CAN 1011: Data Communication

- Data Link Control Protocols

Contents

- Flow Control
  - Stop-and-Wait flow control
  - Sliding-Window flow control
- High-Level Data Link (HDLC) Protocol
  - Basic Characteristics
  - Frame Structure

What is Data Link Control?

- The logic or procedures used to convert the raw stream of bits handled by the physical layer into a "reliable" data link
- Performed by the Data Link Control Protocol (Layer)
- Requirements and Objectives:
  - Frame-level synchronization: Recognize frame start and end
  - Flow control: Regulate sending of frames to match the ability of RX to absorb them
  - Error control: Retransmission of damaged or unacknowledged frames
  - Addressing: Identify stations on a multi-point link
  - Allow control information to go with data on same link
  - Link management: To initiate, maintain, and terminate data exchange

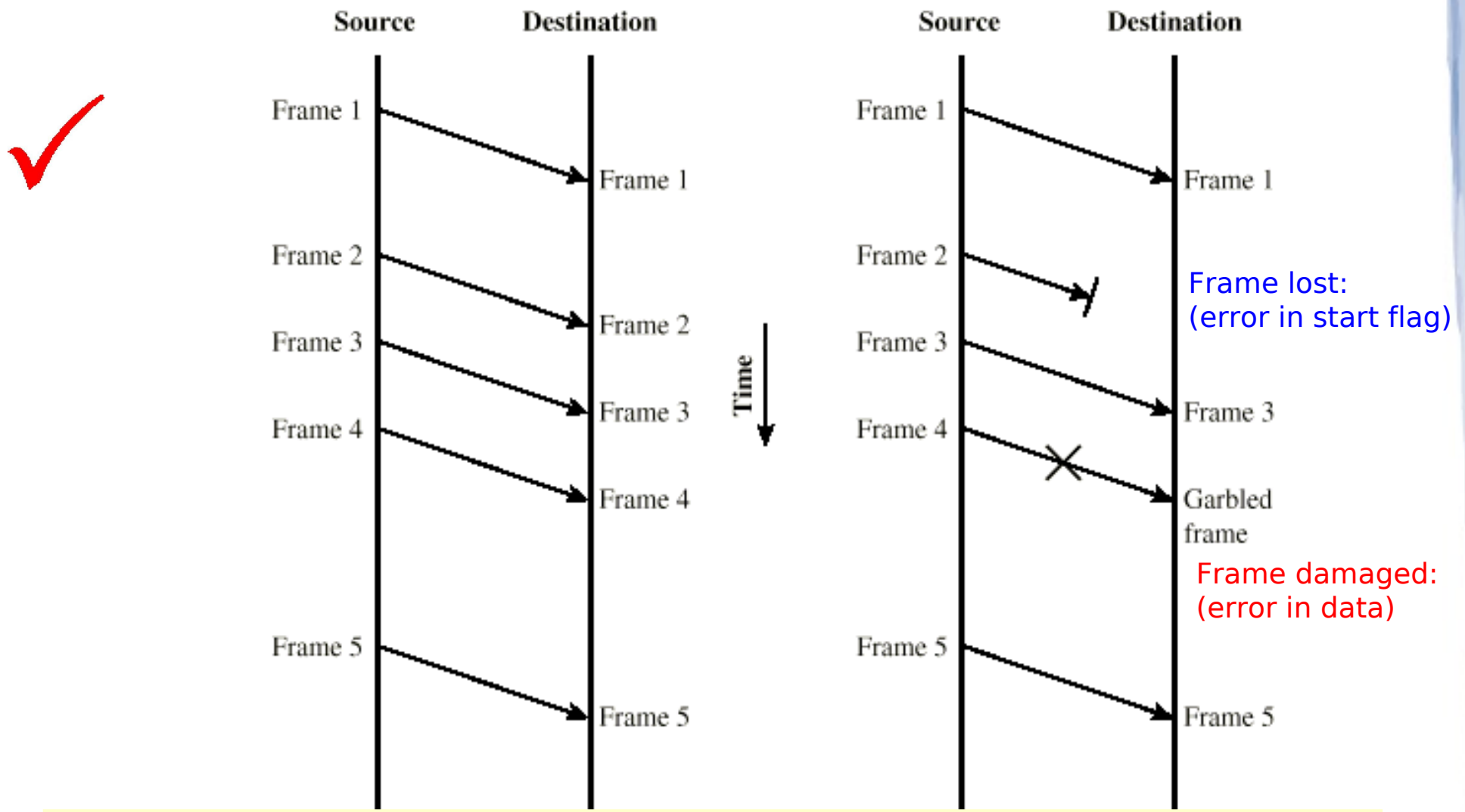
Flow Control

- Required to avoid the TX overwhelming the RX by the flow of data it sends
- RX does *not* 'absorb' the received data instantly!
- It *buffers* (temporarily stores) the data it receives in a finite-size buffer to do some processing before sending it upward to higher layers
- Without flow control, the RX buffer may overflow and data gets lost…

Flow Control over a link (assume no error)

- For now, assume:
  - No frames lost (loss over a single link is a kind of error in recognizing the frame start)
  - No frames arrive in error
  - Frames arrive in the same order they were sent, after a propagation delay

