Module 5

(DLC/LLC Sub Layer)

(Data Link Layer and Medium Access Sub Layer: Error Detection and Error Correction - Fundamentals, Block coding, Hamming Distance, CRC; Flow Control and Error control protocols - Stop and Wait, Go back – N ARQ, Selective Repeat ARQ, Sliding Window, Piggybacking, Random Access, Multiple access protocols -Pure ALOHA, Slotted ALOHA, CSMA/CD,CDMA/CA; Wired LAN, Wireless LANs, Connecting LANs and Virtual LANs)

Dr. Nirnay Ghosh

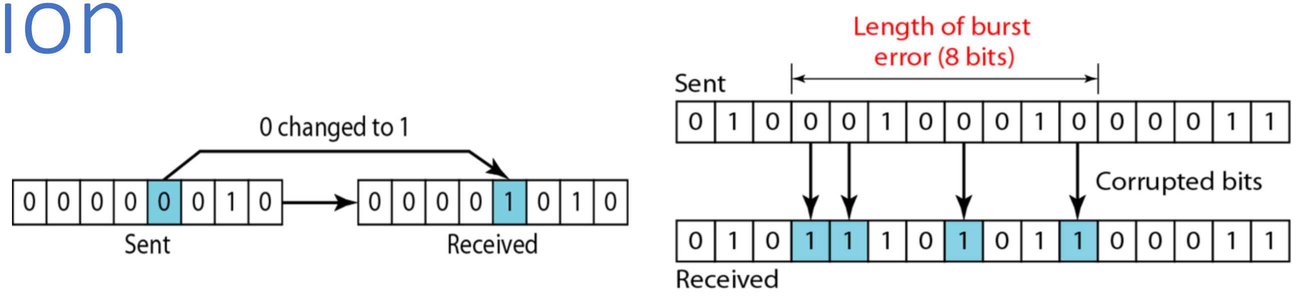
Assistant Professor

Department of Computer Science & Technology

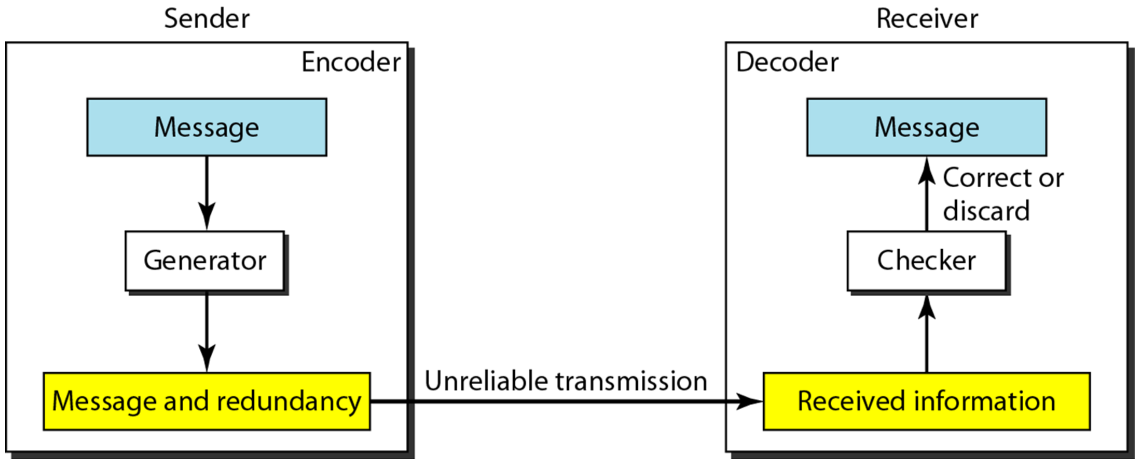
IIEST, Shibpur

Error Detection and Correction

- Interference occurs during transmission
  - Changes the shape of the signal
- Types of errors: single-bit error, burst error
- Redundancy: send some extra bits with data to detect or correct errors
  - Added by the sender and removed by the receiver.
  - Achieved through block coding
  - Creates a relationship between the redundant bits and the actual data bits
  - Verified by the receiver
- Error detection: less overhead than error correction



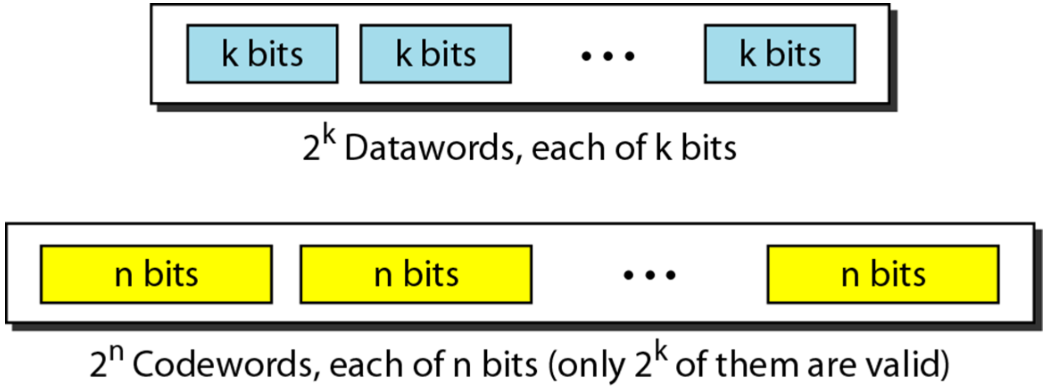
Types of Errors: (i) Single-bit Error (ii) Burst Error



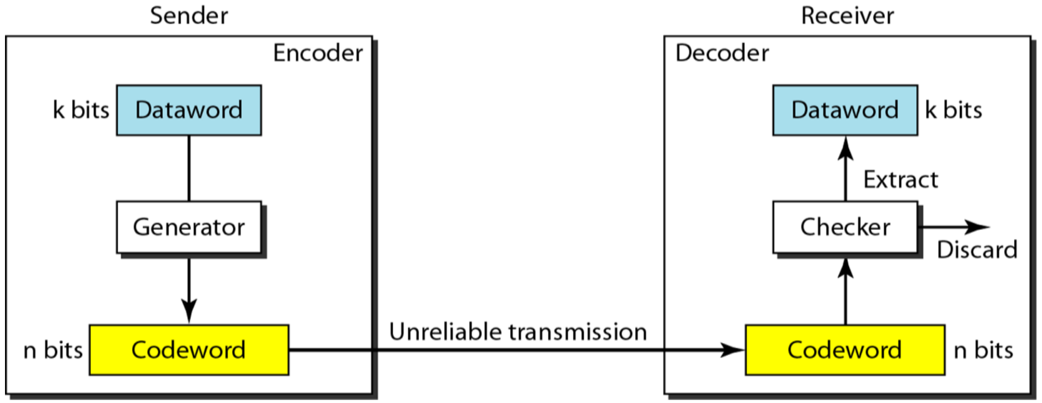
The Structure of Encoder and Decoder

Block Coding

- Datawords (*k*-bits each); codewords (*n*-bits each); $n > k$
- One-to-one: the same dataword is always encoded as the same codeword.
- Two conditions for error detection:
  - The receiver has (or can find) a list of valid codewords.
  - The original codeword has changed to an invalid one.



