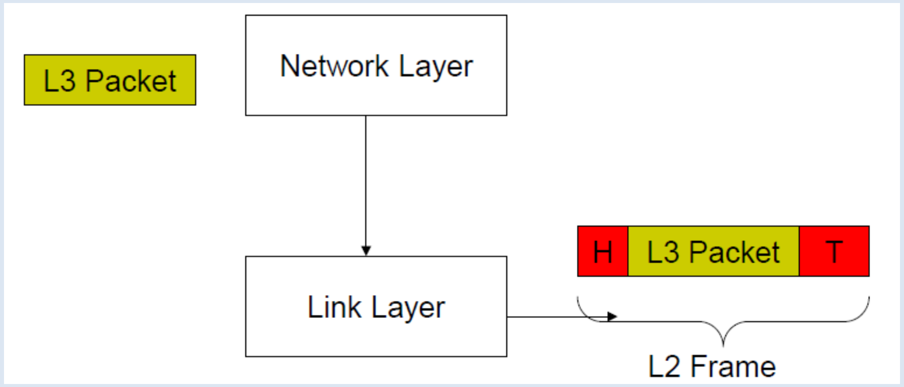
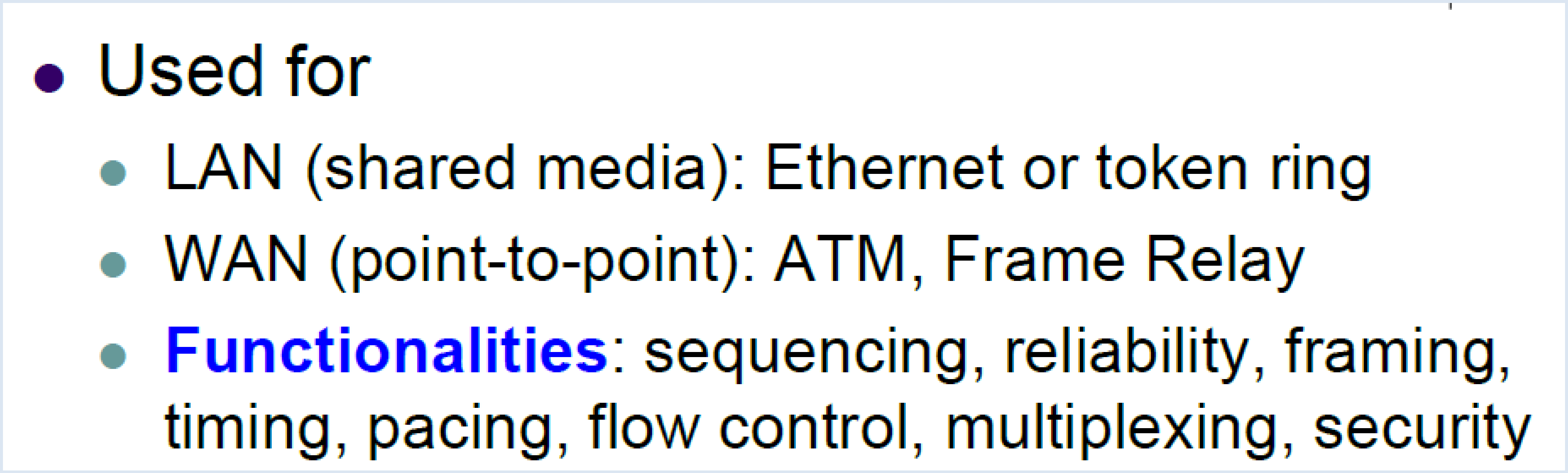
# Data Communication
# Chapter 7
# Data Link Control and Protocol

Himal Acharya
Course Instructor

# Data Link Layer

- Link layer is the second layer (L2) of the OSI layer architecture
- It provides a well-defined service interface to the network layer (L3)





- Used for
  - LAN (shared media): Ethernet or token ring
  - WAN (point-to-point): ATM, Frame Relay
  - **Functionalities**: sequencing, reliability, framing, timing, pacing, flow control, multiplexing, security

## Data Link Layer Design Issues

- DLL makes use of the service provided by the physical layer (L1) to implement ***reliable*** information transmission
  - Need reliable transmissions over a noisy channel requires:
    - The sender breaks the bit stream into frames and appends information necessary **for the receiver** to detect frames received in error → ***FRAMING*** and ***ERROR DETECTION***
    - ***Framing:*** technique to identify the beginning and end of a block of information
    - *Error Detection*: A mechanism by which **the sender** knows that a packet is lost or received in error

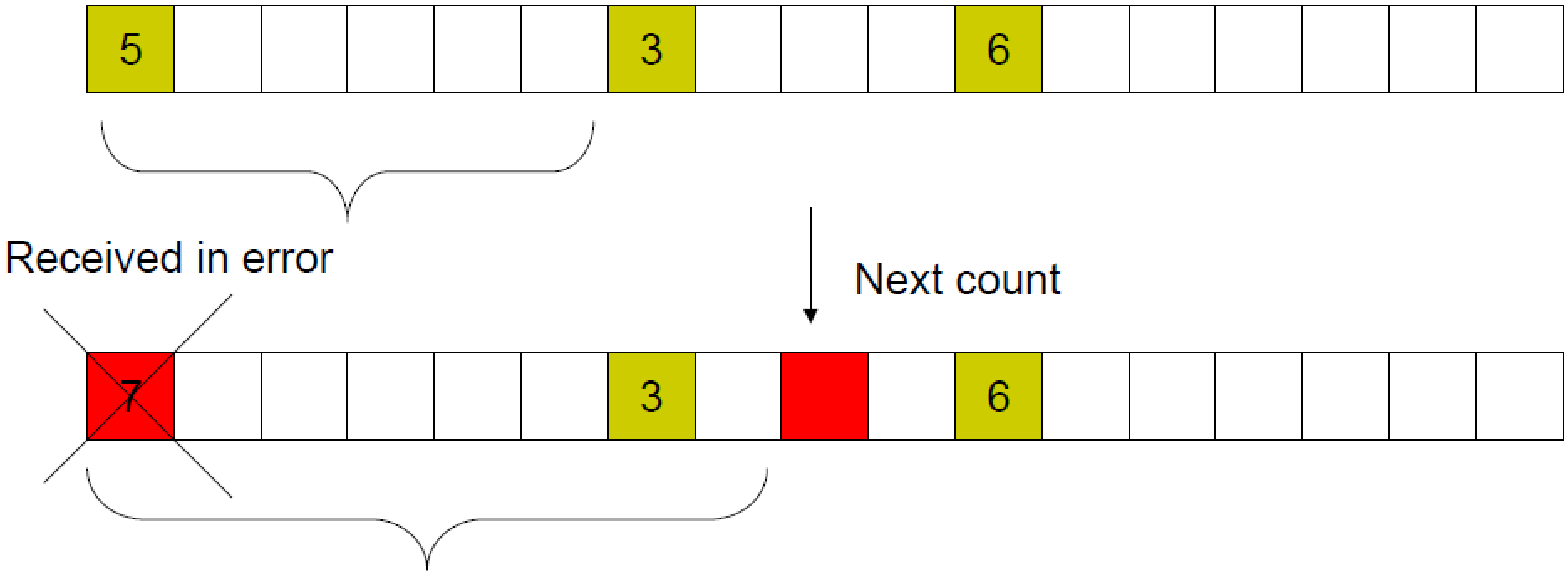
      *Error Control:* Techniques to rectify the error (e.g., retransmission)

