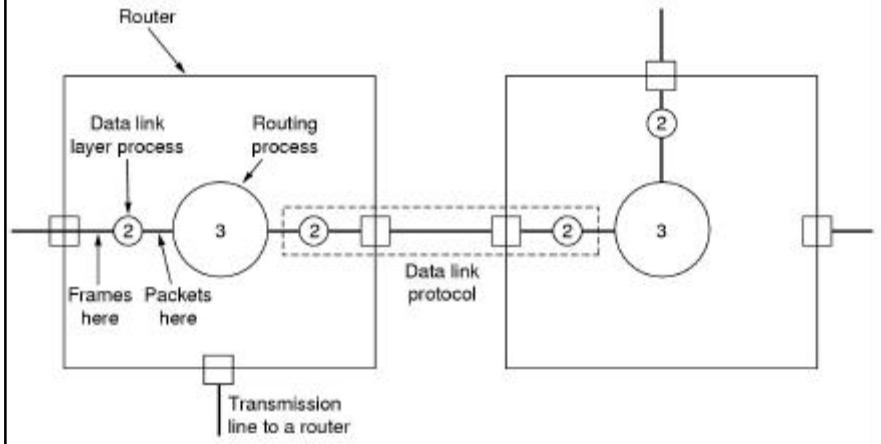
Computer Networks

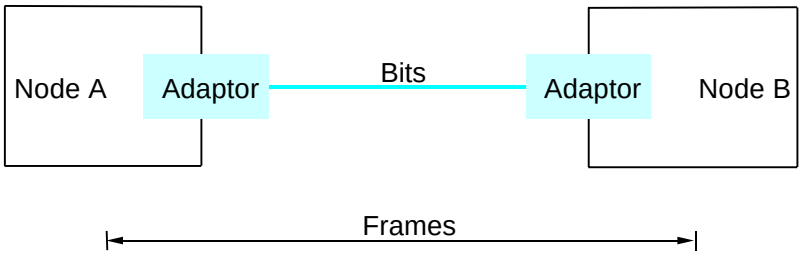
The Data Link Layer

Data Link Layer

- Framing
  - Byte oriented protocols (e.g. IBM BISYNC, ARPANET IMP-IMP, DEC DDCMP)
  - Bit oriented protocols (e.g.IBM SDLC, ISO HDLC, CCITT X.25 LAPB)
  - Clock-based (TDM, SONET)
- Error Detection & Correction: Parity checking, CRC, Hamming codes
- Flow Control & Reliable Transmission
  - Stop-and-wait
  - Sliding Window
    - go-back-n
    - selective repeat

