

Computer network HW2

-Retransmission + Congest control

DATE : 2018/11/28

Introduction



Introduction

Target

- Application layer reliable transfer / congestion control
- Implement TCP by **UDP**
- Socket Programming

| UDP | TCP |
|----------------------------------|---|
| Unreliable Unordered delivery | Reliable In-order Delivery Congestion Control |

Introduction

You need to implement three components : the sender, receiver and agent.



Introduction

Sender / Receiver

- Send / receive file by UDP
- Provide reliable transmission
- Congestion control

Agent

- Forward Data & ACK packets
- Randomly drop data packet
- Compute loss rate

Requirement

Requirement

Reliable Transmission

- Data & ACK
- Time out & Retransmission(***Go-Back-N***)
- Sequence number
- Completeness and correctness of transmitted file

Requirement

Congestion control [*sender side*]

- Slow start
 - Send single packet in the beginning
 - When below the threshold, congestion window increase exponentially until packet loss, i.e., $1 \rightarrow 2 \rightarrow 4 \rightarrow 8 \rightarrow \dots$
 - When larger than or equal to the threshold, congestion window increase linearly until packet loss, i.e., $16 \rightarrow 17 \rightarrow 18 \rightarrow 19 \rightarrow \dots$
- Packet loss / time out
 - Set threshold to $\max(\lfloor \frac{\text{congestion window}}{2} \rfloor, 1)$
 - Set Congestion window to 1
 - Retransmit
 - From the first “un-ACKED packet”

Requirement

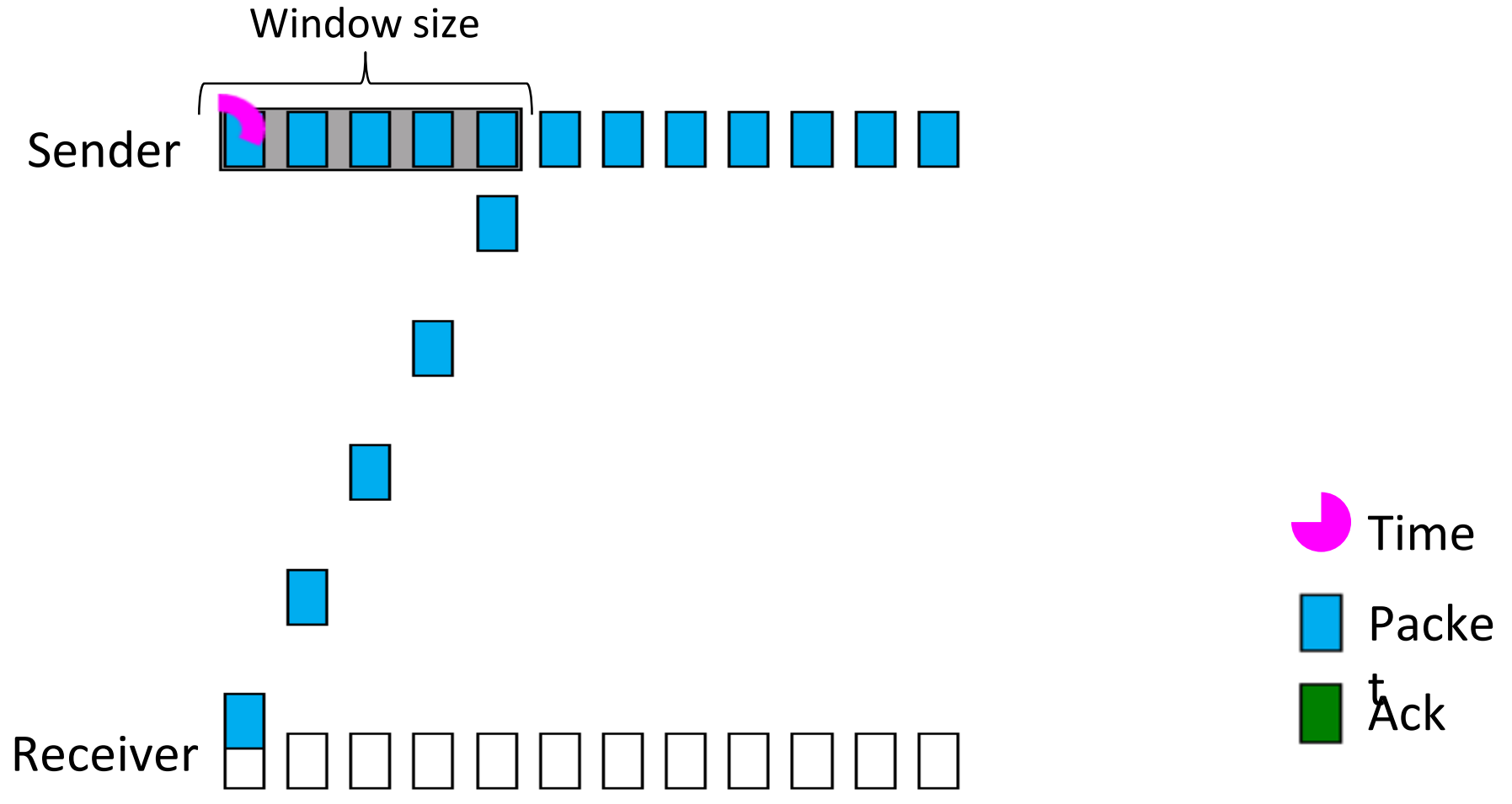
Buffer handling [*receiver side*]

- Buffer Overflow
 - Drop packet if “out of range” of buffer
- Flush (write) to the file
 - Only when *both buffer overflows and all packets in range are received*.

