CS 313 Introduction to Computer Networking & Telecommunication

Data Link Layer Part II – Sliding Window Protocols

Part 2 - Topics

- Sliding Window Protocols
- Go Back N Sliding Window Protocol
- Selective Repeat Sliding Window Protocol

Data Frame Transmission

- Unidirectional assumption in previous elementary protocols

  ⇒ Not general

- Full-duplex - approach 1
  - ❑Two separate communication channels
    - ➢Forward channel for data
    - ➢Reverse channel for acknowledgement

      ⇒ Problems: 1. reverse channel bandwidth wasted

      2. cost

Data Frame Transmission

- Full-duplex - approach 2
  - ❑Same circuit for both directions
  - ❑Data and acknowledgement are intermixed
  - ❑How do we tell acknowledgement from data?

    "*kind*" field telling data or acknowledgement
  - ❑Can it be improved?
- Approach 3
  - ❑Attaching acknowledgement to outgoing data frames

    $\Rightarrow$ Piggybacking

Piggybacking

- Temporarily delaying transmission of outgoing acknowledgement so that they can be hooked onto the next outgoing data frame
- Advantage: higher channel bandwidth utilization
- Complication:
  - How long to wait for a packet to piggyback?
  - If longer than sender timeout period then sender retransmit

$\Rightarrow$ Purpose of acknowledgement is lost

Piggybacking

- Solution for timing complexion
  - ❑ If a new packet arrives quickly

    ⇒ Piggybacking
  - ❑ If no new packet arrives after a receiver ack timeout

    ⇒ Sending a separate acknowledgement frame

Sliding Window Protocol

- We are going to study three bidirectional *sliding window protocols* (max sending window size, receiving window size)
  - One-bit sliding window protocol (1, 1)
  - Go back N (>1, 1)
  - Selective repeat (>1, >1)
- Differ in efficiency, complexity, and buffer requirements

Sliding Window Protocol

- Each outbound frame contains an *n*-bit sequence number
  - Range: 0 - MAX_SEQ (MAX_SEQ = 2^n - 1)
  - For stop-and-wait, *n* = __. Why?
- At any instance of time
  - Sender maintains a set of sequence numbers of frames *permitted to send*
    - These frames fall within *sending window*
  - Receiver maintains a set of sequence numbers of frames *permitted to accept*
    - These frames fall within *receiving window*

Sliding Window Protocol

- Lower limit, upper limit, and size of two windows *need not be the same*
- Fixed or variable size
- Requirements
  - Packets delivered to the receiver's network layer must be in the same order that they were passed to the data link layer on the sending machine
  - Frames must be delivered by the physical communication channel in the order in which they were sent

Sending Window

- Contains frames can be sent or have been sent but not yet acknowledged – *outstanding* frames
- When a packet arrives from network layer
  - Next highest sequence number assigned
  - Upper edge of window advanced by 1
- When an acknowledgement arrives
  - Lower edge of window advanced by 1

Sending Window

- If the maximum window size is *n*, *n* buffers is needed to hold unacknowledged frames
- Window full (maximum window size reached)

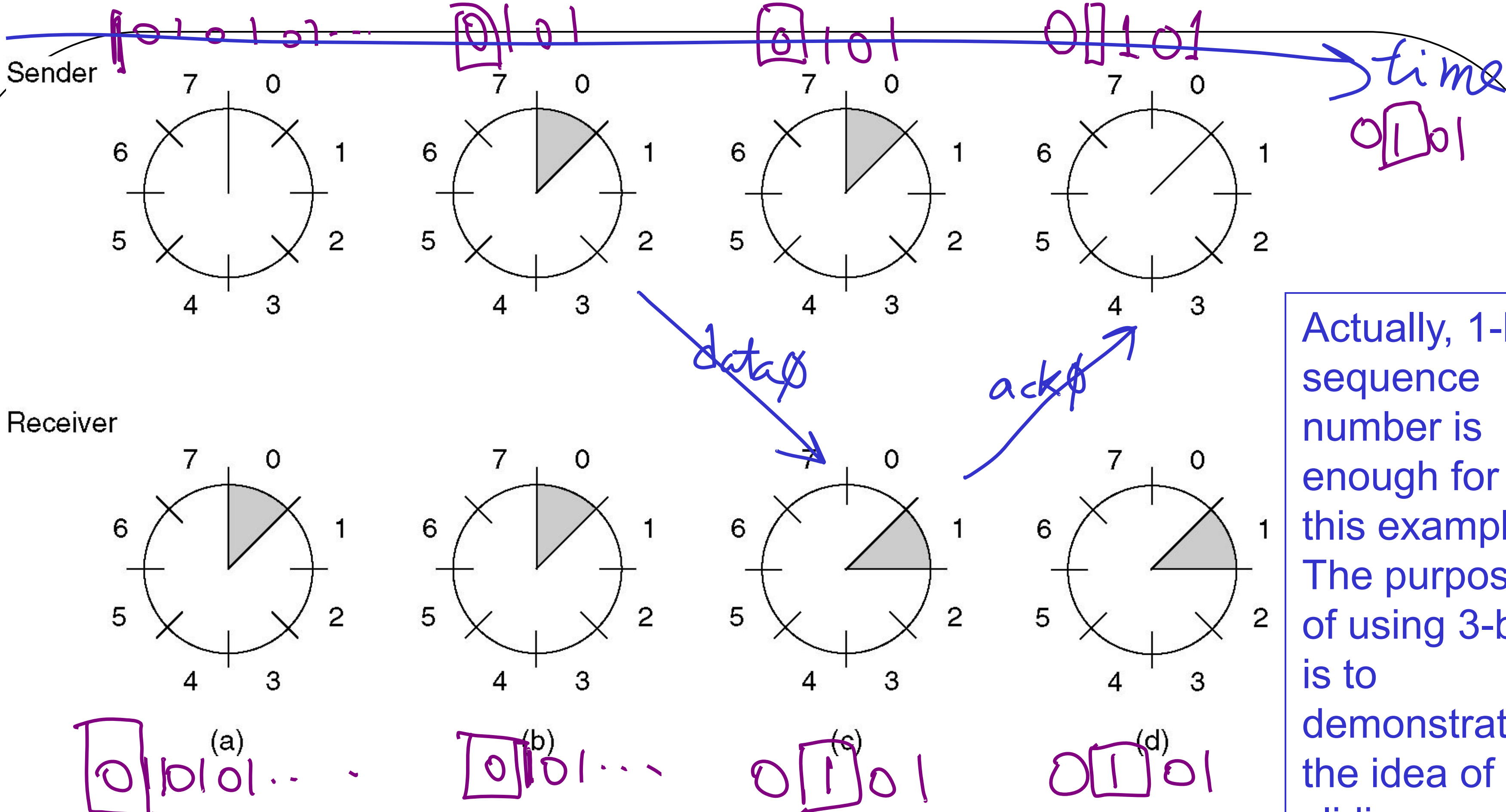
  $\Rightarrow$ shut off network layer

Receiving Window

- Contains frames may be accepted
- Frame outside the window → discarded
- When a frame's sequence number equals to lower edge
  - ❑Passed to the network layer
  - ❑Acknowledgement generated
  - ❑Window rotated by 1

Receiving Window

- Contains frames may be accepted
- Always remains at initial size (different from sending window)
- Size
  - =1 means frames only accepted in order
  - >1 not so
- Again, the order of packets fed to the receiver's network layer must be the same as the order packets sent by the sender's network layer



A sliding window of size 1, with a 3-bit sequence number.