The Data Link Layer

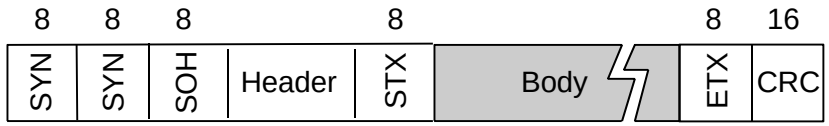


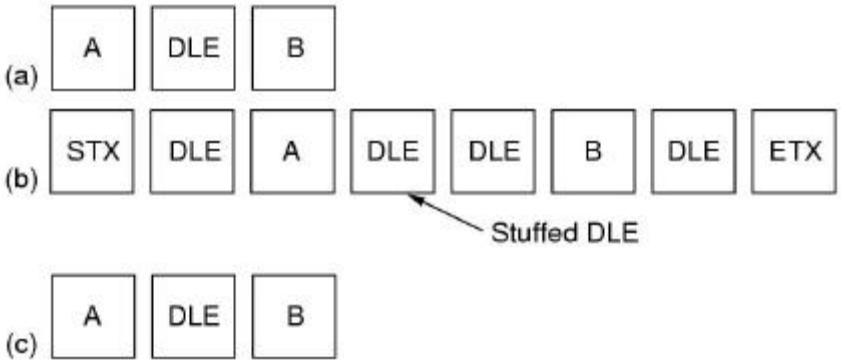


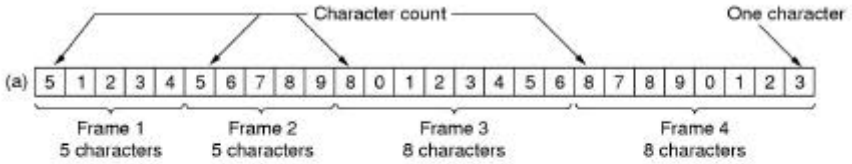
Byte Oriented Protocols

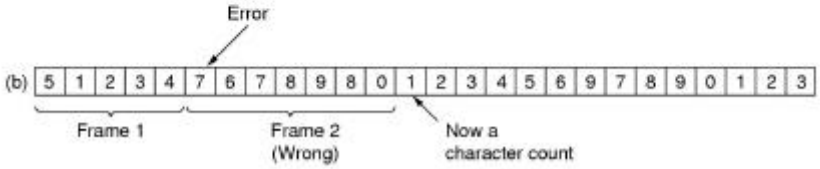
- Sentinel Approach (IBM's BISYNC, ARPANET's IMP-IMP):
  - Enclose packet in special sentinel characters:
    - SOH-STX-ETX for BISYNC: need to escape ETX, variable header
    - DLE/STX-DLE/ETX for IMP-IMP: escape DLE
  - Problem with corruption of sentinel characters
- Byte Counting (DECNET's DDCMP):
  - include number of bytes (COUNT) in header
  - Problem with corruption of COUNT field
- Disadvantage of Byte oriented Protocols: tightly tied to character codes

Byte Oriented Protocols



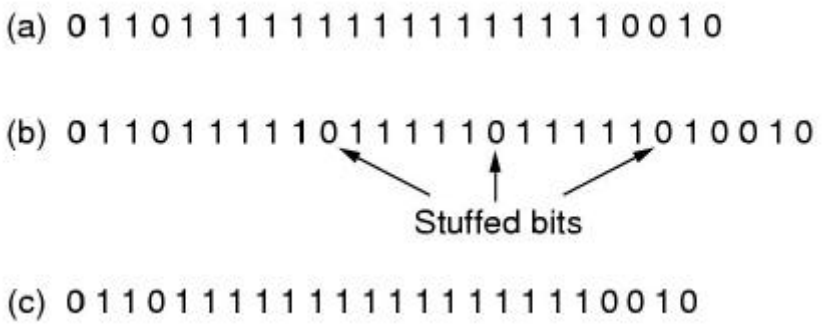




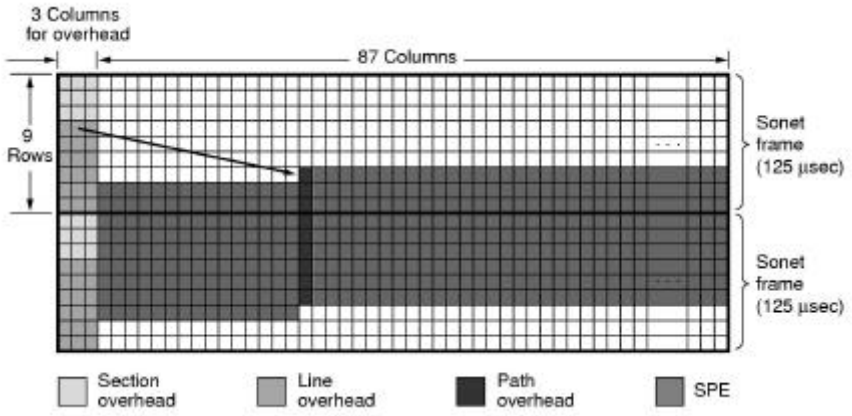


Bit Oriented Protocols

- Variable length bit streams
- Sender:
  - encloses packet (bit stream) : 01111110
  - appends a 0 after each 11111 in body (*bit staffing*).
- Receiver, upon receiving 011111:
  - next bit 0: stuffed bit is removed
  - next bit 1:
    - if next bit 0 (i.e. 01111110): end of frame marker
    - if next bit 1 (i.e. 01111111): error