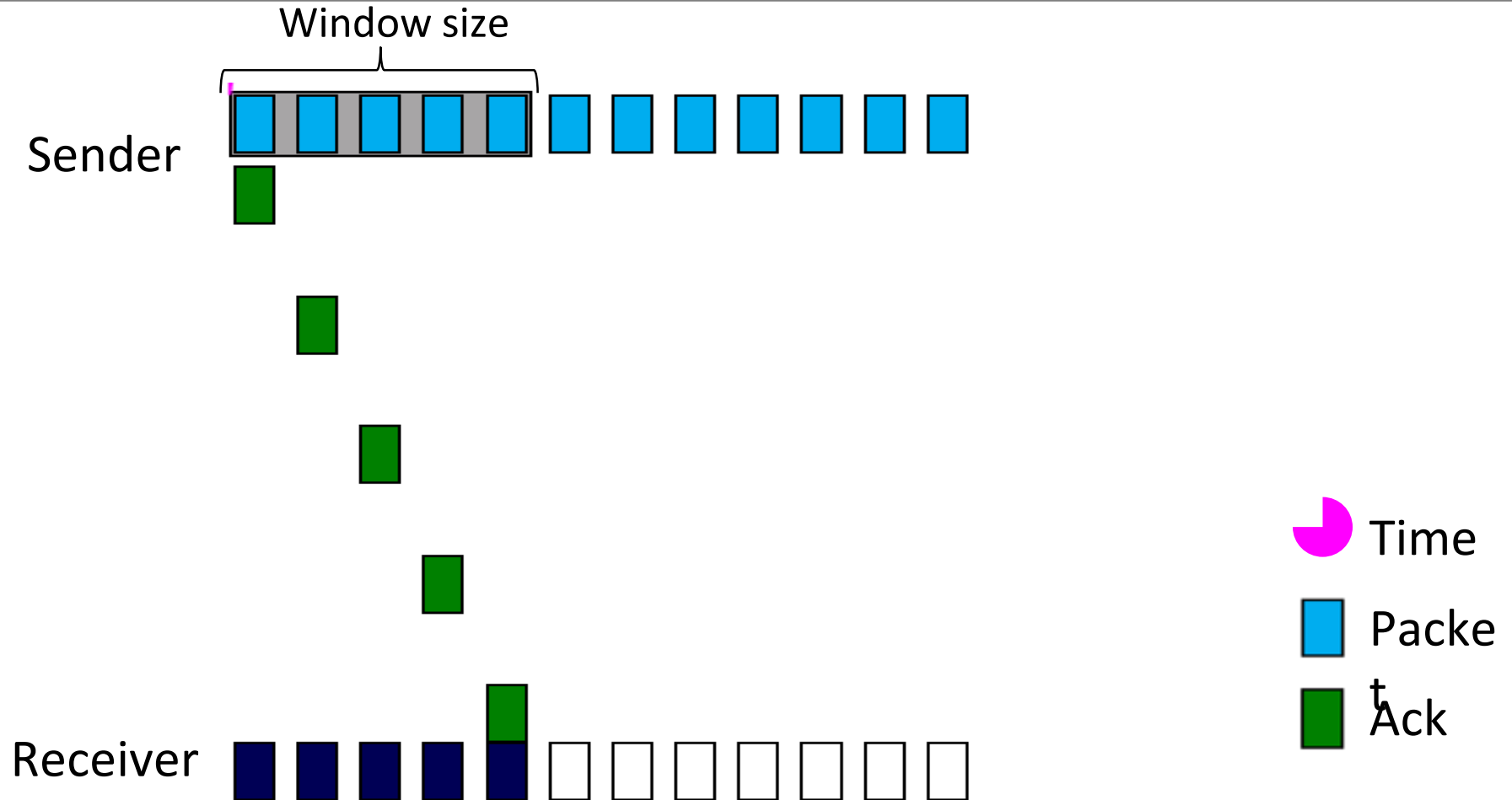
What is Go-Back-N(GBN)?



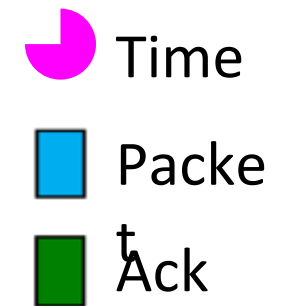
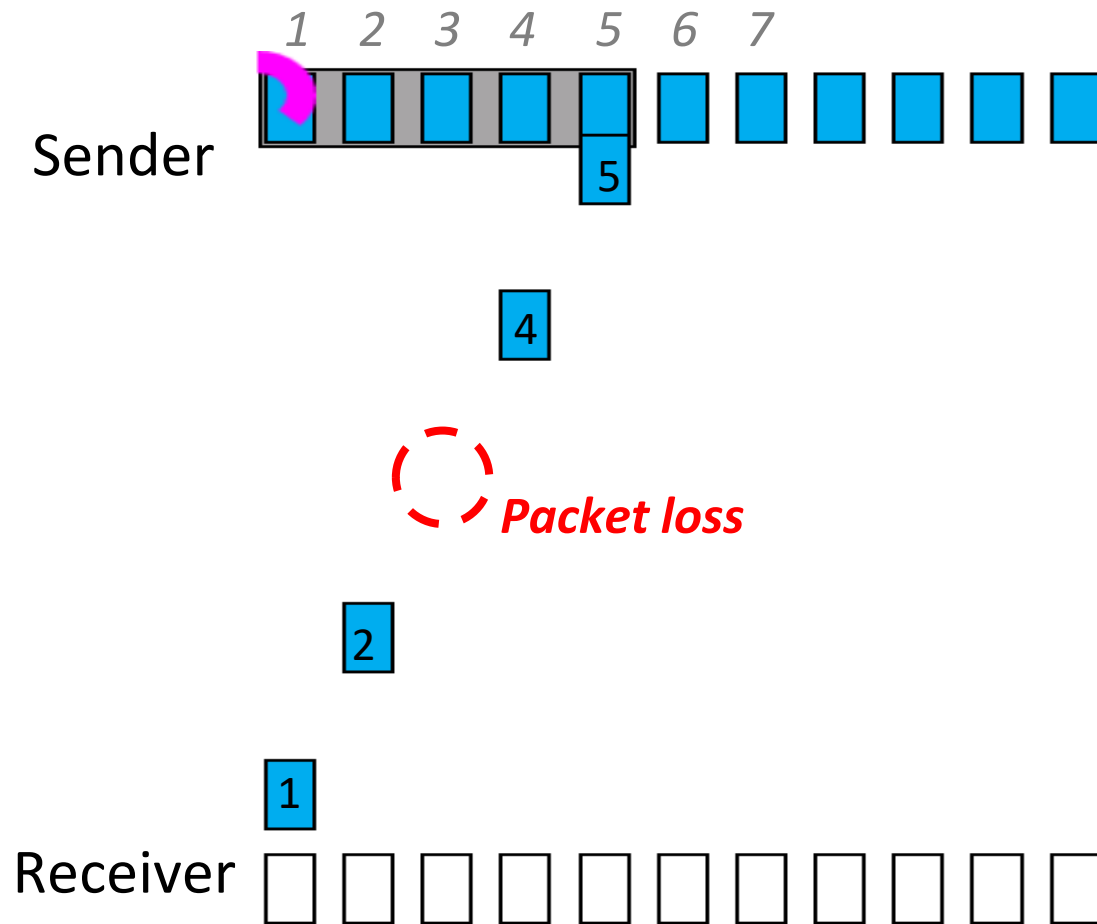
Go-Back-N case 1 (working normally)