(a) Initially.

(b) After the first frame has been sent.

(c) After the first frame has been received.

(d) After the first acknowledgement has been received.

Actually, 1-bit sequence number is enough for this example. The purpose of using 3-bit is to demonstrate the idea of sliding window.

In many textbooks, an array of boxes are used to represent the window.

One Bit Sliding Window Protocol

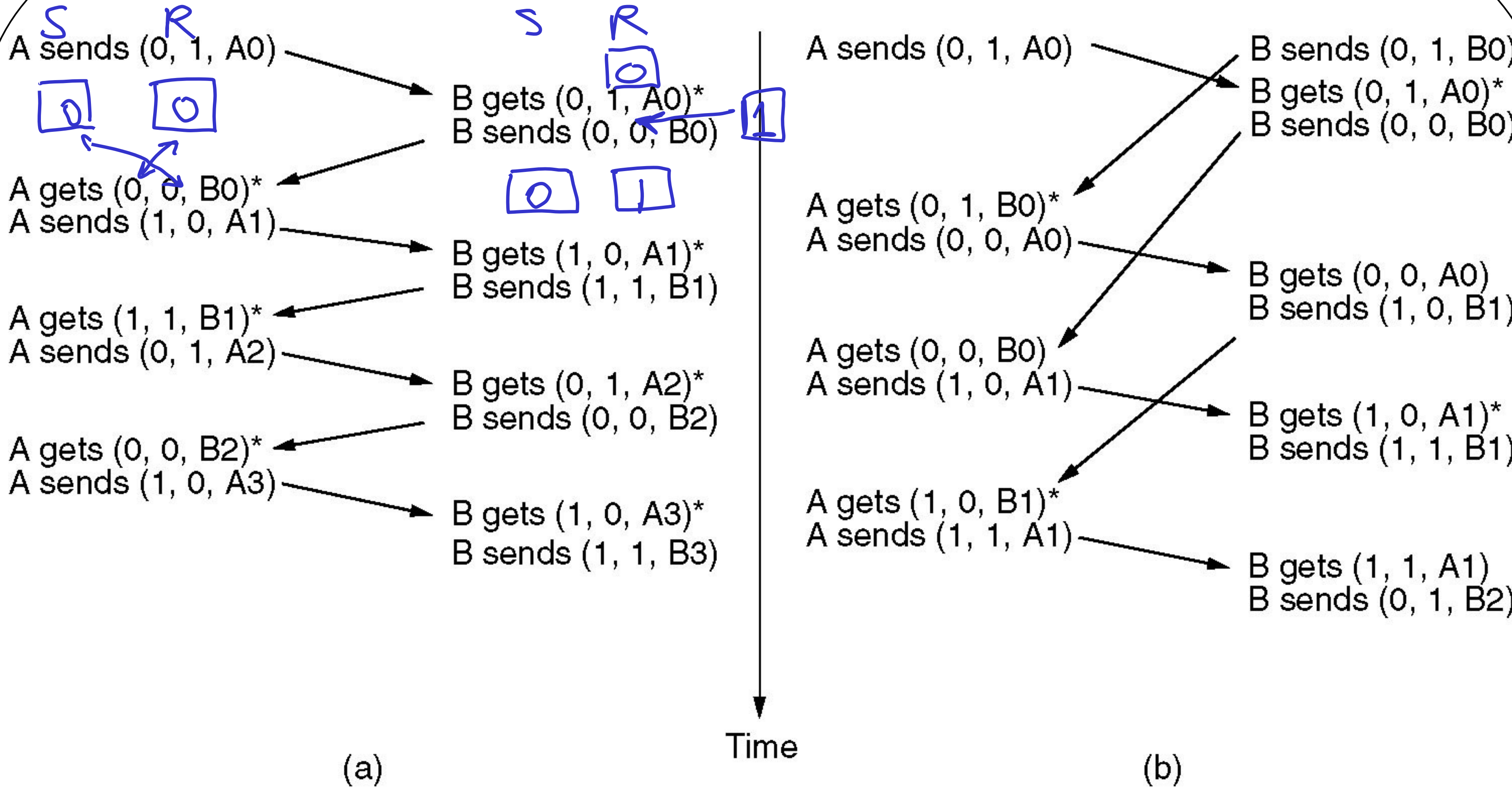
- Sending window size = receiving window size = 1
- Stop-and-wait
- Refer to algorithm in Fig 3-16
- Acknowledgement =

  Sequence number of last frame received w/o error*
- Problem of sender and receiver send simultaneously

*: some protocols define the acknowledgement to be the sequence number expected to receive