Datawords and Codewords in Block Coding



Process of Error Detection in Block Coding

A Code for Error Detection

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Block Coding (Contd...)

- Hamming distance
  - Number of differences between the corresponding bits
  - Apply XOR operation (⊕) on the two words
  - Count the number of 1s in the result.
- Minimum Hamming Distance for Error Detection
  - For detection of up to *s errors* in all cases, the minimum Hamming distance between all valid codewords must be $d_{min} = s + 1$
- Linear Block Code: XOR operation of two valid codewords results a valid codeword

$0 \oplus 0 = 0$ $\quad$ $1 \oplus 1 = 0$

a. Two bits are the same, the result is 0.

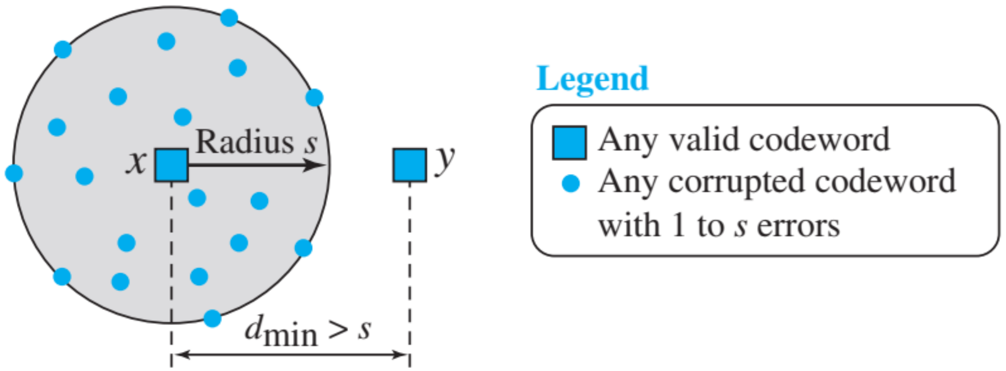
$0 \oplus 1 = 1$ $\quad$ $1 \oplus 0 = 1$

b. Two bits are different, the result is 1.

$$\begin{array}{rccccc} & 1 & 0 & 1 & 1 & 0 \\ \oplus & 1 & 1 & 1 & 0 & 0 \\ \hline & 0 & 1 & 0 & 1 & 0 \end{array}$$

c. Result of XORing two patterns

XORing of Two Single Bits or Two Words



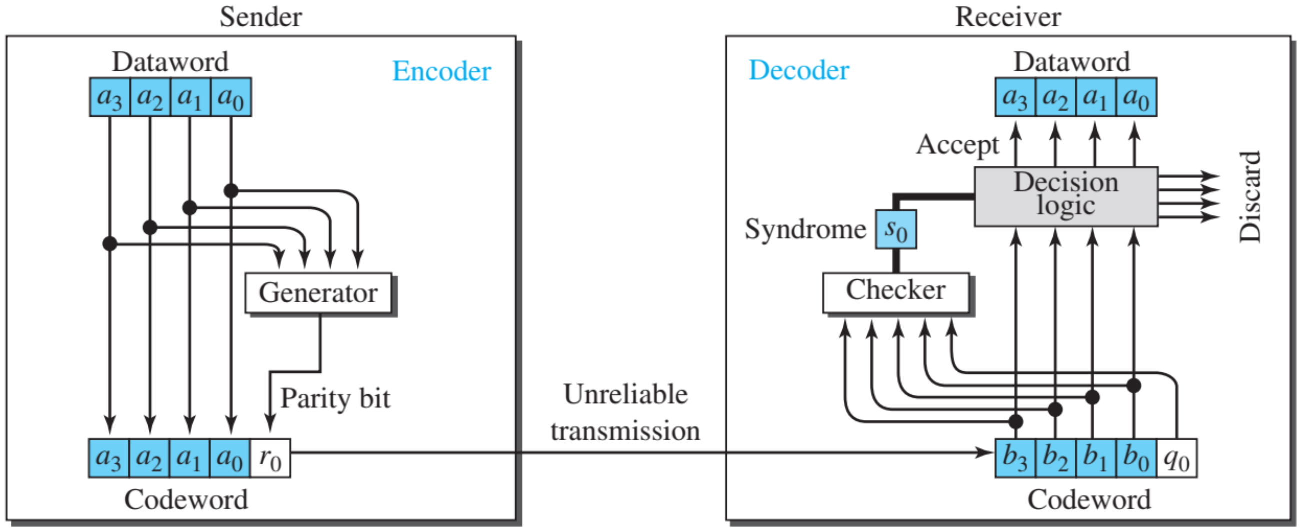
Geometric concept explaining d_{min} in error detection

Block Coding (Contd...)

- Parity-Check Code
  - Most familiar error detection code
  - Parity bit: extra bit selected to make the total number of 1s in the codeword even (even parity).
    - $n = k + 1$
  - The minimum Hamming distance for this category is $d_{min} = 2$
    - Single-bit error-detecting code
- Generator: generate 5-bit codeword by *modulo-2 operation*
- Checker: generates syndrome bit to detect error

Simple Parity Check Code C (5, 4)