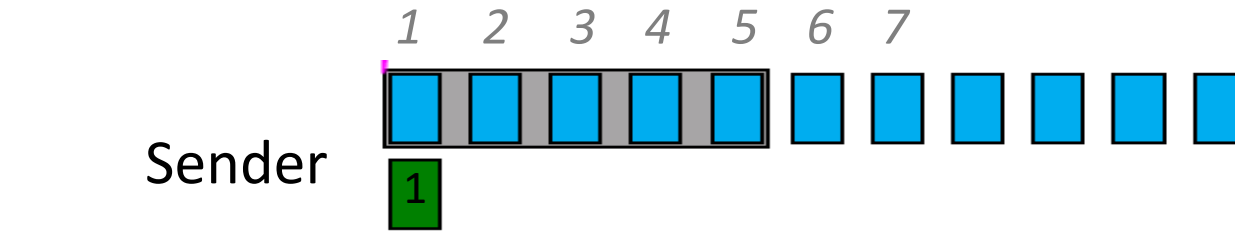
Go-Back-N case 1 (working normally)



Go-Back-N case 2 (packet loss)



Go-Back-N case 2 (packet loss)

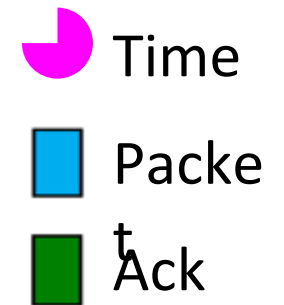


ACK-2 triggered