Case 1: normal case

Case 7: simultaneous start

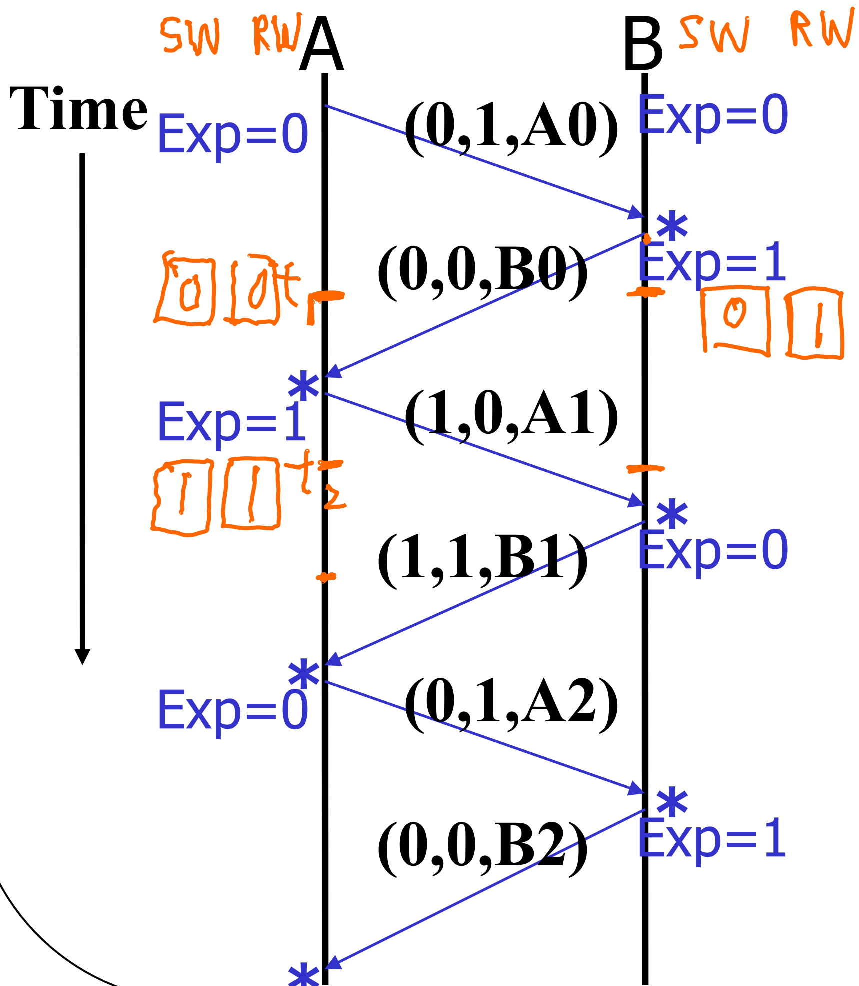


(a) Case 1: Normal case. (b) Case 7: Abnormal case.
The notation is **(seq, ack, packet number)**. An asterisk indicates where a network layer accepts a packet.

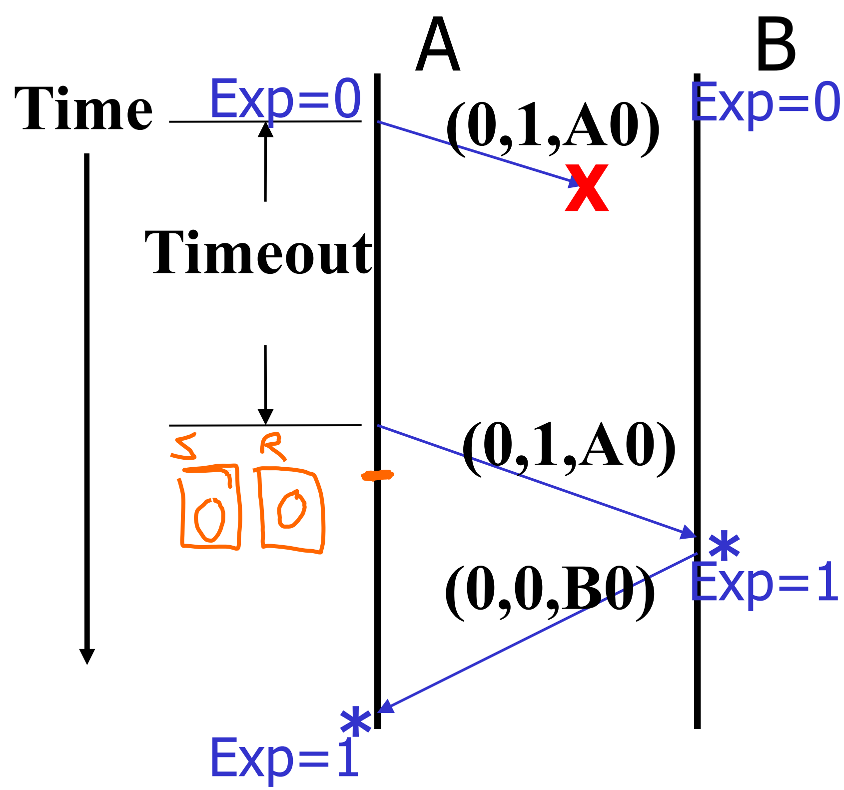
Try to draw the sending windows and receiving windows for A and B!