Model for Frame Transmission over a link

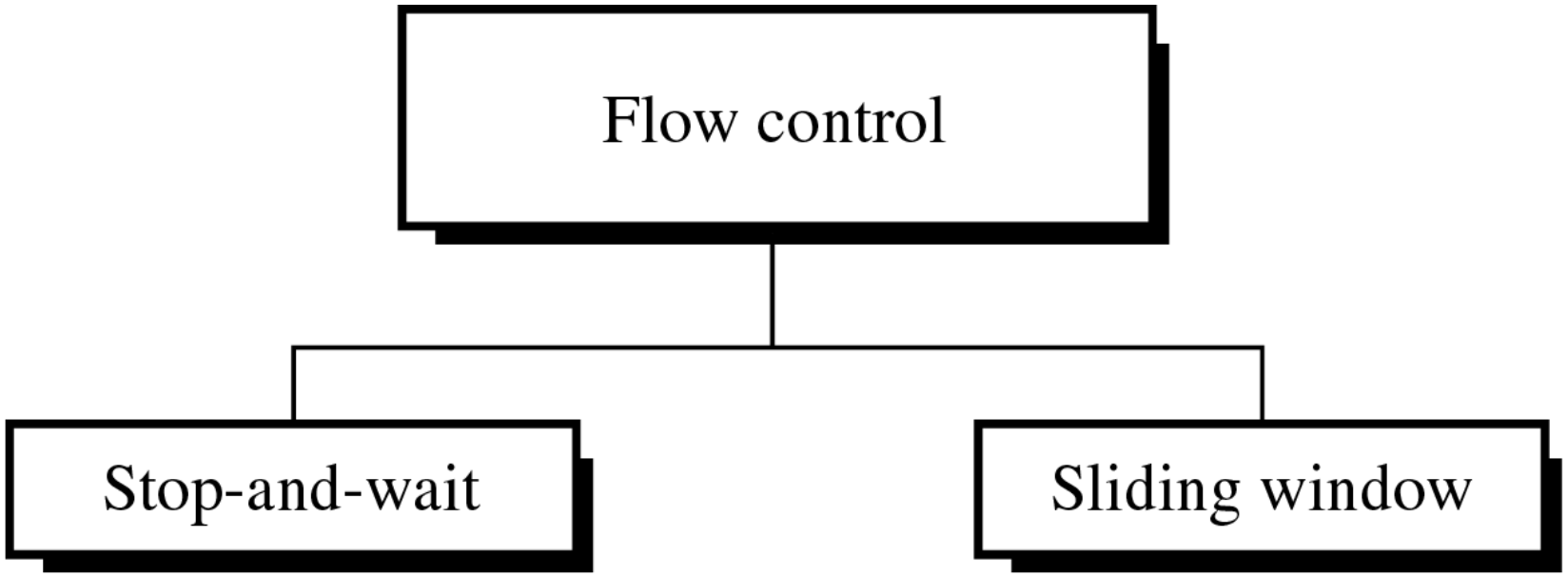


Do you allow only one or multiple frames to travel on the link at any given time?

(a) Error-free transmission

(b) Transmission with losses and errors

Main Flow Control Protocols



Send one frame at a time

Send several frames at a time

Stop and Wait

- Frames sent and acknowledged one at a time:
  1. Source transmits frame and waits for ACK
  2. Destination receives frame and replies with an acknowledgement ACK
  3. When source gets ACK, it sends next frame
- Disadvantages:
  - Destination can stop the flow by not sending ACK

    (but we can use timeout to overcome this)
  - Not efficient - wastes time in waiting until short data frames arrive on long links (frame does not 'fill' the link)

i.e. Would like to make frames long in time (large in size for a given data rate)
$T_f = LT_b = L/R$

Data Fragmentation (smaller frames!)

- However, large blocks of data are often split into several smaller frames- Why?
  - Limited frame buffer size at RX
  - To reduce frame error rate: remember? FER = $1-(1-BER)^F$
  - Errors are detected sooner (when frame arrives)
  - On error, we need to retransmit a smaller amount of data
  - On a shared medium, e.g. a LAN, this ensures that a transmitting station does not occupy the medium for a long time

- → "Stop and wait" is inadequate in such situations where frames are "short"
- Link utilization depends on the frame length in time relative to the link propagation time.

Stop and Wait Link Utilization: The 'a' ratio

- Total number of bits in a frame = L bits, T_b = bit duration
- R = Data rate, bps
- Frame transmission time, t_f sec: Time taken by the TX to emit all the frame bits into medium t_f is large for large frames and low data rates

$$t_f = LT_b = \frac{L}{R}$$

- d = Link physical length, m , V = Velocity of propagation over link, m/s
- Link Propagation time, t_p sec: Time for a bit to traverse the link (link length in time)

$$t_p = \frac{d}{V}$$

- Link length in bits, **B bits/link**: The number of bits that 'fill' the link if data is transmitted continuously
- a = Propagation time / Transmission time = t_p/t_f
-

$$a = \frac{t_p}{t_f} = \frac{\frac{d}{V}}{\frac{L}{R}} = \frac{R[\frac{d}{V}]}{L} = \frac{B}{L}$$

If $t_f = 1$, a = Propagation time t_p

$$B = R\,(\text{b/sec})\,[t_p\,(\text{sec})] = R\,[\frac{d}{V}]$$

Smaller 'a' means better link utilization (with large frames)

Stop and Wait Link Utilization (Efficiency)

- Let us define link utilization, U, as:

$$U = \frac{\text{Useful time during which link is used for transmitting data}}{\text{Elapsed time}}$$

- Will demonstrate on next slide that U is given by:

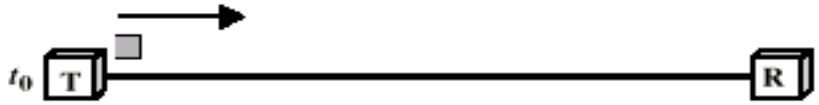
$$U = \frac{1}{2a + 1}$$

- Where 'a' is $$a = \frac{\text{Propagation Time}}{\text{Frame Transmission Time}}$$

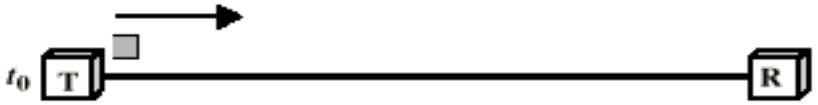
- Utilization is high for small a: U approaches 1, i.e. 100% efficiency
  - Shorter links, Higher propagation velocities
  - Larger frames sizes, Lower data rates
- Efficiency is poor for large a:
  - Longer links, Lower propagation velocities
  - Smaller frames sizes, Higher data rates

Stop and Wait Link Utilization: 2 cases:

Link is longer than frame: $t_p > t_f$: **a > 1**



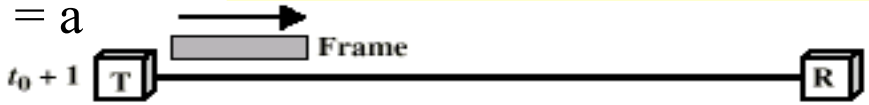
Link is shorter than frame: $t_p < t_f$: **a < 1**



$t_f = 1$

$t_p = a$

Let frame transmission time $t_f = 1 \Rightarrow$ a = Propagation time = t_p



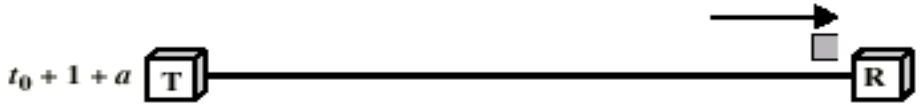
Link 'empty' most of the time !

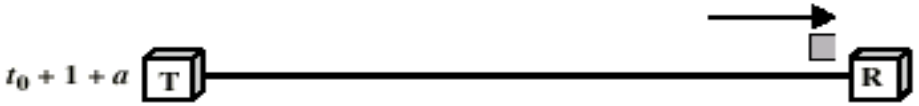
⇒ Underutilized

a < 1

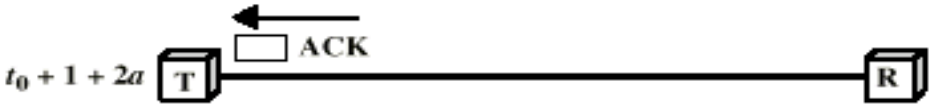
Link 'full' most of the time

⇒ Better utilized





\+ Overhead of an ACK frame in both cases !