- A fast sender can overwhelm a slow receiver resulting in frame loss due to buffer overflow → FLOW CONTROL **may be implemented at DLL**

# Framing

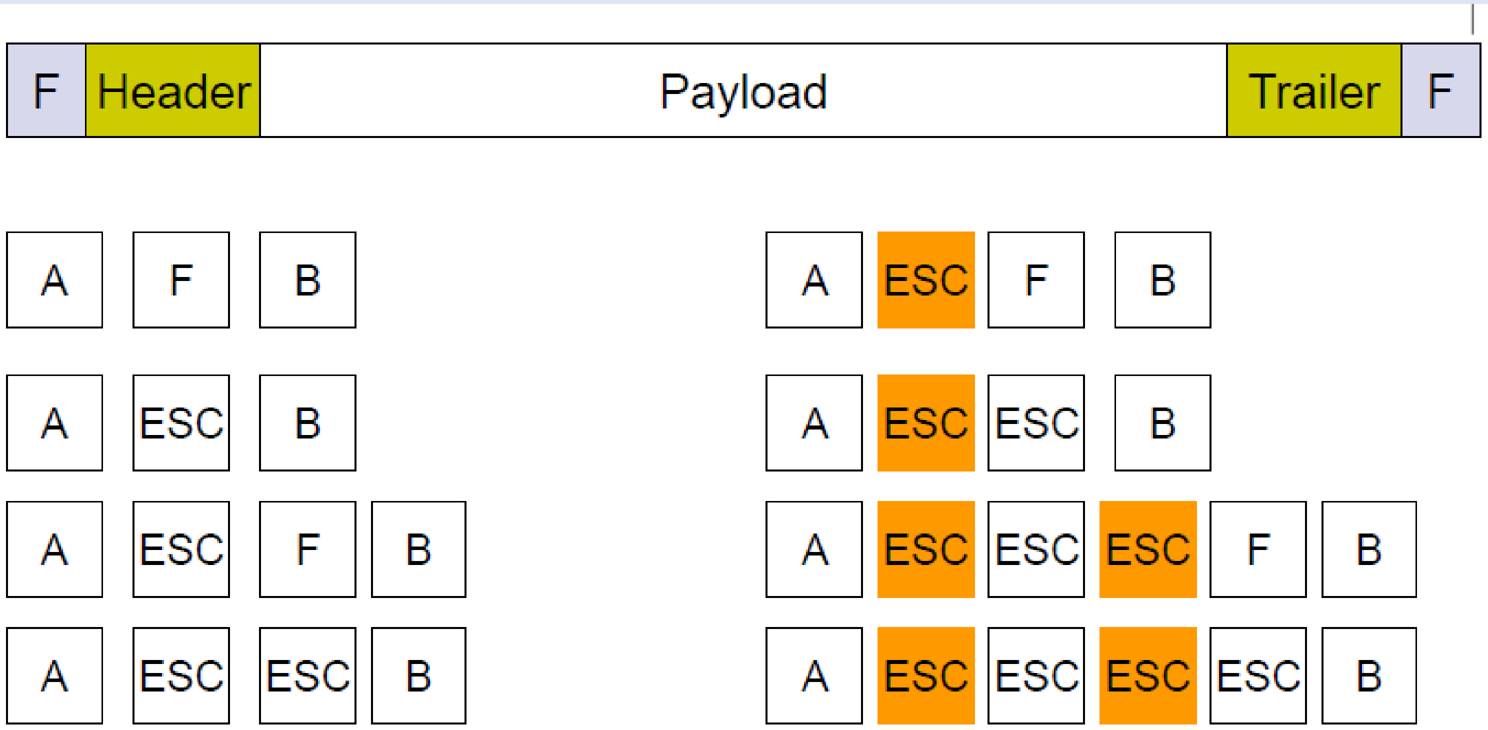
- Purpose: to identify the beginning and end of a block of information
- Main techniques: Character count , Byte stuffing , Bit stuffing

Framing – Character Count



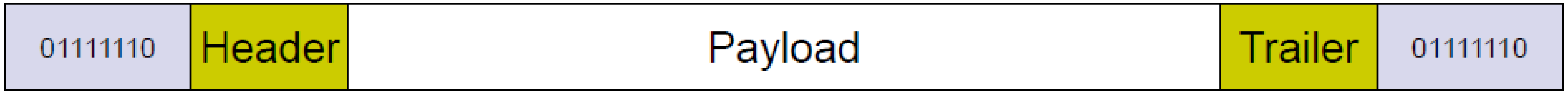
- Uses a field in the frame header to specify the number of characters in the frame (from left to right)
- Transmission errors can cause errors to propagate to multiple frames

# Framing- Byte Oriented



- Each frame starts and ends with a special character (FLAG)
- What happens when the flag character appears in the payload?
  - **Byte stuffing** (or character stuffing) – sender inserts a **special character (ESC)** before each occurrence of flag or ESC patterns **(shown in orange color)**
- Stuffing characters are removed by the receiver

## Framing- Bit Oriented



**0110111111111111111110010**

**011011111011111011111010010**

- Each frame begins and ends with a **special bit pattern** "01111110" (7E in hex)
  - Synchronization errors don't propagate beyond a single frame
- Whenever the sender encounters **5 consecutive 1s** in the data it automatically stuffs a 0 into the outgoing sequence → bit stuffing
  - Stuffing bits are removed by the receiver.

## Flow Control

Flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgment from the receiver. Any receiving device has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data. Thus, the flow of data must not be allowed to overwhelm  the receiver.