One Bit Sliding Window Protocol

- Case 1: no error



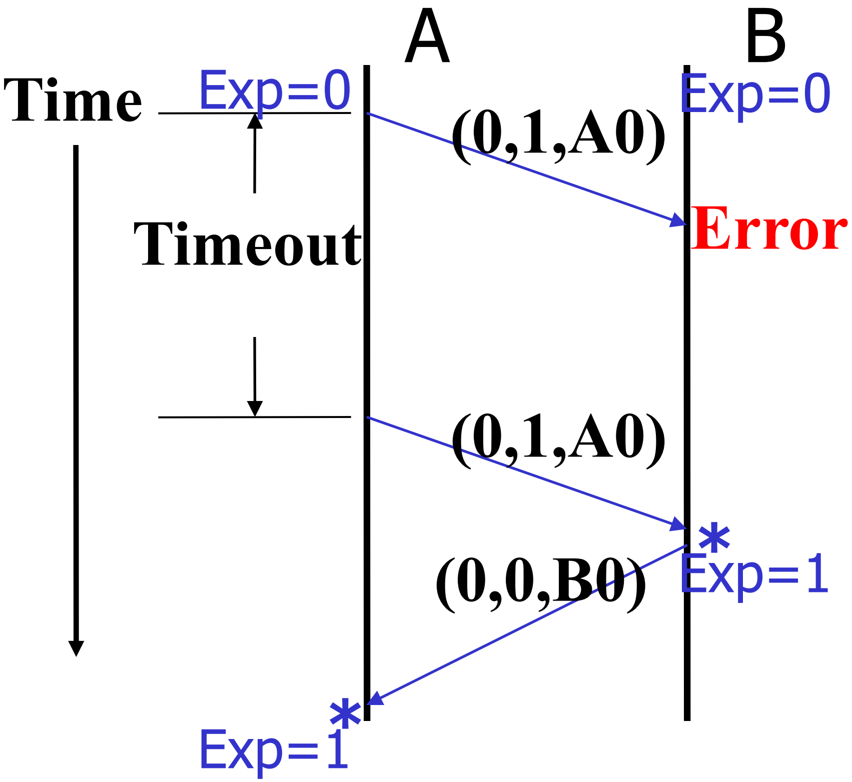
- Case 2: data lost



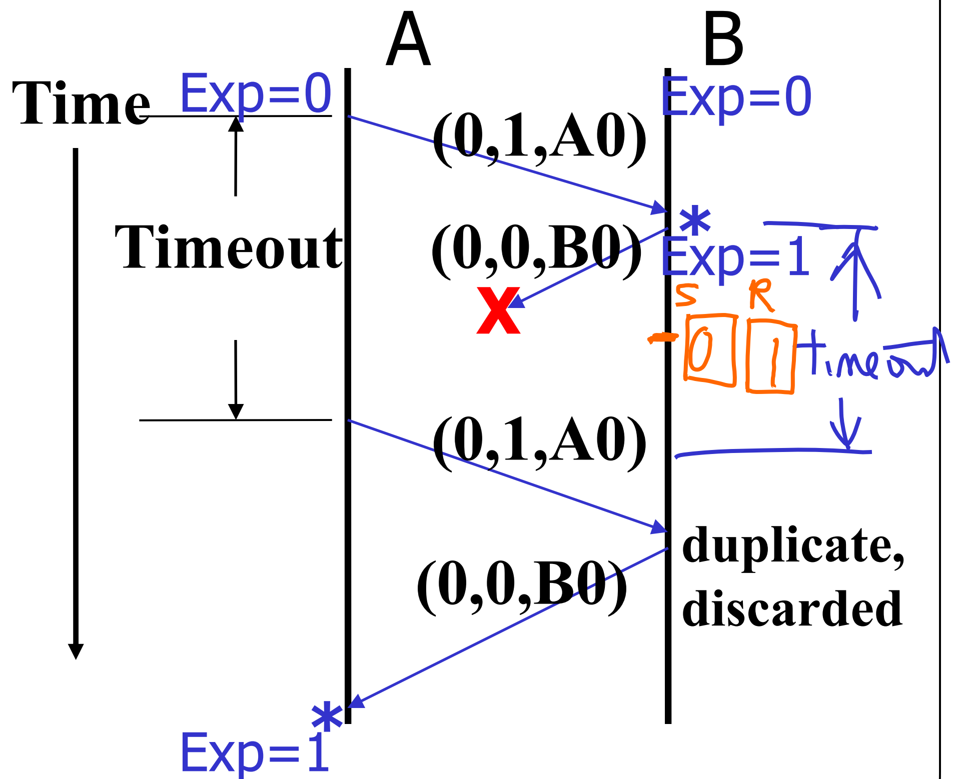
Try to draw the sending windows and receiving windows for A and B!

One Bit Sliding Window Protocol

- Case 3: data error



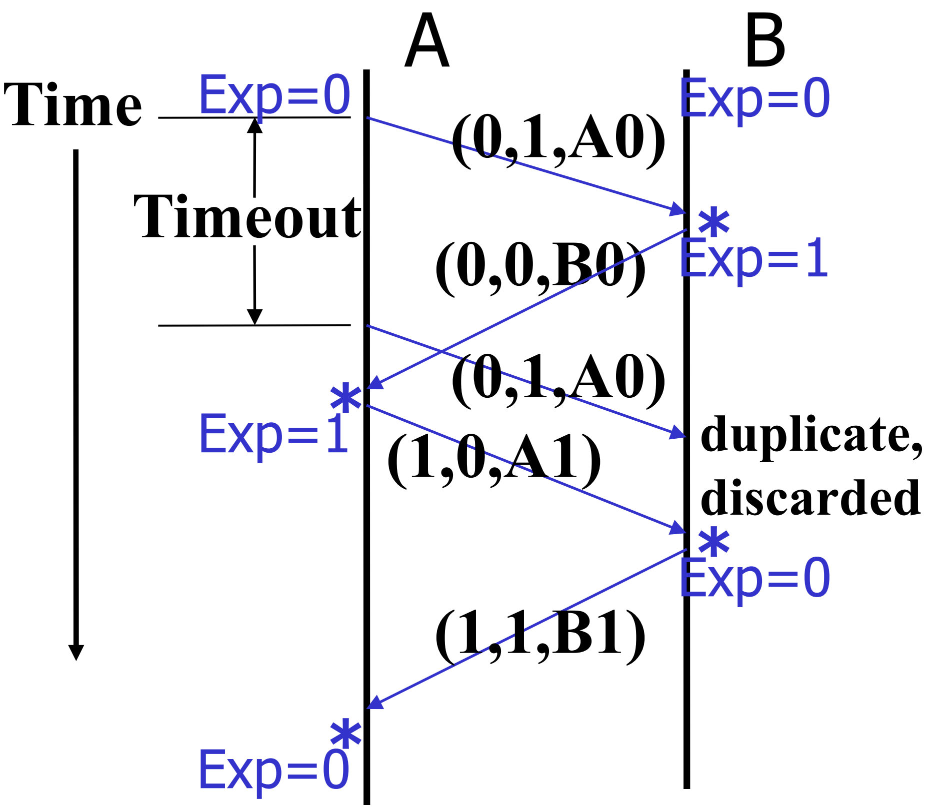
- Case 4: ack. lost



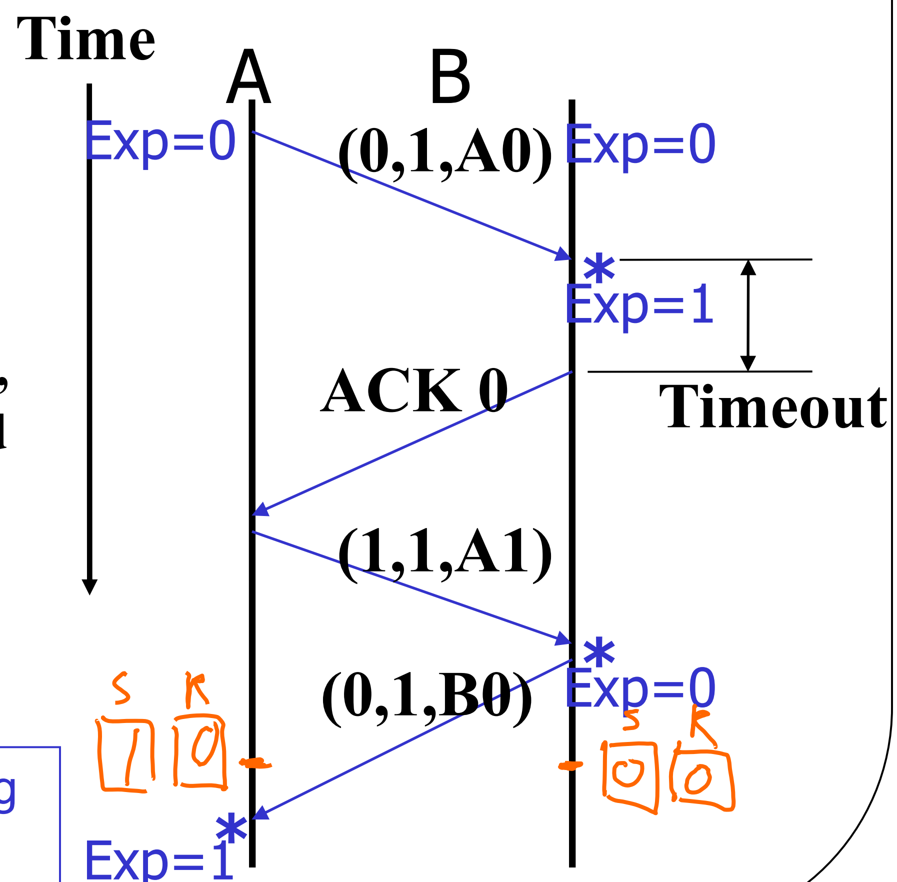
Try to draw the sending windows and receiving windows for A and B!

