

sender

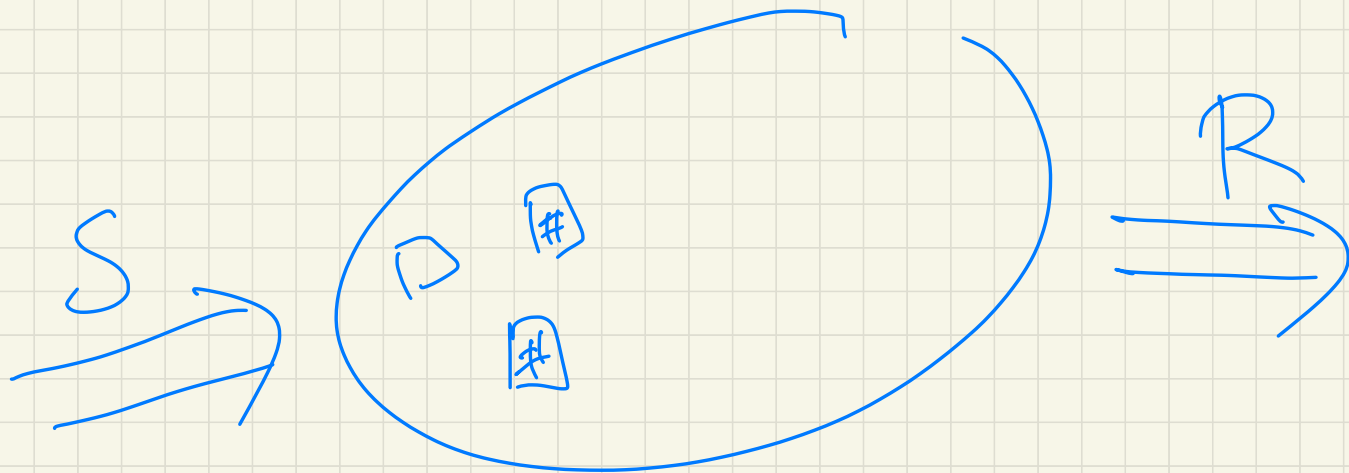
receiver

# PEER-TO-PEER CONNECTIVITY


## Service Models



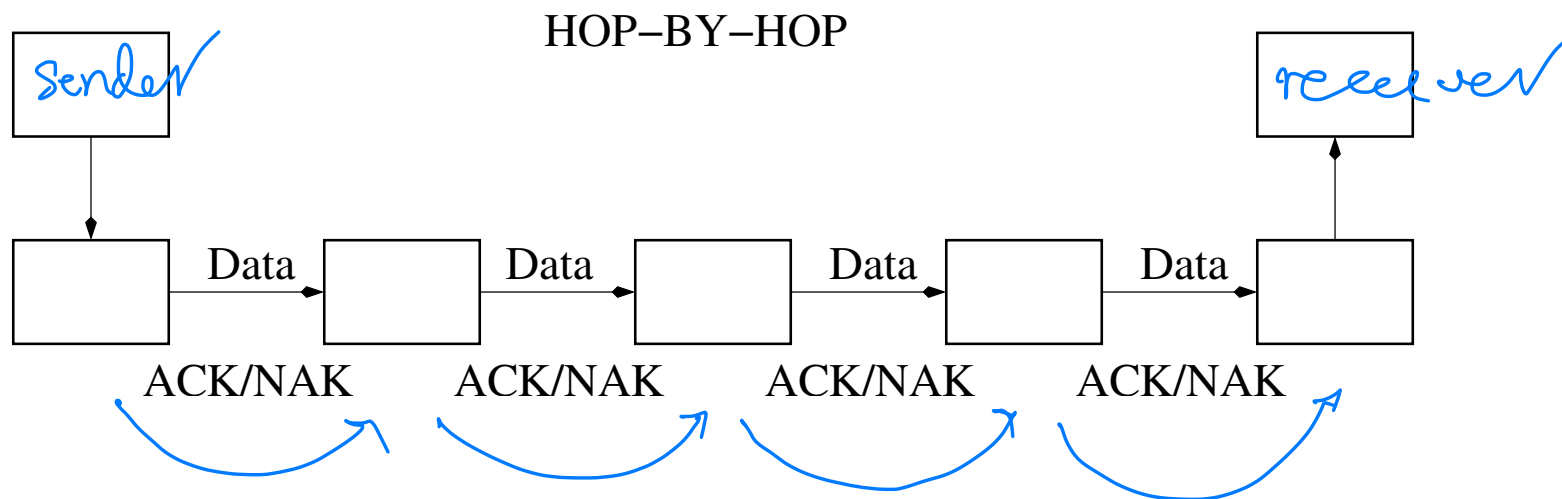
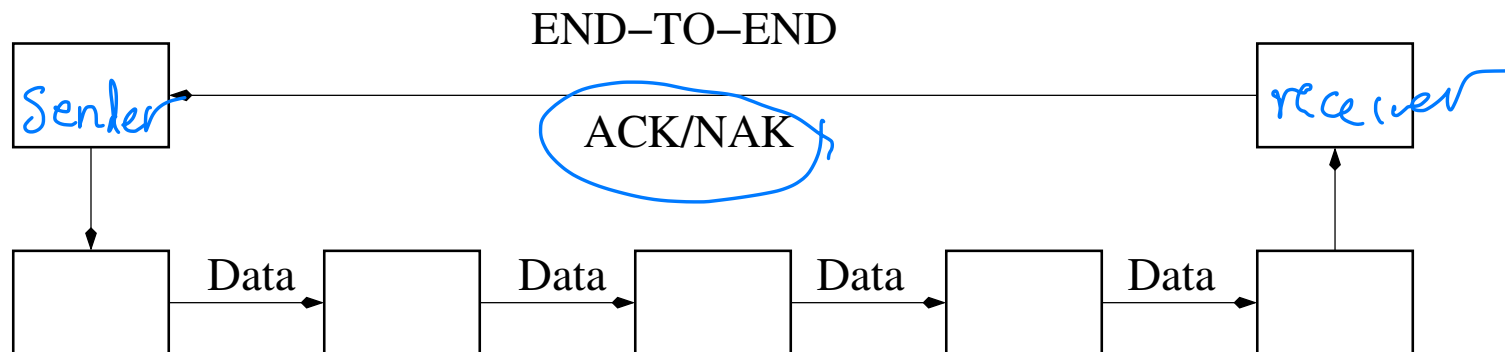
- There are two types of service models.
  - **Connction-oriented:** A connection setup procedure precedes the transfer of information. This initializes state information between the two peer nodes. In the data transfer phase the state information provides a context used by the peer processes in the exchange of data.
  - **Connectionless:** There is no connection setup procedure. Instead, self-contained blocks of information are transmitted and delivered using address information in each packet. In its simplest case, no ACK is provided. So this is more appropriate when transfer is reliable.
- QoS parameters may also be added, e.g., reliability in terms of probability of errors, probability of loss, etc.



## End-to-End Requirements

1. Arbitrary message size.
  2. Reliability and sequencing.
  3. Pacing and flow control.
  4. Timing.
  5. Addressing.
  6. Security features (Privacy, authentication, integrity).
- 

## End-to-End and Hop-by-Hop



## Need for Coordination

- Three issues must be addressed:

1. **Line Discipline:**

Coordinates the link systems. Typically we have half-duplex transmission between two communicating peers.

