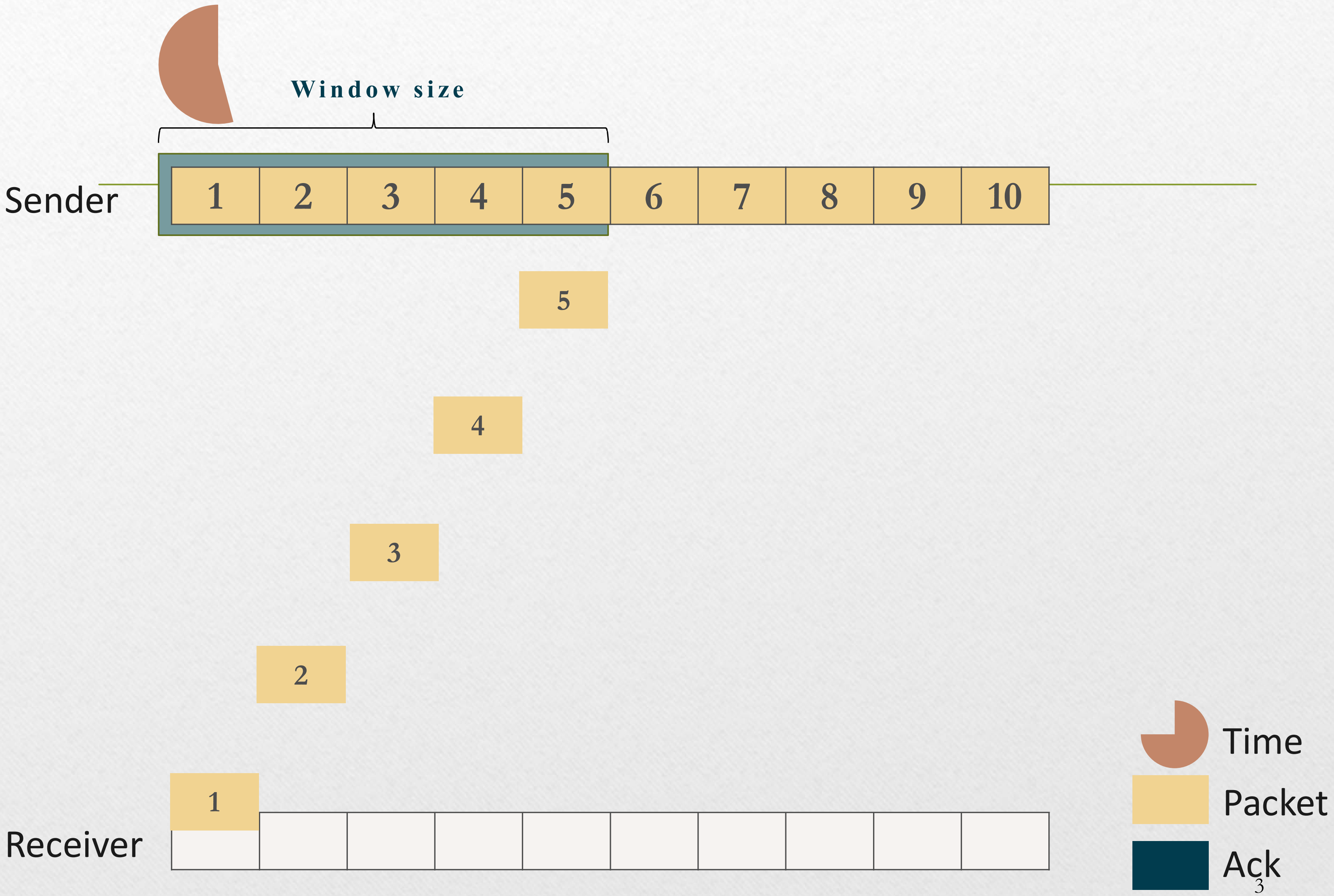
AI–Chun Pang / Instructor

戴維均 陳昇 / T.A.s

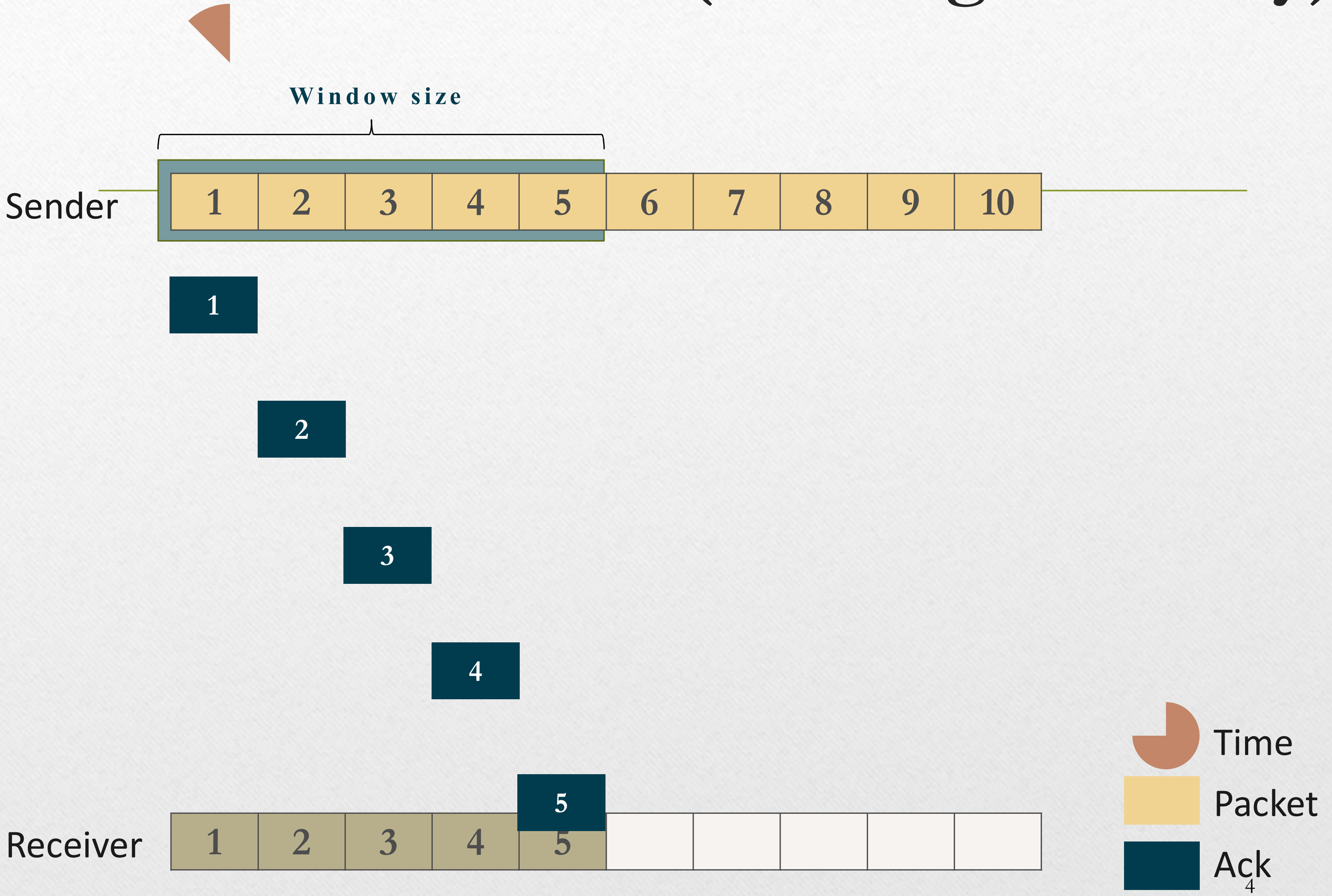
Assignment 3 - Retransmission & Congestion Control

What is Go-Back-N?