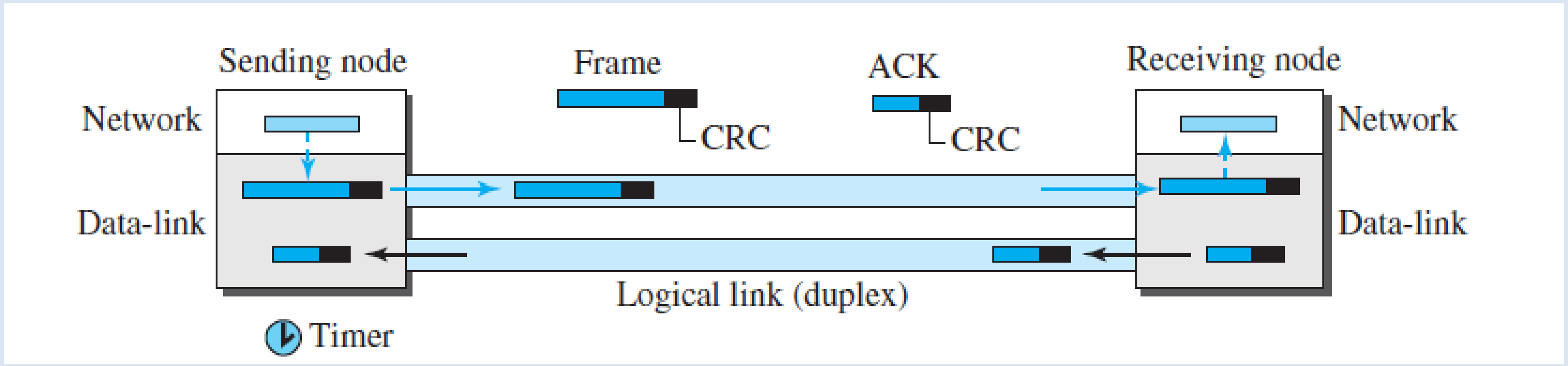
Flow Control Techniques:

- Stop and Wait Flow Control
- Sliding Window Flow Control

## Stop-and-Wait Protocol

- Simple protocol used for flow control
- In this protocol, the sender sends one frame , stops until it receives confirmation from the receiver and then sends the next frame.
- Here, for data frames the communication is unidirectional but auxiliary ACK frames travel from other direction.
- If any frame is not received by the receiver and is lost. Receiver will not send any ACK as it has not received any frame
- Sender will not send the next frame as it will wait for the ACK for the previous frame which it had sent .
  - A deadlock situation can be created
  - Solved by the timer. If the time expires, the sender resends the previous frame. The sender needs to keep a copy of the frame until its ACK arrives. Discards copy to send the next frame.

# Stop and Wait Protocol



## Sliding Window Flow Control

- This protocol allows multiple frames to be in transit
- Destination allocates buffer space for n frames. Source is allowed to send n frames at a time without waiting for any acknowledgments
- Receiver sends acknowledgment with sequence number of anticipated frame
- Sender maintains list of sequence numbers it can send and receiver maintains list of sequence numbers it can receive.
- The window of the frame has limited range: n
- For n = 3 bit, the sequence number can range from 0 to 7,ie 0 to $2^n$-1. For n=3,ie after sequence number 7, the next number is 0.

## Piggybacking

- Piggybacking is a feature for bidirectional flow control using sliding window protocol.
- If two stations exchange data, each needs to maintain  two windows, one for transmit and one  for receiver.
- Each side needs to send the data and ACK  to the other. The data in one direction is piggybacked with the acknowledgment in the other direction.
- Because piggybacking makes communication at the data link layer more complicated, it is not a common  practice.

## Error Control

- Error control is both error detection and error correction. If any frames are lost or damaged in transmission, then error control allows the receiver to inform and retransmit those frames.
- In the transmission of sequence of frames, there is possibility of two types of errors.
    - ❖ Lost frame: A frame fails to arrive at the other side. For example, a noise damage a frame to the extent that the receiver is not aware that a frame has been transmitted.
    - ❖ Damaged frame: A recognizable frame does arrive, but some of the bits are in error- altered during transmission.
- Error control in the data link layer: Automatic repeat request ARQ

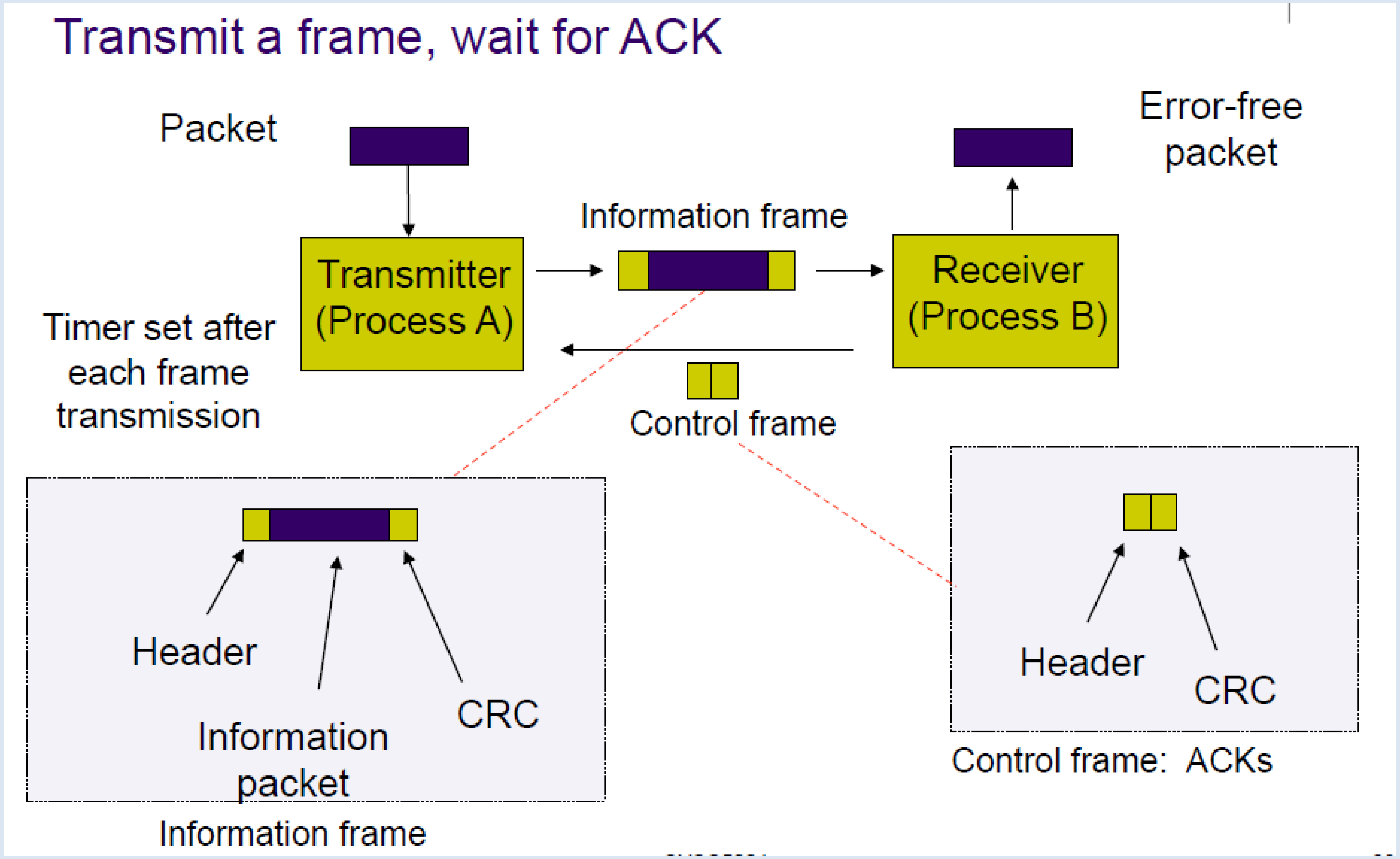
# Automatic Repeat Request ARQ

- *Purpose*: to ensure a **sequence** of information packets is delivered in order and without errors or duplications despite transmission errors & losses
- We will look at:
  - Stop-and-Wait ARQ
  - Sliding Window ARQ
    - Go-Back N ARQ
    - Selective Repeat ARQ