2. **Flow Control:**

Coordinates the amount of data that can be sent and received.

3. **Error Control:**

Receiver must inform sender of lost and/or damaged packets. This is usually handled by ARQs (Automatic Repeat reQuests).

Why do we need error control again

- We examine all three issues in the sequel.

## Peer-to-Peer Strategies

- Peer-to-Peer considerations:

Information Packets

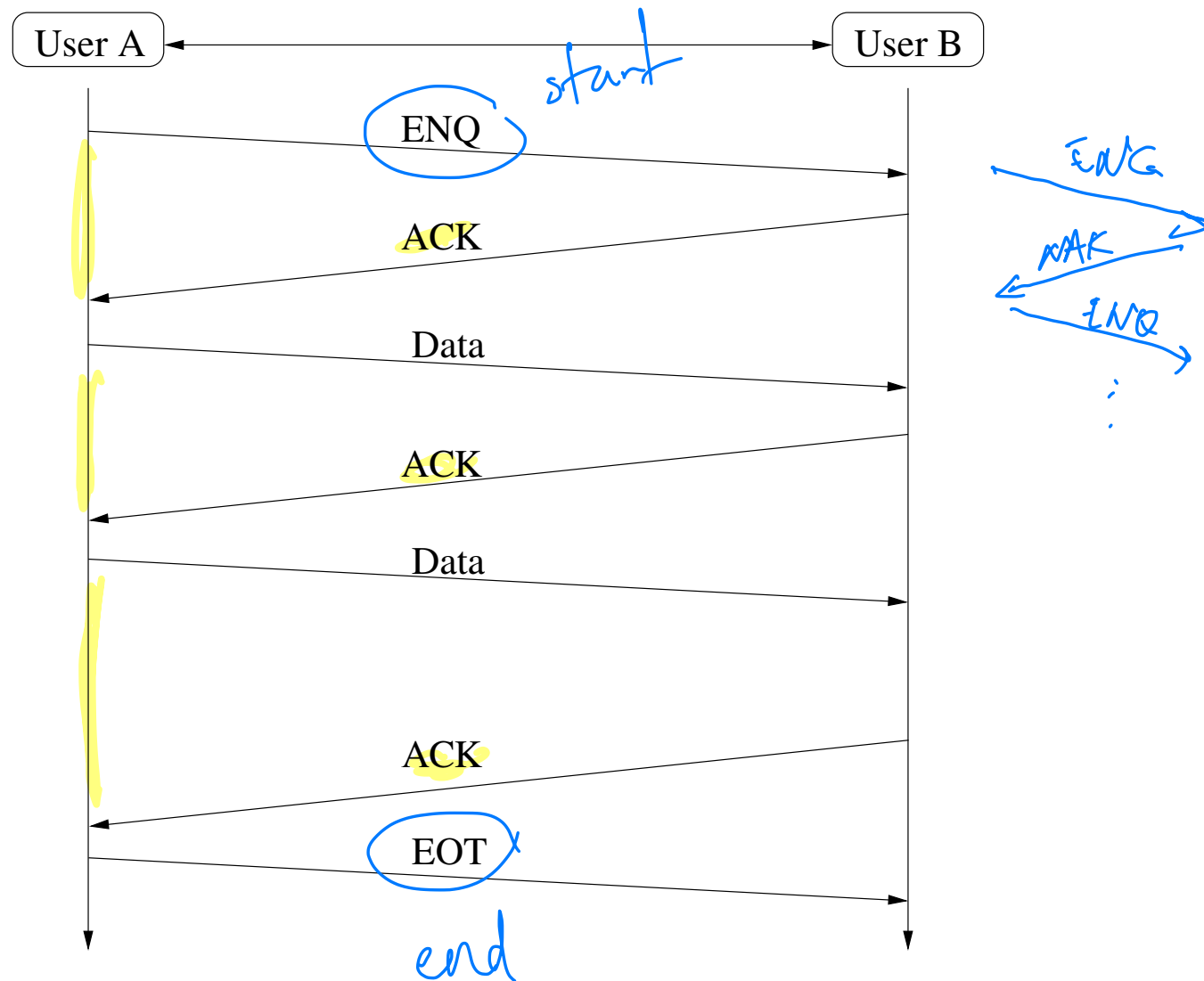


Control Packets

*smaller than  
information packet*

1. Who should start?
2. Is the other peer ready?
3. How much should I send?
4. When do I report problems?

## ENQ/ACK Line Discipline

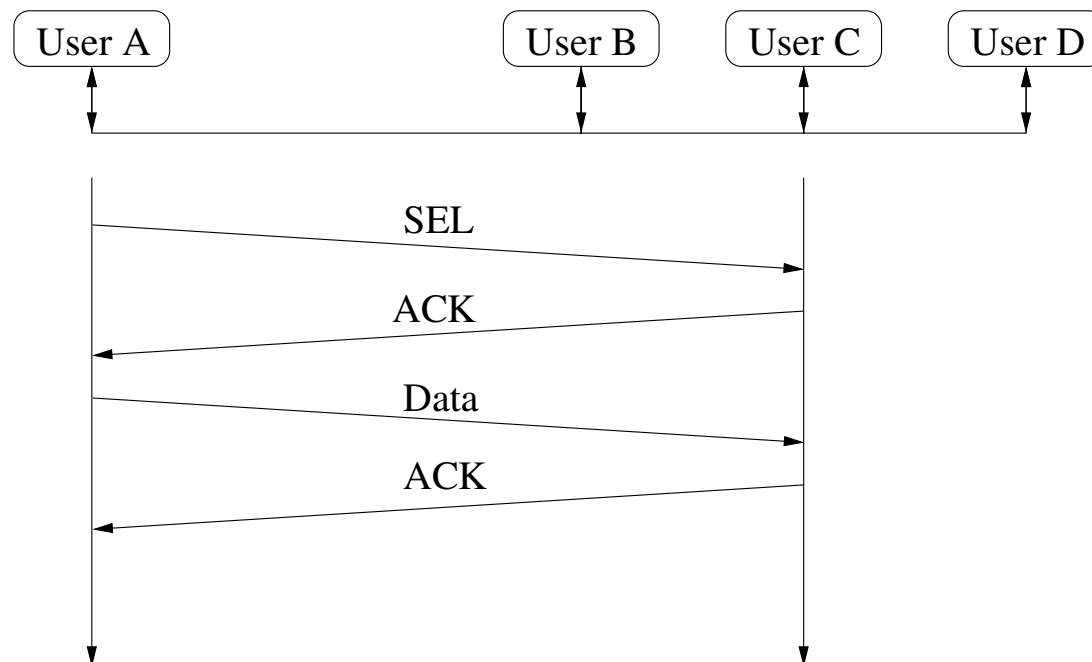




## ENQ/ACK Line Discipline

1. Sender sends packet ENQ to determine if receiver is available.
2. Receiver must answer either with ACK (acknowledgement) or NAK (Negative ACK).
  - (a) If sender receives neither ACK nor NAK it must assume frame ENQ was lost and sends replacement (usually makes three attempts).
  - (b) If sender receives a NAK to ENQ each time after three times it gives up and tries later.
3. After receiving an ACK sender sends Data which are “periodically” acknowledged by the receiver.
4. Once all Data has been sent the sender sends EOT (End of Transmission).