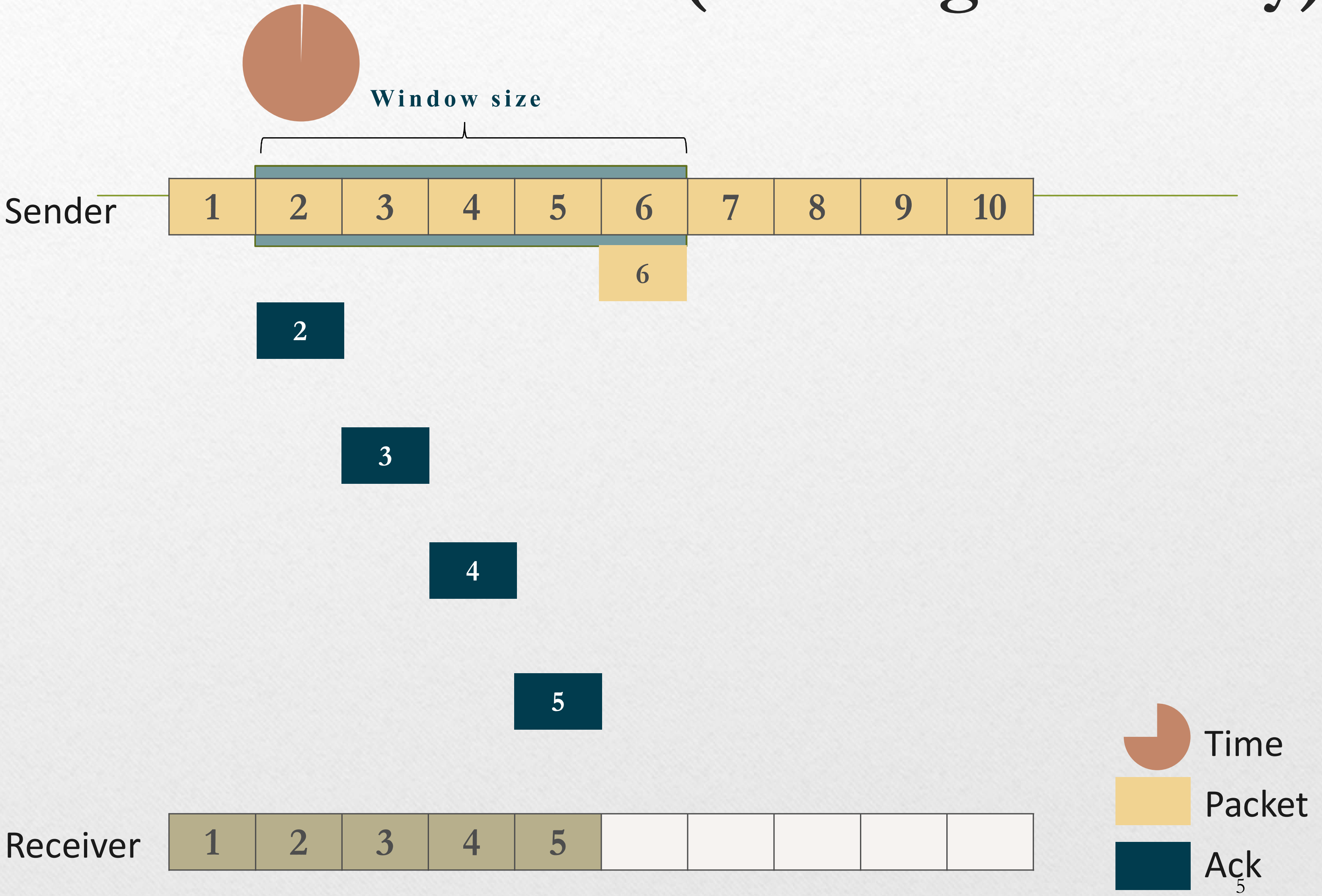
Go-Back-N case 1 (working normally)



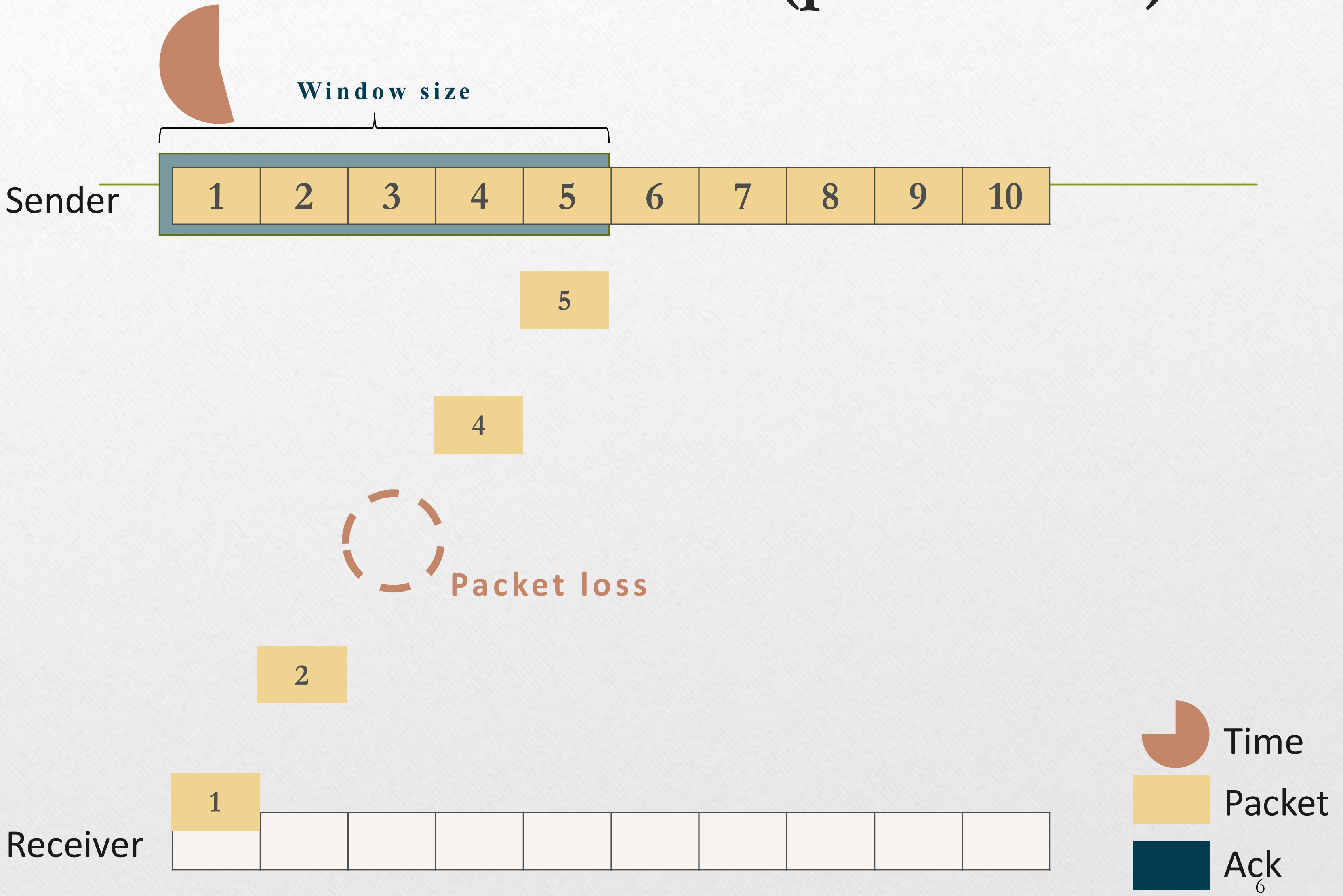
Go-Back-N case 1 (working normally)