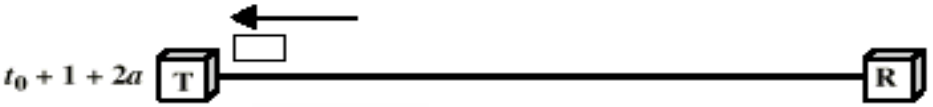


$t_p > t_f$ (a) $a > 1$

$$U = \frac{1}{1 + 2a}$$

1 ← Useful TX time

1 + 2a ← Elapsed time



(b) $a < 1$ $t_p < t_f$

Stop and Wait Efficiency: Example

- Compare the efficiency of stop-and-wait flow control for two links using the parameter 'a':
  - Fame size, L = 1000 characters of 8 bits each, = 8000 bits

Link is longer than frame: a > 1

- Satellite link between 2 ground stations
- d = 2 x 36,000 km, Data rate, R = 1 Mbps
- Typical wave velocity, V = 3 x 10^8 m/s

$$a = \frac{t_p}{t_f} = \frac{B}{L} = \frac{Rd}{LV} = \frac{10^6(2x36x10^6)}{8000(3x10^8)} = 30$$

- Frame TX time, t_f = L/R = 8000/(1x10^6) = 8 ms
- Propagation time, t_p = d/V
  = 2x36x10^6/(3x10^8) = 240 ms
- End of first frame reaches RX after 8+240 = 248 μs from start
- ACK takes 240 ms more to reach TX, i.e. it starts sending 2nd frame after 488 ms
- Utilization = 8/488 = 1.6% (=1/(2a+1))

Link is shorter than frame: a < 1

- 200-m optical fiber link
- Data rate, R = 1 Gbps
- Typical wave velocity, V = 2 x 10^8 m/s
- $$a = \frac{t_p}{t_f} = \frac{B}{L} = \frac{Rd}{LV} = \frac{10^9(200)}{8000(2x10^8)} = 0.125$$
- Frame TX time, t_f = L/R = 8000/(1x10^9) = 8 μs
- Propagation time, t_p = d/V
  = 200/(2x10^8) = 1 μs
- End of first frame reaches RX after 8+1 = 9 μs from start
- ACK takes 1 μs more to reach TX, i.e. it starts sending 2nd frame after 10 μs
- Utilization = 8/10 = 80% (=1/(2a+1))

Sliding Windows Flow Control

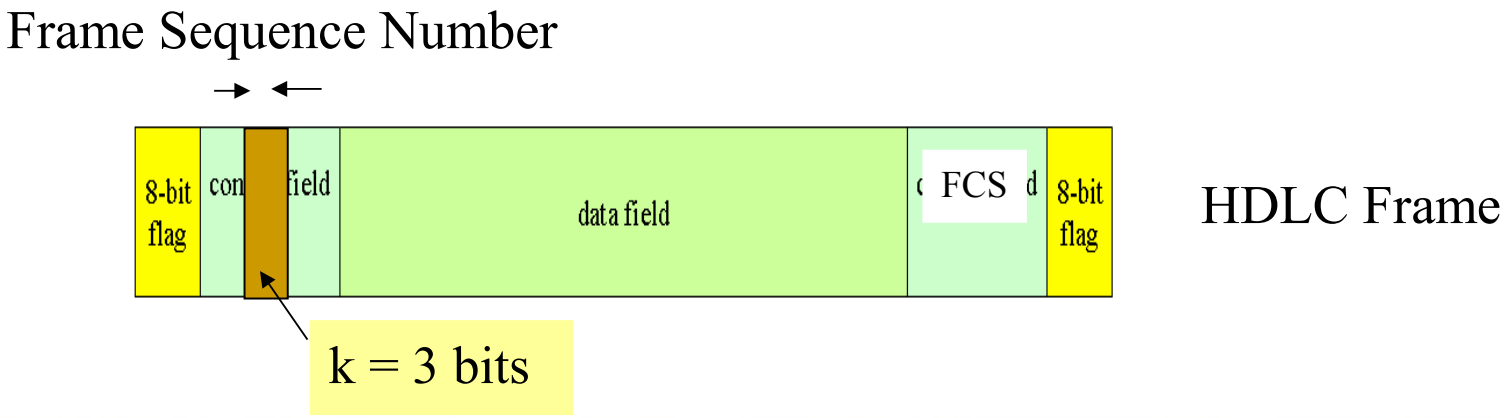
- Avoids the low efficiency of Stop-and-wait when a > 1
- Allows *multiple* frames to be "in transit" simultaneously on the link

- RX keeps a buffer store (in memory) for W frames
- So, TX can send *up to* W frames without waiting for ACK
- Each frame carries a sequence number
- ACK from RX shows the number of next expected frame
- TX keeps a list of frames it can send
- RX keeps a list of frames it expects to receive
- These lists form *sliding windows* at TX and RX that shrink/expand as frames are sent, and ACKs are sent/received
- Hence, Sliding Windows Flow Control

Frame Sequence Numbering

- Frame sequence number is limited by the size of a corresponding field in the frame, e.g. k bits
  - Frames are numbered modulo 2^k
  - e.g. for k = 3, frame sequence # is modulo $2^3 = 8$, i.e.= 0,1,..., 7, 0,1,…
- Window size (W) is limited to a maximum of 2^k-1

i.e. $W_{max} = 7$ in the above example

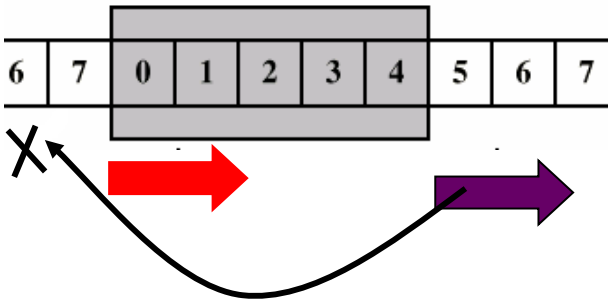


HDLC Frame

Sliding Window Send/Receive Cycle

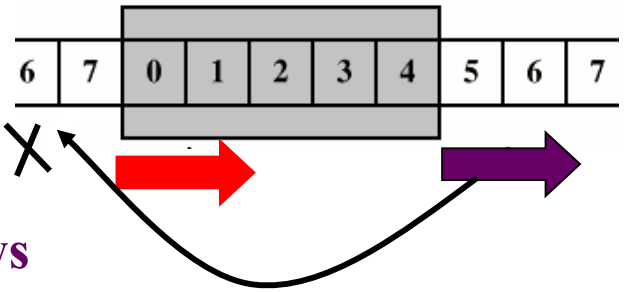
TX

Window covers
frames to be sent



RX

Window covers
frames to be received



Flexible-
(Not rigid) Windows

When you want to ACK some frames:
1. Delete ACKed frames from buffer
2. Expand Window to receive more
3. Send Acknowledgement