Basic elements of ARQ:

- *ACKs* (positive acknowledgments)
- *NAKs* **(**negative acknowledgments)
- *Timeout mechanism*

# Stop-and-Wait ARQ



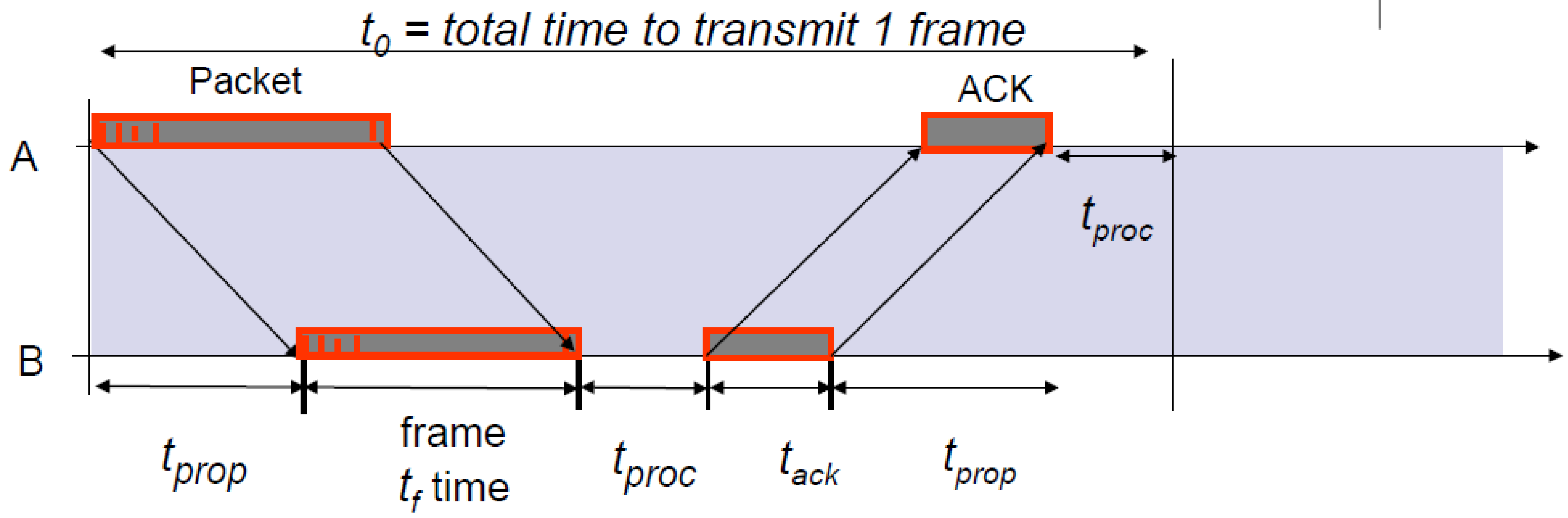
## Sequence Numbers in Stop and Wait ARQ

- Both transmitted frames and ACKs need to have sequence numbers for correct protocol operation

- We can easily show that the frames can be marked as 0,1,0,1,0,1,0 , i.e., with an alternating sequence of 0's and 1's

- This is the reason why the protocol is called Alternating Bit (AB) or Stop-and-Wait protocol
- ACKs are numbered with the sequence number of the frame they correspond to

# Stop-and-Wait Protocol



$$t_0 = 2t_{prop} + 2t_{proc} + t_f + t_{ack}$$

$$= 2t_{prop} + 2t_{proc} + \frac{n_f}{R} + \frac{n_a}{R}$$

bits/info frame

bits/ACK frame

channel transmission rate

Additional time ($t_{proc}$) needed either for B to prepare for ACK or for CRC check.

## Stop and Wait Protocol

- The frames transmitted are numbered alternatively as 0,1,0,1…
- .The receiver sends **ACK** with the same number as the frame
- The transmitter retransmits a frame if a **timeout** is exceeded
- The efficiency of the protocol is defined as the fraction of time spent transmitting new frames
- The efficiency can be computed for error free transmission or for transmission with errors, with $P_f$ the frame error probability