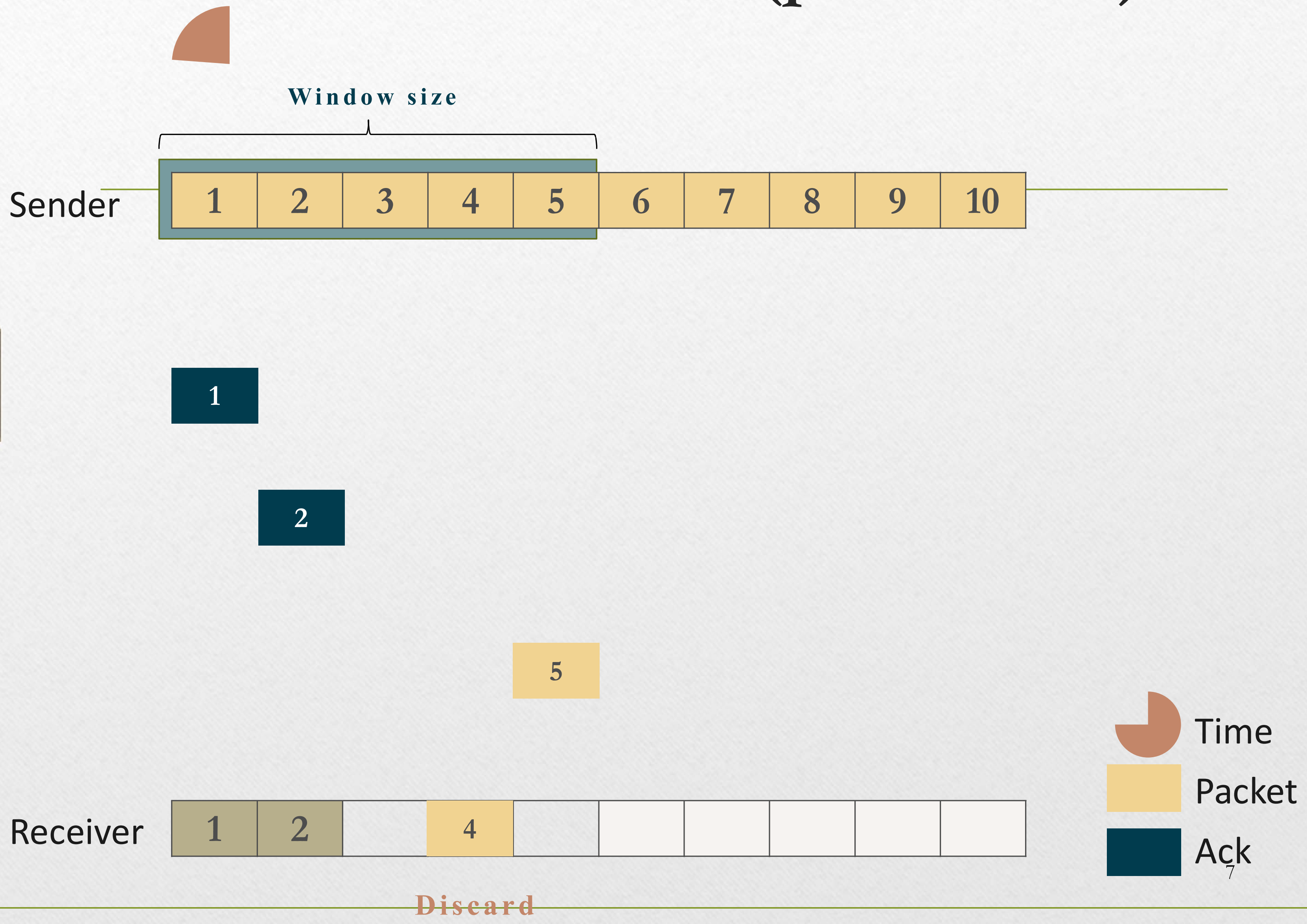
Go-Back-N case 1 (working normally)



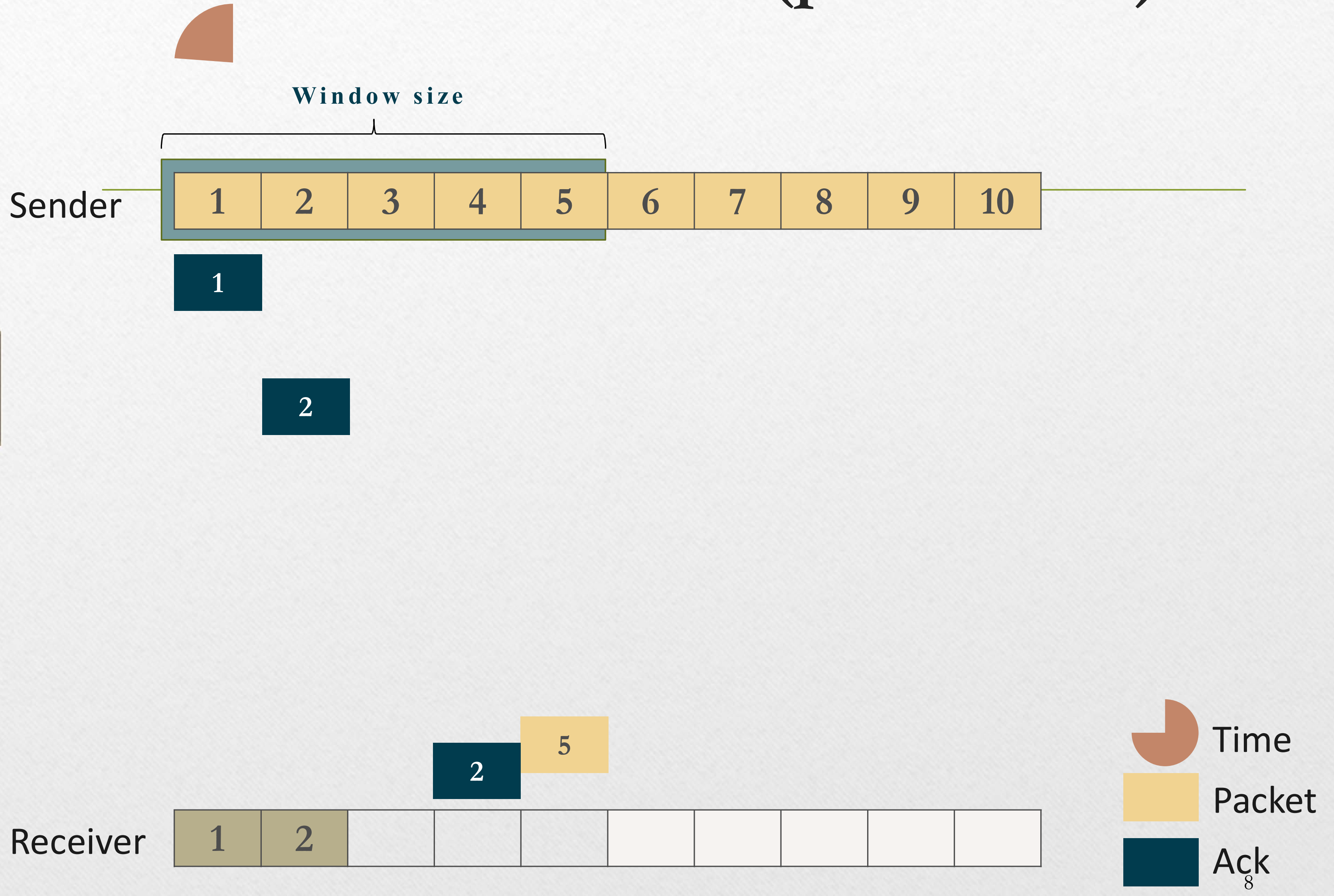
Go-Back-N case 2 (packet loss)