1. Delete ACKed frames from buffer
2. Expand Window to send more

ACKs

When you want to send more:
Send frames

Shrink window past received frames

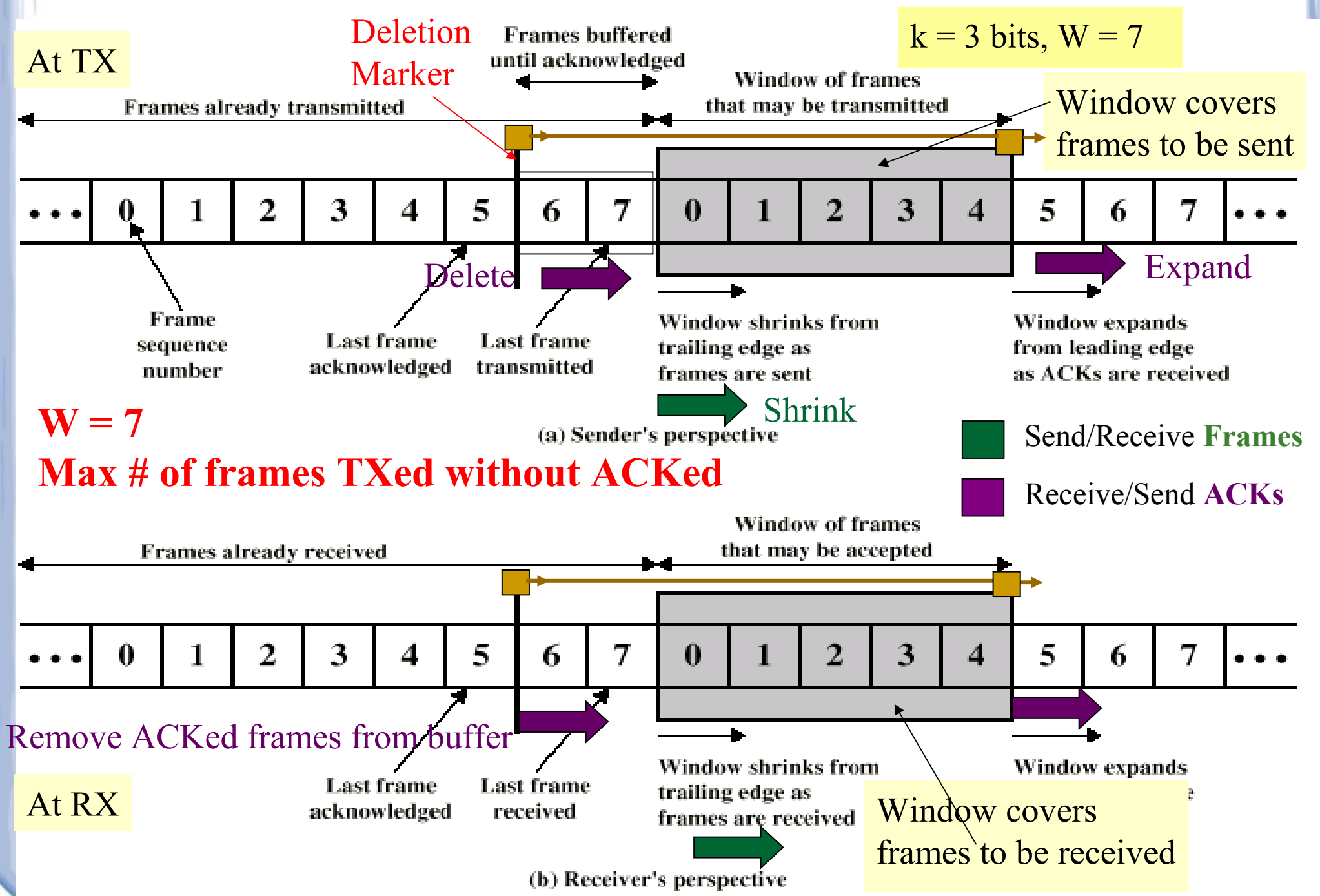
Frames

Shrink window past sent frames

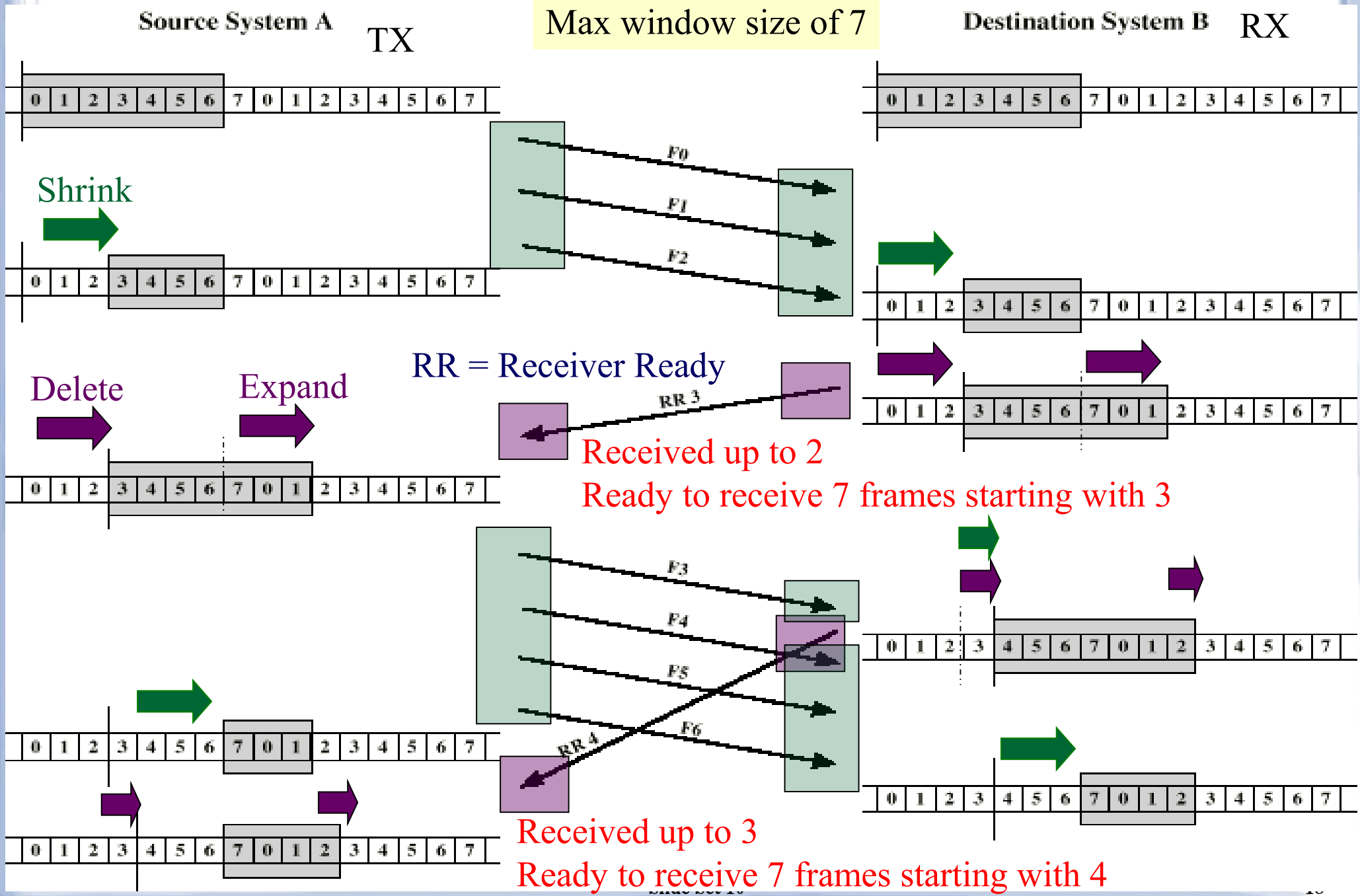
Receive Frames

Acknowledging frames is a separate issue from receiving them

Sliding-Window Diagram



Example Sliding Window



Sliding Window Enhancements

- RR n: Positive receive ACK that asks for more (received up to frame n-1 and ready for n)
- Receiver can also acknowledge receiving frames without permitting further transmission:

  (Receive Not Ready RNR)
  - Example RNR 5: “Received frames up to 4, but not ready for 5 and beyond yet”
  - When it becomes ready, RX must send a normal acknowledge (RR 5) later for TX to resume sending frames

Sliding Window Protocol: Efficiency

- Much more efficient than Stop and wait for a>1
- Treats link as *a pipeline* to be filled with several frames in transit simultaneously- not just one by one
- With window size W and assuming no error, link utilization, U, is given by

$$U = \begin{cases} 1 & W \geq (2a + 1) \\ \dfrac{W}{2a + 1} & W < (2a + 1) \end{cases}$$

where a = Propagation time/Frame transmission time = tp/tf

- i.e. Sliding window protocol can achieve 100% utilization for W ≥ (2a + 1).
- The smaller the W needed for this the better! (Why?). This requires a small a (so small a is still advantageous!)

Sliding Window Efficiency: Example

Shorter links are better (small a)

- Compare the efficiency of Sliding Window flow control for two links using the parameter 'a':
  - Fame size, L = 1000 characters of 8 bits each, = 8000 bits

- Satellite link between 2 ground stations
- d = 2 x 36,000 km, Data rate, R = 1 Mbps
- Typical wave velocity, V = 3 x 10^8 m/s