Clock-based Protocols (e.g. SONET)



Parity Checking

- One additional bit to make number of 1's even (or odd)
  - Detection of odd (even) bit errors: even (odd) number of errors are not detected (50% of burst errors)
- Two additional bits for even and odd numbered bits
  - Detection of single or even/odd pair bits
- Burst error detection
  - Basic idea: Distribute the bits of a message in different frames
  - Guaranteed to detect errors due to bursts whose duration is less than the time to send one frame
  - Detects longer burst errors with probability $1- (1/2)^n$

Parity Checking

Figure 4.1 Detecting Single Bit Errors Using Parity Checking

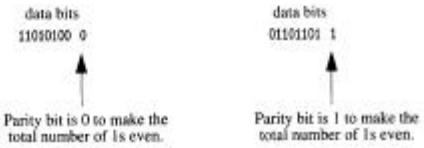


Figure 4.2 Detecting Consecutive Double-Bit Errors Using Parity Checking

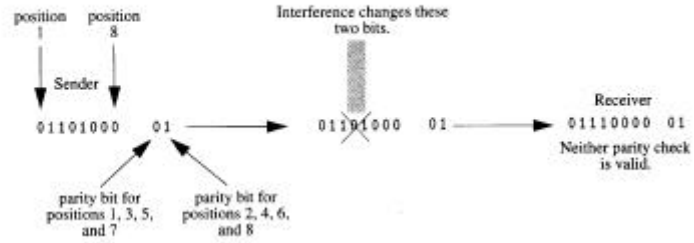


Figure 4.3 Detecting Burst Errors Using Parity Bits

Sender: Row (frame) number		Parity bit for one row
1	01101	1
2	10001	0
3	01110	1
4	11001	1
5	01010	0
6	10111	0
7	01100	0
8	00111	1
9	10011	1
10	11000	0
Column number	12345	6

Burst error occurs and destroys column four, making it all zeroes.

Receiver: Row number		Parity bit for one row
1	01101	1
2	10001	0
3	01100	1*
4	11001	1
5	01000	0*
6	10101	0*
7	01100	0
8	00101	1*
9	10001	1*
10	11000	0
Column number	12345	6

* Parity bit is not correct

Cyclic Redundancy Checks

- Based on polynomial division
  - $b_n\, b_{n-1}\, b_{n-2} \ldots b_1\, b_0 \rightarrow b_n\, x^n + b_{n-1}\, x^{n-1} + b_{n-2}\, x^{n-2} + \ldots b_1\, x^1 + b_0$
- Steps:
  - Given a bit string B, append a number(= degree of G(x)) of 0s to the end of it: B(x)
  - Divide B(x) by a agreed-on *generator* polynomial G(x) and get the remainder R(x)
  - Define T(x)=B(x)-R(x). (note: T(x)/G(x) generates a 0 remainder)
  - Transmit T->T(x) which will arrive as T'(x).
    - If T'(x)/G(x) generates a 0 remainder OK. Otherwise error!

CRC Analysis (I)

- On what conditions T'(x)/G(x) will yield a 0 remainder?
  - Changing T(x) is like adding to it an unknown E(x): T'(x)=T(x)+E(x)=>T'(x)/G(x)=[T(x)+E(x)]/G(x)= T(x)/G(x)+E(x)/G(x):
  - Undetected transmission errors correspond to errors for which G(x) is a factor of E(x)
    - If G(x) has at least 2 terms all single-bit errors are detected
    - If x is not a factor of G(x) then all burst errors having length <=to the degree of G(x) are detected.
    - If x+1 is a factor of G(x), then all burst errors damaging an odd number of bits are detected
    - ...