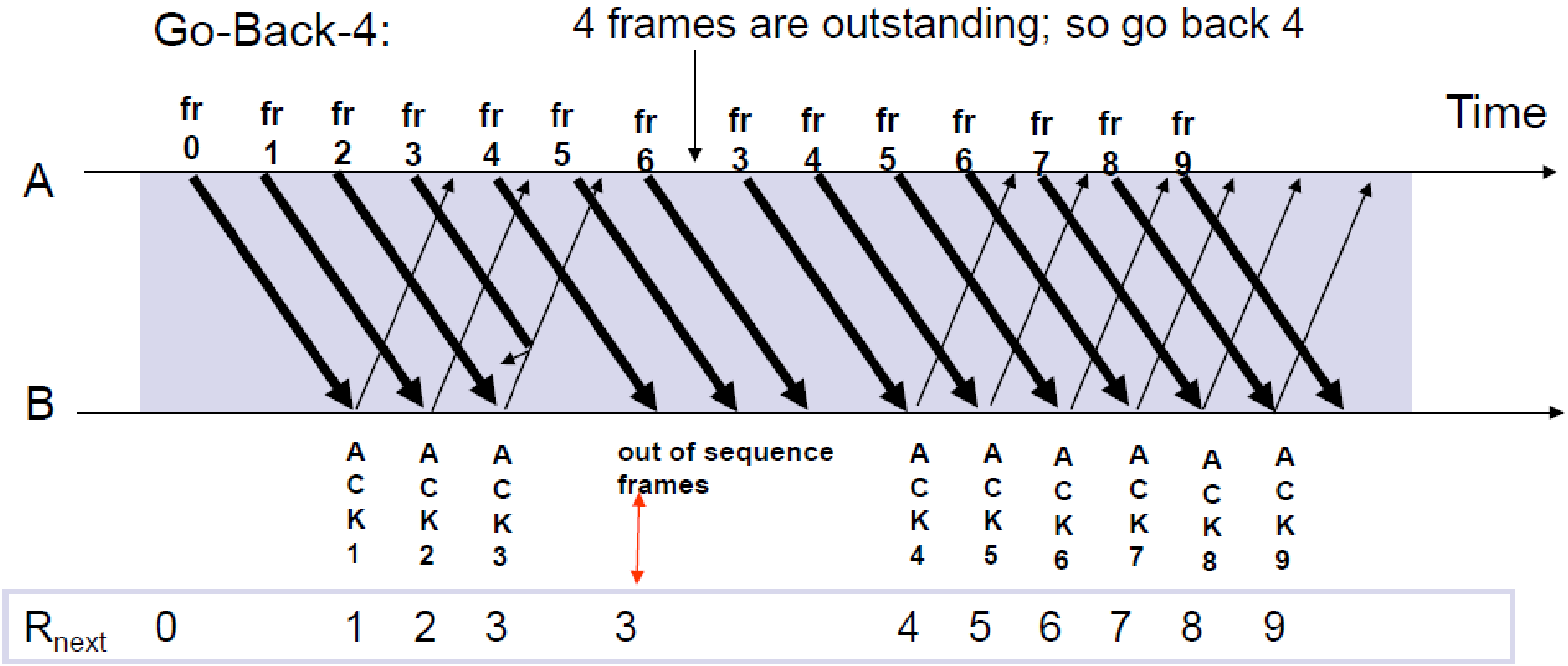
### Application of Stop-and-Wait ARQ

- IBM *Binary Synchronous Communications protocol* (Bisync): character-oriented data link control
- *Xmodem*: modem file transfer protocol
- *Trivial File Transfer Protocol* (RFC 1350): simple protocol for file transfer over UDP

# Go-Back-N ARQ

- Improve Stop-and-Wait by **not waiting**!
- **Keep channel busy** by continuing to send frames Idea: change **STOP** signs to **traffic lights**
- Allow a window of up to $W_s$ outstanding frames. For example, if the sending window size is 4 ($2^2$), then the sequence numbers will be 0,1,2,3,0,1,2,3,0,1, and so on.
- Use *m*-bit sequence numbering for numbering finite number of frame in a sequential manner.
- If **ACK for oldest** frame arrives before window is exhausted, we can continue transmitting
- If window is exhausted, pull back and **retransmit all N outstanding frames**
- Alternative: Use timeout

# Go-Back-N ARQ



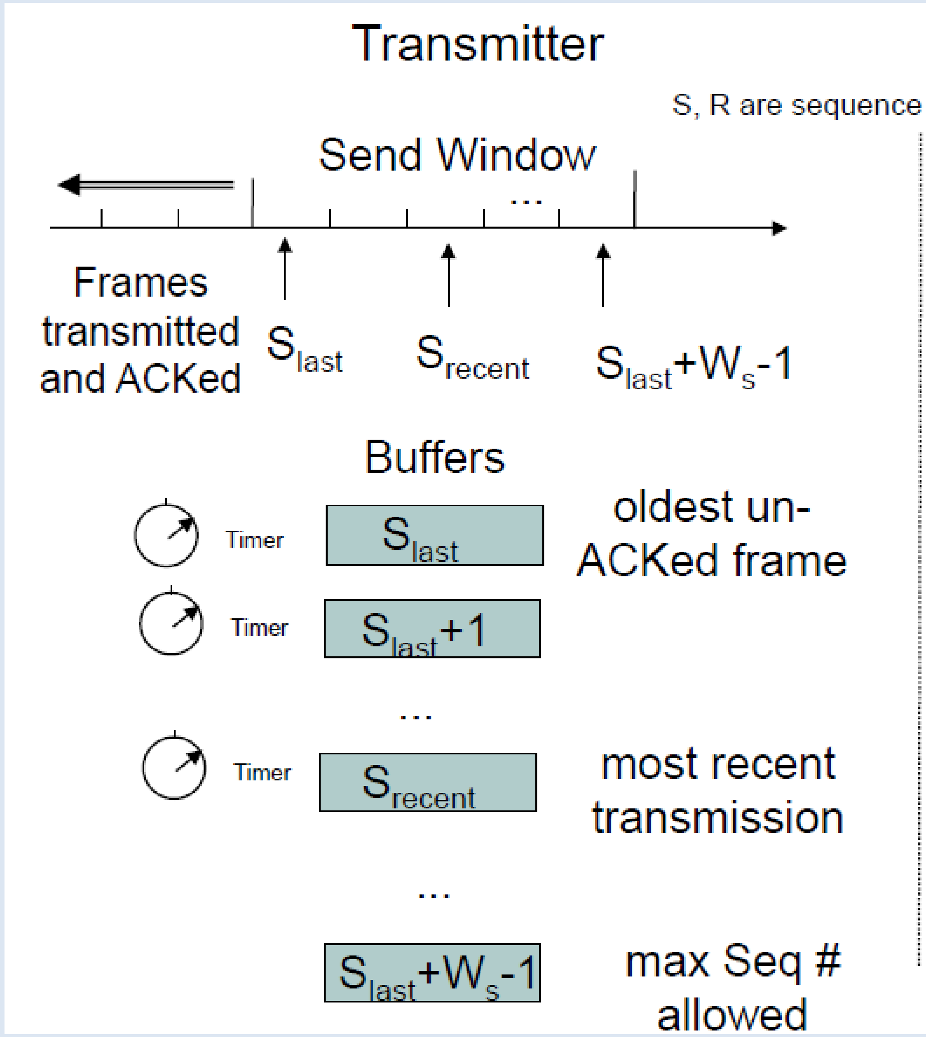
- Frame transmissions are ***pipelined*** to keep the channel busy
- Frame with errors and subsequent out-of-sequence frames are ignored
- Transmitter is forced to go back when window of 4 is exhausted

## Go-Back-N with Timeout

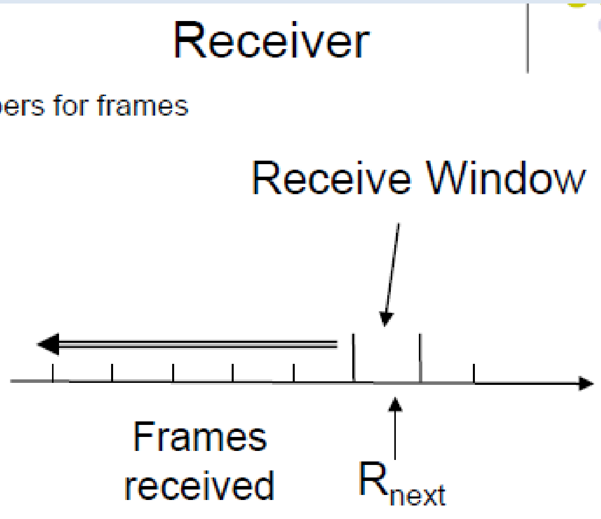
Problem: If a frame is lost and source does not have frame to send, then window will not be exhausted and recovery will not commence

- Use a timeout with each frame When timeout expires, resend all outstanding frames