One Bit Sliding Window Protocol

- Case 5: early timeout



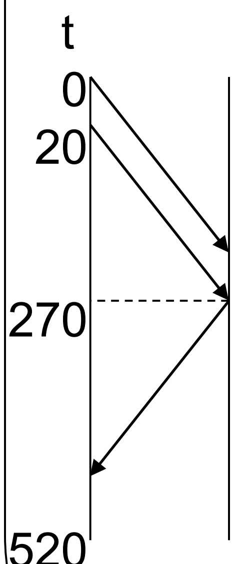
- Case 6: outgoing frame timeout



Try to draw the sending windows and receiving windows for A and B!

Performance of Stop-and-Wait Protocol

- Assumption of previous protocols:
  - Transmission time is negligible
  - False, when transmission time is long
- Example - satellite communication



  - channel capacity: 50 kbps, frame size: 1kb
    round-trip propagation delay: 500 msec
  - Time:

t=0	start to send 1st bit in frame
t=20 msec	frame sent completely
t=270 msec	frame arrives
t=520 msec	best case of ack. received

  - Sender blocked 500/520 = 96% of time
    - Bandwidth utilization 20/520 = 4%

Performance of Stop-and-Wait Protocol

- If channel capacity = b, frame size = L, and round-trip propagation delay = R, then bandwidth utilization = _____
- Conclusion:
  - Long transit time + high bandwidth + short frame length $\Rightarrow$ disaster

Performance of Stop-and-Wait Protocol

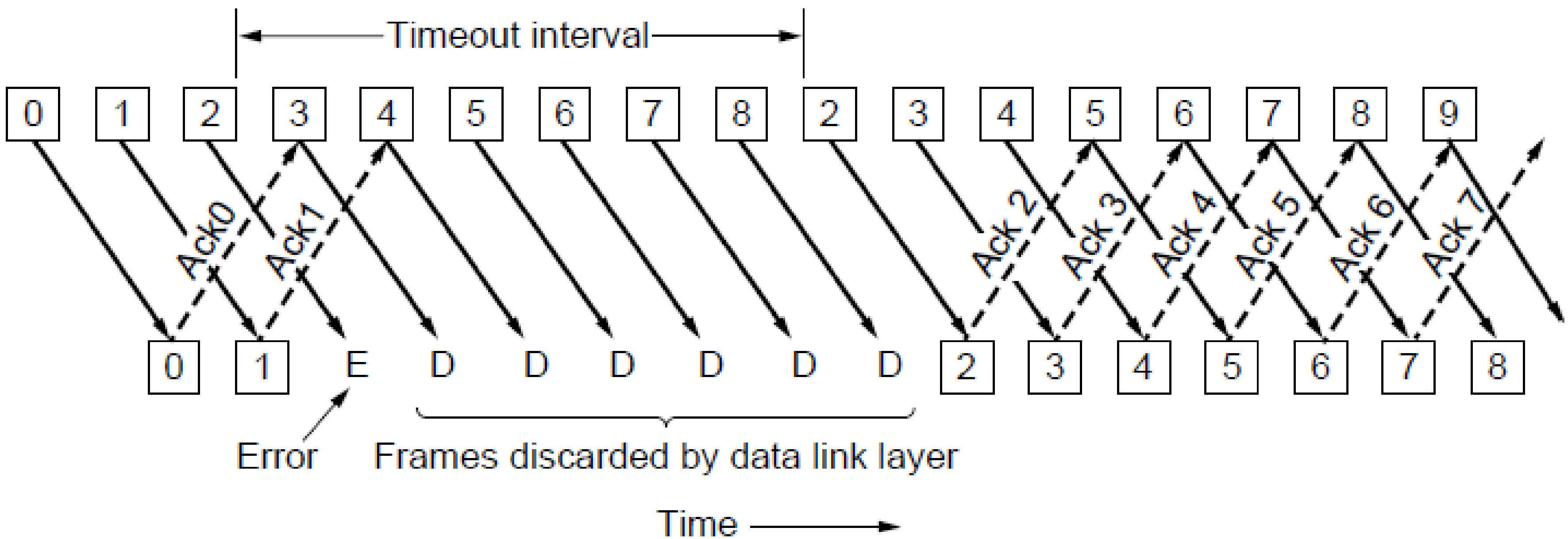
- Solution: Pipelining
  - Allowing *w* frames sent before blocking
- In our example, for 100% utilization
  - *w* = __, max window size = __
  - sequence number = __ bits
- Problem: errors
- Solutions
  - Go back *n* protocol (GNP)
  - Selective repeat protocol (SRP)

Acknowledge *n* means frames *n*,*n*-1,*n*-2,… are acknowledged (i.e., received correctly)