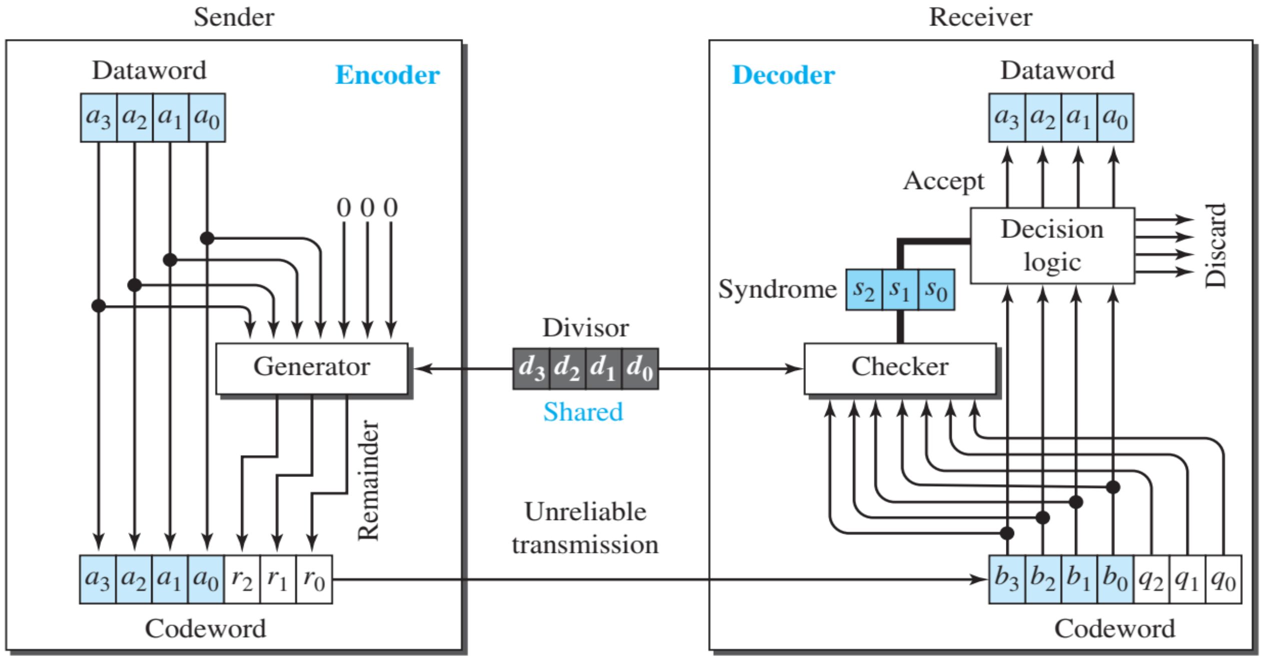
Dataword	*Codeword*	*Dataword*	*Codeword*
0000	**00000**	1000	**10001**
0001	**00011**	1001	**10010**
0010	**00101**	1010	**10100**
0011	**00110**	1011	**10111**
0100	**01001**	1100	**11000**
0101	**01010**	1101	**11011**
0110	**01100**	1110	**11101**
0111	**01111**	1111	**11110**



Encoder and Decoder for Simple Parity-Check Code

Cyclic Codes

- Special linear block codes
- Generates a new codeword by cyclic shift of the given codeword
- Cyclic Redundancy Code (CRC): used to correct errors in networks (LANs, WANs, etc.)
- Encoder: dataword (*k*-bits); codeword (*n*-bits)
  - Generator – performs modulo-2 binary division to generate check bits (remainder)
    - The divisor is predefined agreed upon
- Decoder:
  - Checker – performs modulo-2 binary division to generate syndromes
    - Syndrome = 000 → uncorrupted codeword; dataword accepted



CRC Encoder and Decoder

A CRC Code with C (7, 4) and divisor 1011

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

Cyclic Codes Analysis

Dataword: $d(x)$ **Codeword:** $c(x)$ **Generator:** $g(x)$ **Syndrome:** $s(x)$ **Error:** $e(x)$

In a cyclic code,

1. If $s(x) \neq 0$, one or more bits is corrupted.
2. If $s(x) = 0$, either
   a. No bit is corrupted, or
   b. Some bits are corrupted, but the decoder failed to detect them.

$$\textbf{Received codeword} = c(x) + e(x)$$

$$\frac{\textbf{Received codeword}}{g(x)} = \frac{c(x)}{g(x)} + \frac{e(x)}{g(x)}$$

- In a cyclic code, those e(x) errors that are divisible by g(x) are not caught.
- Single-bit Error: If the generator has more than one term and the coefficient of x_0 is 1, all single-bit errors can be caught.

- Two Isolated Single-Bit Errors

Difference: $j - i$

0	1	0	1	1	1	0	1	0	1	0	0	0	0	1	1
x^{n-1}			x^j							x^i					x^0

- Error $e(x) = x^i(x^{j-i} + 1)$
- If $g(x)$ has more than one term and one term is x^0, it cannot divide x^i
- To detect error: $g(x)$ must not divide $(x^{j-i} + 1)$
- If $g(x)$ cannot divide $x^t + 1$ (t between 0 and $n - 1$), then all isolated double errors can be detected.

Cyclic Codes Analysis (Contd…)

- Odd number of error
  - A generator that contains a factor of $x + 1$ can detect all odd-numbered errors.
- Burst Errors:
  - General form of error $e(x) = (x^j + . . . + x^i) = x^i(x^{j-i} + ...+ 1)$
  - To detect a single error (minimum condition for a generator), the generator cannot divide x^i.
  - For term $(x^{j-i} + ...+ 1)$ three cases:

- All burst errors with $L \leq r$ will be detected.
- All burst errors with $L = r + 1$ will be detected with probability $1 - (1/2)^{r-1}$.
- All burst errors with $L > r + 1$ will be detected with probability $1 - (1/2)^{r}$.

- L: length of the error
  - $L-1 = j - i$

- Criteria for good polynomial generator

A good polynomial generator needs to have the following characteristics:

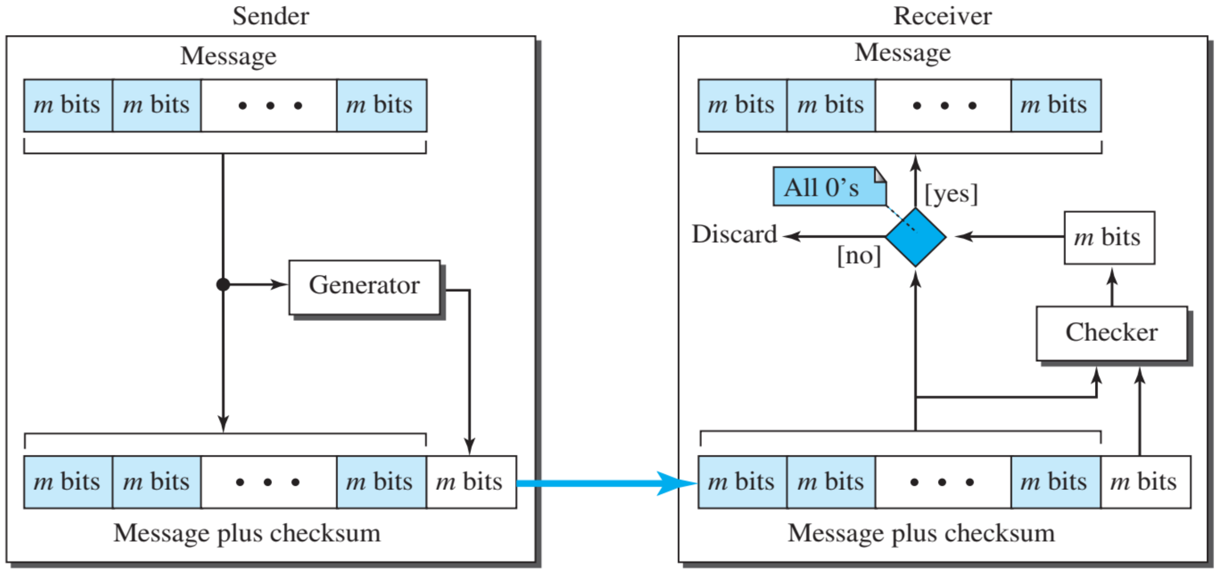
1. It should have at least two terms.
2. The coefficient of the term x^0 should be 1.
3. It should not divide $x^t + 1$, for t between 2 and $n - 1$.
4. It should have the factor $x + 1$.