CRC Analysis (II)

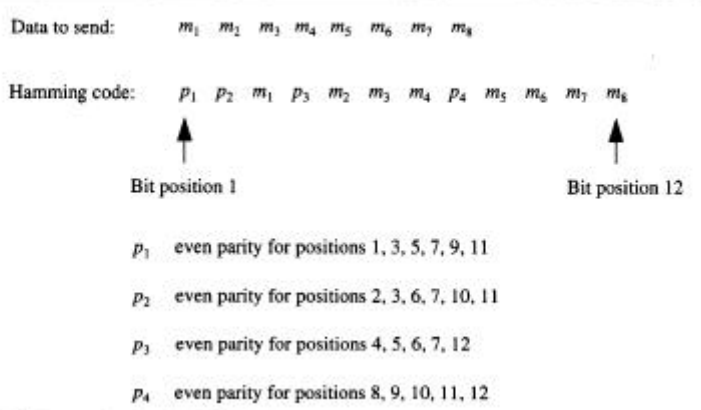
- If x is not a factor but x=1 is a factor then CRC detects:
  - all burst errors of length r < degree of G(x)
  - all burst errors affecting an odd number of bits
  - all burst errors of length r+1, with probability $(2^{r-1}-1)/2^{r-1}$
  - All burst errors of length > r+1 with probability $(2^{r}-1)/2^{r}$
- Typical G(x) polynomials:
  - CRC-12: $x^{12}+ x^{11}+ x^3+ x^2+ x+1$
  - CRC-16: $x^{16}+x^{15}+ x^2+1$
  - CRC-CCITT: $x^{16}+ x^{12}+ x^5+1$
  - CRC-32: $x^{32}+ x^{26}+ x^{23}+ x^{22}+ x^{16}+ x^{12}+ x^{11}+ x^{10}+ x^8+x^7+ x^5+ x^4+ x^2+ x +1$

Hamming Codes

- Based on the concept of Hamming distance: the number of bit positions in which 2 bit strings differ
- Error correcting code
- Uses *r* parity bits to detect and correct singe bit errors: for m data bits, $m+r+1<=2^r$
  - parity bits are placed in positions *k*: $k=2^i$, i=0,1,2...
  - (data/parity) bit in position k contributes to the parity bits in positions i where: $k=\sum_x i$, $i=2^x$
  - Inverted bit is in position *i*, where $i=\sum k$, *k=position of parity bit with wrong value*
- Burst errors as in simple parity

Hamming Codes

FIGURE 4.9 **Hamming Code for Single-Bit Error Correction**



Hamming Codes

FIGURE 4.10 Bit Stream Before Transmission

Data:	0	1	1	0	0	1	1	1
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8

Hamming code:	0	1	0	1	1	1	0	1	0	1	1	1
	p_1	p_2	m_1	p_3	m_2	m_3	m_4	p_4	m_5	m_6	m_7	m_8

FIGURE 4.11 Parity Checks of Frame After Transmission

	Position Number
• Indicates bits checked using p_1: parity check fails.	1
▲ Indicates bits checked using p_2: parity check succeeds.	0
⌐ Indicates bits checked using p_3: parity check fails.	1
⌙ Indicates bits checked using p_4: parity check succeeds.	0

Error is in position 0101 = 5

TABLE 4.2 Bit Position Errors and Associated Parity Errors

Erroneous bit position	Invalid parity checks	$p_4p_3p_2p_1$
no error	none	0000
1	p_1	0001
2	p_2	0010
3	p_1 and p_2	0011
4	p_3	0100
5	p_1 and p_3	0101
6	p_2 and p_3	0110
7	p_1, p_2 and p_3	0111
8	p_4	1000
9	p_1 and p_4	1001
10	p_2 and p_4	1010
11	p_1, p_2 and p_4	1011
12	p_3 and p_4	1100