Go Back *n* Protocol

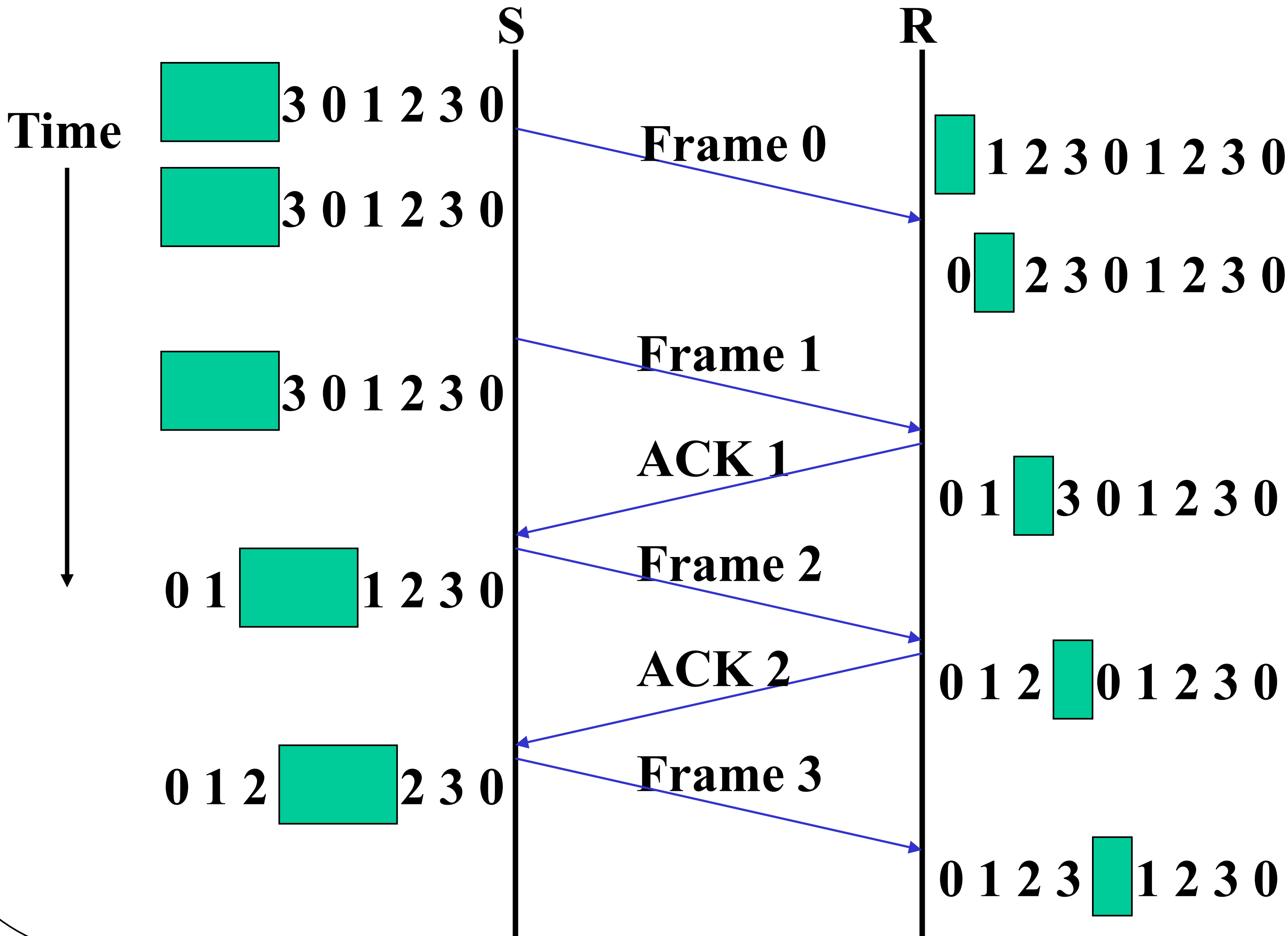
- Receiver discards all subsequent frames following an error one, and send no acknowledgement for those discarded
- Receiving window size = 1 (i.e., frames must be accepted in the order they were sent)
- Sending window might get full
  - If so, re-transmitting unacknowledged frames
- Wasting a lot of bandwidth if error rate is high

Go Back *n* Protocol



(a)

Go Back *n* Protocol



Go Back *n* Protocol

- What is the maximum sending window size?
- Maximum sending window size of = MAX_SEQ, not MAX_SEQ+1
  - With *n*-bit sequence number,
    MAX_SEQ = $2^n - 1$,
    maximum sending window size = 2^n - 1
  - e.g., for 3-bit window, MAX_SEQ = 7, so window size = 7 although max. size could be 8
- Why?

Go Back *n* Protocol - Window Size

- Suppose 3-bit window is used and max sending window size = MAX_SEQ+1 = 8
  - ❑ Sender sends frames 0 through 7
  - ❑ Piggybacked ack 7 comes back
  - ❑ Sender sends anther 8 frames w/ sequence numbers 0 through 7
  - ❑ Another piggybacked ack 7 comes back
  - ❑ Q: Did all second 8-frames arrive successfully or did all of them get lost?
  - ❑ Ack 7 for both cases $\Rightarrow$ Ambiguous

    $\Rightarrow$ Max. window size = 7

Go Back *n* Protocol Implementation

- Sender has to buffer unacknowledged frames
- Acknowledge *n* means frames $n, n-1, n-2$, ... are acknowledged (i.e., received correctly) and those buffers can be released
- One timer for each *outstanding* frame in sending window