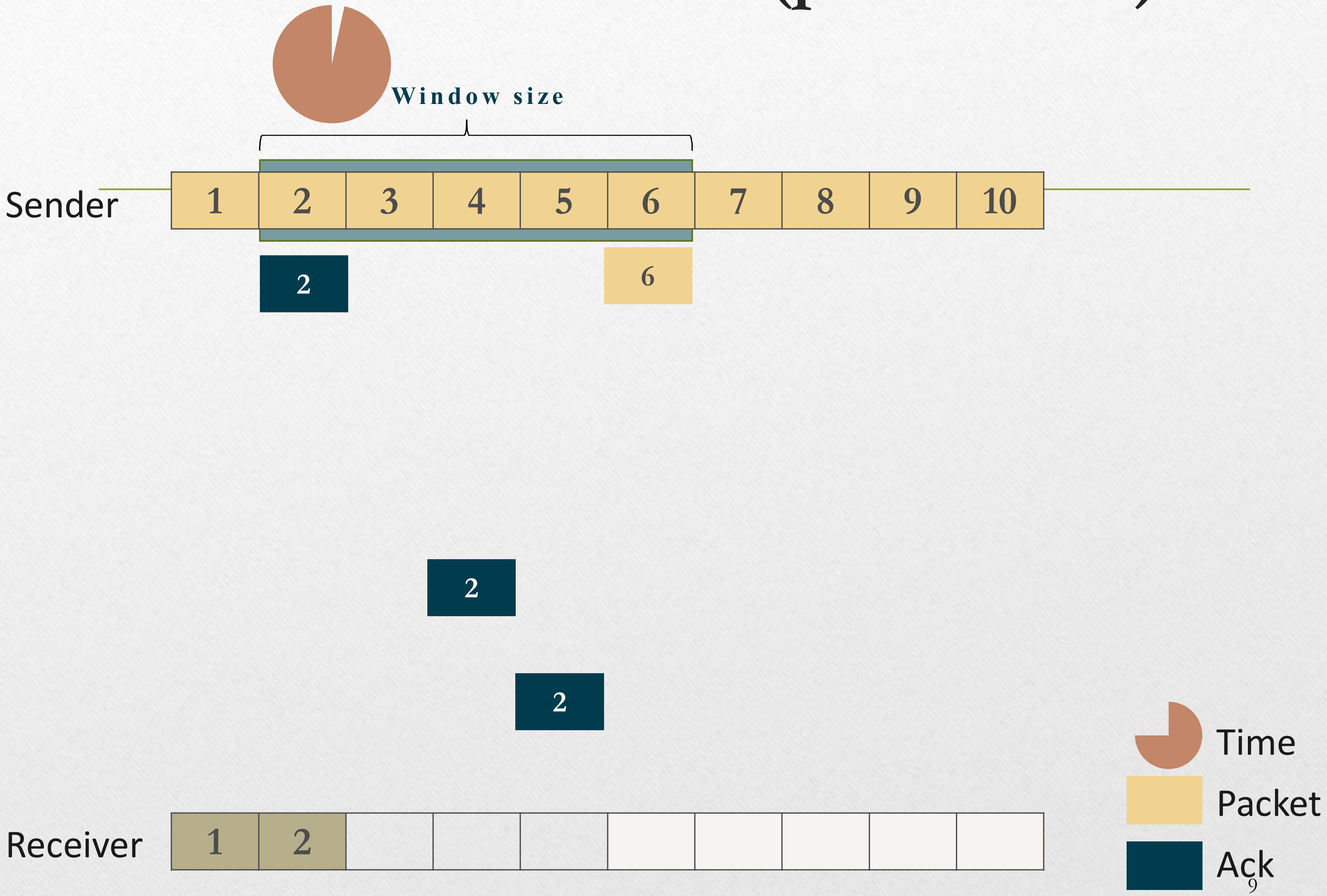
Go-Back-N case 2 (packet loss)



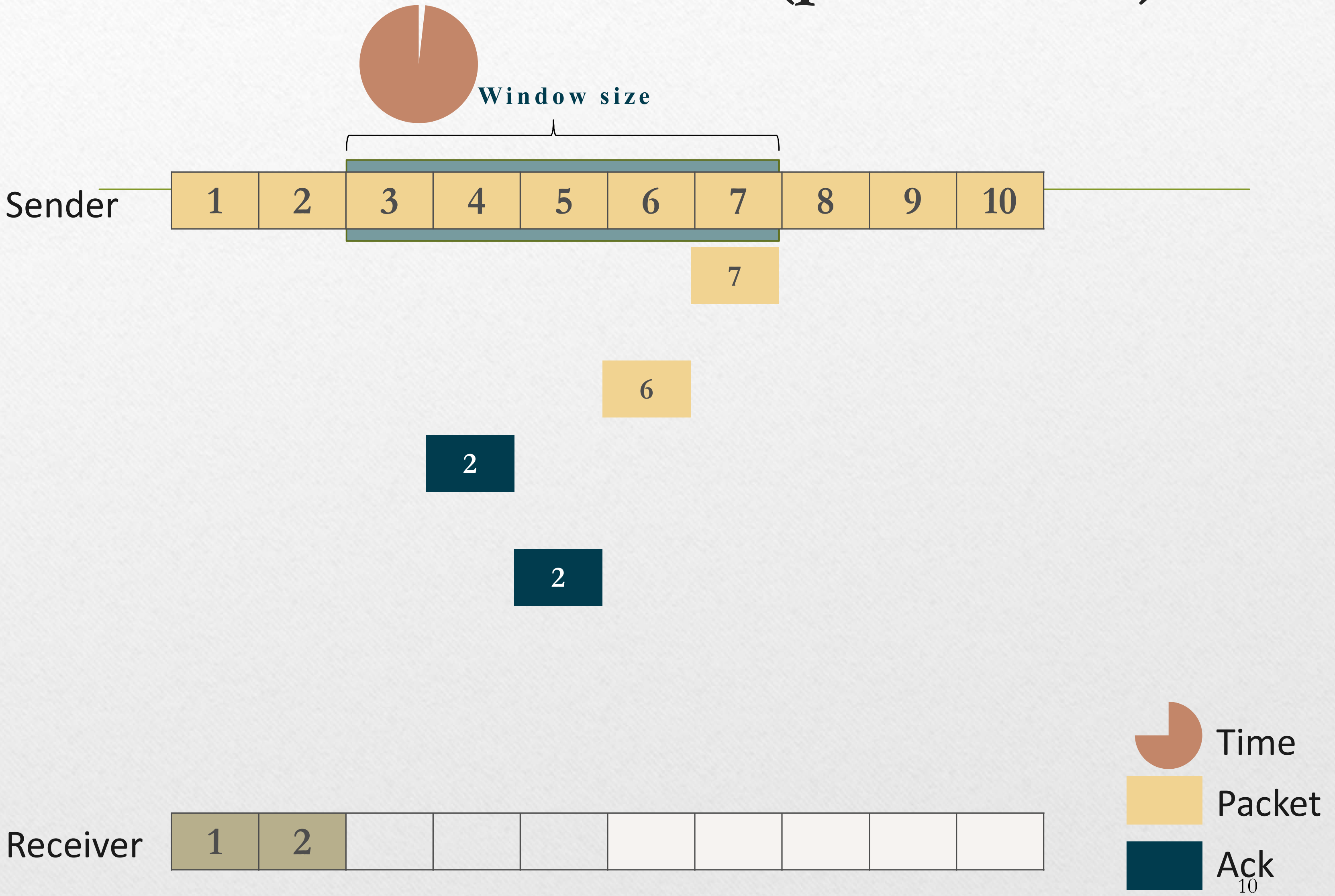
Go-Back-N case 2 (packet loss)