Standard Polynomials used in Different Networks

Name	*Polynomial*	*Used in*
CRC-8	$x^8 + x^2 + x + 1$ 100000111	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$ 11000110101	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$ 10001000000100001	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ 100000100110000010001110110110111	LANs

Checksum

- Error detection code – applied to messages of arbitrary length
- Used in the network and transport layers
- Checksum unit can be attached anywhere in the message

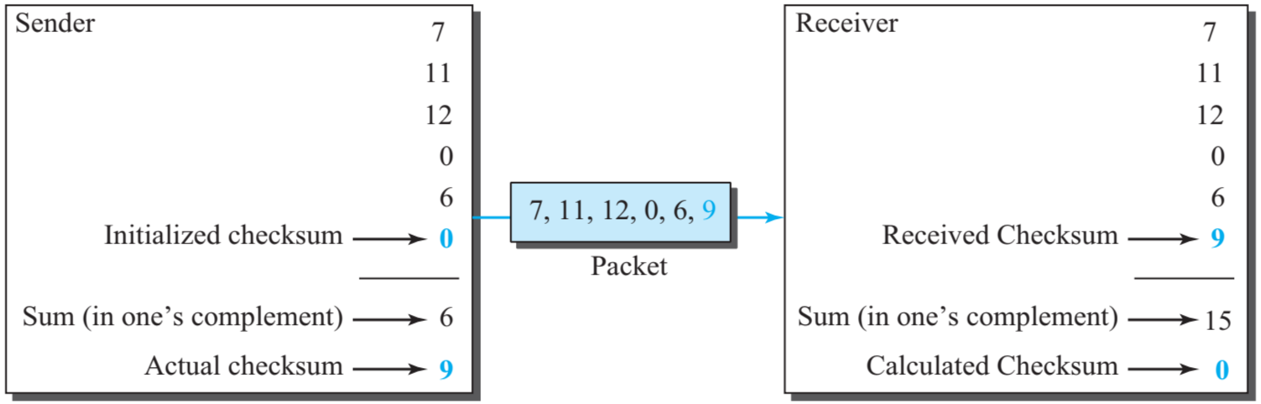


- Underlying mechanism for checksum calculation
  - One's complement addition
  - Complement
- Traditionally the Internet has used 16-bit checksum

Procedure to Calculate Traditional Checksum

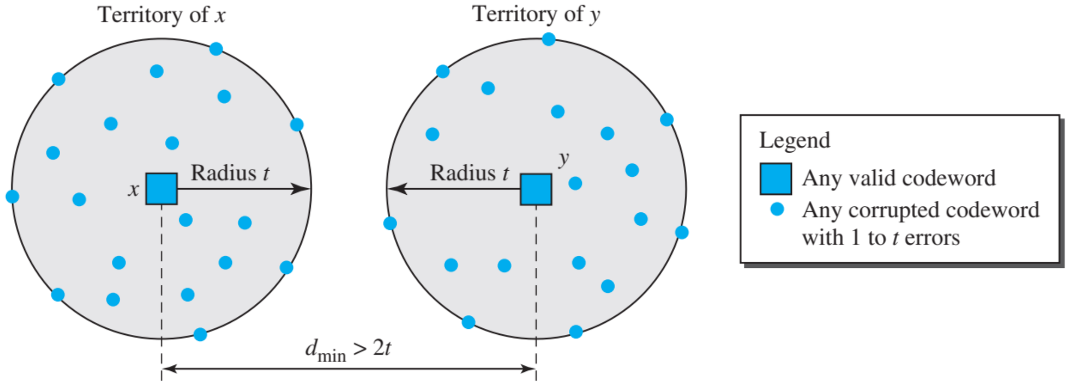
Sender	*Receiver*
1. The message is divided into 16-bit words.	1. The message and the checksum are received.
2. The value of the checksum word is initially set to zero.	2. The message is divided into 16-bit words.
3. All words including the checksum are added using one's complement addition.	3. All words are added using one's complement addition.
4. The sum is complemented and becomes the checksum.	4. The sum is complemented and becomes the new checksum.
5. The checksum is sent with the data.	5. If the value of the checksum is 0, the message is accepted; otherwise, it is rejected.

Example to Show the Calculation of Checksum



Forward Error Correction

- Packets get lost or corrupted during transmission
- Regenerate/retransmit – high delay
- Forward Error Correction (FEC): error correction schemes at the receiver

