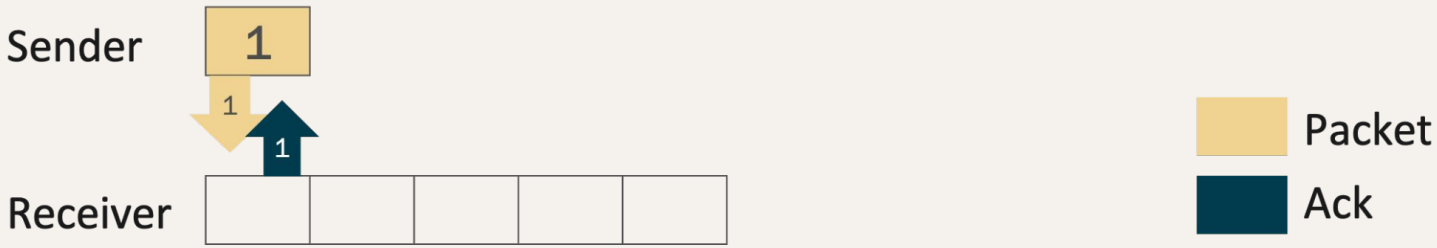
Go-Back-N case 2 (packet loss)



Go-Back-N with Congestion Control

Go-Back-N + Congestion Control

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

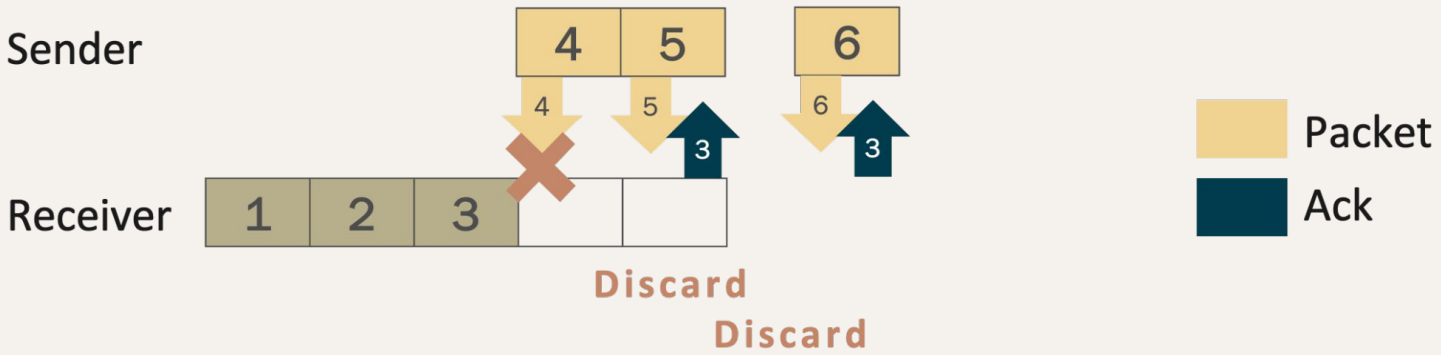


Go-Back-N + Congestion Control

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

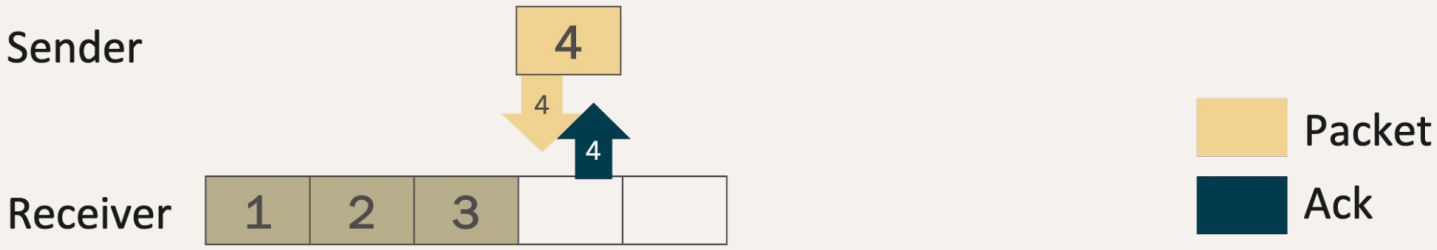
Go-Back-N + Congestion Control