by packet-2

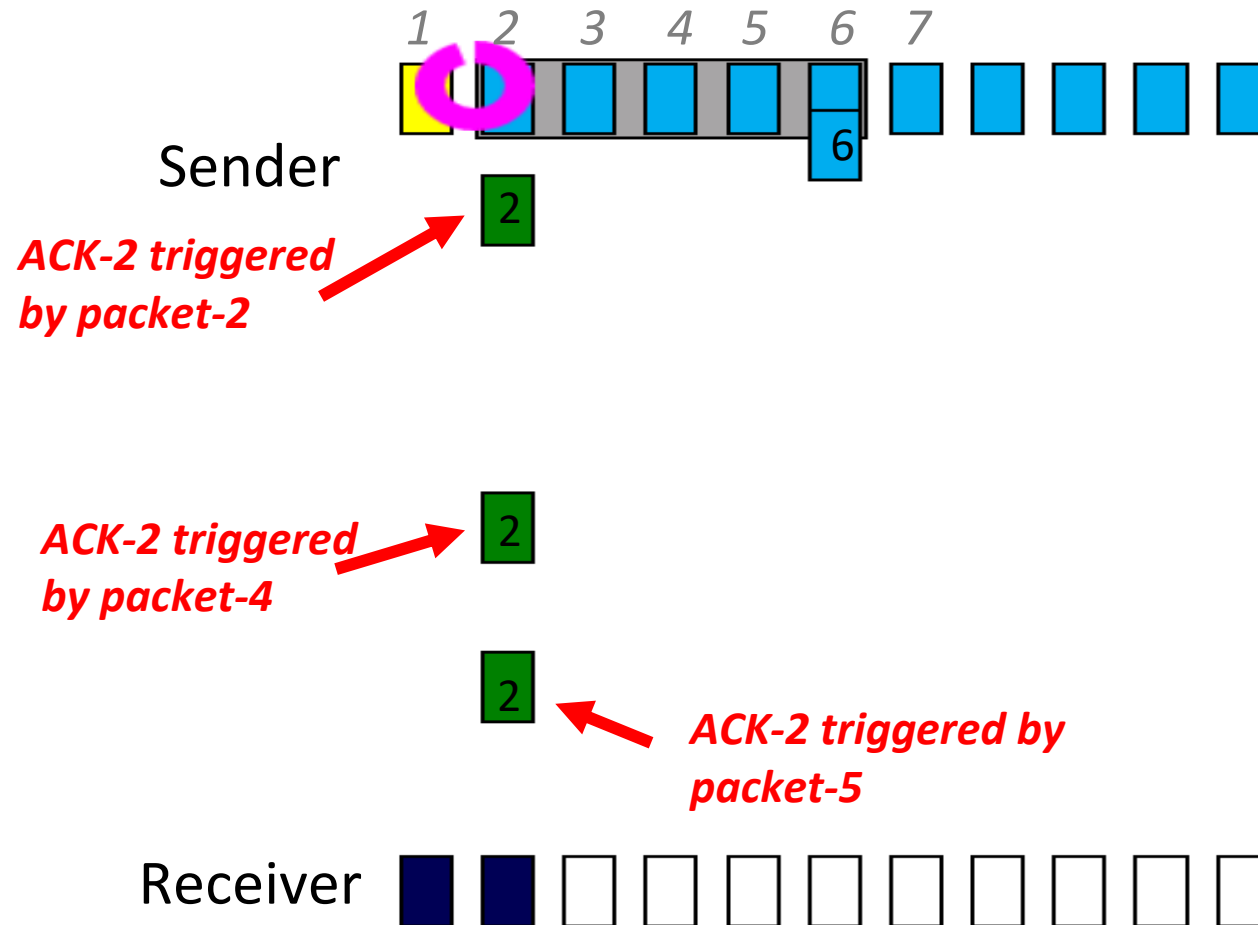
ACK-2 triggered
by packet-4

ACK-2 triggered by
packet-5

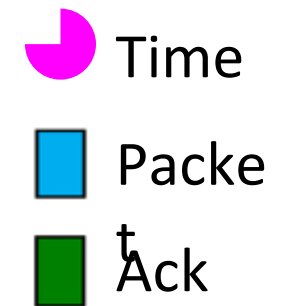
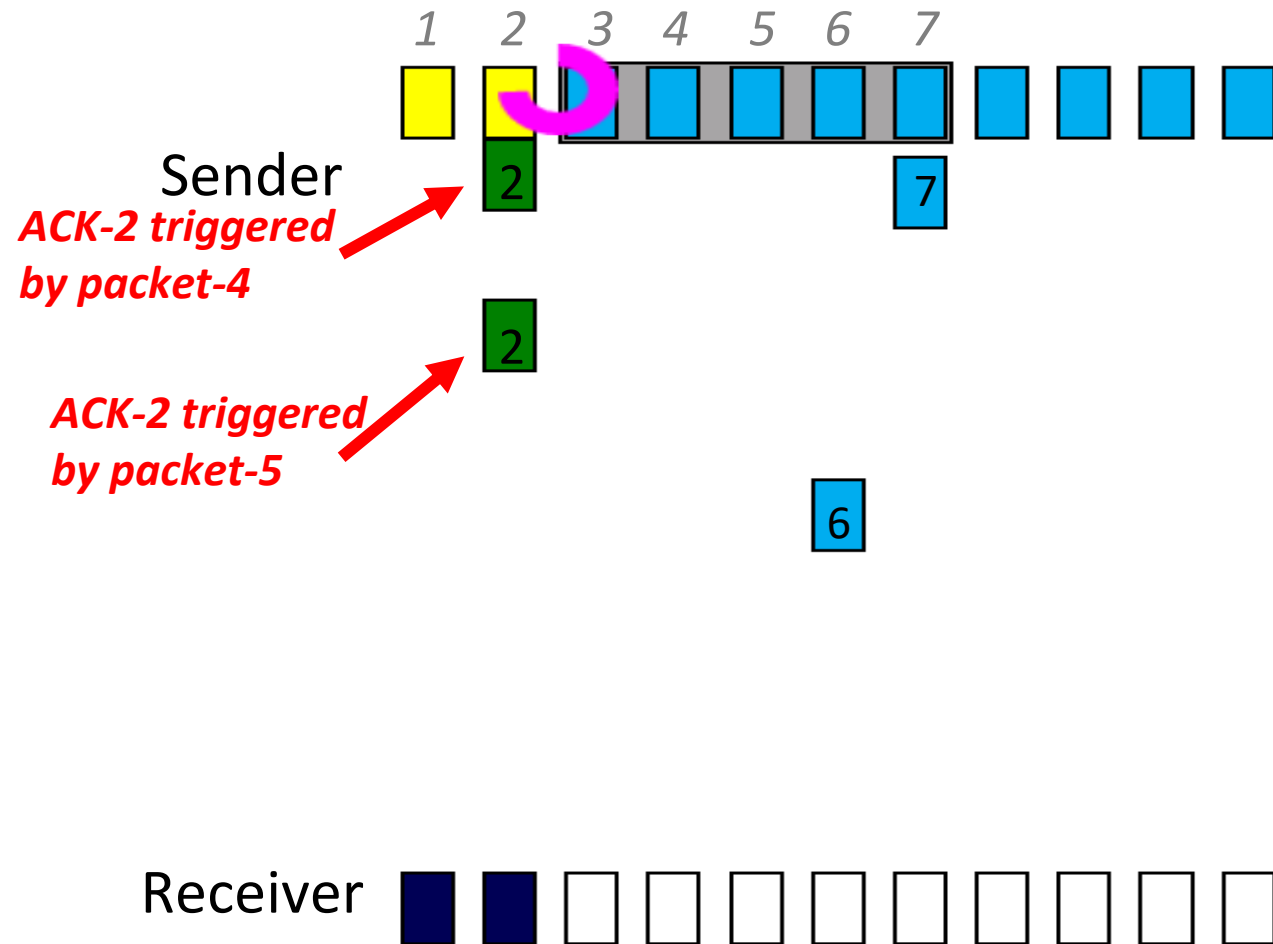
Receiver