- Frame TX time, t_f = L/R = 8 ms
- Propagation time, t_p = d/V = 240 ms
- a = t_p / t_f = 30
- 100 % link utilization is achieved with window size W:

  W ≥ (2 a+1) ≥ (2 x 30 +1) ≥ 61

  W = 61, k = 6 bit Large window, large buffers at TX, RX)
- For k = 3 bits, W = 7:

  Utilization U = W/(2a+1) = 7/(61) = 11.5% > 1.6% for Stop and wait.

- 200-m optical fiber link
- Data rate, R = 1 Gbps
- Typical wave velocity, V = 2 x 10^8 m/s
- Frame TX time, t_f = L/R = 8 μs
- Propagation time, t_p = d/V = 1 μs
- a = t_p / t_f = 0.125
- 100 % link utilization is achieved with window size W:

  W ≥ (2 a+1) ≥ (2 x 0.125 +1) ≥ 1.25

  i.e. W = 2 (A window of just 2 frames!)

  - easily achieved in practice!

High-Level Data Link Control Protocol (HDLC)

- To satisfy various applications, HDLC defines:
  - Three types of communicating stations
  - Two types of link configurations, and
  - Three data transfer modes of operation.

High-Level Data Link Control Protocol (HDLC)

Station types:

- Primary Station (PS): (e.g. computer)
  - Responsible for controlling link operation
  - Control frames issued by the PS are called *commands*
- Secondary Station (SS): (e.g. terminal, sensor)
  - Operates under the control of a primary station
  - Control frames issued by the SS are called *responses*
- Combined Station (CS):
  - Issues both commands and responses

High-Level Data Link Control Protocol (HDLC)

Link configurations:

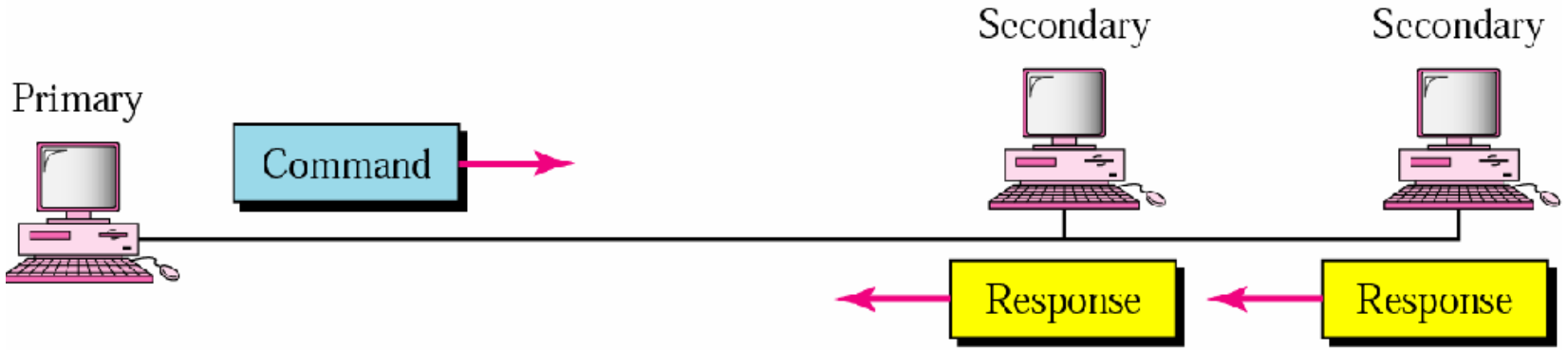
Determined by the types of stations on the link

- Unbalanced (different status): e.g. One primary station plus one or more secondary stations
- Balanced (same status): e.g. Two combined stations

High-Level Data Link Control Protocol (HDLC)

Data Transfer modes: (Who can send?, what? and when?)