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



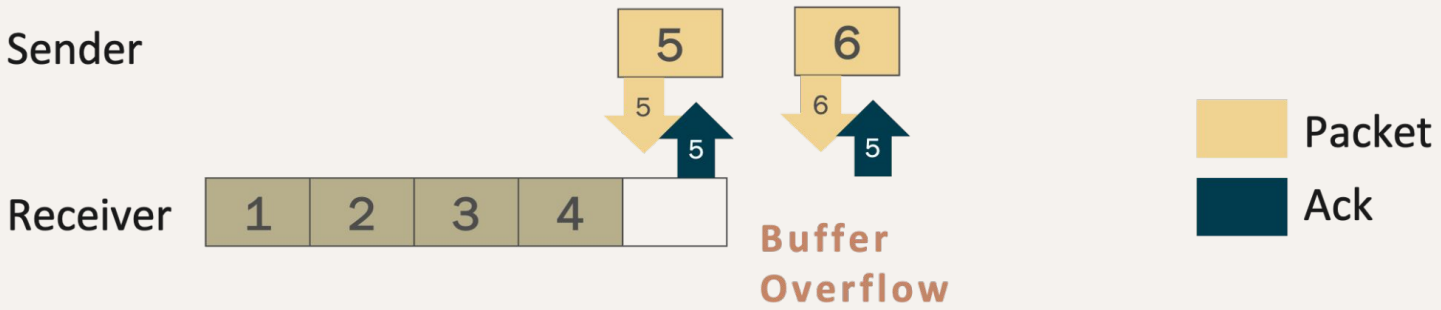
Go-Back-N + Congestion Control

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



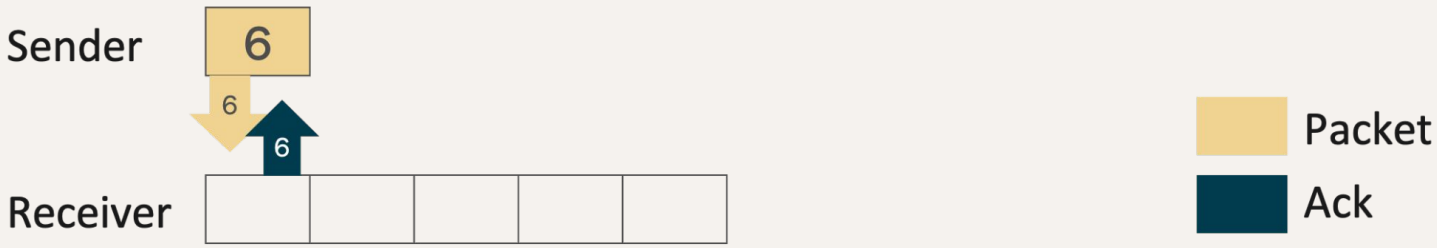
Go-Back-N + Congestion Control

- Sender sends Data 5, 6
- Congestion window = 2. Threshold = 1
- Receiver sends ACK 5, drops Data 6, sends ACK 5, flush buffer()