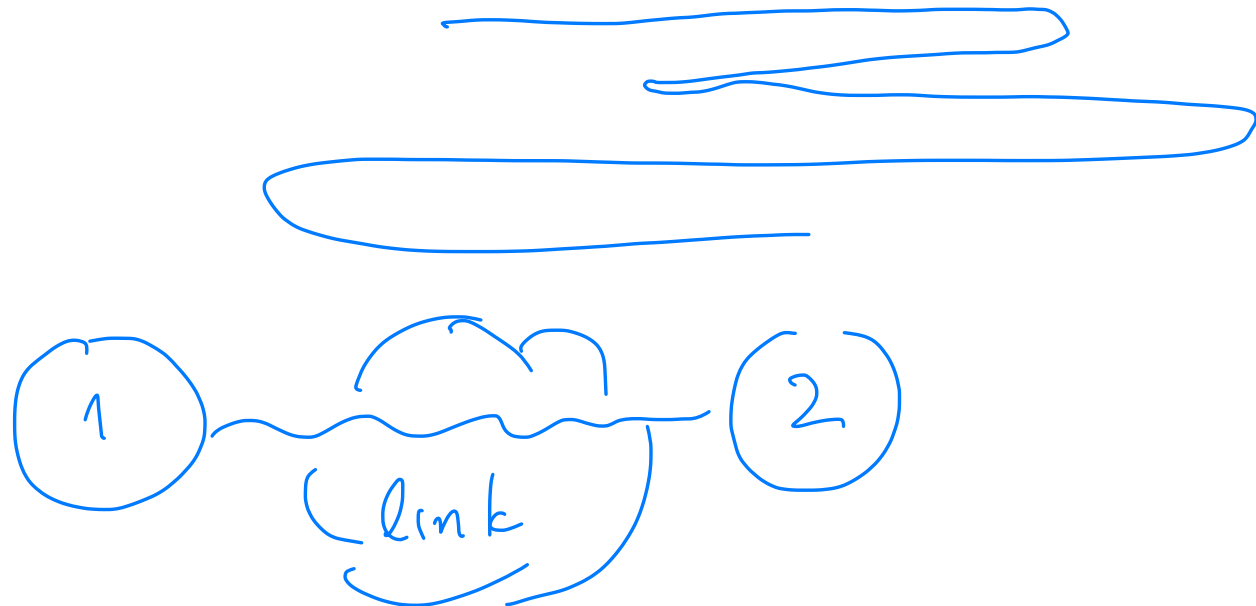
## Poll/Select Line Discipline



1. In SEL User A selects the user it wants to talk to.
2. One of Users B, C, D must respond with ACK.
3. Poll is used by User A to solicit transmissions.

## LINK (FLOW) CONTROL MECHANISMS

- How is the flow controlled?
- Several flow control mechanisms are in use.
- We discuss two:
  - Stop-and-Wait
  - Sliding Window



## STOP-AND-WAIT

- Basic Algorithm:

- **Source:**

- Transmits frame

- **Destination:**

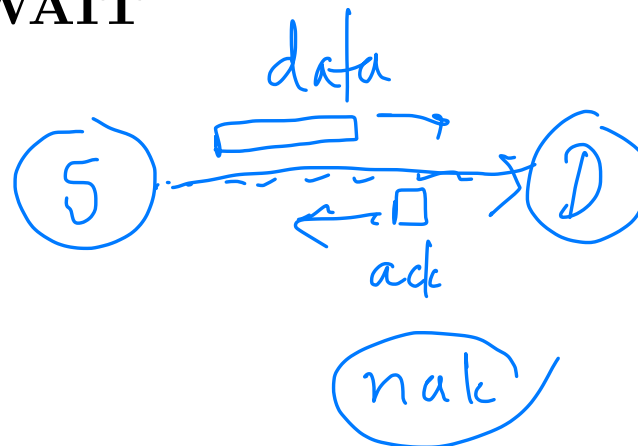
- \* receives frame

- \* indicates willingness to accept another frame by sending back ACK to the frame just received.

- **Source:**

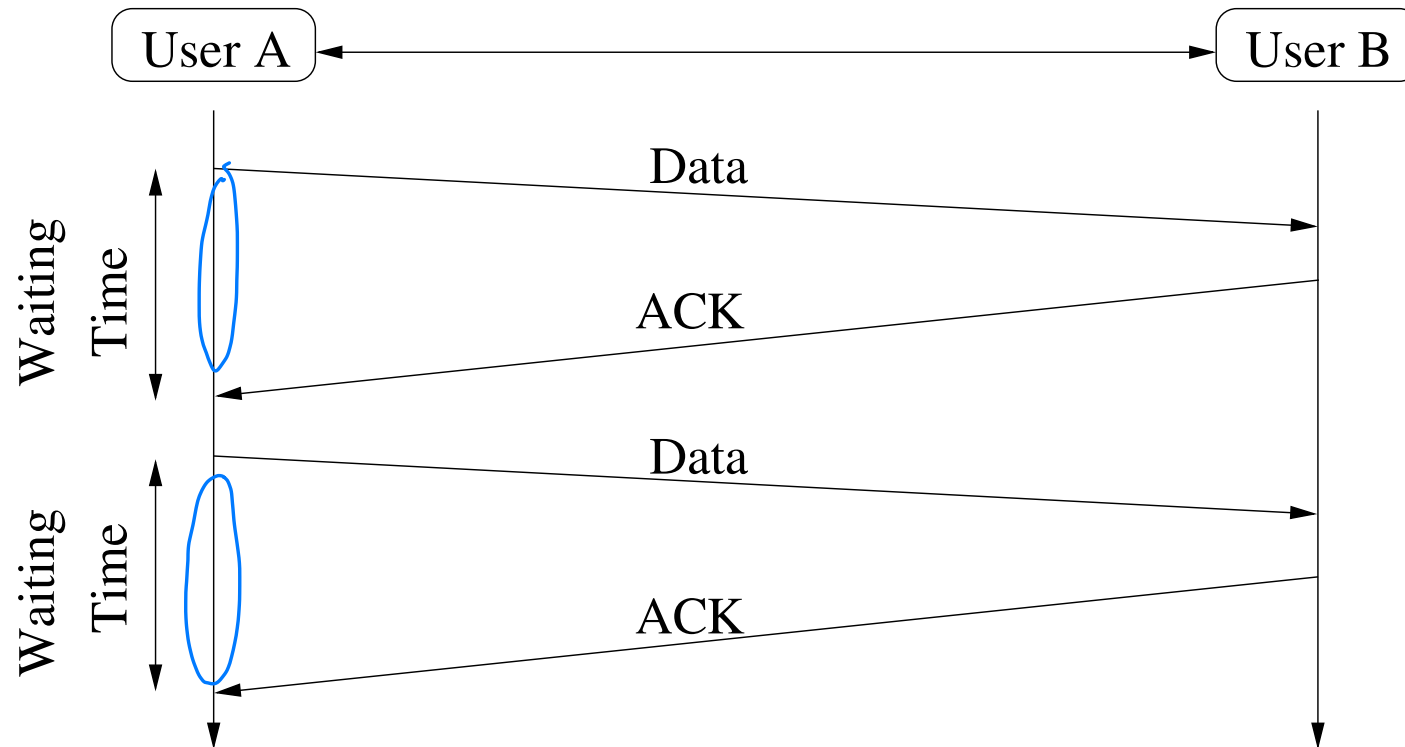
- Must wait until it receives ACK before it sends new frame.