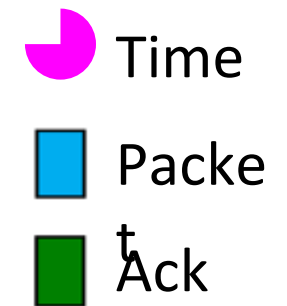
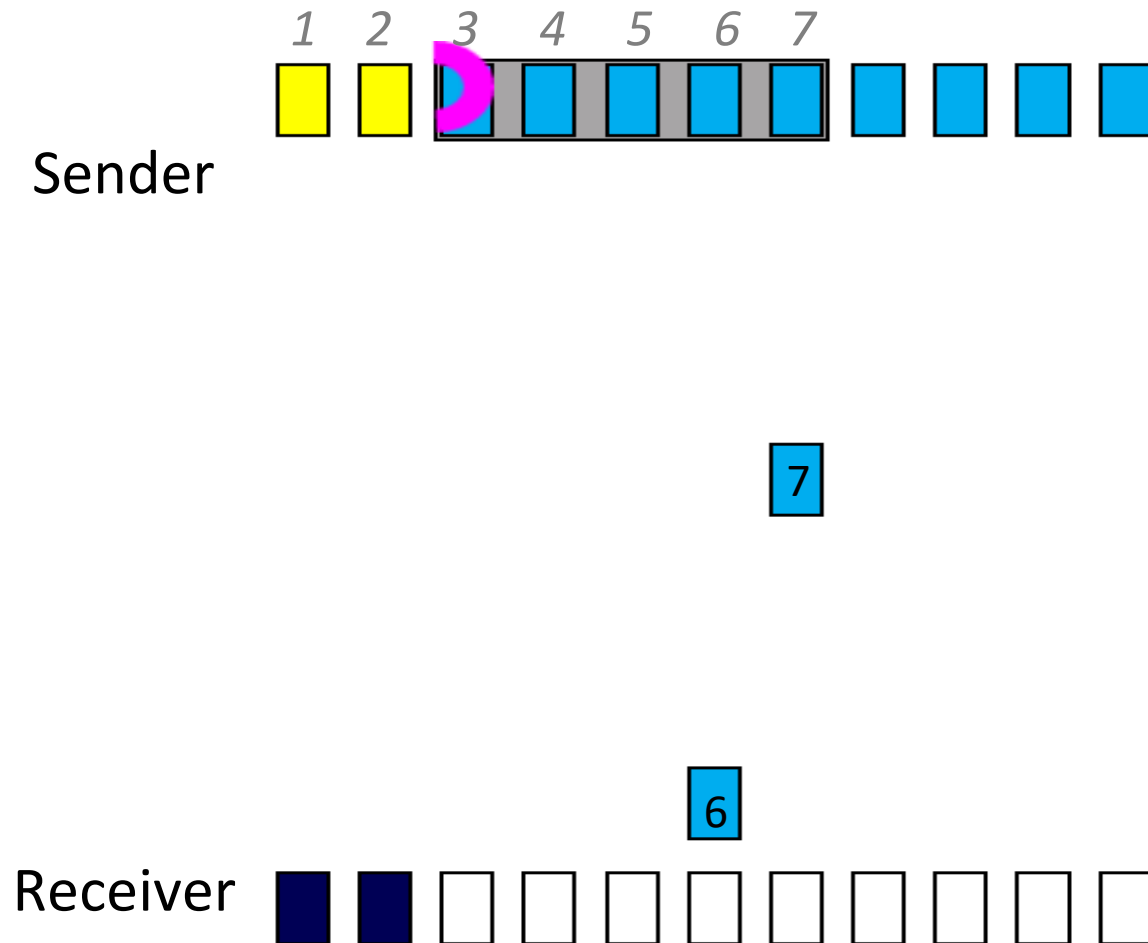
Go-Back-N case 2 (packet loss)



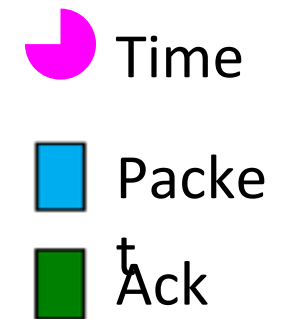
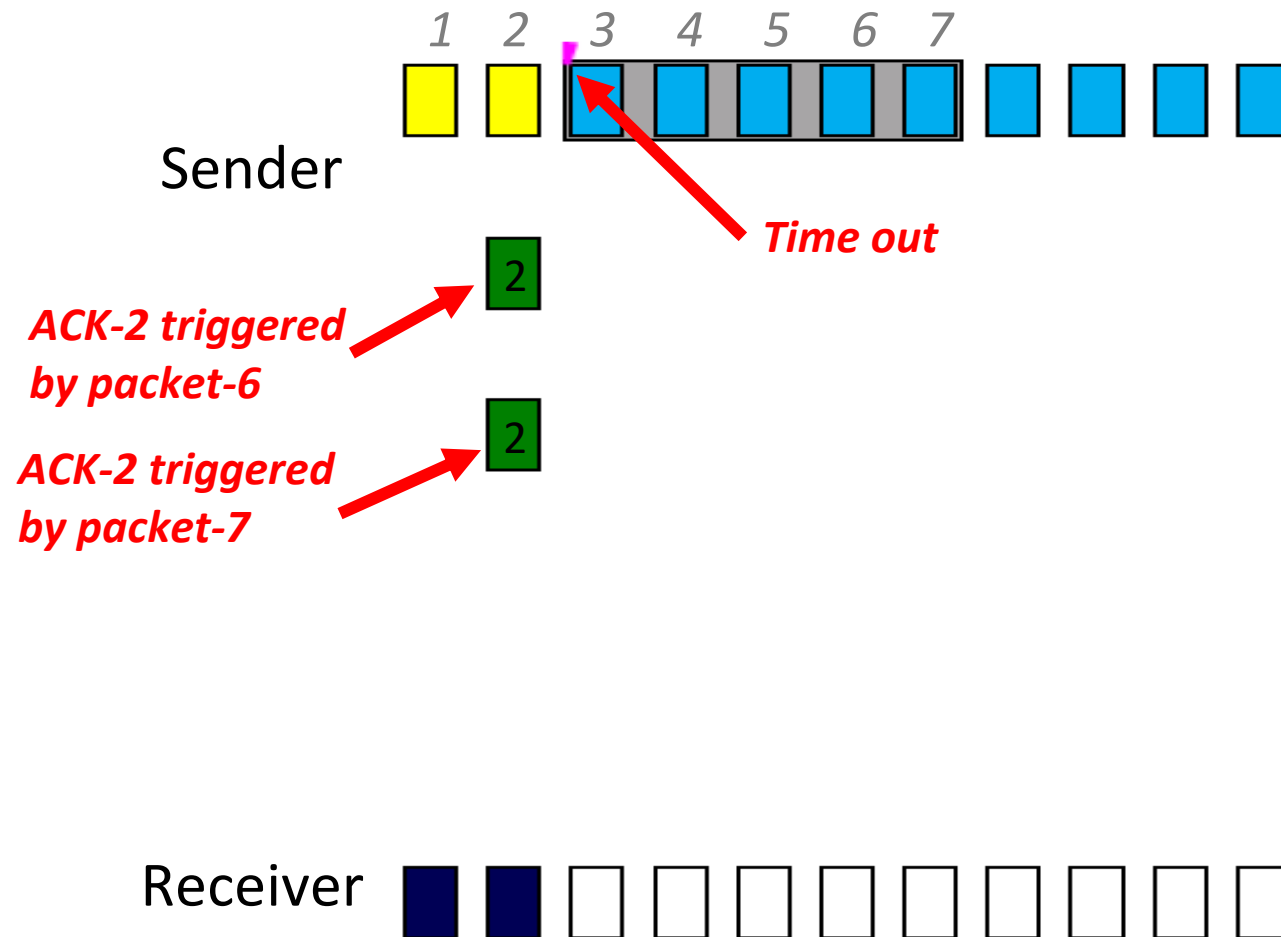
Go-Back-N case 2 (packet loss)