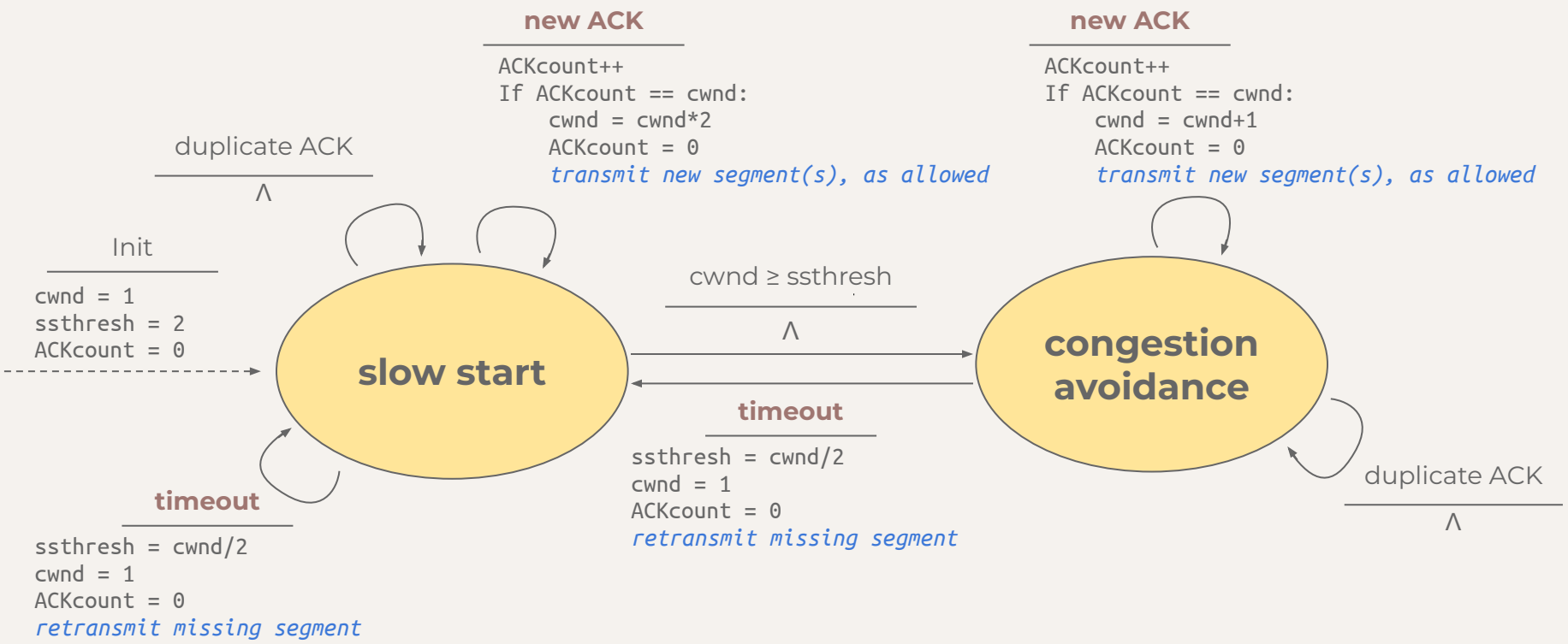


Go-Back-N + Congestion Control

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



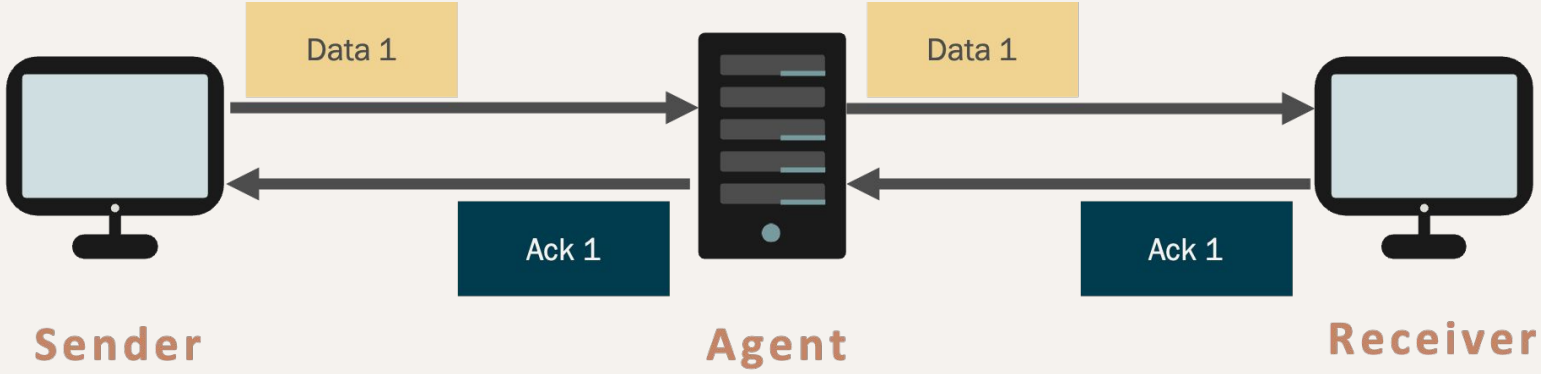
Go-Back-N + Congestion Control (Sender)



Assignment 3 Announcement

Specification (1/12)

- Implement three components: sender, receiver and agent.



Specification (2/12)

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

Specification (3/12)

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

Specification (4/12)

- **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 equal to or 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 (5/12)

- **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 (corrupted/out of order/buffer overflow), flush`
  - Agent:
    - `get, fwd, data, ack, fin, finack, sequence number, drop, corrupt, error rate`

Specification (6/12)

- Show Message for **sender**

tab characters

format: `send\tdata\t#%d,\twinSize = %d`

```
send    data    #1,     winSize = 1
recv    ack     #1
send    data    #2,     winSize = 2
send    data    #3,     winSize = 2
recv    ack     #2
recv    ack     #2
time    out,            threshold = 1
resnd   data    #3,     winSize = 1
recv    ack     #3
send    data    #4,     winSize = 2
send    data    #5,     winSize = 2
...
send    data    #1000,  winSize = 2
send    fin
recv    ack     #1000
recv    finack
```

Data #3 was corrupted

Assume there are 1000 packets in total

Might be "`resnd`"

Specification (7/12)