- Thus by withholding ACK the destination can stop the flow of data.



every  
packet  
sent  
is  
acknowledged

## STOP-AND-WAIT



Waiting times: typically they are random

$$LB \leq W \leq UB$$

$$5 \cdot UB$$

## INEFFICIENCIES

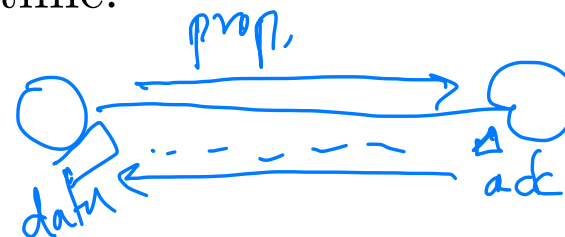
- Stop-and-Wait rarely used in practice for data transfers
- Only one frame at time is in transit
- If propagation time is long relative to the transmission time then the line will be idle for most of the time.

- For example,
  - the frame rate is

$$\frac{1}{T_{frame}}$$

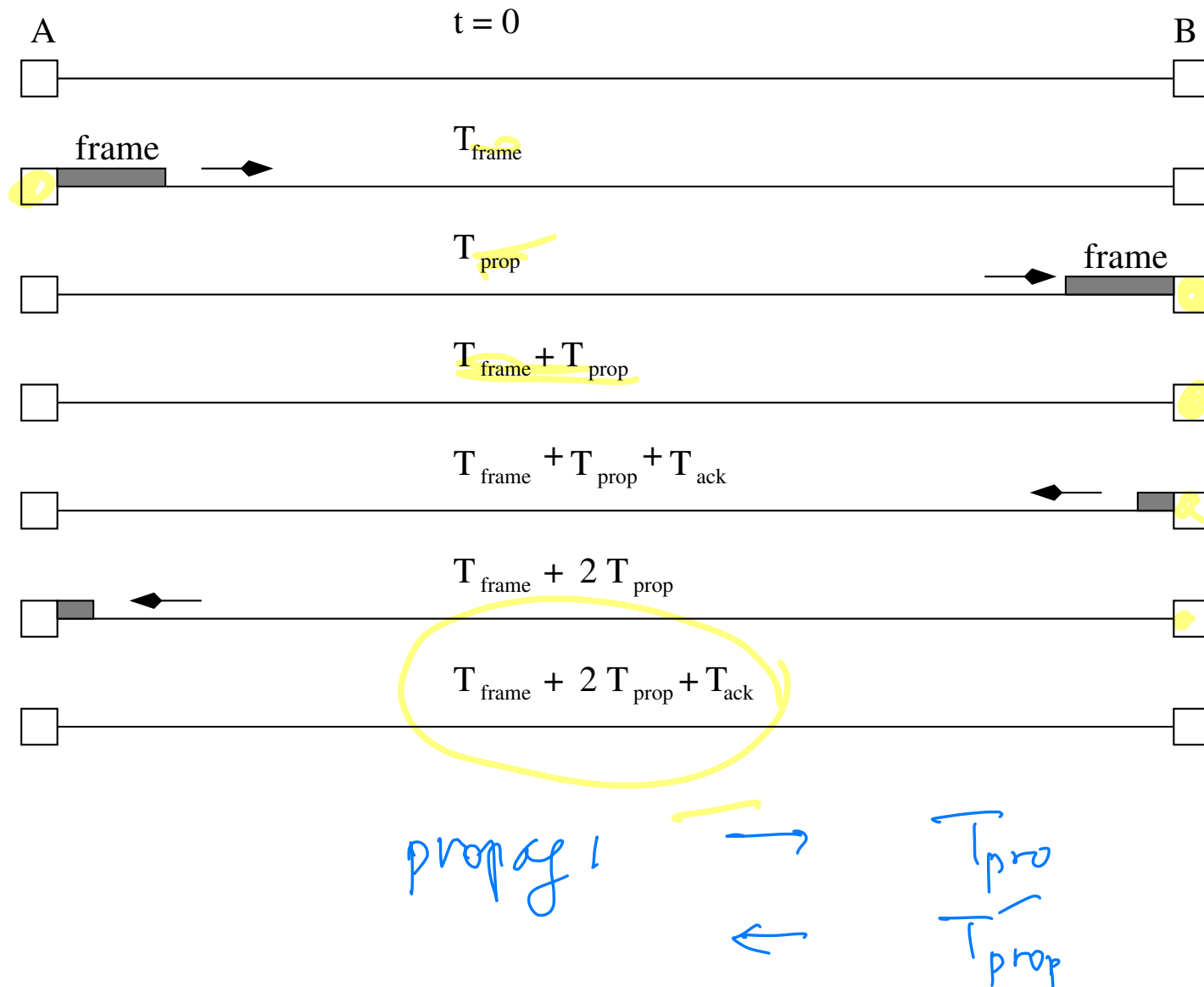
- but in fact the actual rate achieved is reduced to

$$\frac{1}{T_{frame} + 2 \cdot T_{prop} + T_{ack}}$$

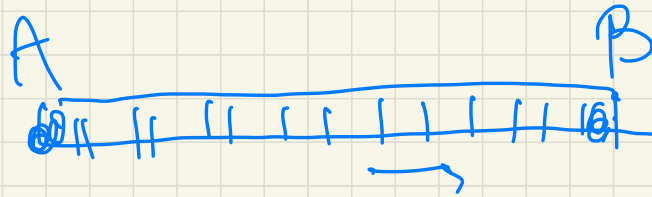


RTT = Round Trip Time / 2 (?)

## Utilization of Links in Stop-and-Wait



$T_{prop}$ : is measured from  
the time the last bit of  
the frame "left" A to the  
time the first bit of the  
frame arrived at B





Efficiency of stop and wait

$$T_{\text{frame}} + 2T_{\text{prop}} + T_{\text{ack}}$$

$$= 1 + \frac{T_{\text{frame}}}{2T_{\text{prop}} + T_{\text{ack}}}$$

overhead

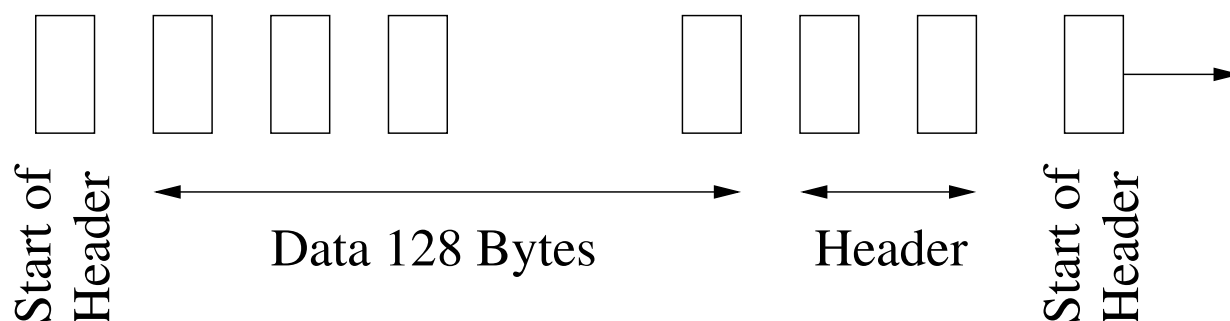
## Asynchronous Protocols (Usually Slow!)