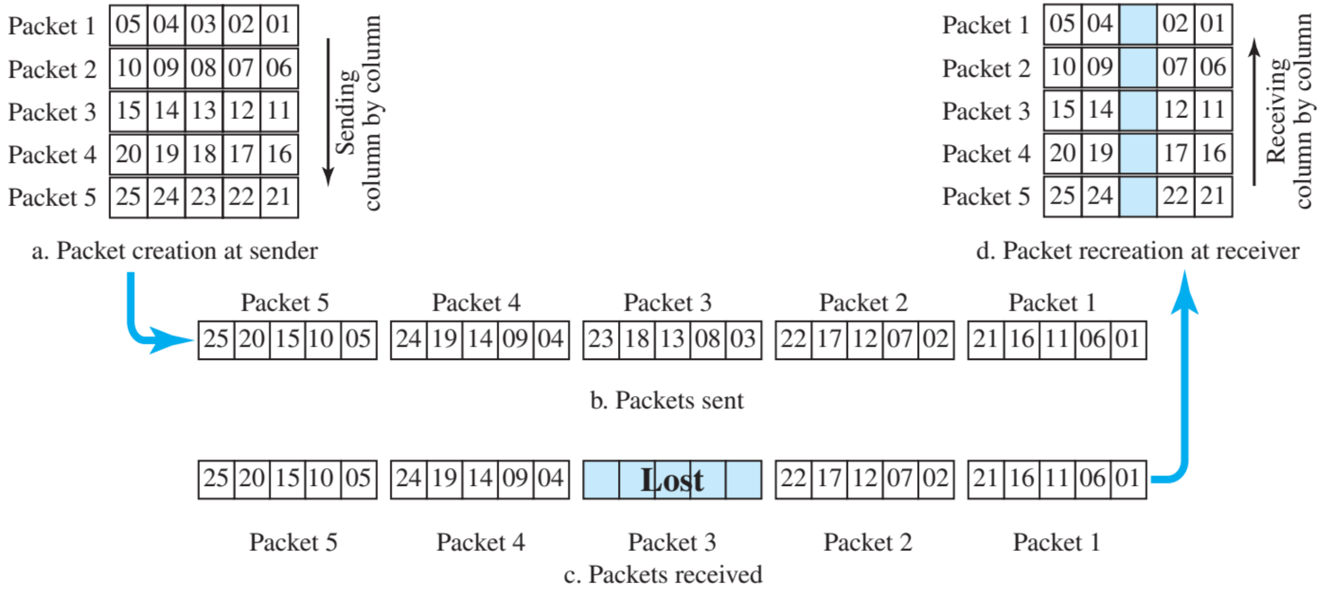


Humming Distance of Error Correction (to correct t errors, we need to have $d_{min} = 2t + 1$)

- Using XOR operation

$$\mathbf{R} = \mathbf{P}_1 \oplus \mathbf{P}_2 \oplus \ldots \oplus \mathbf{P}_i \oplus \ldots \oplus \mathbf{P}_N \quad \rightarrow \quad \mathbf{P}_i = \mathbf{P}_1 \oplus \mathbf{P}_2 \oplus \ldots \oplus \mathbf{R} \oplus \ldots \oplus \mathbf{P}_N$$

  - Divide a packet into *N* chunks
  - Create the exclusive OR of all the chunks and send *N + 1* chunks
  - If any chunk is lost or corrupted, it can be created at the receiver site.
  - Limitation: overhead

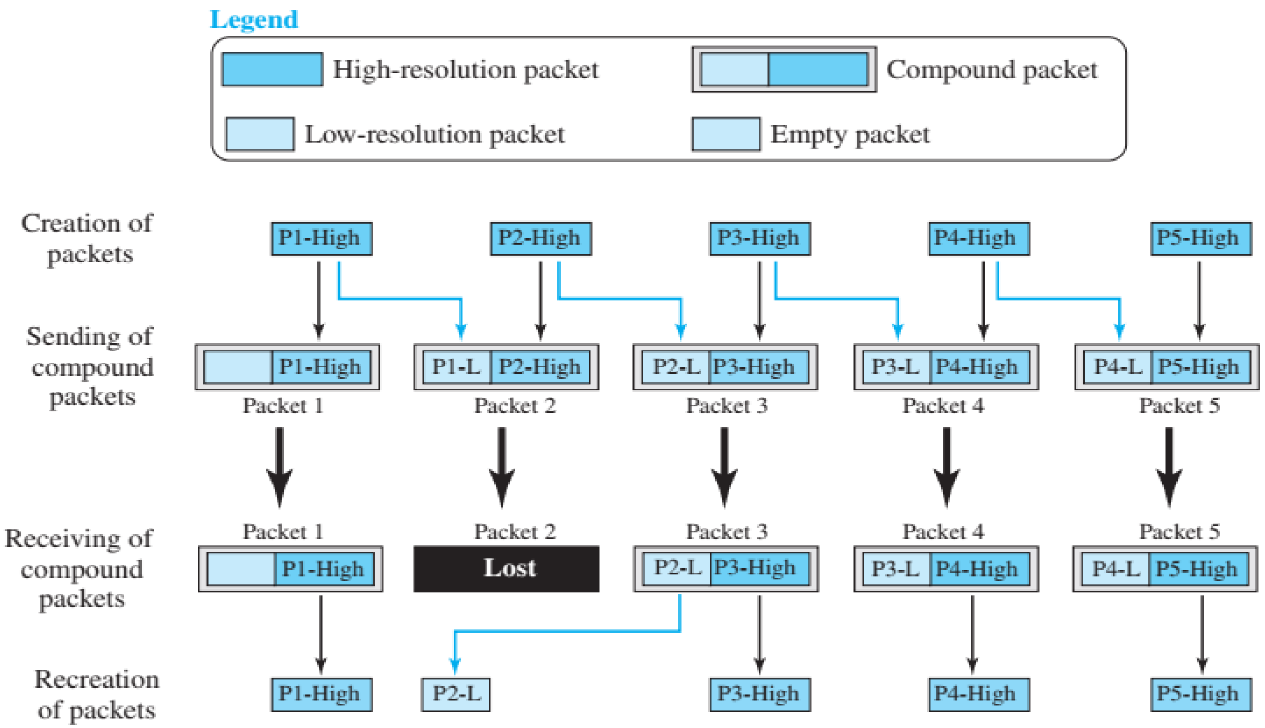


Chunk Interleaving (Used for multimedia data)

Forward Error Correction (Contd…)

- Combining Hamming Distance and Interleaving
  - Create *n*-bit packets that can correct *t*-bit errors.
  - Interleave *m* rows and send the bits column by column
  - Correct burst errors up to *m* × *t*-bit errors.

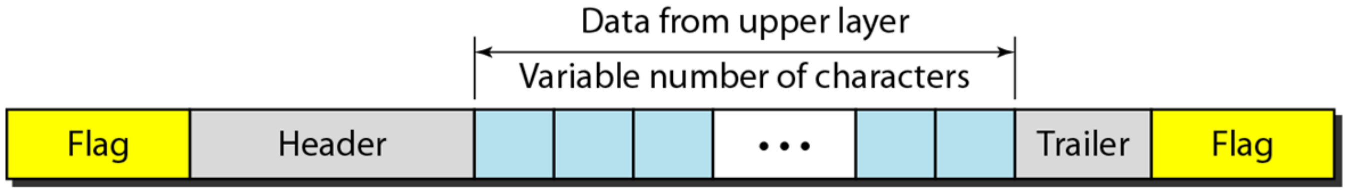
- Compounding High- and Low-Resolution Packets
  - Useful for multimedia data
  - First packet has an empty low resolution section
  - Last packet cannot be recovered