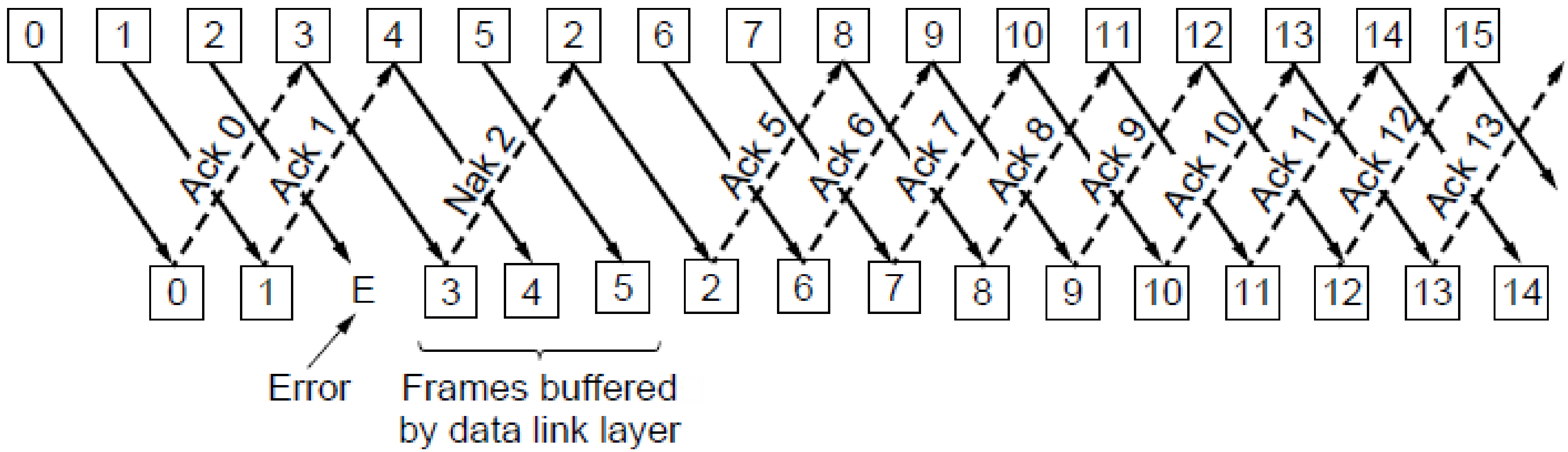
Select Repeat Protocol

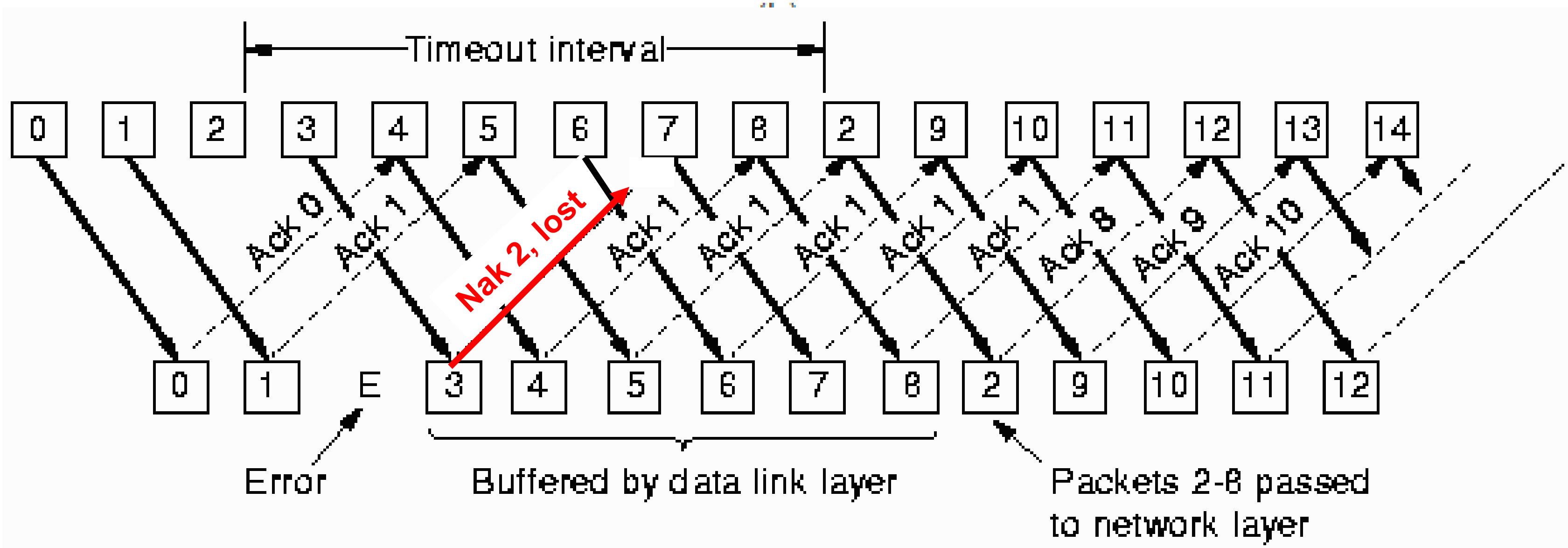
- Receiver stores correct frames following the bad one
- Sender retransmits the bad one after noticing
- Receiver passes data to network layer and acknowledge with the highest number
- Receiving window > 1 (i.e., any frame within the window may be accepted and buffered until all the preceding one passed to the network layer
- Might need large memory

Negative Acknowledgement (NAK)

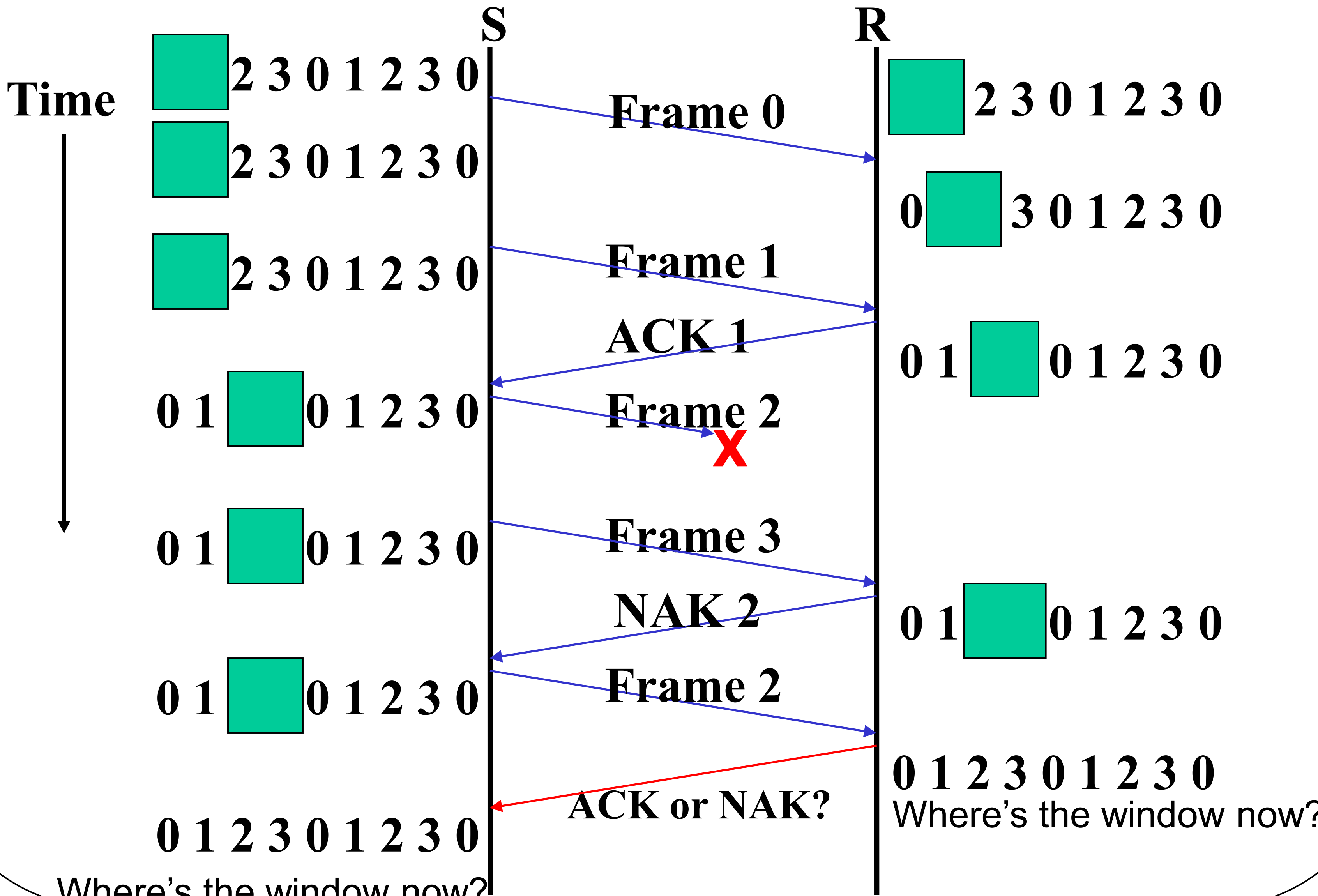
- SRP is often combined with NAK
- When error is *suspected* by receiver, receiver request retransmission of a frame
  - Arrival of a damaged frame
  - Arrival of a frame other than the expected
- Does receiver keep track of NAK?
- What if NAK gets lost?
- To nak, or not to nak: that is the question

Selective Repeat with NAK





Selective Repeat with NAK



Select Repeat Protocol Implementation

- Receiver has a buffer for each sequence number within receiving window
- Each buffer is associated with an "arrived" bit
- Check whether sequence number of an arriving frame within window or not
  - If so, accept and store
- Maximum window size = ? Can it be MAX_SEQ ?