Hamming Codes

Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

Flow Control

FIGURE 5.1 **Flow Control Using Signaling**

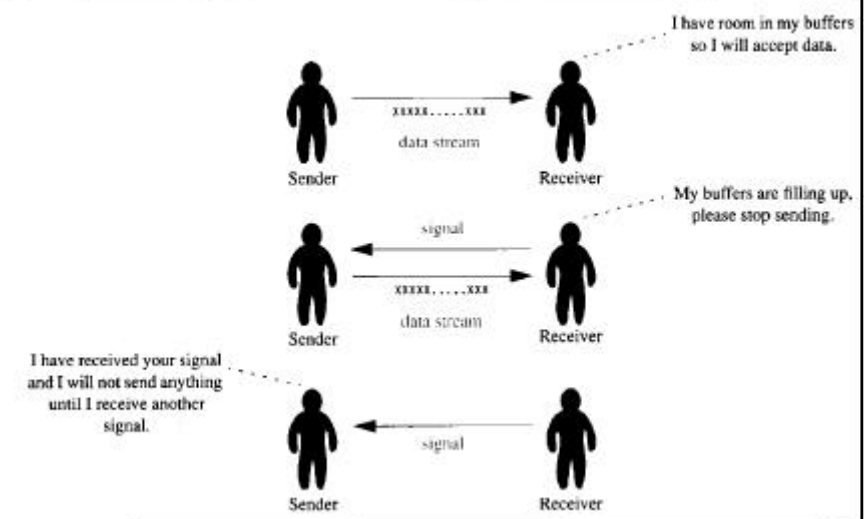
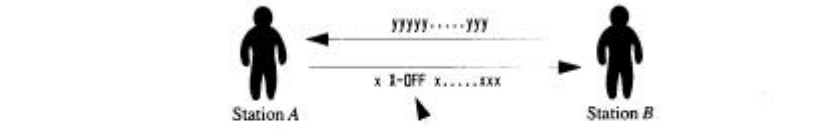


FIGURE 5.2 **Flow Control Using Inband Signaling**

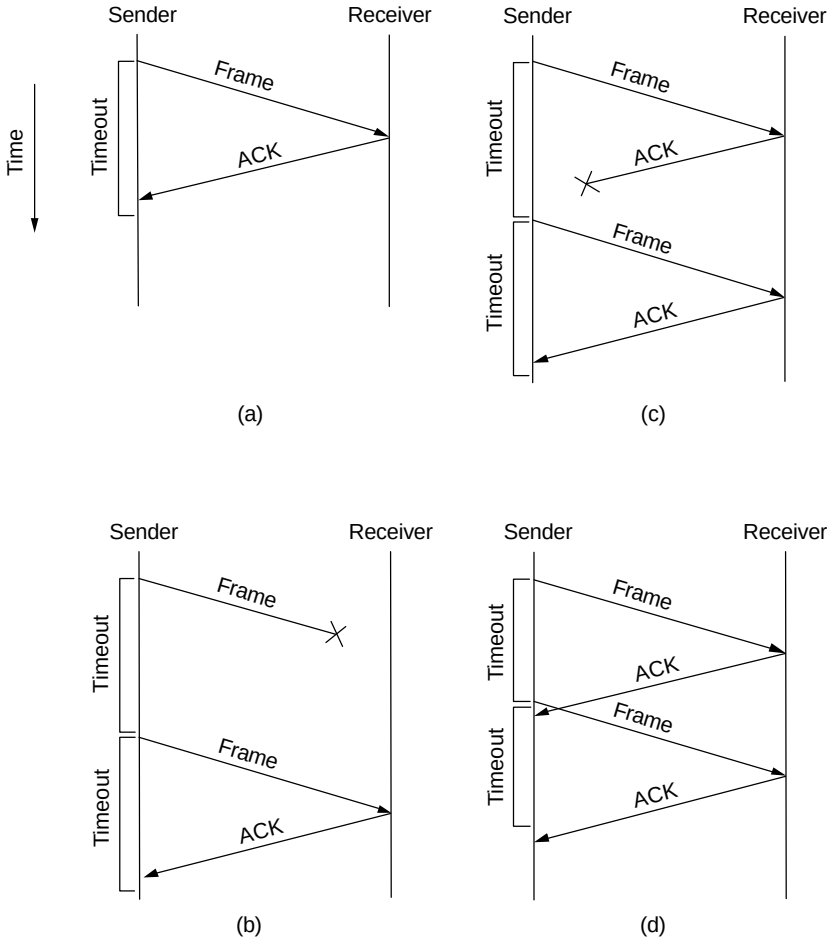


X-OFF character (ASCII code DC3). When *B* gets it, *B* will stop transmitting.