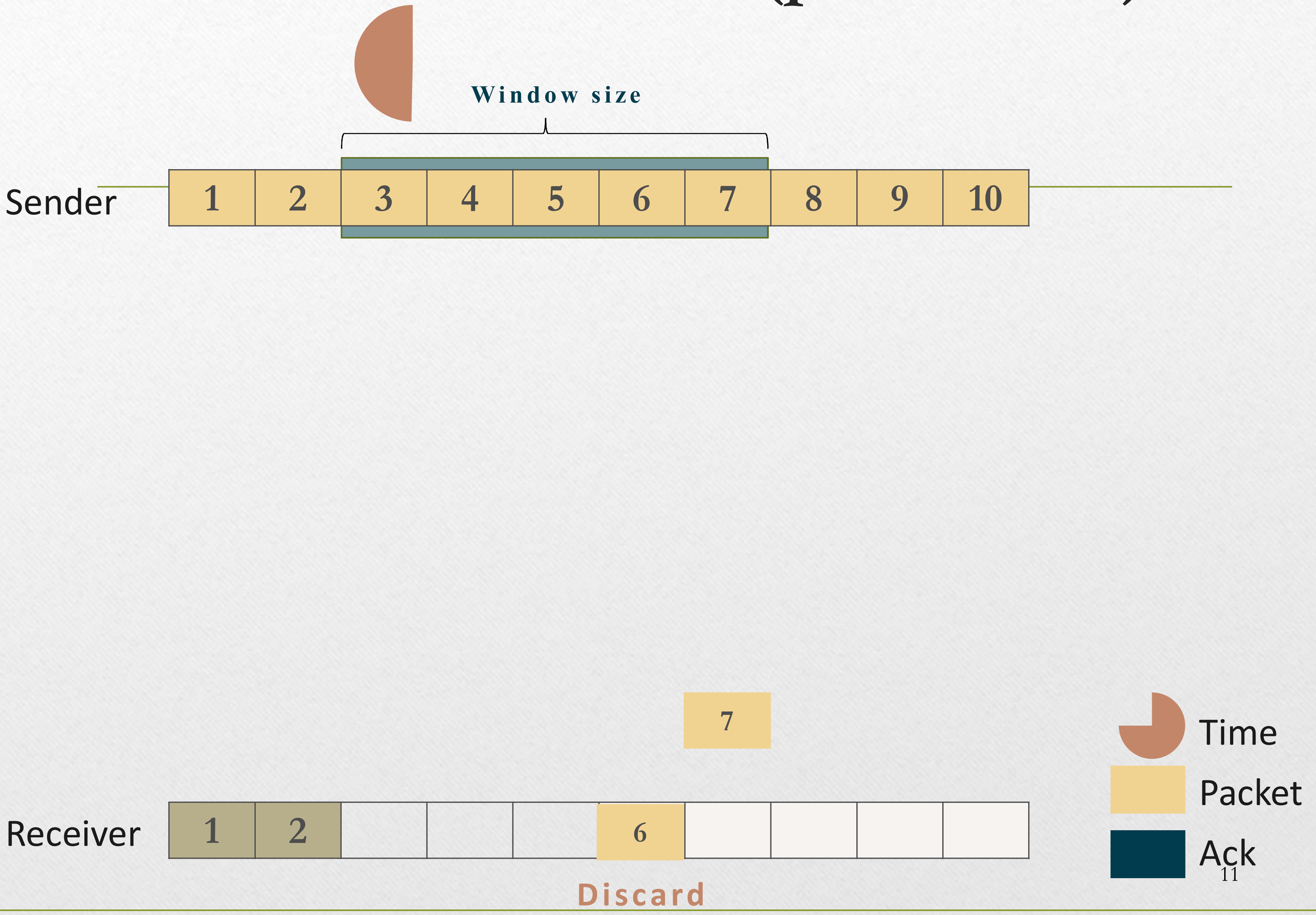
Go-Back-N case 2 (packet loss)



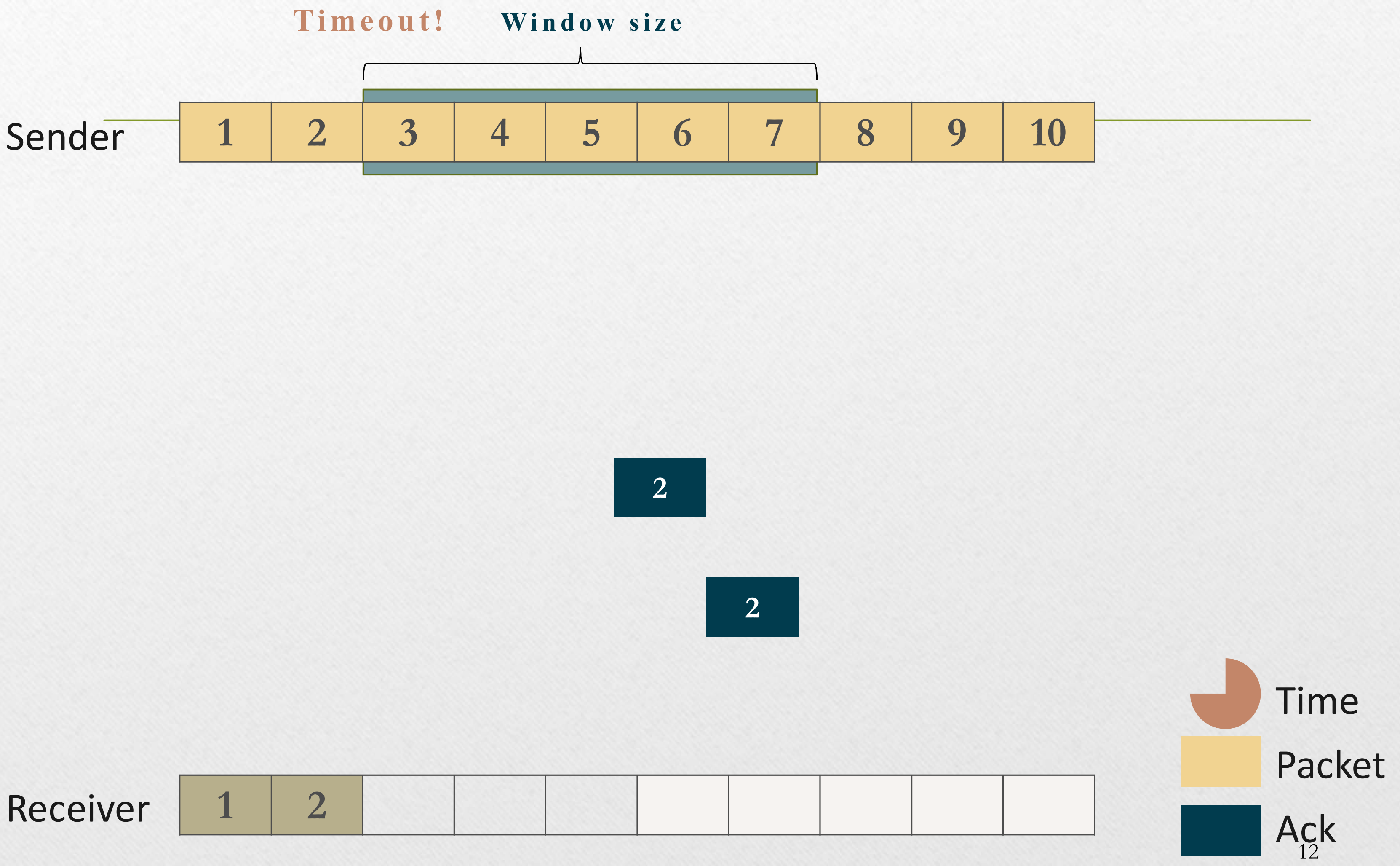
Go-Back-N case 2 (packet loss)