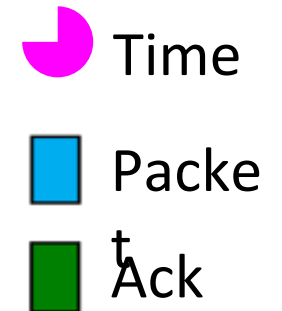
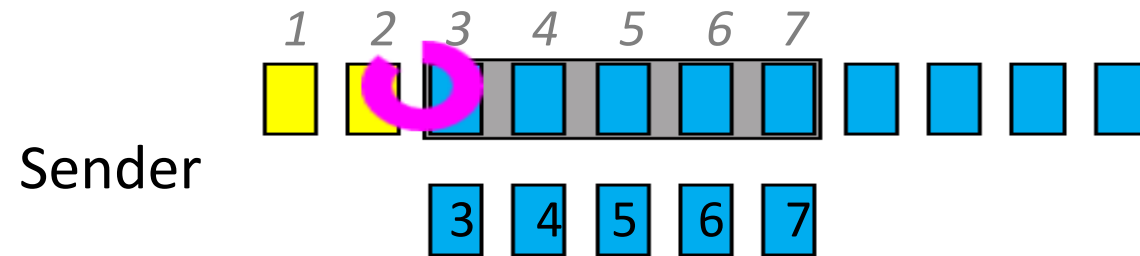
Go-Back-N case 2 (packet loss)



Go-Back-N case 2 (packet loss)

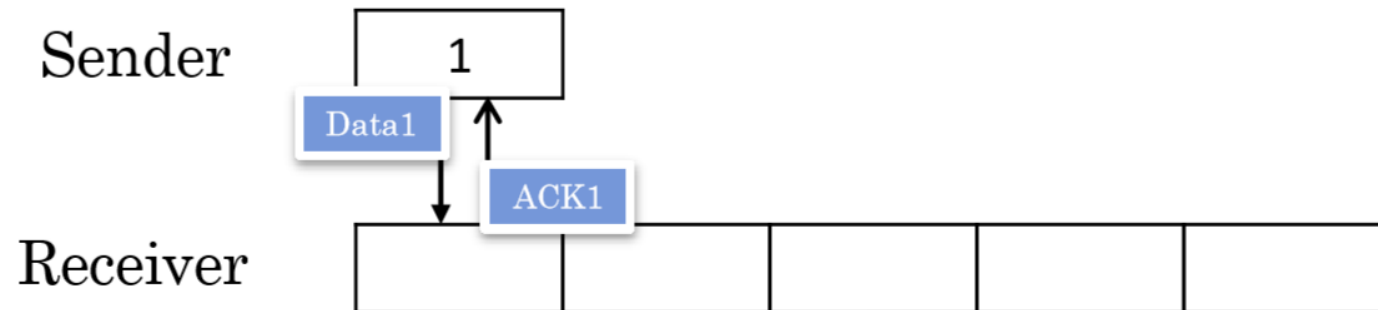


Go-Back-N case 2 (packet loss)



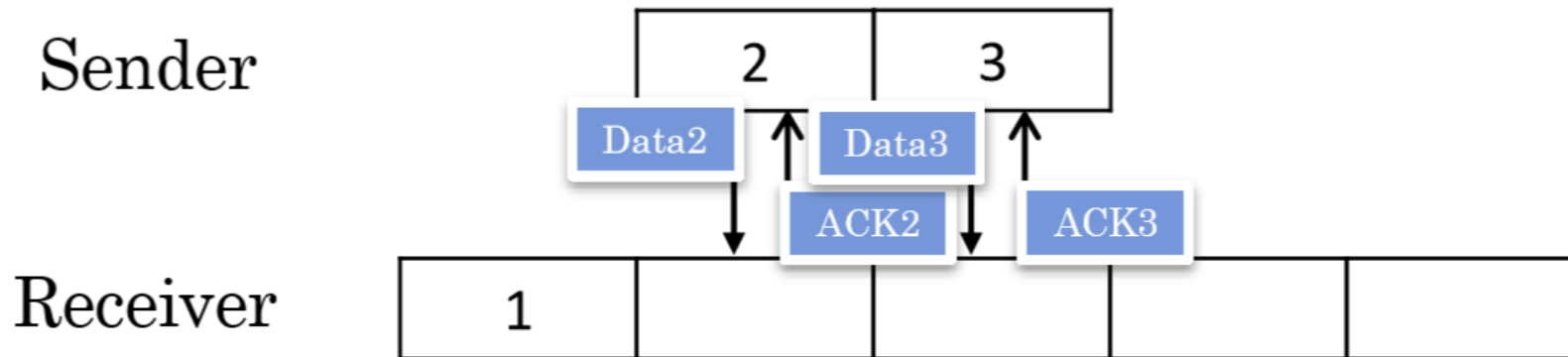
Example (***GBN + Congestion control***)