Select Repeat Protocol - Window Size

- Suppose 3-bit window is used and window size = MAX_SEQ = 7

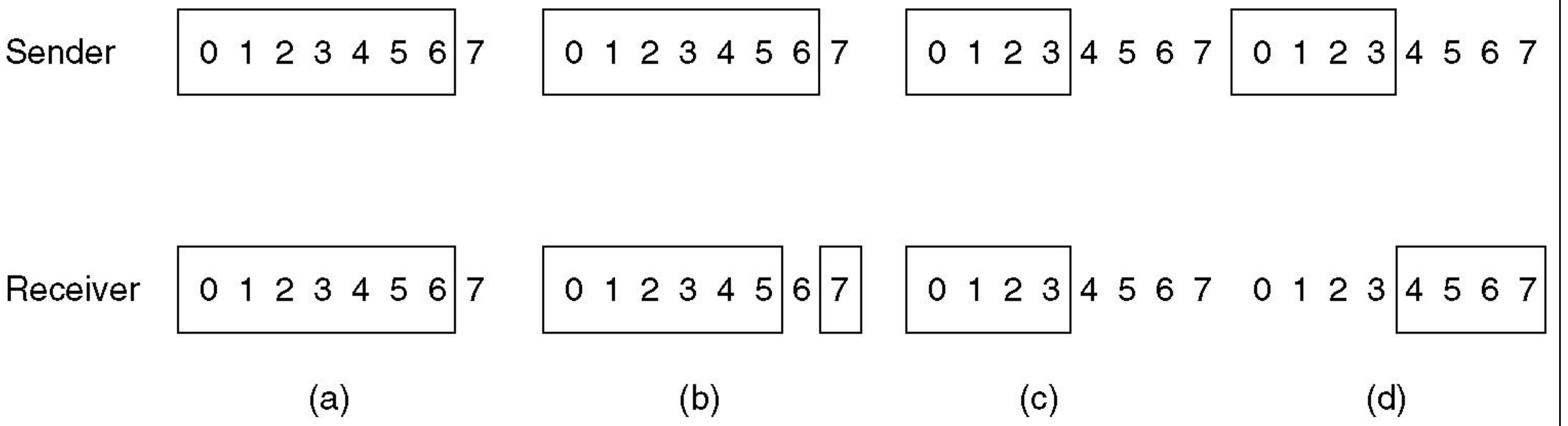
sender	receiver
0 1 2 3 4 5 6 sent	0 1 2 3 4 5 6 accepted
	0 through 6 to network layer
	all acknowledgements lost
0 retransmitted	**0** accepted
ack 6 received	
7 sent	7 accepted
	7 and **0** to network layer

Select Repeat Protocol - Window Size

- Problem is caused by new and old windows overlapped
- Solution
  - ❑Window size=(MAX_SEQ+1)/2
  - ❑E.g., if 4-bit window is used, MAX_SEQ = 15
    $\Rightarrow$ window size = (15+1)/2 = 8
- Number of buffers needed

  = window size

Select Repeat Protocol



(a) Initial situation with a window size seven.

(b) After seven frames sent and received, but not acknowledged.

(c) Initial situation with a window size of four.

(d) After four frames sent and received, but not acknowledged.

Acknowledgement Timer

- Problem
  - If the reverse traffic is light, effect?
  - If there is no reverse traffic, effect?
- Solution
  - Acknowledgement timer:

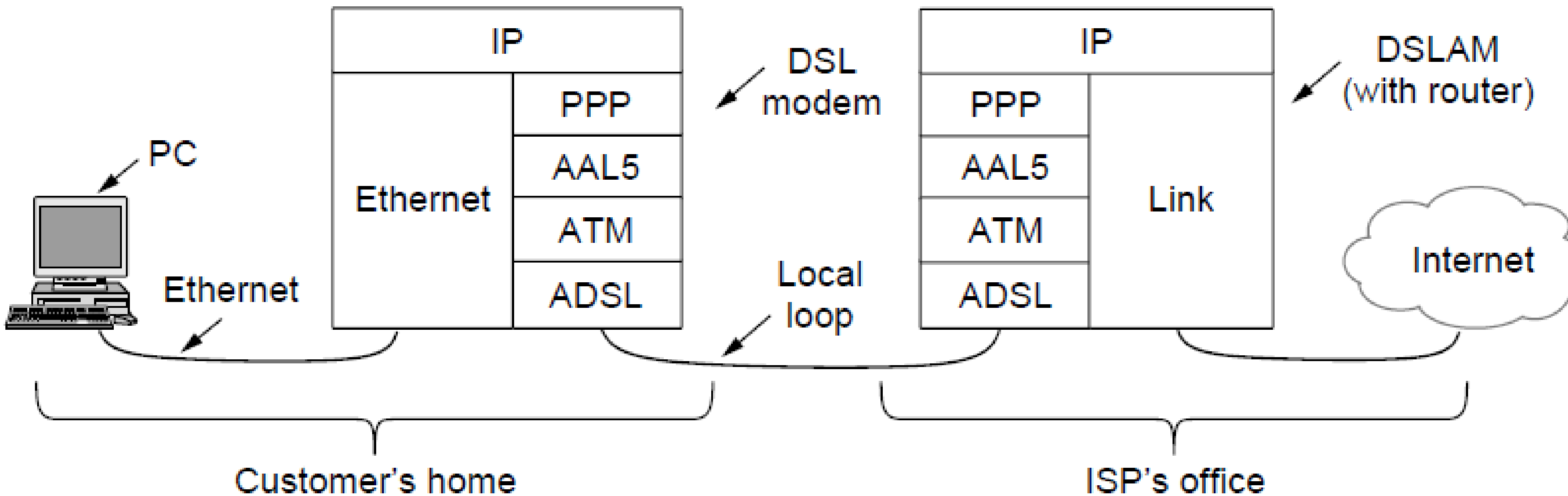
    If no reverse traffic before timeout

    send separate acknowledgement
  - Essential: ack timeout < data frame timeout

    Why?

Example: ADSL

- ADSL protocol stacks



- ATM (Asynchronous Transfer Mode)

ADSL

- PPP (Point-to-Point Protocol) full frame format for unnumbered mode operation

Bytes	1	1	1	1 or 2	Variable	2 or 4	1
	Flag 01111110	Address 11111111	Control 00000011	Protocol	Payload	Checksum	Flag 01111110

- AAL5 (ATM Adaptation Layer 5) frame carrying PPP data

Bytes	1 or 2	Variable	0 to 47	2	2	4
	PPP protocol	PPP payload	Pad	Unused	Length	CRC
	AAL5 payload			AAL5 trailer		