# Go-Back-N Transmitter and Receiver



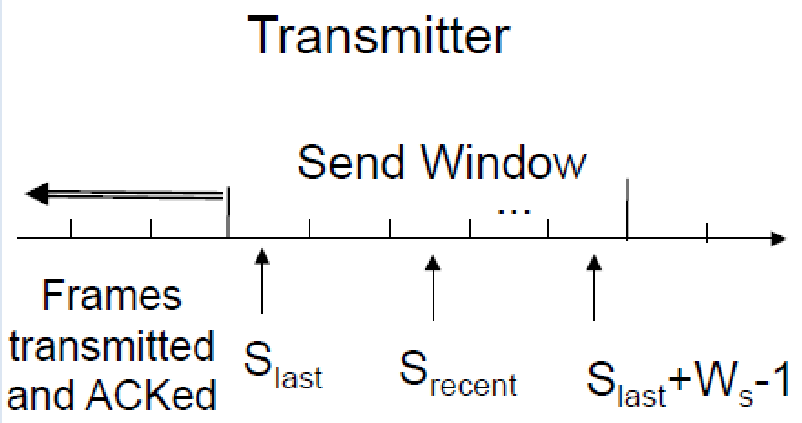
S, R are sequence numbers for frames



Receiver will only accept a frame that is error-free and that has sequence number **$R_{next}$**

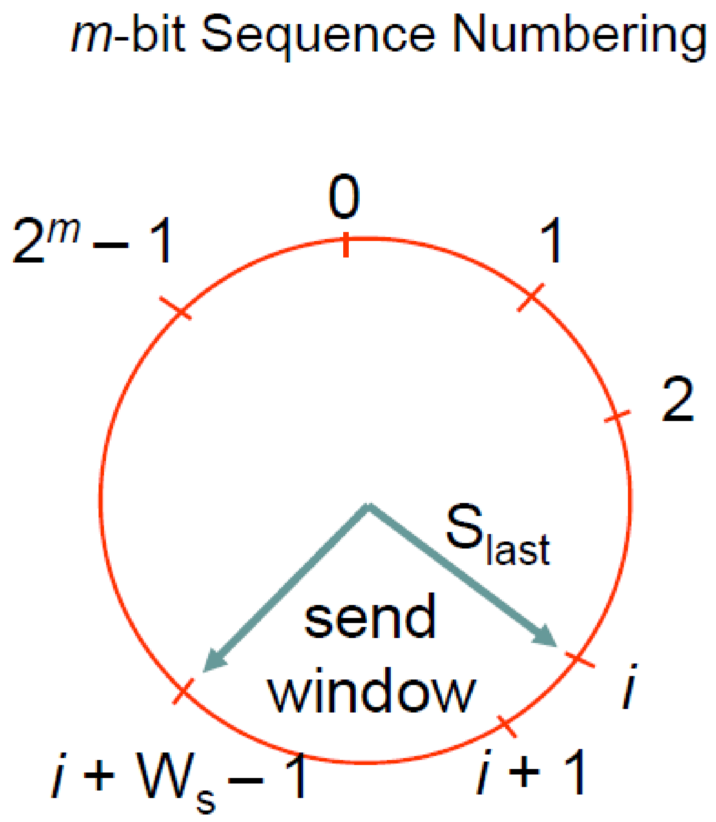
When such frame arrives $R_{next}$ is incremented by one, so the *receive window slides forward* by one

# Sliding Window Operation



Transmitter waits for error-free ACK frame with sequence number $S_{last}$

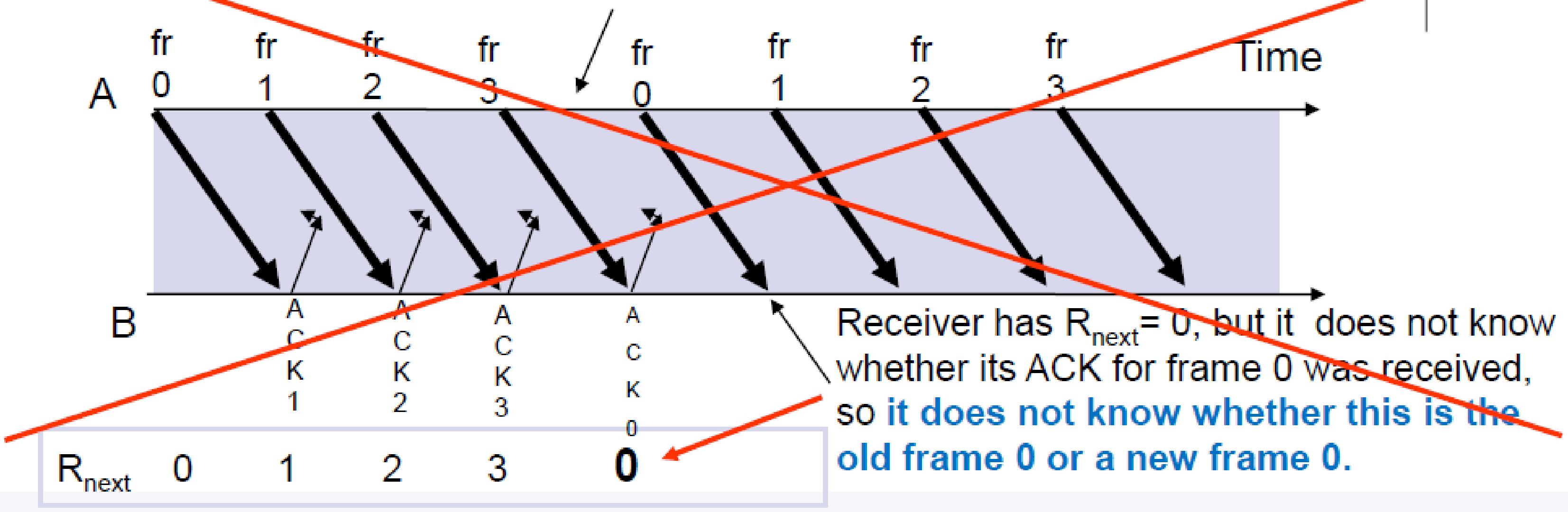
When such ACK frame arrives, $S_{last}$ is incremented by one, and the *send window slides forward* by one