Stop-and-Wait

- Receiver acknowledges each frame
- Sender waits for ack before it sends the next frame
- Disadvantages: Poor efficiency
  - delay * bandwidth: (delay = RTT or RTT/2)
  - channel utilization:
    - the percentage of time the channel is transmitting data frames
  - effective data rate:
    - The actual number of <u>data</u> bits sent per unit of time.
- *Exercise: Derive formulas for the above 2 metricsfor the stop-and-wait protocol*

Stop-and-Wait



Sliding Window: go-back-n

- Sender assigns consecutive (modulo 2^k) sequence numbers to each frame: 0 to 2^k -1
- **Sender maintains window** buffer of size 2^k -2 (why?).
  - Everytime a frame is sent, it is buffered.
  - Frames are removed as they are acknowledged
  - If frame not acknowledged it is retransmitted **along with all outstanding frames** (go-back-n)
- Receiver expects to receive frames in order (modulo 2^k):
  - If frame is damaged **or out of order** it is discarded
- Receiver does not acknowledge each received frme explicitly. Ack piggybacks data frames and **refers to the most recently received frame**. Ack timeouts if no data.

go-back-n

FIGURE 5.7 A Sliding Window Protocol

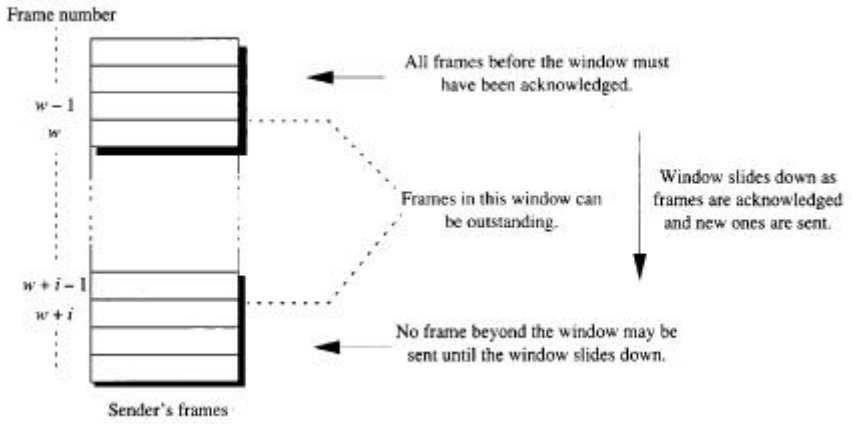


FIGURE 5.9 Typical Frame Format

go-back-n

FIGURE 5.10 Protocol Failure When Window Size Equals 2^K

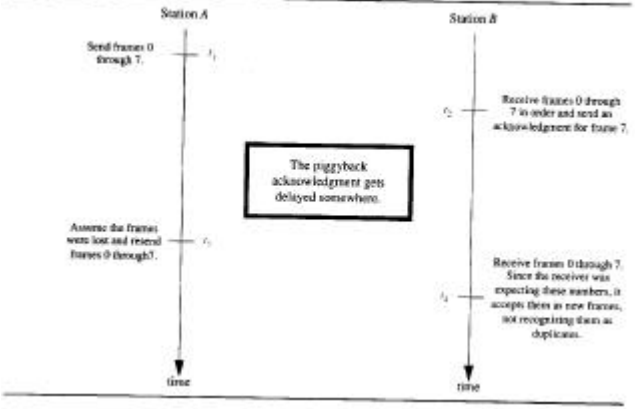
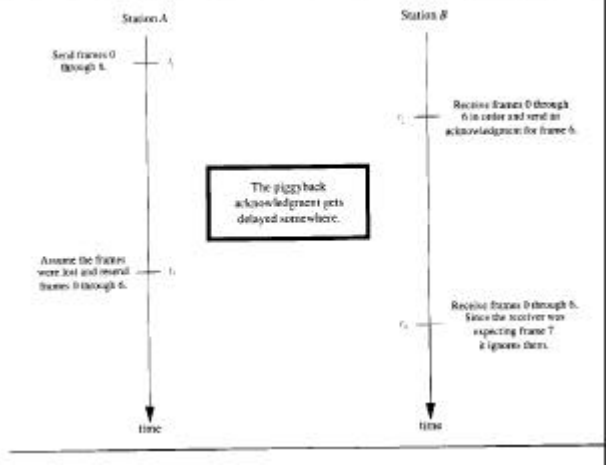


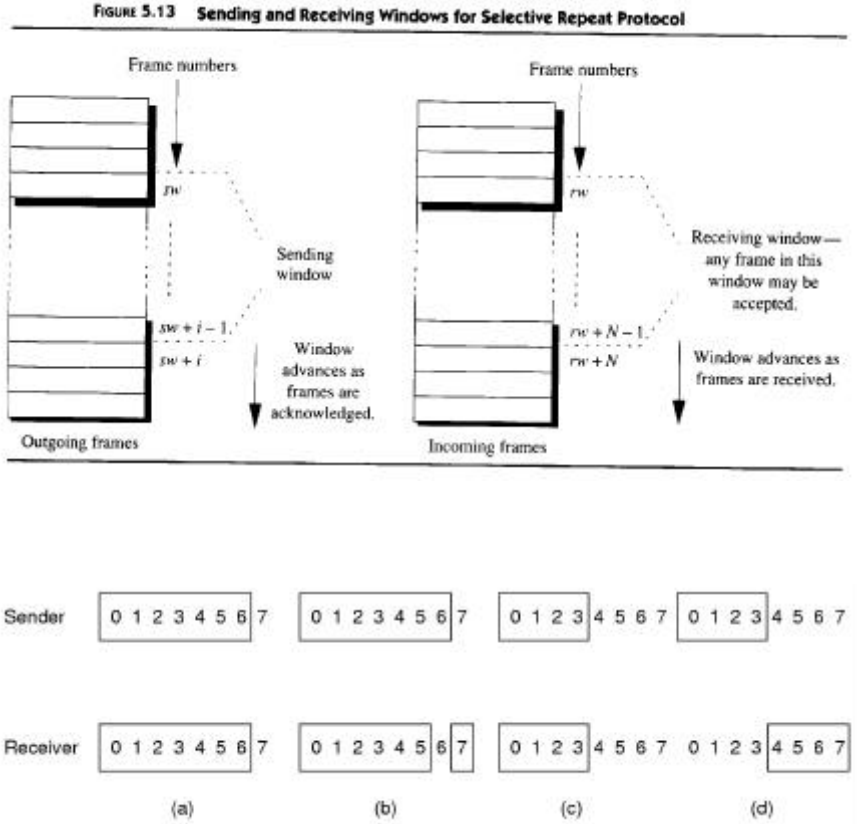
FIGURE 5.11 Protocol Success When Window Size Equals $2^K - 1$



Sliding Window: Selective Repeat

- Similar to go-back-n **but**:
  - Receiver
    - Utilises **window too**, which defines which frames can be received
    - Out of order frames that are in window are buffered even if they are out of order
    - Nack sent for damaged or delayed expected frame
    - Ack sent for frame f_i, where i is the largest sequence number such as all frames $f_j, j<i$ have been received
  - Sender
    - Retransmits only timed out and Nack-ed frames
- Window sizes w:
  - $w_{sender} + w_{receiver} <= 2^k$