Based on Stop-and-Wait.

- XMODEM
- YMODEM
- ZMODEM
- BLAST
- KERMIT

## XMODEM: Example<sup>a</sup>

- The XMODEM frame is depicted below.
- Start of Header is 1 Byte, Header is 2 Bytes, and Data is 128 Bytes.



- The header was followed by the 128 bytes of data, and then a single-byte checksum.
- The complete packet was thus 132 bytes long, containing 128 bytes of payload data, for a total channel efficiency of about 97%

---

<sup>a</sup>Developped in 1977

### **Example: Xmodem ARQ<sup>a</sup>**

- Xmodem used to be a popular modem transfer protocol.
- Information transmitted in fixed-length blocks consisting of a 3-byte header, 128 bytes of data and a 1-byte checksum.
- Header has a Start-of-header character, a 1-byte sequence number and a 2s complement of the sequence number.
- **Example: Bitsync ARQ**
  - Bitsync is IBM's Binary Synchronous Communications protocol.
  - It is character-oriented and uses the ASCII character set.

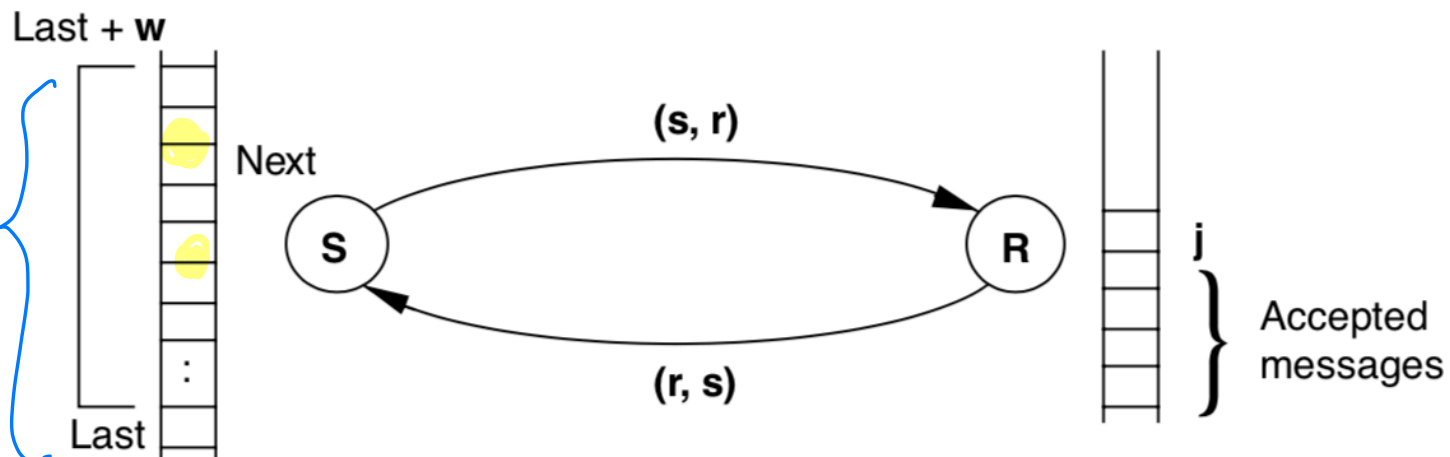
---

<sup>a</sup>ARQ stands for Automatic Repeat Request

## Idea of Sliding Window

TCP

- Sliding window protocol is a widely used transport layer protocol that implements a reliable channel between a pair of processes:  $S$  (sender) and  $R$  (receiver).



$S, R$  will monitor the quality of the line using "Little's law" to determine how many should be sent!

If Sender sends 500 packet

- 1) Sender will number packet
- 2) Expect to receive ACKs  
eg. # 1, 2, 3, ..., 500

Receiver

Packet 237 nak

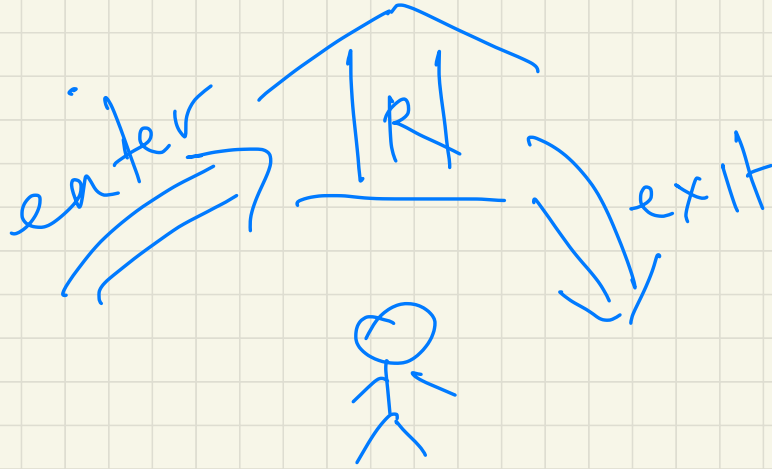
Sender: interprets 1...236 have  
been received ok.

## Sliding Window: What Does It Do?

- It handles both omission failures and message reordering caused by an unreliable channel.
- It detects the loss or reordering of messages (and acknowledgments) using timeouts (with limited accuracy since message propagation delays are arbitrarily large but finite) and resolves it by retransmissions.
- It has a mechanism to improve the transmission rate, and restore the message order at the receiving end without overflowing its buffer space. } Little's Law

How are all these things accomplished? This is what TCP is all about!

Little's Law provides control measure.