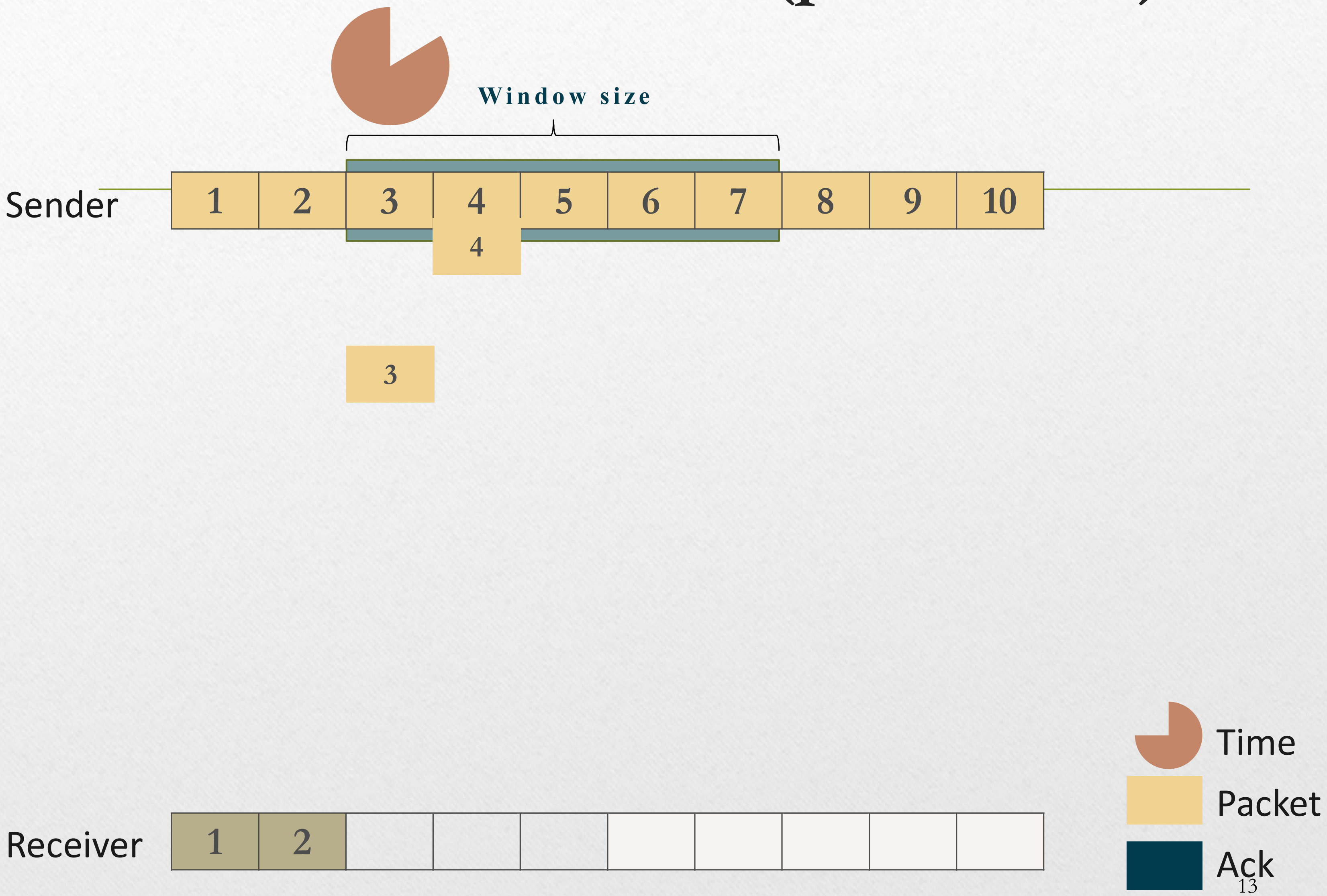
Go-Back-N case 2 (packet loss)



Go-Back-N case 2 (packet loss)



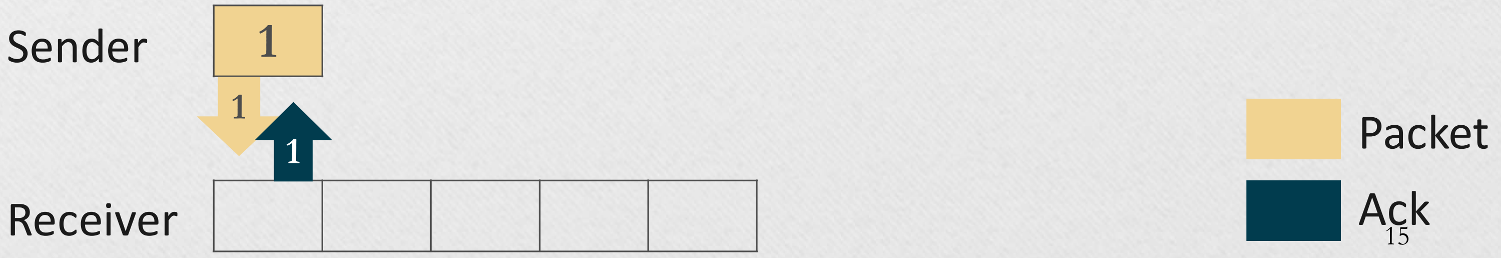
Go-Back-N case 2 (packet loss)