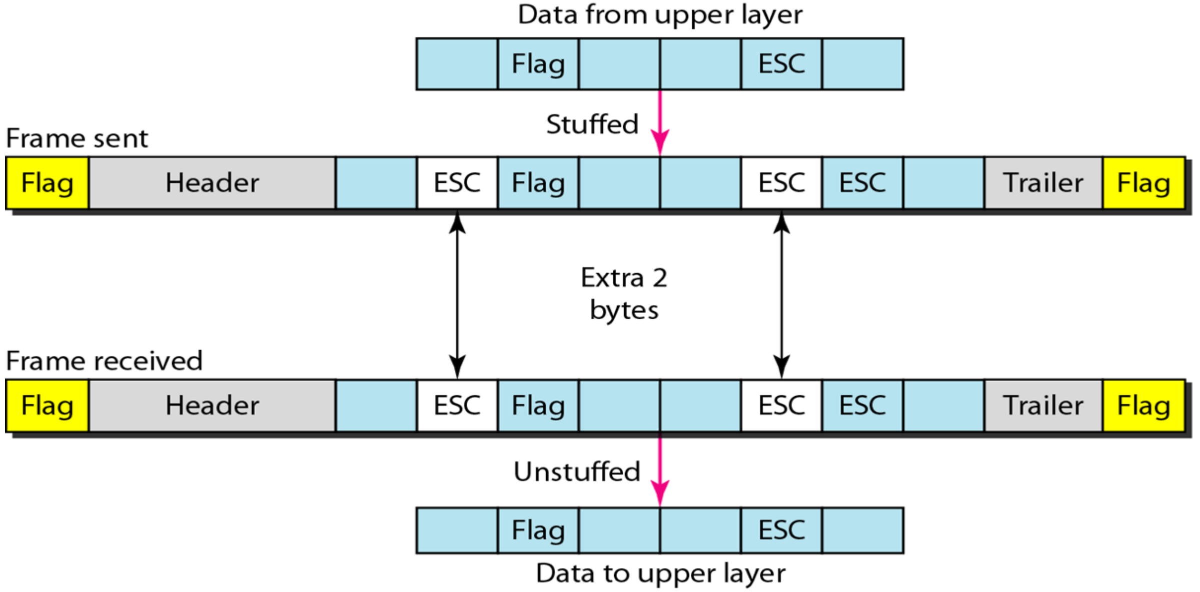
Compounding High and Low Resolution Packets

DLC Services

- Framing : packing bits into frames so that each frame is distinguishable
- Adding a sender address and a receiver address
- Receiver address: defines where the packet should go
- Sender address: helps in acknowledgement
- Frame size : variable size in LANs
  - Character-oriented approach
    - Flag acts as a frame delimiter
    - Byte stuffing: to prevent the receiver from reaching end of the frame early
    - Limitation: more bit patterns same as flag after one or more *escape characters*



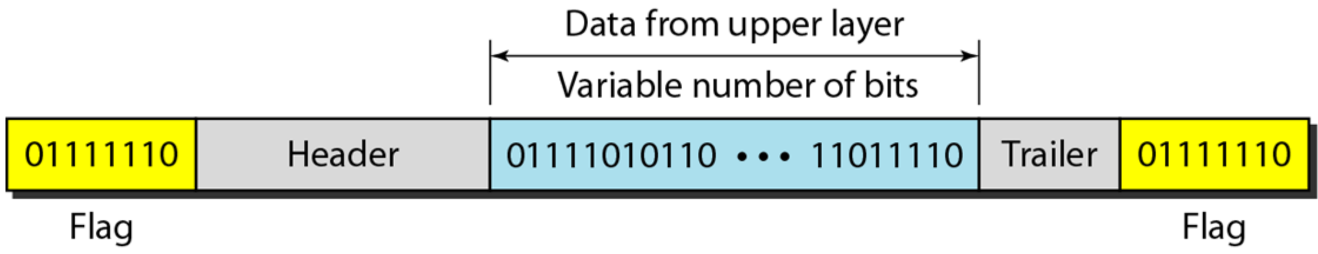
Character-oriented Framing



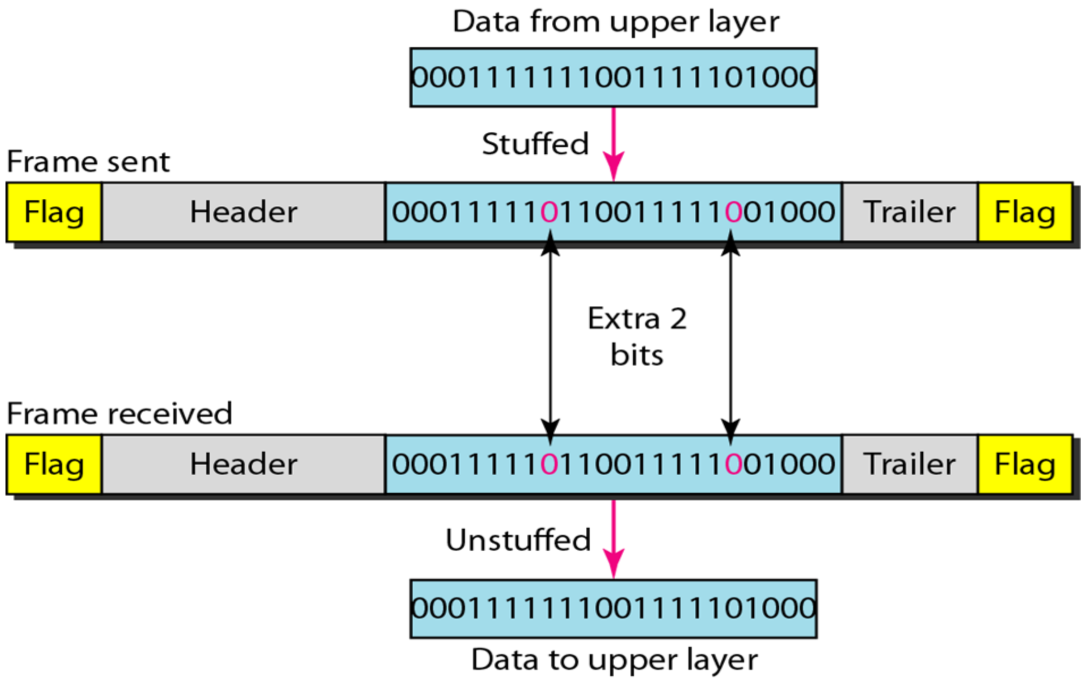
Byte Stuffing and Unstuffing in Character-Oriented Framing

DLC Services (Contd…)

- Frame size : variable size in LANs
  - Bit-oriented approach
    - Special 8-bit flag: 01111110
    - Bit stuffing: to prevent the receiver from reaching end of the frame early
    - Adding a 0 after 011111
- Flow control
  - Sender should not overwhelm the receiver
  - Prevent buffer overflow
  - Restrict the amount of data that the sender can send before waiting for acknowledgment.
- Combination of Flow and Error control
  - Based on automatic repeat request
  - Retransmission of data