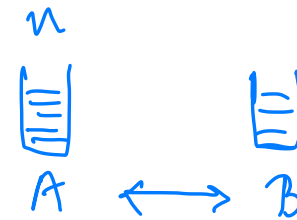


## Sliding Window: How Does it Work?

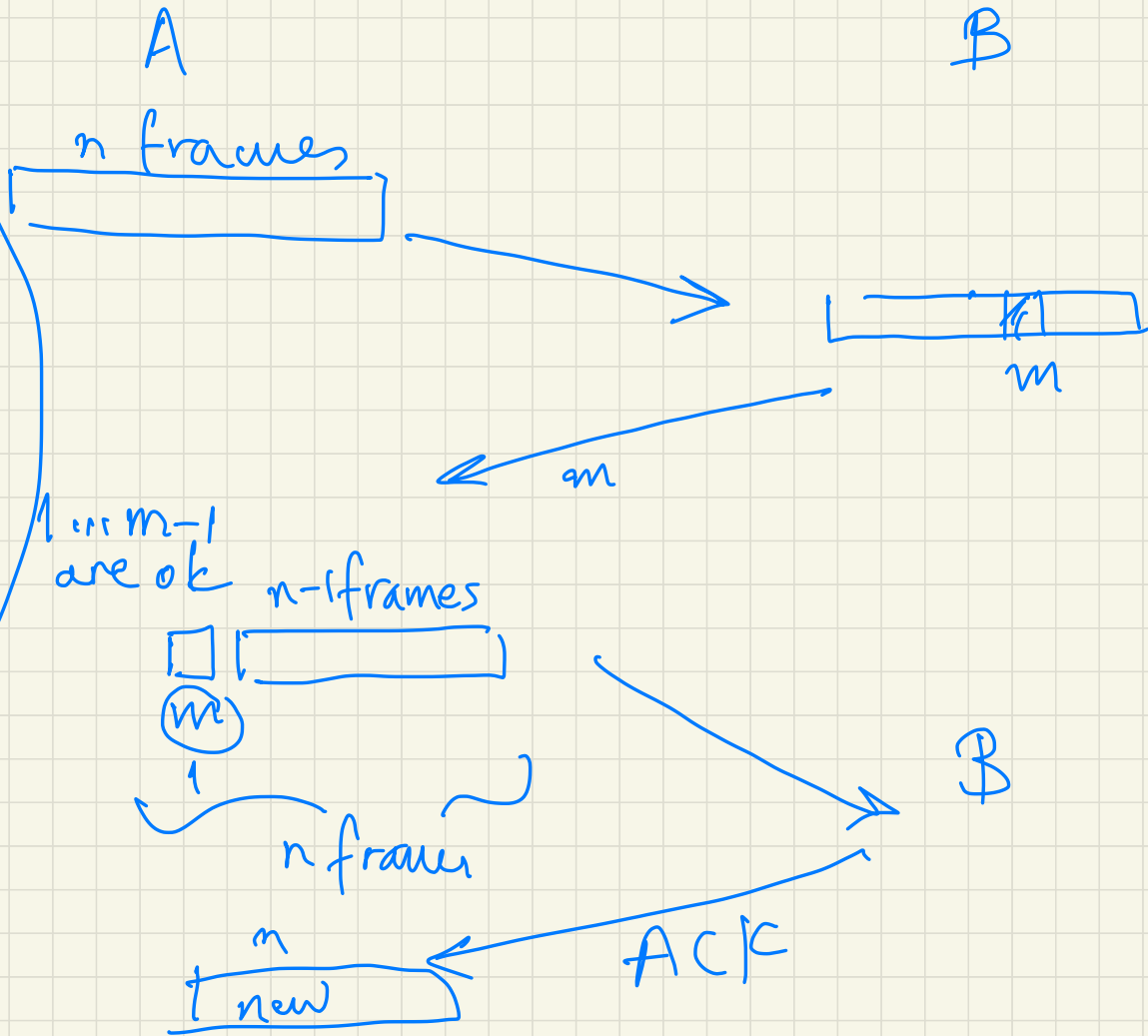
- The sender continues the send action without receiving the acknowledgments of at most  $w$  outstanding messages ( $w > 0$ ), where  $w$  is called the window size. If no acknowledgment to the previous  $w$  messages is received within an expected period of time, then the sender resends those  $w$  messages.
- The receiver anticipates the sequence number  $j$  of the next incoming message. If the anticipated message is received, then  $R$  accepts it, sends the corresponding acknowledgment back to  $S$ , and increments  $j$ . Otherwise,  $R$  sends out an acknowledgment corresponding to the sequence number  $j - 1$  of the previous message accepted by it.

## SLIDING WINDOW (General Concept)

- Main problem in Stop-and-Wait is that only one frame at a time can be in transit.
- Assume we have two stations  $A$ ,  $B$ .
  - $B$  has buffer space for  $n$  frames
  - $A$  is allowed to send  $n$  frames without waiting for ACK
  - Each frame is labeled with a sequence number
  - $B$  acknowledges a frame by sending an ACK that includes a sequence number announcing it is ready to receive next  $n$  frames beginning with number specified
  - sequence numbers occupy a field in the frame; for a  $k$ -bit field the range is  $0..2^k - 1$  and frames are numbered modulo  $2^k$ .