*Assume m bits are used for marking packets and Acks*

*Rule: **Maximum Allowable Window Size** is $W_s = 2^m - 1$*

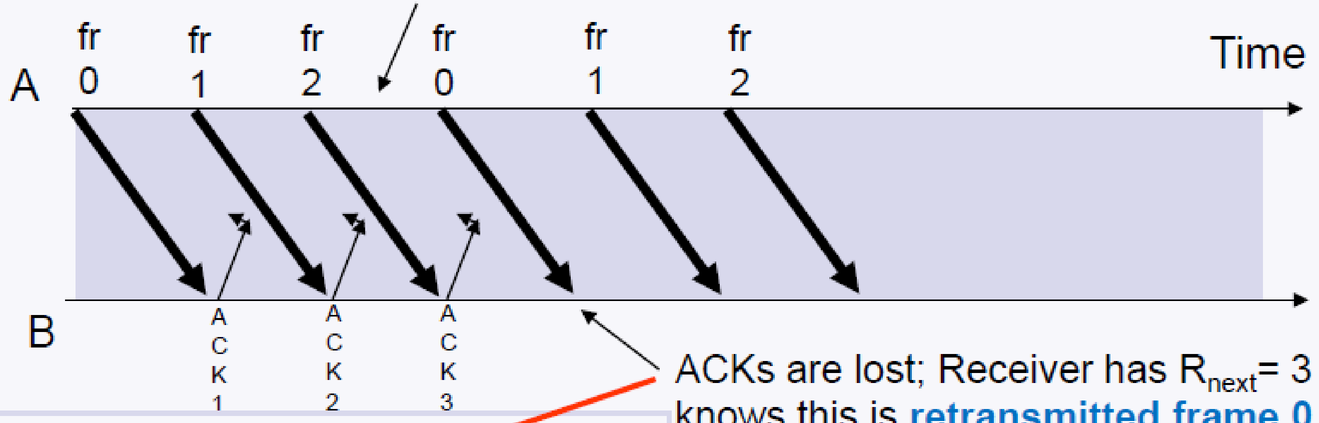
m=2, $2^2 = 4$, **Go-Back - 4: Transmitter goes back 4**



| $R_{next}$ | 0 | 1 | 2 | 3 | **0** |
|---|---|---|---|---|---|

Receiver has $R_{next} = 0$, but it does not know whether its ACK for frame 0 was received, so **it does not know whether this is the old frame 0 or a new frame 0.**

m = 2, $2^2 = 4$, **Go-Back-3: Transmitter goes back 3 (window size is 4 – 1 = 3)**

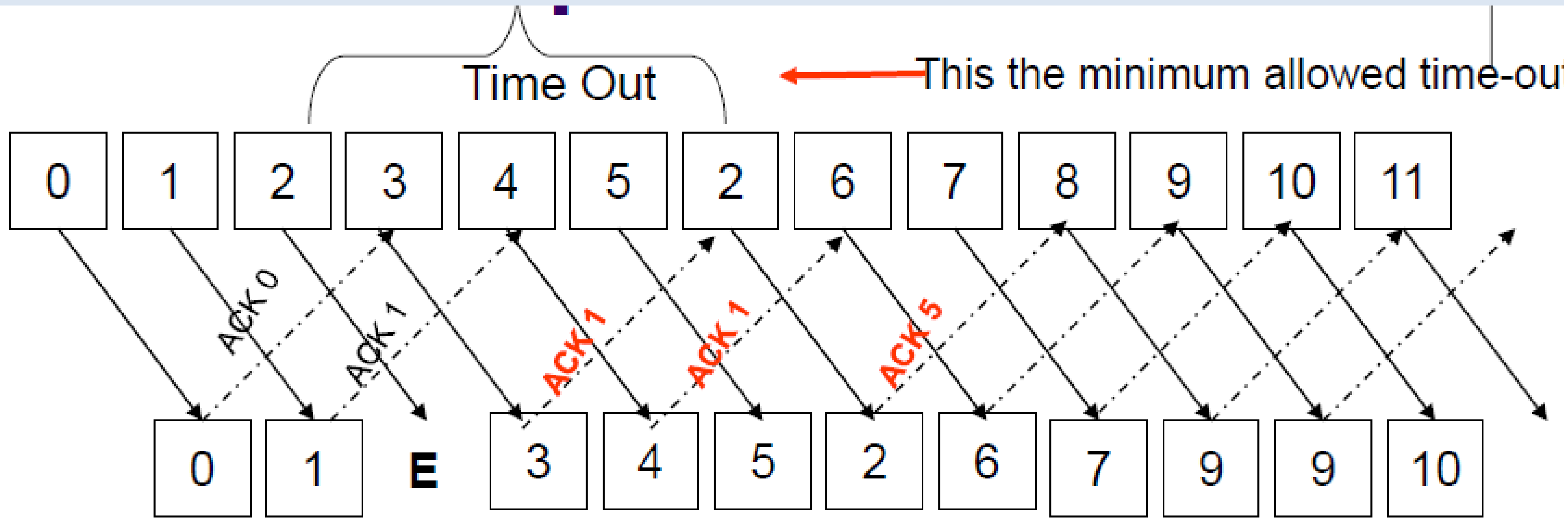


| $R_{next}$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|

OK!

ACKs are lost; Receiver has $R_{next} = 3$ , so it knows this is **retransmitted frame 0** and it rejects the old frame 0

# Selective ARQ



- Go-Back-N ARQ inefficient because ***multiple frames are resent*** when errors or losses occur
- Selective Repeat retransmits ***only an individual frame***
  - **Timeout** causes individual corresponding frame to be resent
  - NAK causes retransmission of oldest un-acked frame (the fact that receiver sends ACK1(this is a NACK) after receiving frame 3 results in frame 2 retransmission earlier than the timeout!)

## Data Link Protocols

- Asynchronous Protocols
  - ▶ treat each character in a bit stream independently
  - ▶ use start and stop bits to frame the data units
  - ▶ less expensive and less complicated equipment
  - ▶ used primarily in modems

- Synchronous Protocols
  - ▶ take the whole bit stream and chop it into characters of equal size
  - ▶ faster than asynchronous transmission

## Asynchronous Data Link Control Protocols - I

- XMODEM (simple, less reliable error checking)
- XMODEM-CRC (more reliable)
- XMODEM-1K (more efficient)
- YMODEM (reliable, multiple files transfer)
- YMODEM-G (fast)
- ZMODEM (fast, good failure recovery)
- X.PC (packet switching network, multiple sessions on one circuit)

## Asynchronous Data Link Control Protocols - II

- KERMIT (reliable, fast file transfer, PC & mainframe)
- Serial Line Internet Protocol (SLIP)
  - Full-duplex
  - IP over asynchronous dial-up or leased lines
  - No error correction
- Point-to-point Protocol (PPP)
  - PC to a TCP/IP network
  - Full-duplex for synchronous and asynchronous transmission
  - Authentication, compression, error correction, & packet sequencing

## Synchronous Data Link Protocols - Classification

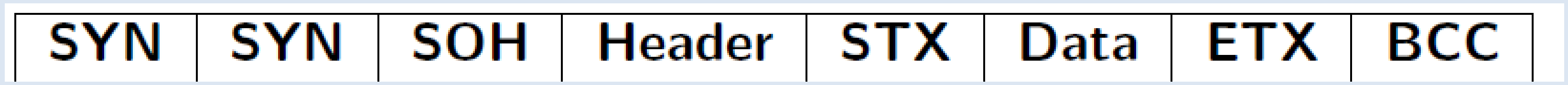
- Character-oriented protocols
  - **Special character** for start and end of message
  - Binary Synchronous Communication Protocol (BSC or BISYNC)
- Byte-count-oriented protocols
  - Special character for start of the header, **count field**, message, block check character (BCC)
  - DEC's Digital Data Communication Message Protocol (DDCMP)
- Bit-oriented protocols
  - Use **flag character** for start and end of message
  - IBM's Synchronous Data Link Control (SDLC)
  - ISO's High-Level Data Link control (HDLC)