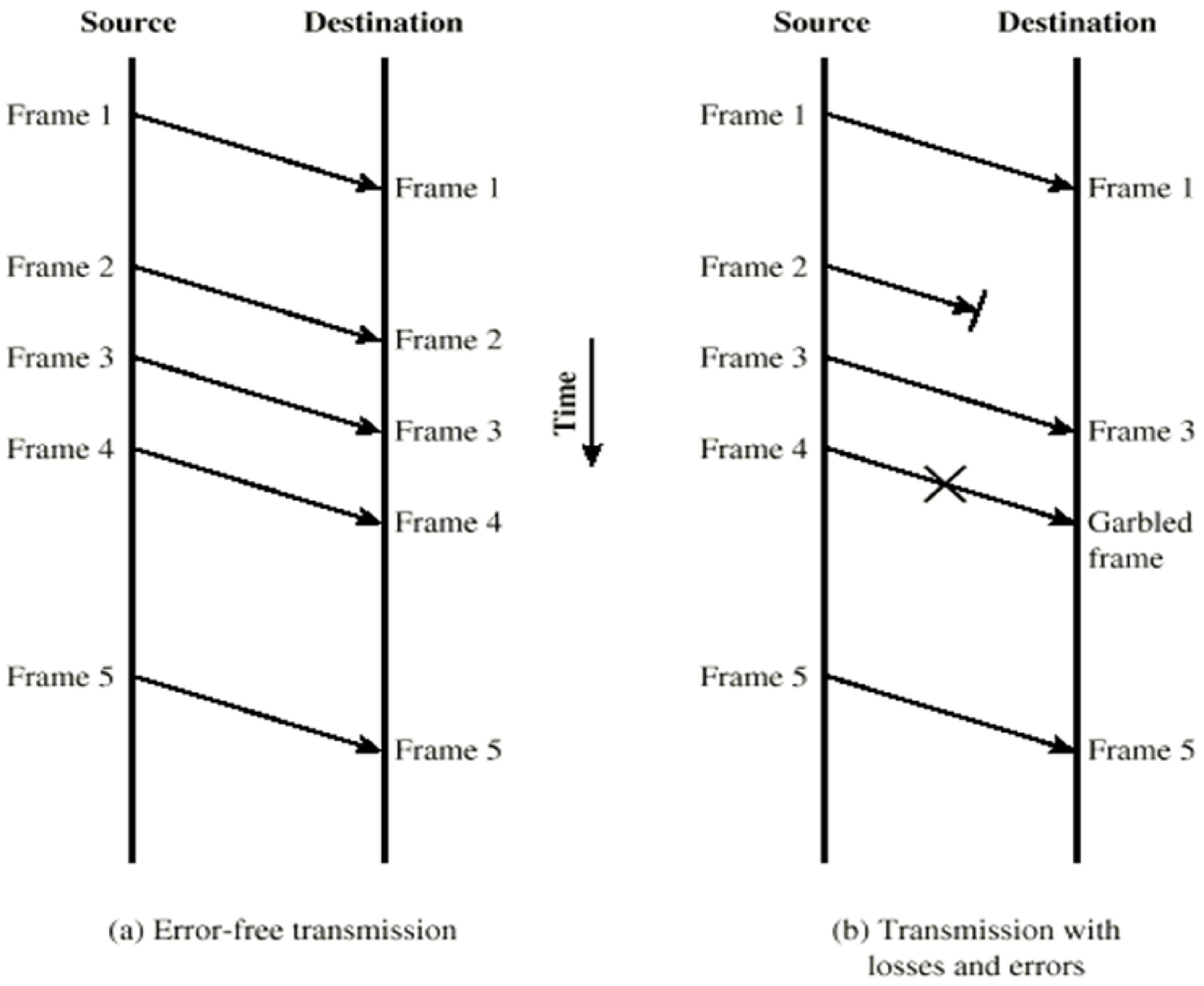
Bit-oriented Framing



Bit Stuffing in Bit-Oriented Framing

Flow Control

- Transmission time: time taken to emit all bits into medium
- Propagation time: time for a bit to traverse the link
- Stop-and-Wait Flow Control
  - Source transmits frame
  - Destination receives frame and replies with acknowledgement
  - Source waits for ACK before sending next frame
  - Destination can stop flow by not sending ACK



Model of Frame Transmission

Flow Control

- Stop-and-Wait Flow Control
  - Link utilization
    - *a > 1*: link is always under utilized
    - *a < 1*: link is inefficiently utilized
  - Not suitable for links with very high data rates or if the distances between sender and receiver is very long
- Sliding Window Flow Control
  - Allow multiple frames to be in transit
  - Receiver has buffer *W* long
  - Sender can send up to *W* frames without *ACK*; Each frame is numbered
  - *ACK* includes sequence number of next frame expected
  - Sequence number range: 0 to $2^k - 1$ (k: size of the sequence number field)
  - Frame numbering: modulo 2^k