Every frame  
is numbered

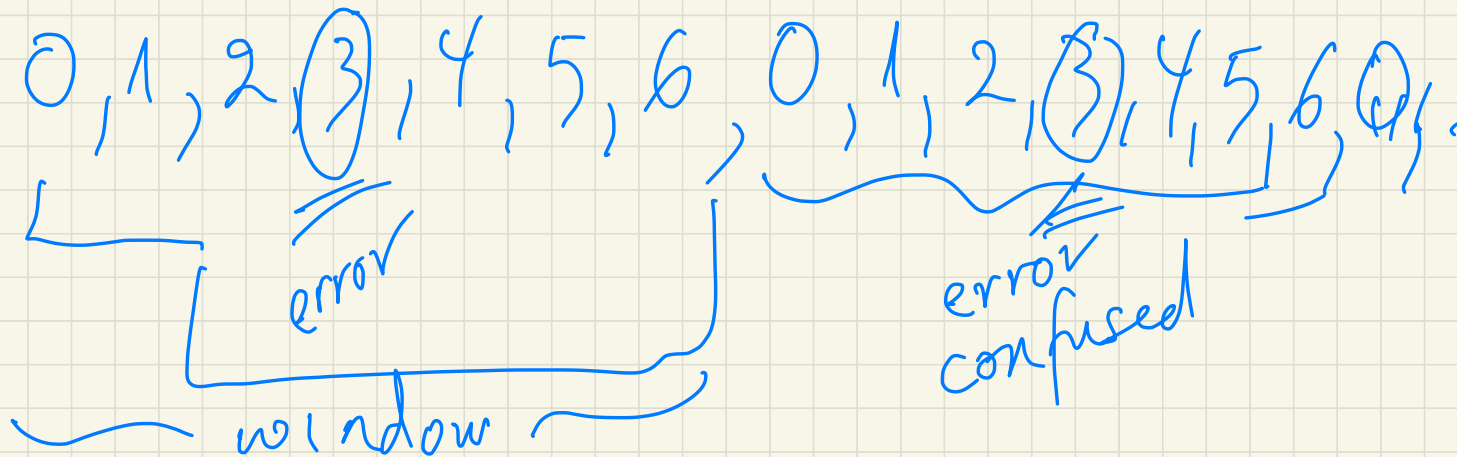


mod  $N$  arithmetic

$0, 1, \dots, N-1, 0, 1, \dots, N-1, \dots$

$$N = 7$$

$0, 1, 2, \textcircled{3}, 4, 5, 6, 0, 1, 2, \textcircled{3}, 4, 5, 6, 0, \dots$



error

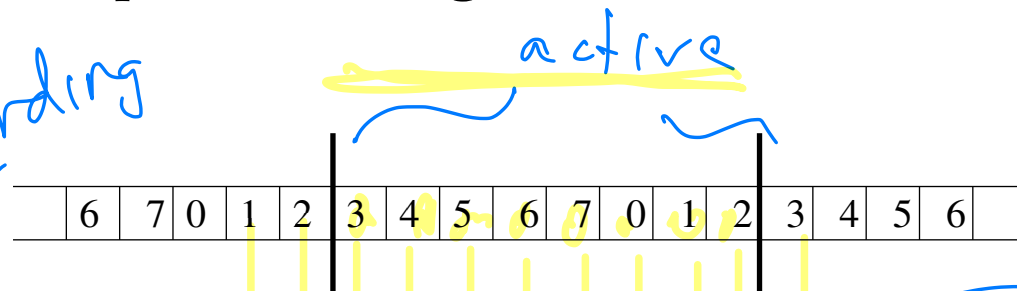
error confused

window

## Example of Sliding Window of Size 8

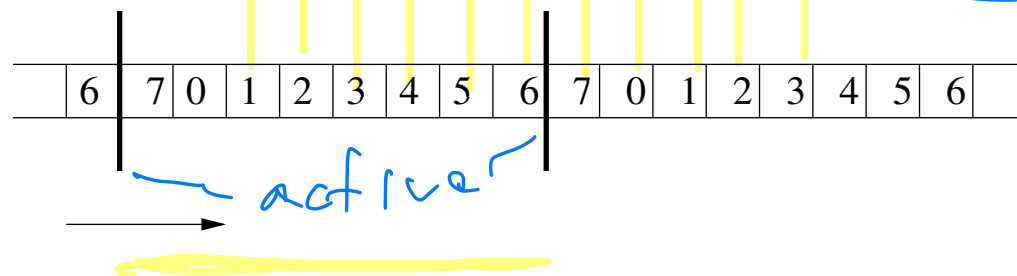
- **Sender:**

State according  
to the Sender



- **Receiver:**

State of  
Receiver



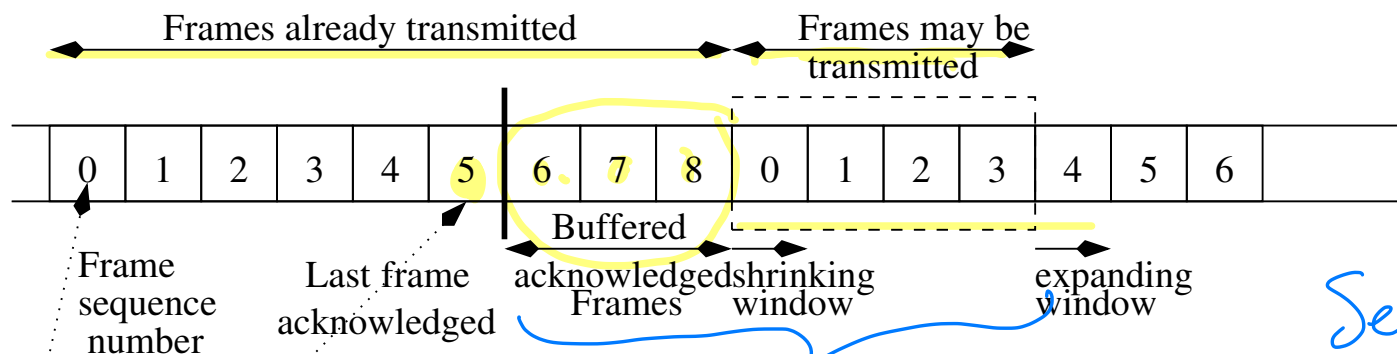
$$N = 8$$

Must packet "tags" using  
modulo arithmetic, modulo  
some number  $N$ ,

## Example of Sliding Window

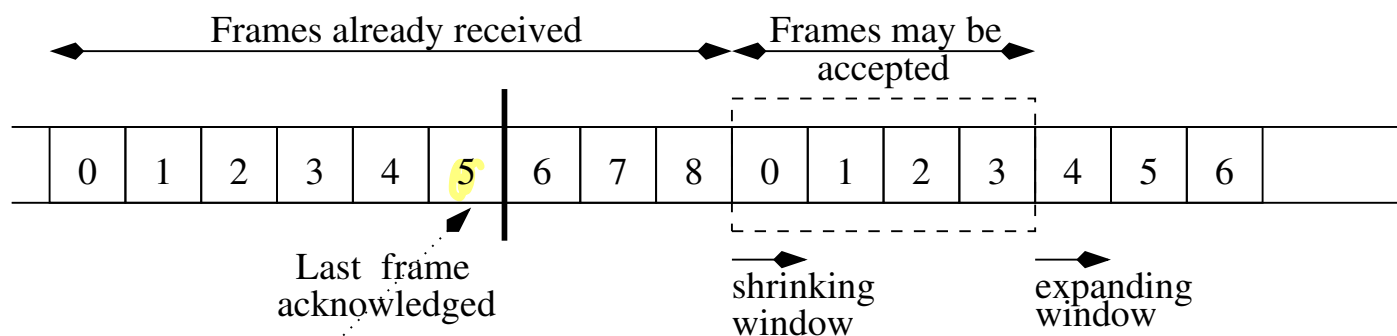
- $B$  could receive frames labeled 2, 3, 4.
- It does not have to acknowledge frames received so far until frame labeled 4 has arrived.
- It then returns an ACK[5].
- $A$  maintains a list of numbers that it is allowed to send.
- $B$  has a separate list of numbers that it is prepared to receive.