## Binary Synchronous Communications

- By IBM
- For 6-bit transcode (SBT), ASCII, EBCDIC
- SYN at start and middle of transmission
- Point to point and multipoint (polling)
- ARQ approach for error checking (ACK1, ACK0, NAK)
- Pros:
  - Transparency and non-transparency modes
  - Efficient, understandable, and widely used
  - Point-to-point & multipoint operations
- Cons:
  - Code dependent
  - Half-duplex protocol
  - Cumbersome for transparency mode

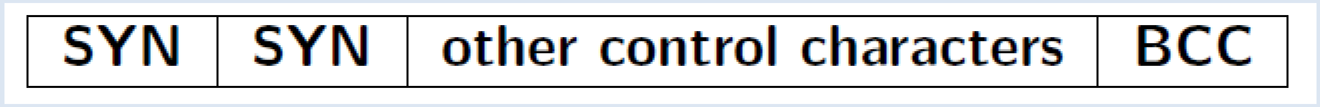
# Binary Synchronous Control BSC protocol

- Data Frame



| SYN | SYN | SOH | Header | STX | Data | ETX | BCC |
|---|---|---|---|---|---|---|---|

- Control Frame: to exchange information between communicating devices.

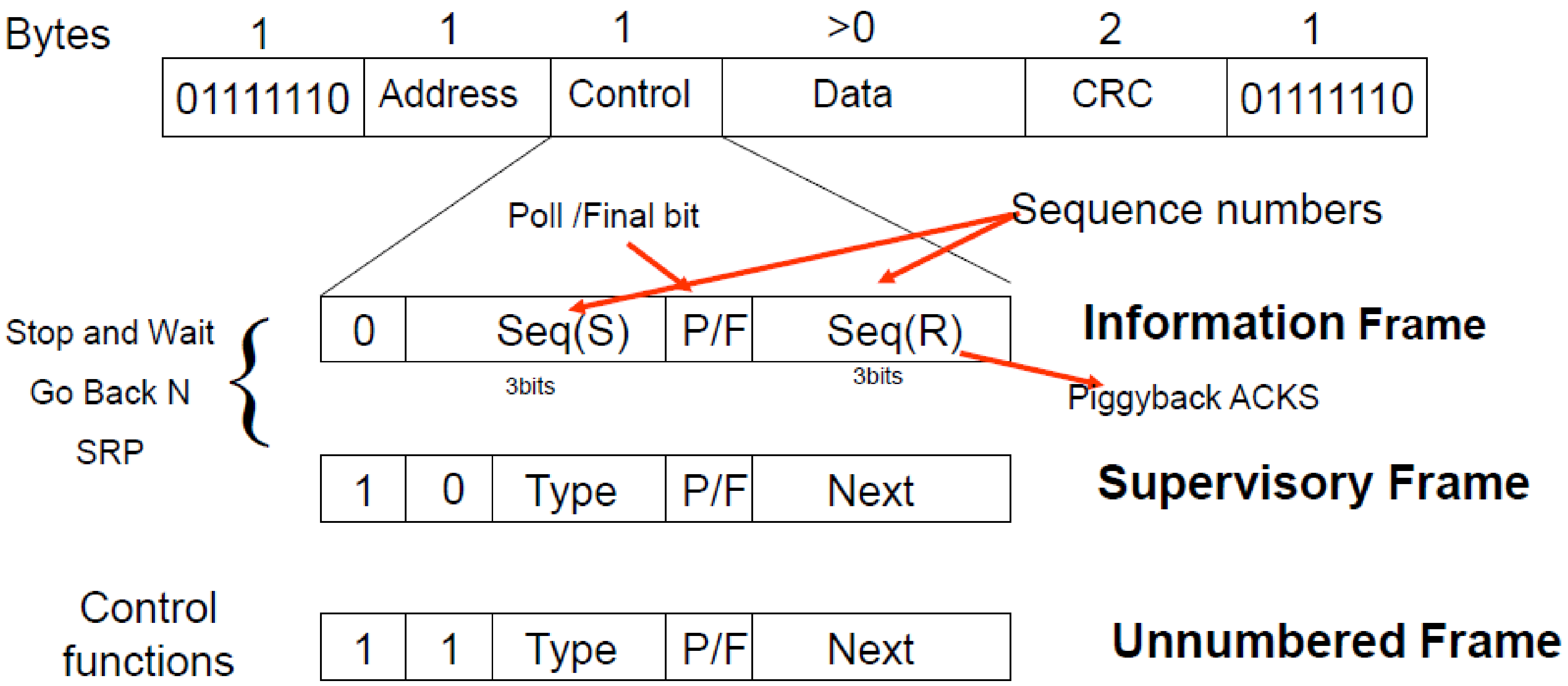


| SYN | SYN | other control characters | BCC |
|---|---|---|---|

- SYN: character alert the receiver to the arrival of a new frame and provide a bit pattern for time synchronization
- SOH: Start of Header
- Header: includes the address of source and destination device and sequence number of frame
- STX (start text): inform receiver next byte is data
- ETX (end text): transition of data and more control character
- BCC (block check count): for error detection

# High-Level Data Link Protocol (HDLC)

- HDLC- High Level Data Link Control
  - Provides a rich set of standards for operating a data link over bit synchronous PHY layers
  - HDLC is a **bit-oriented** protocol (bit by bit synchronization)

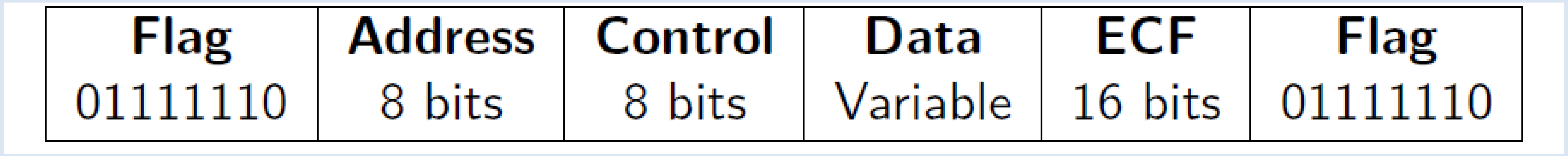


# HDLC Supervisory Fames

- Type 0 → Receiver Ready (Acknowledgment)
- Type 1 → Reject (Negative ACK)
  - *Next* indicates the first frame in sequence not received correctly
- Type 2 → Receiver not Ready
  - It acknowledge all frame up to but not including *Next*
  - No buffers
  - Can be used for flow control
- Type 3 → Selective Repeat
  - It calls for retransmission of only the frame specified

# Synchronous Data Link Control SDLC

- Subset of HDLC



| Flag | Address | Control | Data | ECF | Flag |
|---|---|---|---|---|---|
| 01111110 | 8 bits | 8 bits | Variable | 16 bits | 01111110 |

- ECF: Error Control field: 16 bits for correction