- Show Message for **agent**

```
get     data    #1
fwd     data    #1,     error rate = 0.0000
get     ack     #1
fwd     ack     #1
get     data    #2
fwd     data    #2,     error rate = 0.0000
get     data    #3
corrupt data    #3,     error rate = 0.3333
fwd     data    #3
get     ack     #2
fwd     ack     #2
get     ack     #2
fwd     ack     #2
```

```
get     data    #3
fwd     data    #3,     error rate = 0.2500
get     ack     #3
fwd     ack     #3
get     data    #4
drop    data    #4,     error rate = 0.4000
get     data    #5
fwd     data    #5,     error rate = 0.3333
...
get     fin
fwd     fin
...
get     finack
fwd     finack
```

Specification (8/12)

- Show Message for **receiver**
- Check a packet in the following order:
  a. **out of order** or not
  b. **corrupted** or not
  c. **buffer overflow** or not

```
recv    data    #1
send    ack     #1
recv    data    #2
send    ack     #2
drop    data    #3      (corrupted)
send    ack     #2
recv    data    #3
send    ack     #3
drop    data    #5      (out of order)
send    ack     #3
...
recv    data    #256
send    ack     #256
drop    data    #257    (buffer overflow)
send    ack     #256
flush
recv    data    #257
...
recv    data    #1000
send    ack     #1000
recv    fin
send    finack
flush
```

Specification (9/12)

- **Packet structure**
  - The format used for transmission should be the same as the right side:
  - `fin`: 0 or 1
  - `syn`: 0 or 1 (just make it 0)
  - `ack`: 0 or 1
  - `checksum`: we will use `crc32()` in `zlib.h` to calculate checksum

```
typedef struct {
    int length;
    int seqNumber;
    int ackNumber;
    int fin;
    int syn;
    int ack;
    unsigned long checksum;
} HEADER;

typedef struct {
    HEADER header;
    char data[1000];
} SEGMENT;
```

Specification (10/12)

- **Settings**
  - Sender
    - Default threshold: 16
    - Default window size: 1
  - Receiver
    - Default packet data size (payload): 1000 bytes
    - Default buffer size: 256 (# of packets)
  - Agent
    - Default packet data size (payload): 1000 bytes
  - Default time out: 1 sec

Specification (11/12)

- You are required to write a Makefile for compilation.

```
$ make sender            // To compile sender code
$ make agent             // To compile agent code
$ make receiver          // To compile receiver code
```

- After compilation, there will be 3 binary files named "`sender`," "`agent`," and "`receiver`."

Specification (12/12)

- Execute the following commands in different terminals and **in sequence**.

```
$ ./agent <agent port> <sender IP>:<sender port> <receiver IP>:<receiver port> <error rate>
$ ./receiver <receiver port> <agent IP>:<agent port>
$ ./sender <sender port> <agent IP>:<agent port> <.mpg filename>
```

- The error rate will be a floating point number between 0 and 1.
- The .mpg file will be placed **in the same folder as the sender**.

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 and corrupt data packet (5%)
  - Compute error rate (5%)
- **Show Message** **(10%)**
  - Show message correctly (10%)
- **Report** **(15%)**
  - Explain your program structure (5% * 3)

    (including **3 flow charts** for **sender**, **agent**, and **receiver**)

Submission (1/2)

- Requirements
  - Your report can be a **pdf** or **clear image** file. Submit it to **Gradescope**.
  - Please put all the source code (i.e., without your report, the video file, and the execution file) into a folder named **<studentID>_hw3** and compress the folder as a **.zip** file. Submit your **.zip** file to **NTU COOL**.

```
B09902999_hw3.zip
`-- B09902999_hw3       (<== folder)
    `-- your source code and Makefile
```

  - You should pass the sanity check with script `sanity-check.py`, or you will get **10 points** penalty.

Submission (2/2)

- 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 14th, 2022
  - Penalty for late submission is **20 points per day**.

Sample Codes

- We will provide sample codes for your reference
  - *agent.cpp*
  - *crc32.cpp*
  - *Makefile*
  - *video.mpg*
  - *sanity-check.py*

Supplementary Materials

crc32()

- Compute CRC-32 Checksum.

```
#include <zlib.h>
unsigned long crc32(unsigned long crc, const Bytef * buf, unsigned int len);
```

- **crc**
  - The previous value for the checksum.
  - In this homework, we can set it to `0L`.
- **buf**: Specifies the buffer to contain the data to be added to this checksum.
- **len**: Specifies the size of `buf`.

`waitKey` in OpenCV

Because a small `waitKey` parameter in openCV leads to some bugs, we suggest you use **1000** for `waitKey` in your program.

```
imshow("Video", client_img);
char c = (char)waitKey(1000);
if(c==27)
    break;
```

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

TA Email: ntu.cnta@gmail.com