## Sliding Window Perspectives



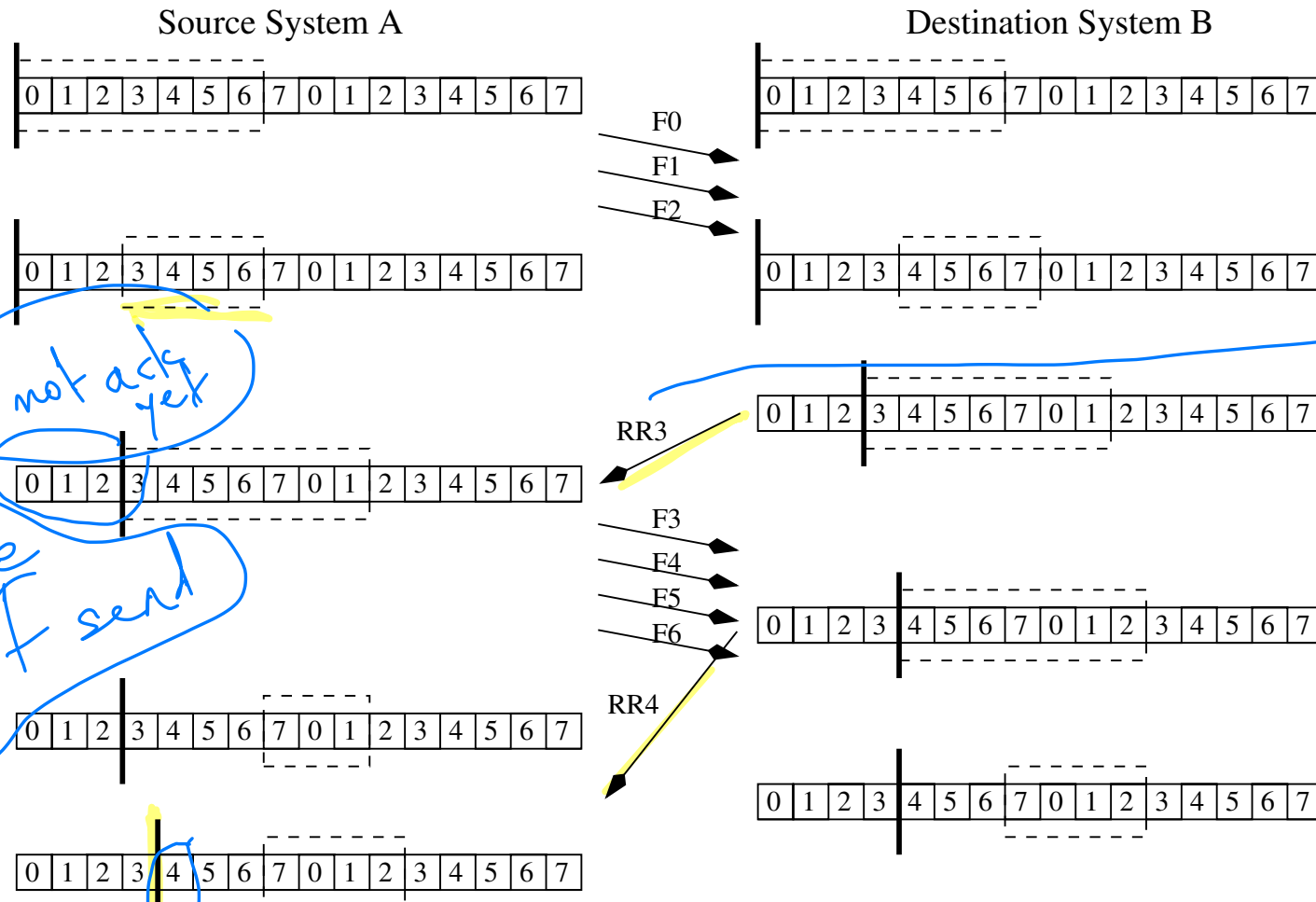
SENDER'S PERSPECTIVE

no more than 8 packets in "limbo"



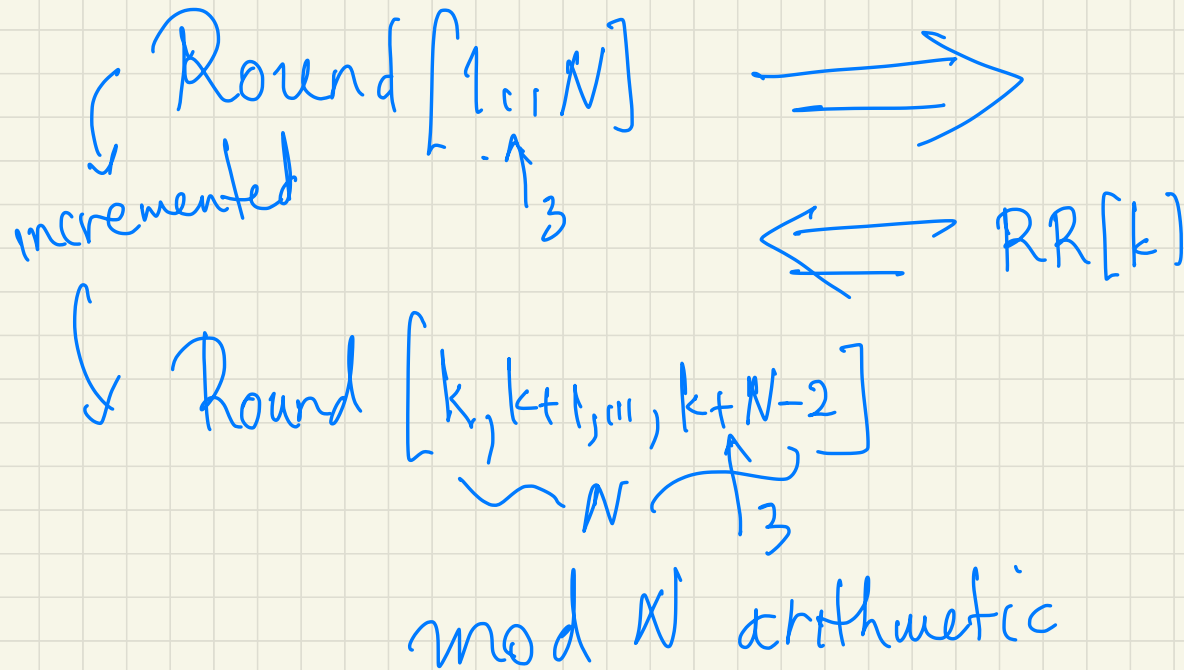
RECEIVER'S PERSPECTIVE

## Example of Sliding Window Protocol



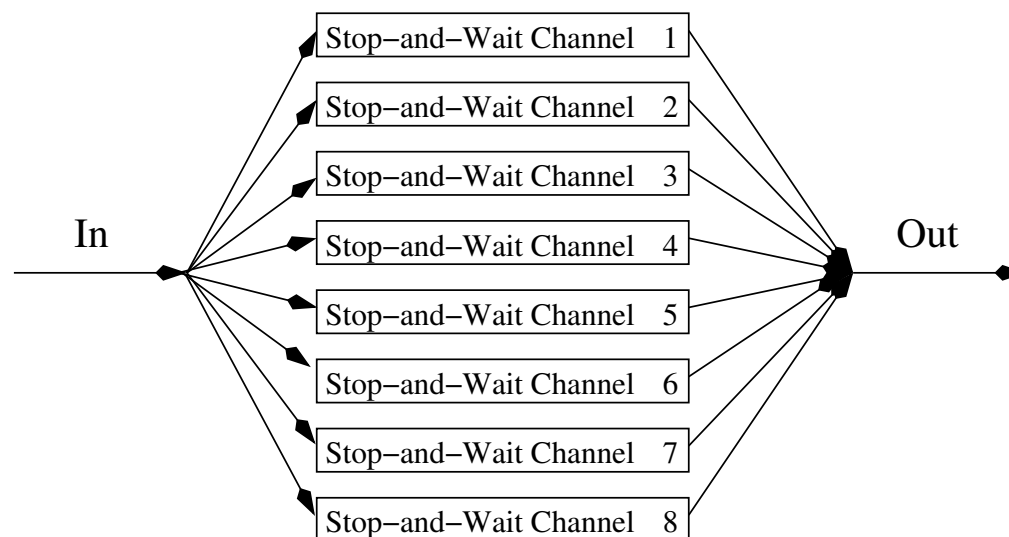


Sliding window protocol is in rounds



## Exercises<sup>a</sup>

1. ARPANET ARQ uses Eight Stop-and-Wait pipes in parallel.



- (a) Incoming packets assigned to one of the channels according to some strategy (e.g., round robin).
- (b) If all channels are busy packet awaits outside the DLC.
- (c) If a channel's turn for transmission comes up before an ACK is received, the packet is sent again.

---

<sup>a</sup>Do not submit

Define the strategies precisely and indicate how this multiple stop-and-wait protocol could be made to work. How does it compare to standard stop-and-wait? How does it compare to sliding window?

2.