Go-Back-N with Congestion Control

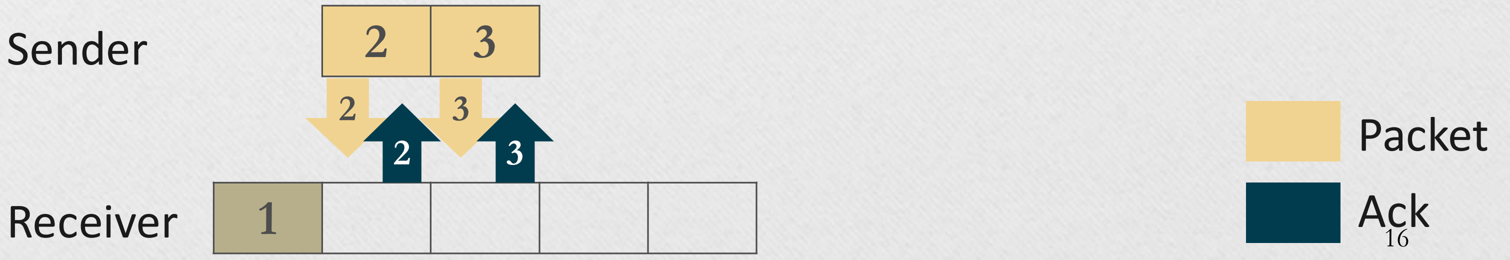
Go-Back-N + Congestion Control

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



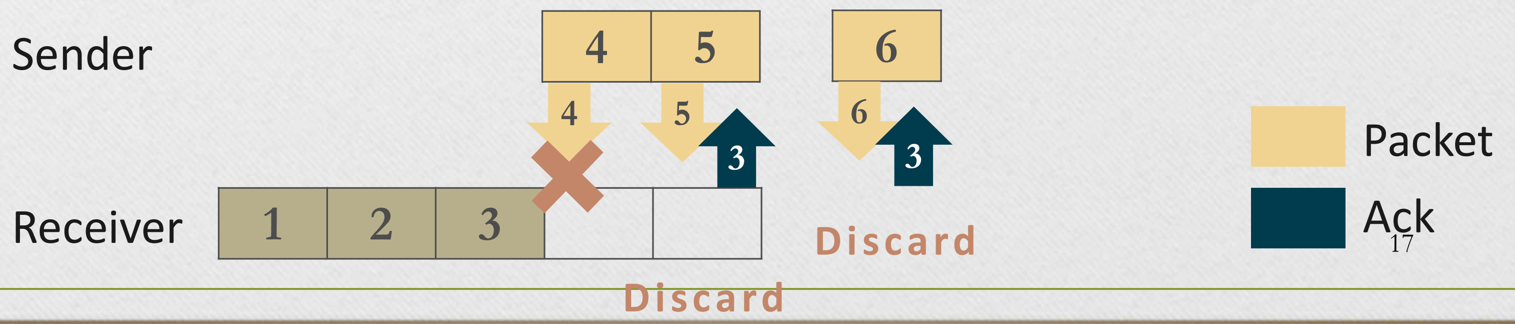
Go-Back-N + Congestion Control

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



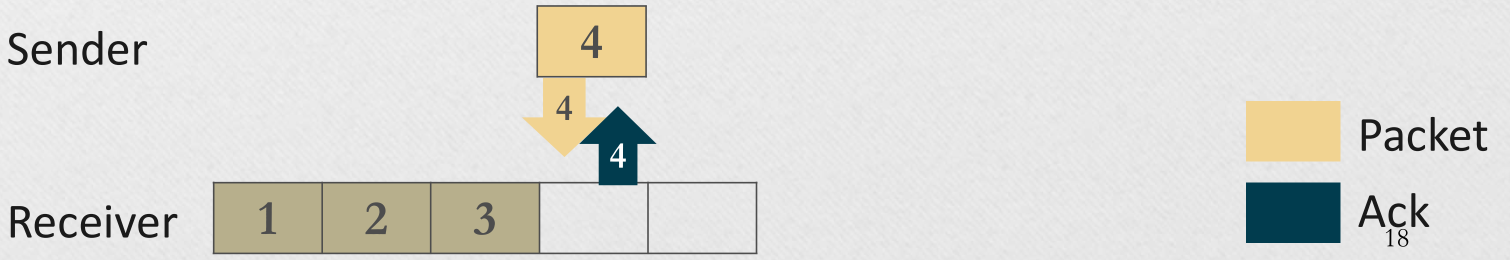
Go-Back-N + Congestion Control

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



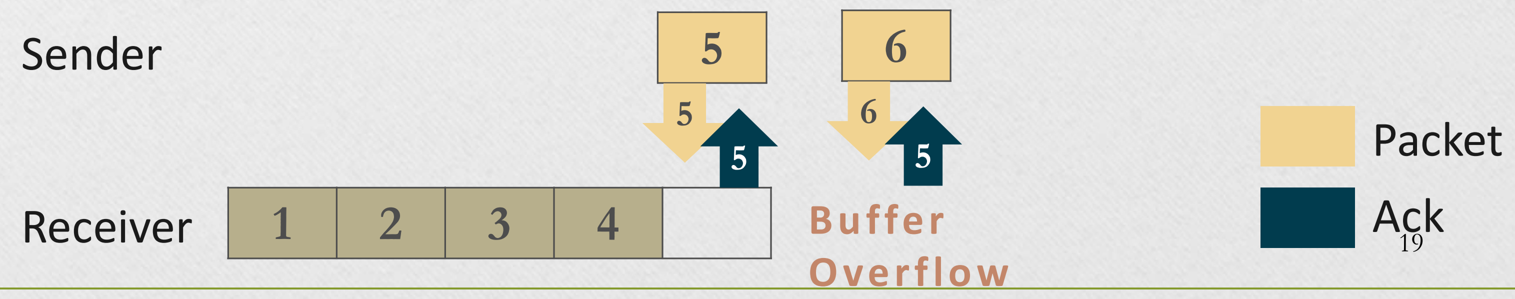
Go-Back-N + Congestion Control

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



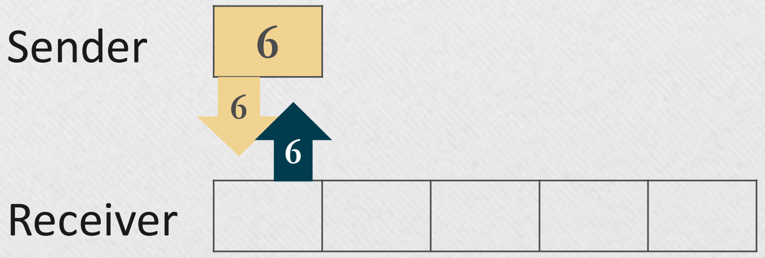
Go-Back-N + Congestion Control

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



Go-Back-N + Congestion Control

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

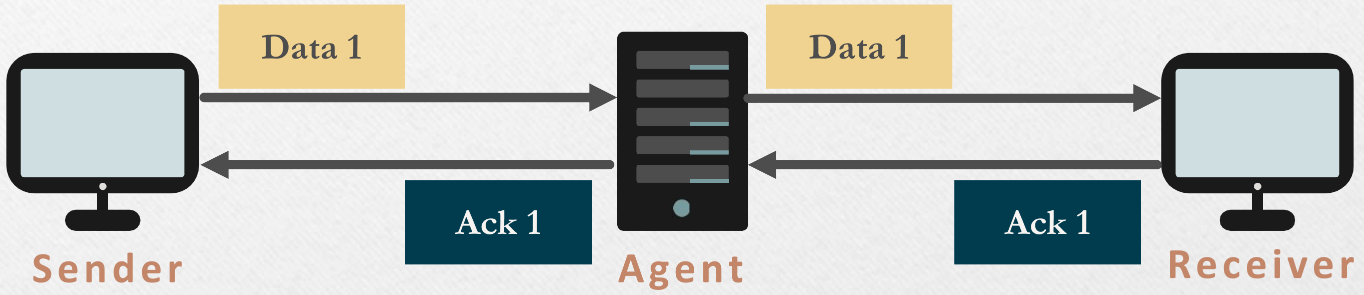


🟨	Packet
🟦	Ack

Assignment 3 Announcement

Specification (1/10)

- **Implement three components: sender, receiver and agent.**



- **Sender / Receiver**
  - Send / receive **video frame** by UDP
  - Provide reliable transmission
  - Congestion control
- Agent
  - Forward Data & ACK packets
  - **Randomly drop data packets**
  - Compute loss rate

Specification (2/10)

- **Reliable Transmission**
  - Data & ACK
  - Time out & Retransmission(Go-Back-N)
  - Sequence number
  - Completeness and correctness of transmitted file
- **Buffer handling** [receiver side]
  - Buffer Overflow:

    Drop the packets during out of buffer
  - Flush (write) to the file:

    Only when buffer overflows or all packets in range are received.

Specification (3/10)

- **Congestion Control** (sender sider)
  - 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 $\max\left(\left\lfloor \frac{window\ size}{2} \right\rfloor, 1\right)$
    2. Set window size to 1
    3. Retransmit – from the first "unACKed packet"

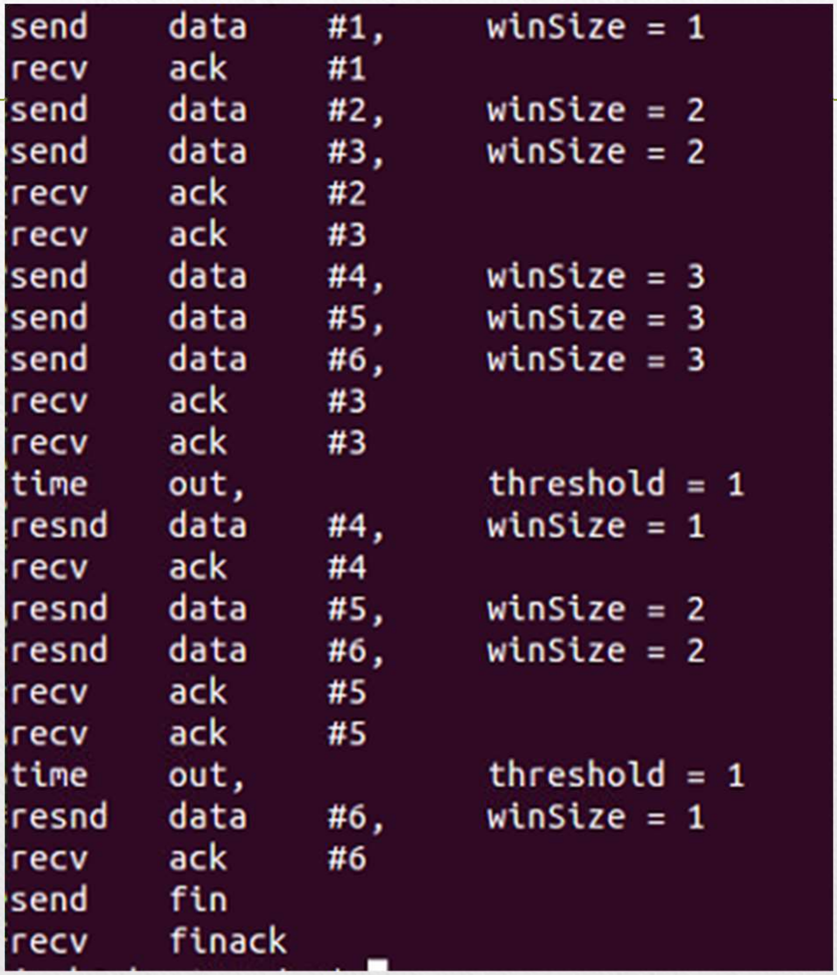
Specification (4/10)