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



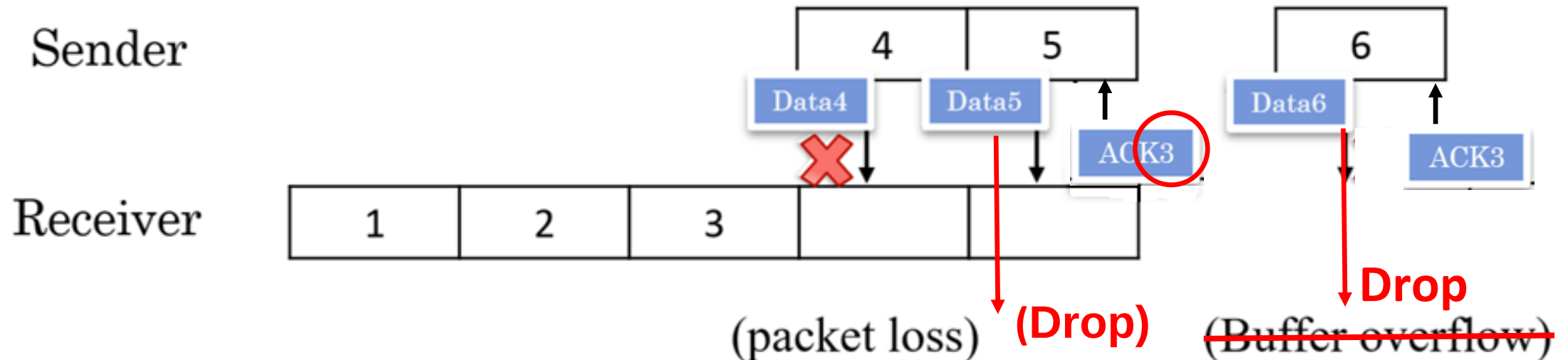
Example (***GBN + Congestion control***)

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



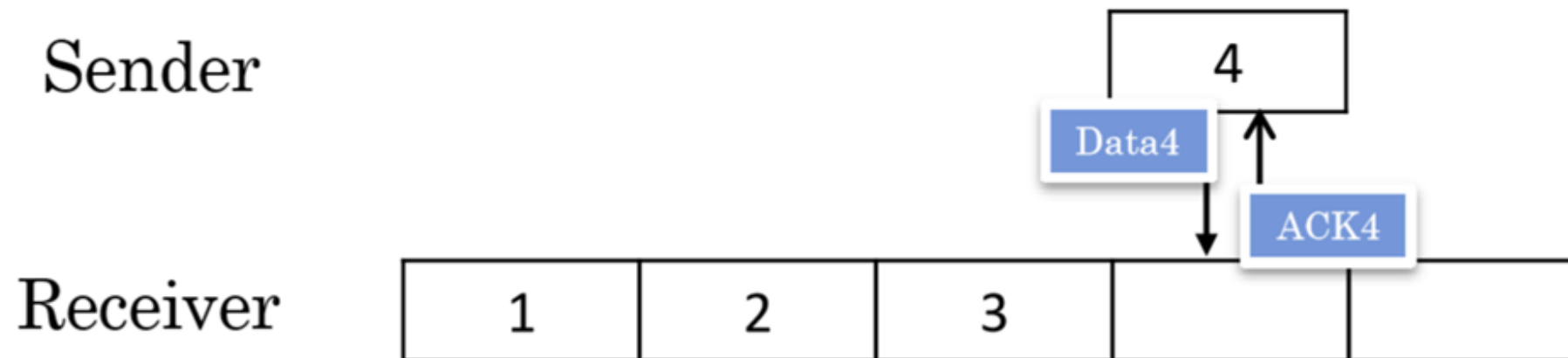
Example (***GBN + Congestion control***)

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



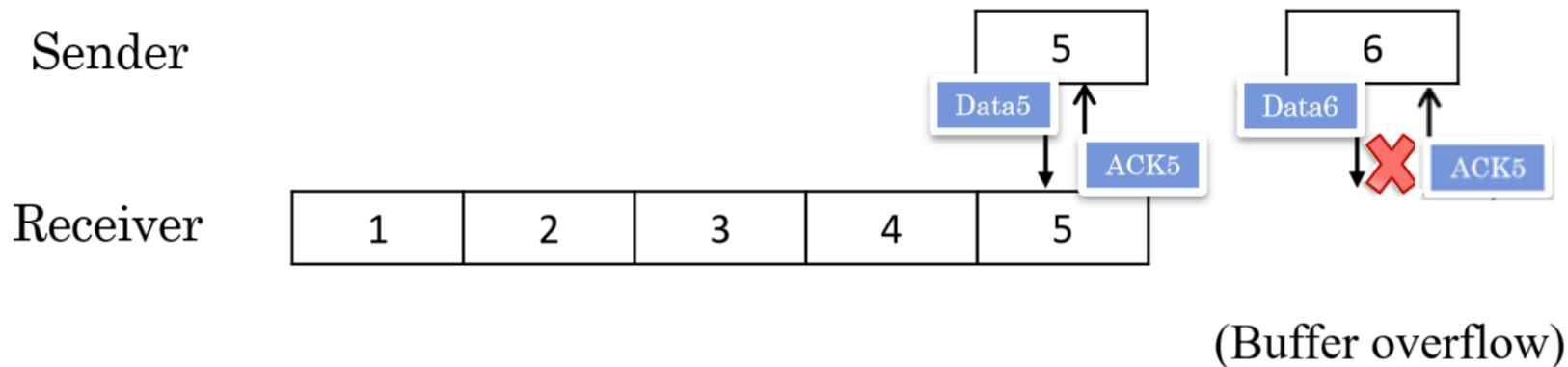
Example (***GBN + Congestion control***)