- For unbalanced link configuration: (Response Modes)
  - Normal Response Mode (NRM)
    Secondary may send data only in response to a command from primary
    e.g. Computer (PS) connected to a number of terminals (SS) over a multi drop line. Computer polls terminals for data



  - Asynchronous Response Mode (ARM) Secondary may send data without explicit permission from primary. Primary still retains link control (initialization, error recovery, logical disconnection, …)
    (Rarely used)

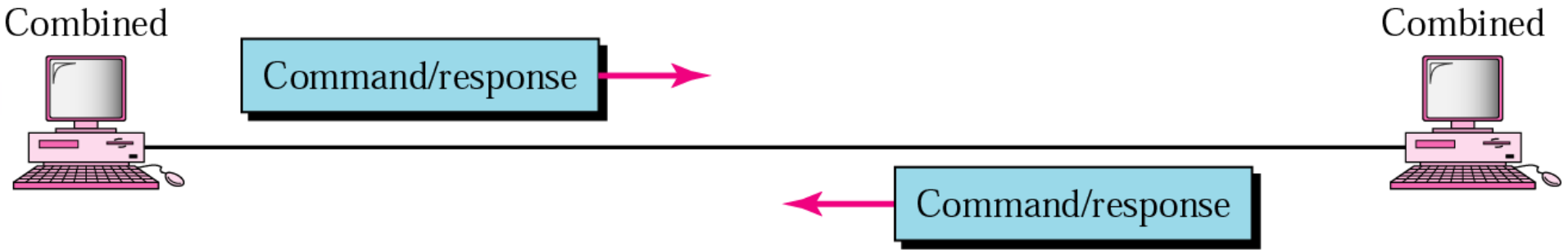
High-Level Data Link Control Protocol (HDLC)

Data Transfer modes: (Who can send?, what? and when?)

- For balanced link configuration
  - Asynchronous Balanced Mode (ABM) Either combined stations may send data without obtaining permission from the other station

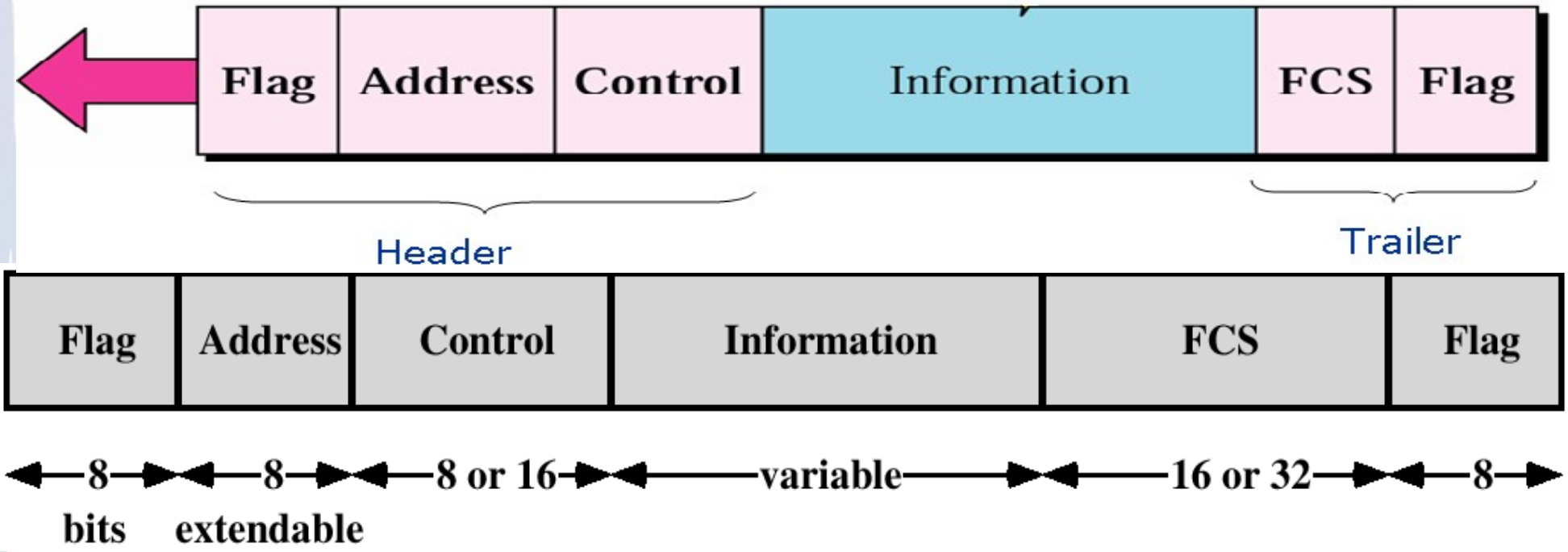
    Most widely used (no polling involved)

    e.g. full duplex point-to-point



HDLC – Frame Structure

- HDLC Uses Synchronous Transmission, i.e. large frames
- Frame consists of the following 'fields':
  - Flag fields at start and end: for frame-level sync
  - Address field: for addressing in multi-point links
  - Control field: for Flow and Error control
  - Information field: (payload data or link management data)
  - FCS field: for error detection