- **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 (5/10)

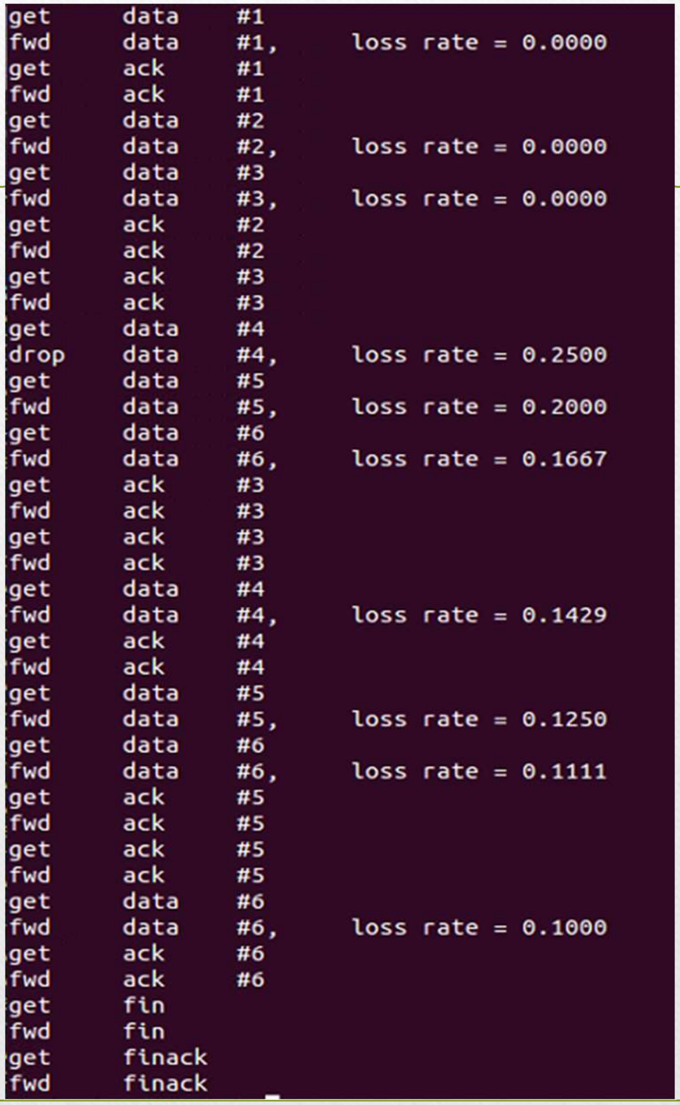
- **Show Message**
  - Sender:



```
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 (6/10)

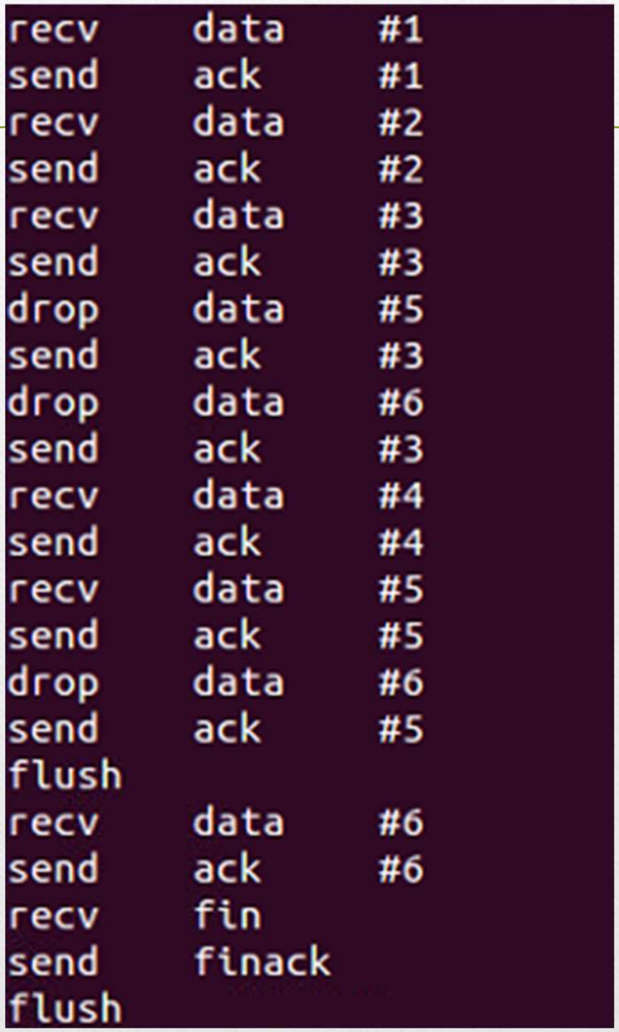
- **Show Message**
  - Agent:



```
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 (7/10)

- **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 (8/10)

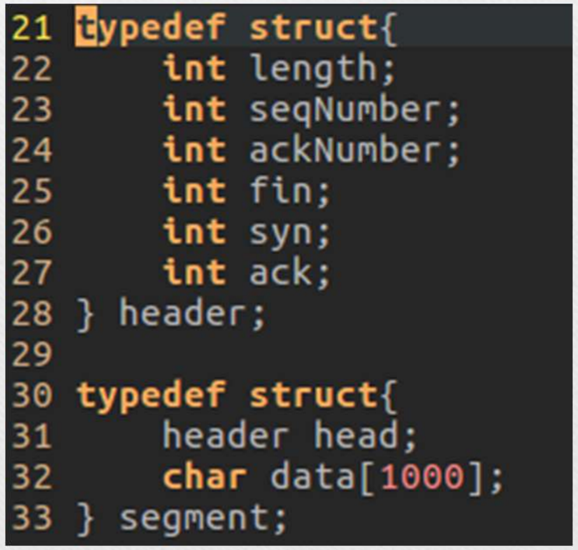
- **Show Message**

  - 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 (9/10)

- **Settings**
  - Sender
    - Arguments: IP, Port, path of source file,… etc.
    - Default threshold:16
  - Receiver
    - Arguments: IP, port, … etc.
    - Default buffer size: 32 segments
  - Agent
    - Arguments: IP, port, loss rate, … etc.
  - Data packet size (payload): 4KB
  - Time out:
    Less than or equal to 1 sec ($\leq 1\ sec$)

Specification (10/10)

- Makefile
  - You are required to write a Makefile for compilation.
  - Thus, the commands should be:
    - $make server // for server code
    - $make agent // for agent code
    - $make receiver // for receiver code
  - After the compilation, there should be 3 executables: server, agent and receiver.

Grading Policy (1/2)

This assignment accounts for 10% of the total score.

• Video Streaming		(15%)
- Correctly play the sample video in HW2		
- Transmit raw frames	(5%)	
- Transmit encoded frames	(10%)	
- Correctly play resolution-unknown videos	(5%)	
• Reliable transmission		(20%)
• Congestion control		(25%)
• Buffer handling		(10%)
• Agent		(9%)
- Randomly drop data packet	(5%)	
- Compute loss rate	(4%)	
• Show Message		(9%)
- Show message correctly	(3% * 3)	
• Report		(12%)
- How to execute your program	(3%)	
- Explain your program structure (including 3 flow charts for sender, agent and receiver)	(3% * 3)	

Grading Policy (2/2)

- **Submission**
  - Your report format must be in **".pdf"** format and named "report.pdf", or else **you will get 0 point** in the part.
  - Please put all the files **into a folder** named **hw3_<student id>**, compress the folder as a **.zip** file, and then submit the **.zip** file to NTU Cool. The zip filename is **hw3_<student id>.zip** .
  - If we **cannot compile or execute your code,** you will **have a chance to demo your results** in your own environment.
  - The penalty for **wrong format** is **10 points**.
  - **No plagiarism is allowed. A plagiarist will be graded 0.**
- **Deadline**
  - Due Date: 23:59:59, January 5th, 2021
  - Penalty for late submission after hard deadline is "10% per day".