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



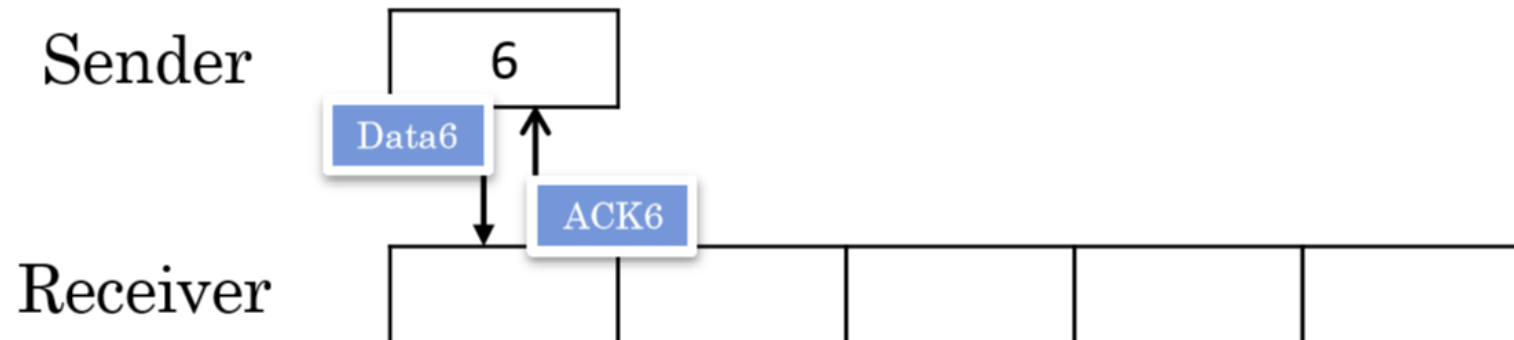
Example (***GBN + Congestion control***)

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



Example (***GBN + Congestion control***)

- Sender sends Data 6
- Congestion Window =1; Threshold =1
- Receiver sends ACK 6
- And so on...



Requirement

Agent

- Forward data and ACK packets
- Randomly drop data packet [***DO NOT DROP ACK PACKETS***]
- Compute loss rate



Requirement

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

Requirement

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
```

Requirement

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
```

Requirement

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
```

Requirement

The format used for transmission should be the same as follow:

```
21 typedef struct{
22     int length;
23     int seqNumber;
24     int ackNumber;
25     int fin;
26     int syn;
27     int ack;
28 } header;
29
30 typedef struct{
31     header head;
32     char data[1000];
33 } segment;
```

fin: 0 or 1
syn: 0 or 1 (just make it 0)
ack: 0 or 1

Requirement

Settings

- Sender
 - Arguments: IP, Port, path of source file,... etc.
 - Default threshold: **16**
 - Input file may include media file or text file, etc.(e.g. ***./sender text.txt***)
- Receiver
 - Arguments: IP, port ,path of destination file, ... etc.
 - Default buffer size: **32 segments**
 - Output file name: **result.??** (Filename Extension is the same as the input file)
- Agent
 - Arguments: IP, port, loss rate, ... etc.

Requirement

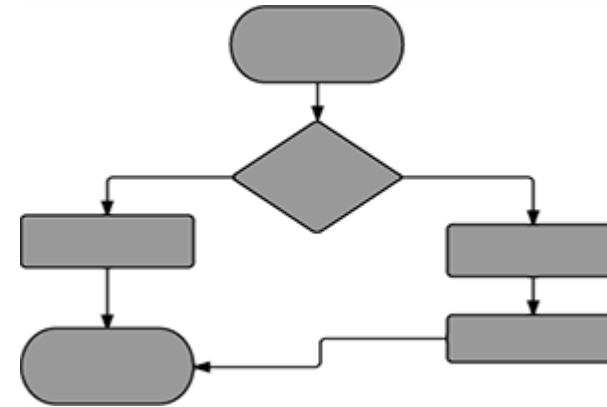
Settings

- File Size
 - More than 0.5 MB (500 KB)
- Data packet size (payload)
 - 1 KB
- Time out
 - Less than or equal to 1 sec ($\leq 1 \text{ sec}$)

Requirement

Document

- Format
 - A4, at most 2 pages
 - Digital **PDF file only**, “HW2-Report.pdf”
- Content
 - How to execute your program
 - Explain your program structure(including **3 flow charts** for sender, agent, and receiver)
 - Difficulties and Solutions



Grading and Submission



Grading and Submission

Grading (100%)

Basic requirement (10%)

Socket programming with UDP

Language: No restriction

Without crash

Reliable transmission (20%) (page 7)

Congestion control (25%) (page 8)

Buffer handling (15%) (page 9)

Agent (10%) (page 26)

Message format (5%) (page 27)

Document (5%) (page 33)

Demo (10%) (page 36)

Grading and Submission

Demo (10%)

- Please fill the demo form (will be announced on course website)
- Come to demo on time
- Discount for those are not on time
- You will get ***ZERO*** for this homework if you ***don't*** demo.

Grading and Submission

Submission Deadline

- 2018/12/26 (Wed.) 23:59 (UTC+8)
- Late submission: 20% off per day
- **NOT** accept after 23:59, 12/28, 2018

Naming

- [Student ID].zip Ex: r069xxxxx.zip
- Email subject: [CN2018] Homework2_studentID
- Email: ntu.cnta@gmail.com