Selective Repeat



Selective Repeat

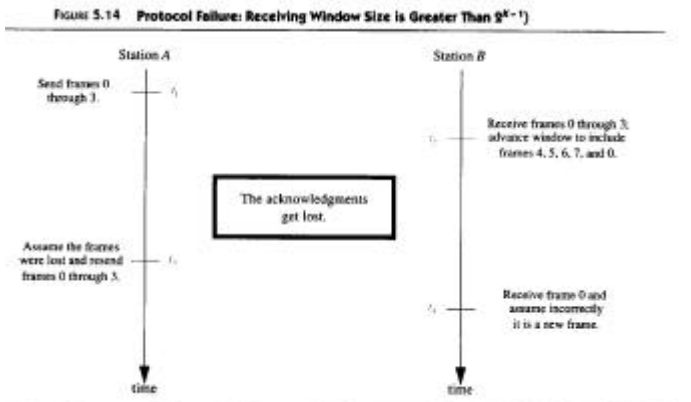
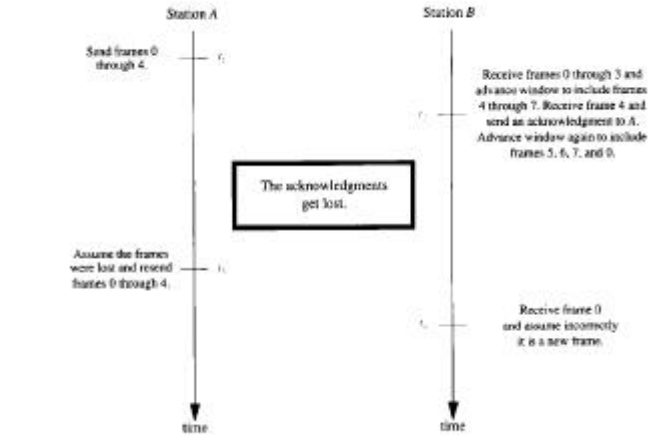
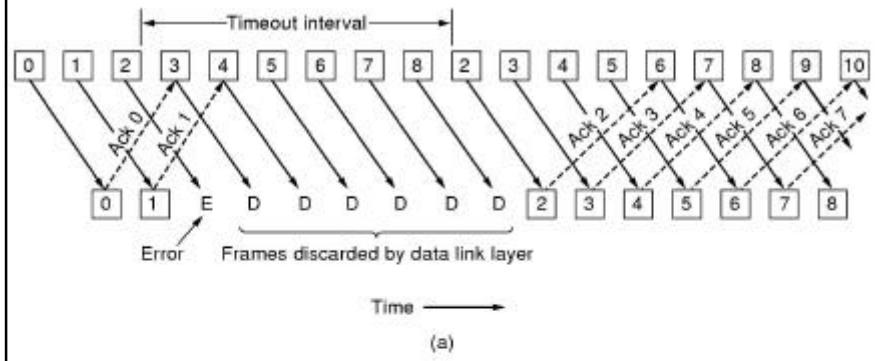


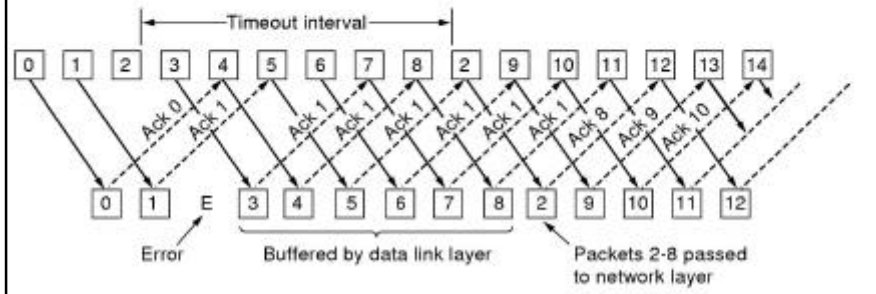
FIGURE 5.13 Protocol Failure: Sending Window Size is Greater Than 2^{k-1}



go-back-n vs Selective Repeat



(a)



(b)