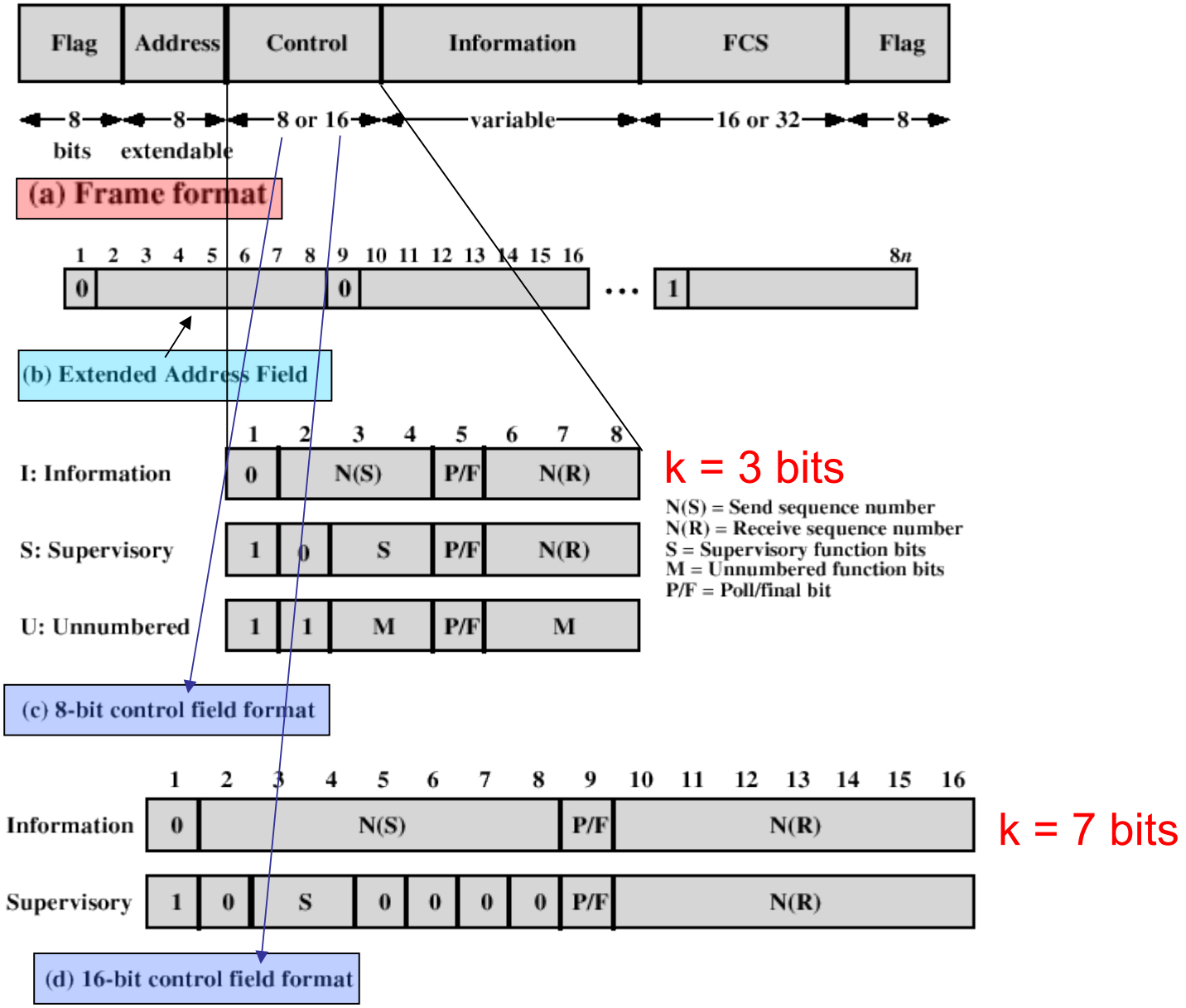


HDLC – Frame Structure

Frame

Address Field

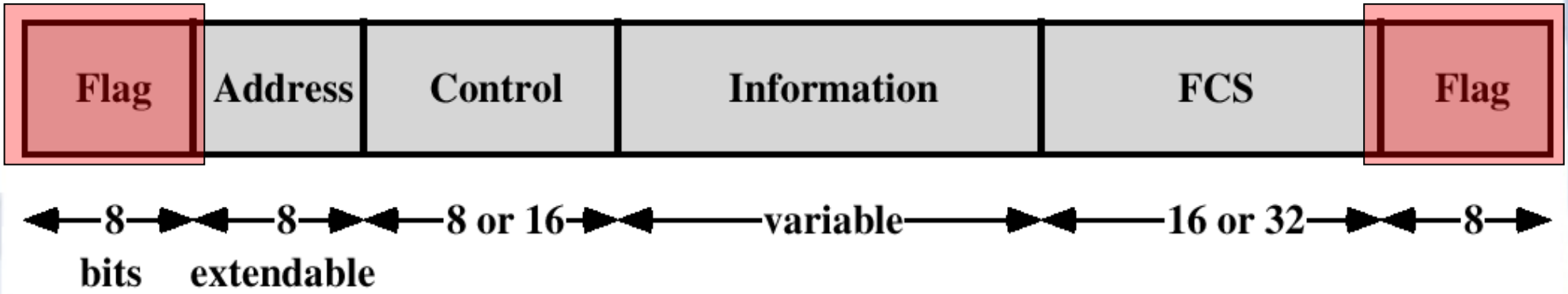
Control Field



HDLC Frame Format

- Flag:
  - Size: 1 Byte
  - Special pattern 0 1 1 1 1 1 1 0 used as frame begin/end and synch.
  - Used for Header and trailer
- Address:
  - Size: 1 Byte (or extendible to more bytes for larger networks)
  - If primary station created the frame, the address is that of the destination secondary station
  - If secondary station created the frame, the address is that of the source secondary station
  - Networks not using “primary/secondary” (e.g. Ethernet) use 2-Byte address (source/destination)
- Control:
  - Size: 1 or 2 Bytes
  - Identifies frame type (I, S, U)
  - Used for error & flow control functions
- Information: Payload of user data (I Frames) **or** link management data (U)
  - Size: Varies (as multiple bytes) from network to network. Always fixed within a network
  - I frames contains user data received from the higher layer (Network layer)
- FCS:
  - Size: 2 or 4 Bytes (depending on the divisor P used)
  - Implements CRC error detection

HDLC – Frame Structure: Flag field



- Flag Field: unique pattern 01111110 at both start and end of frame
  - Used for frame-level synchronization
  - This pattern should not exist in any other part of the frame
  - Two ways to ensure this:
    - Ensure that higher layers avoid using these pattern in the data they generate (causes lack of data transparency)
    - TX uses *bit stuffing* for data (inserts a 0 after each consecutive five 1s) to ensure data transparency at higher layers

HDLC Bit Stuffing:

- At TX:

  Inserts a 0 after each consecutive five 1s of non-flag data

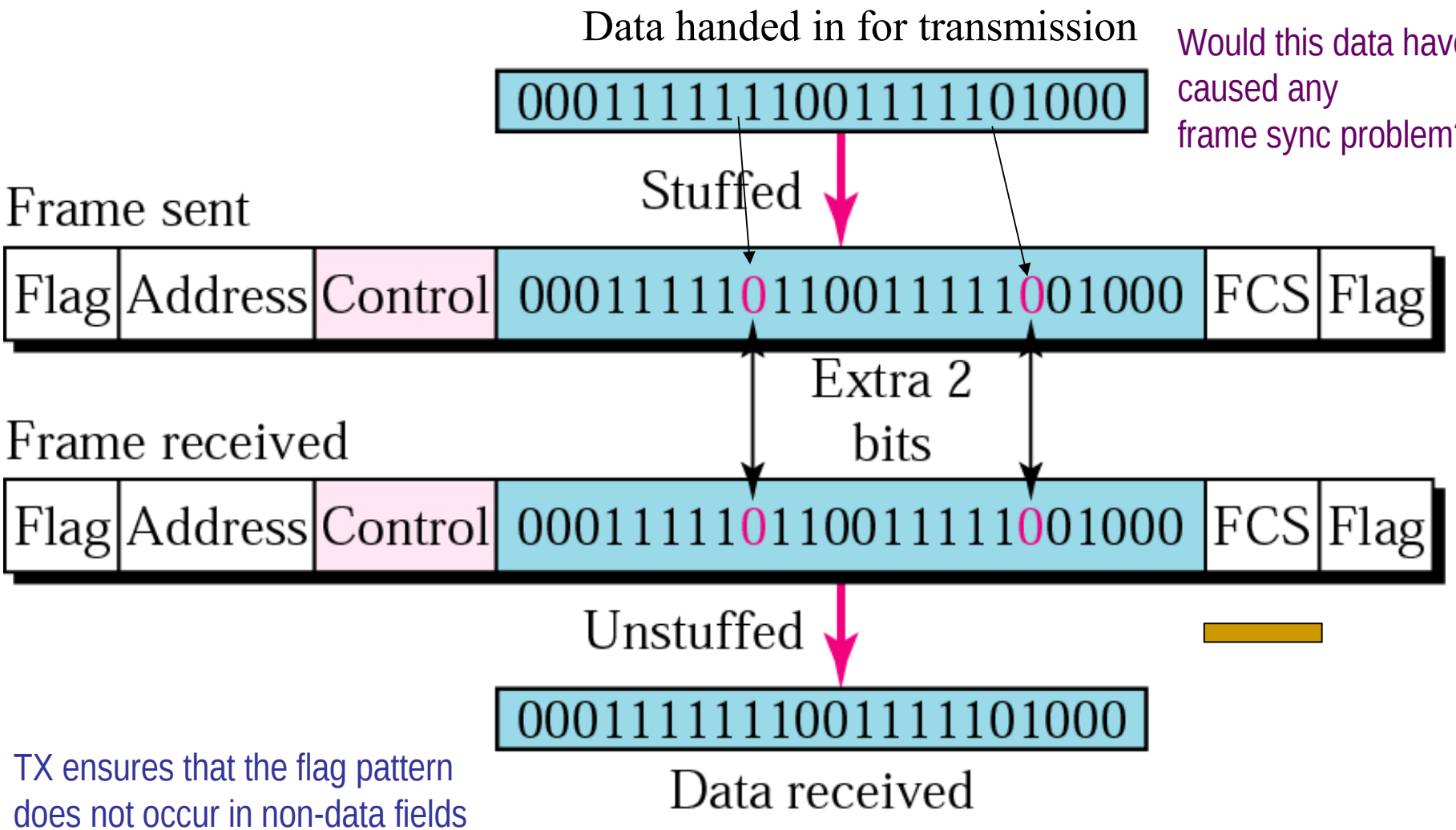
  (e.g. 01111100….or 01111101)

- At RX:

  After detecting the preamble flag, RX monitors incoming bits – when a pattern of five 1s appears; the 6^{th} and the 7^{th} bit are checked:

  - If 00 or 01 $\Rightarrow$ Bit 6 is a stuffed 0 $\Rightarrow$ Remove it
  - If 10 $\Rightarrow$ This is the postamble flag $\Rightarrow$ end of frame
  - If 11 $\Rightarrow$ ABORT (Error- there must have a 0 after every five 1's)

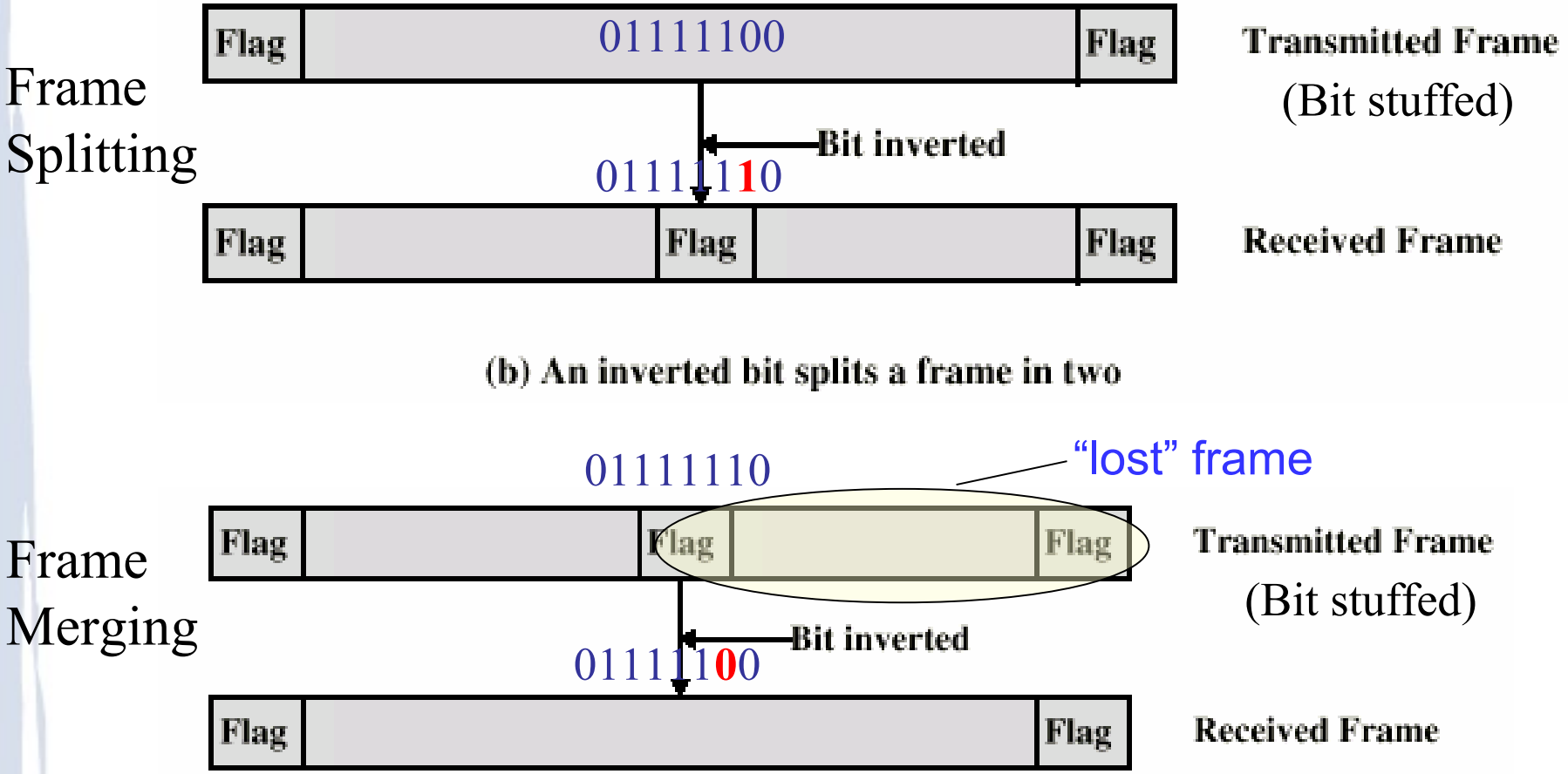
HDLC Bit Stuffing & Removal



TX ensures that the flag pattern does not occur in non-data fields

HDLC Bit Stuffing

- Problems: Single-bit errors could split a frame into 2 or merge two frames into 1.



HDLC Frame Types

- HDLC defines three types of frames
- Frame type determined by first bits in the control field
  - **User Information frames** (**I**-frames): 8 or 16 bit control field
    - Used to transport user data
    - In a duplex system, they can also carry control messages regarding previously received data to be carried along with data TXed (piggybacking)
  - **Supervisory frames** (**S**-frames): 8 or 16 bit control field
    - Used to transport control information for the data transfer (ACKs) only- Short frame, **Carries No Data**
  - **Unnumbered frames** (**U**-frames): 8 bit control field
    - Convey link management functions
    - Does **not** contain a frame sequence number (N or S) (unnumbered)
    - Can carry Management data

HDLC I, S & U Frames Format

Arrow indicates that the first transmitted field is the Header flag, then the address, then the control…