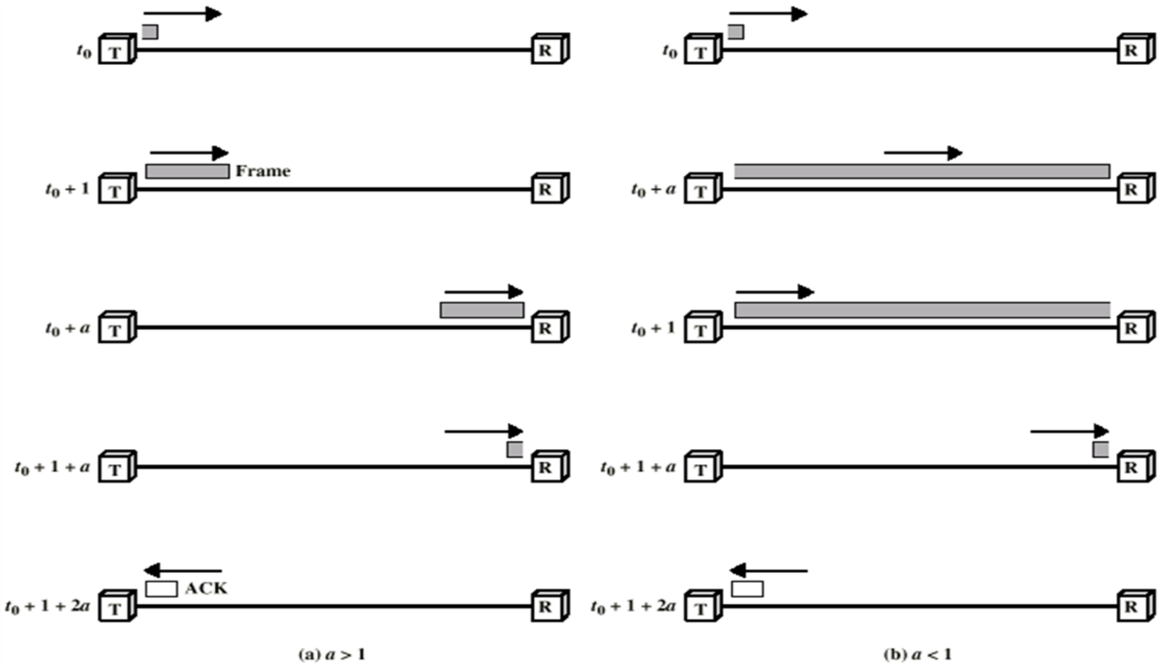
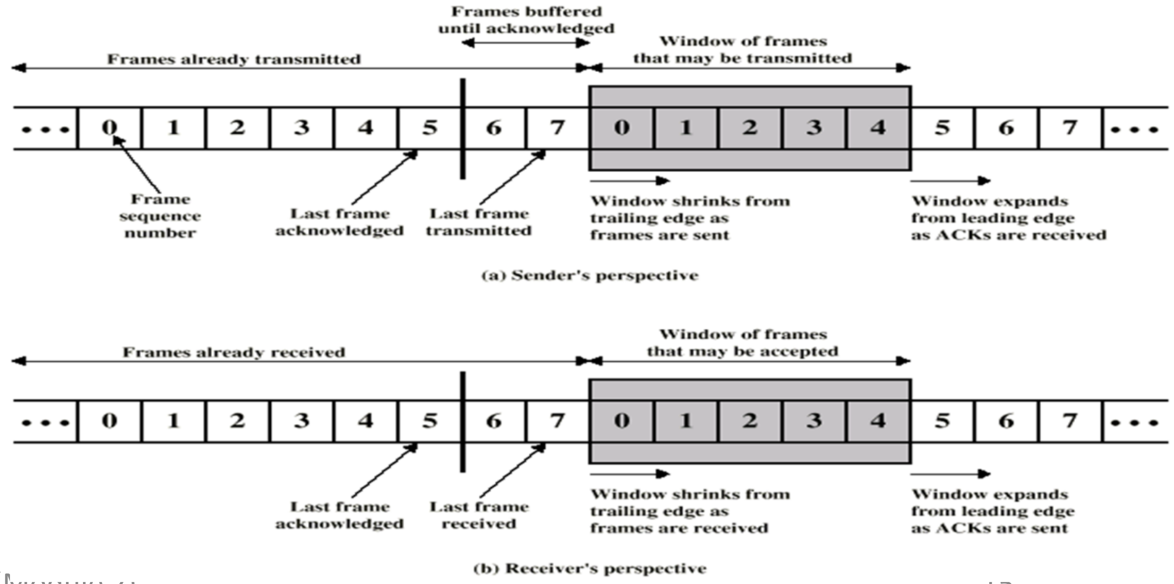


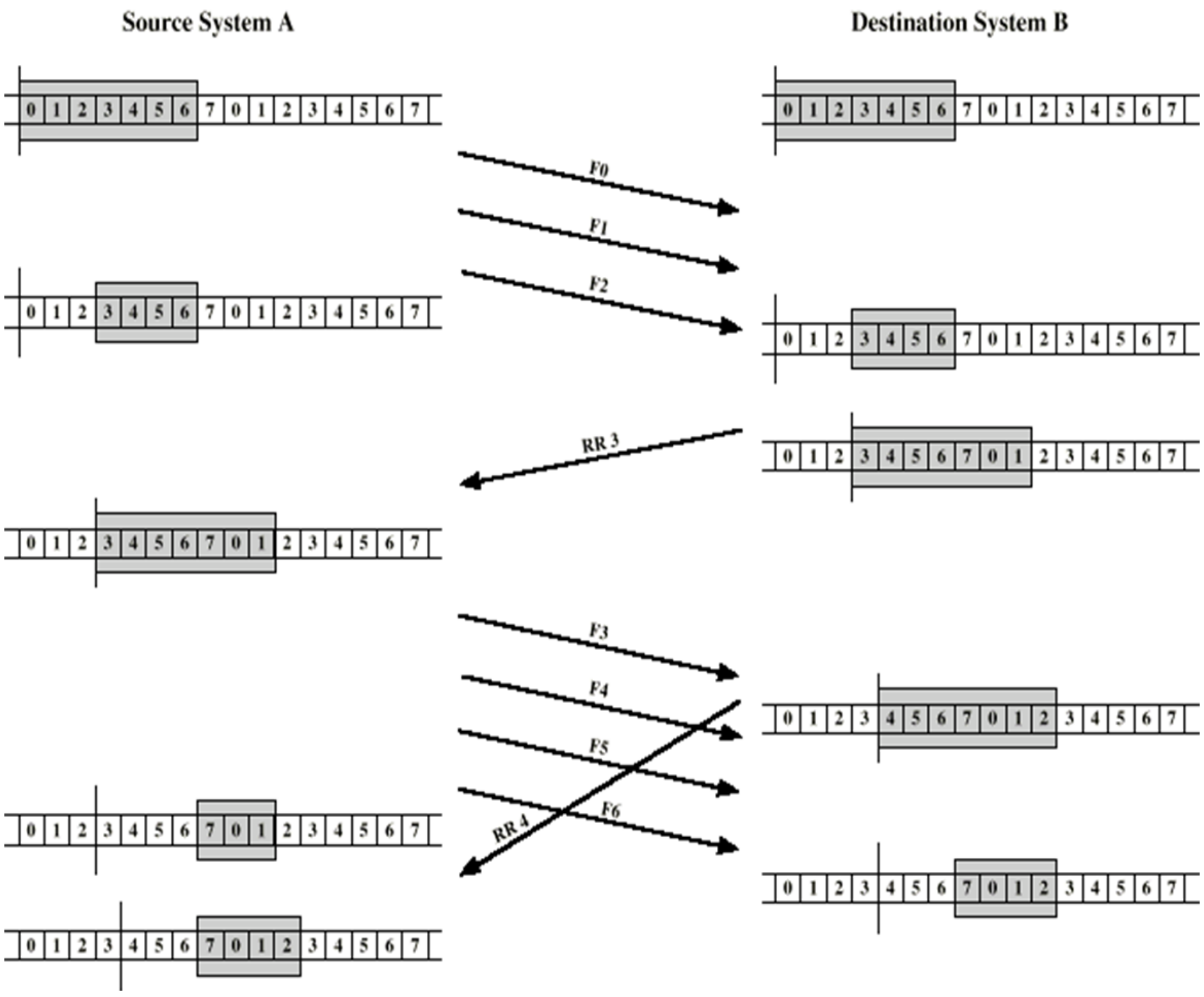
Figure 7.2 Stop-and-Wait Link Utilization (transmission time = 1; propagation time = a)



Sliding Window Design

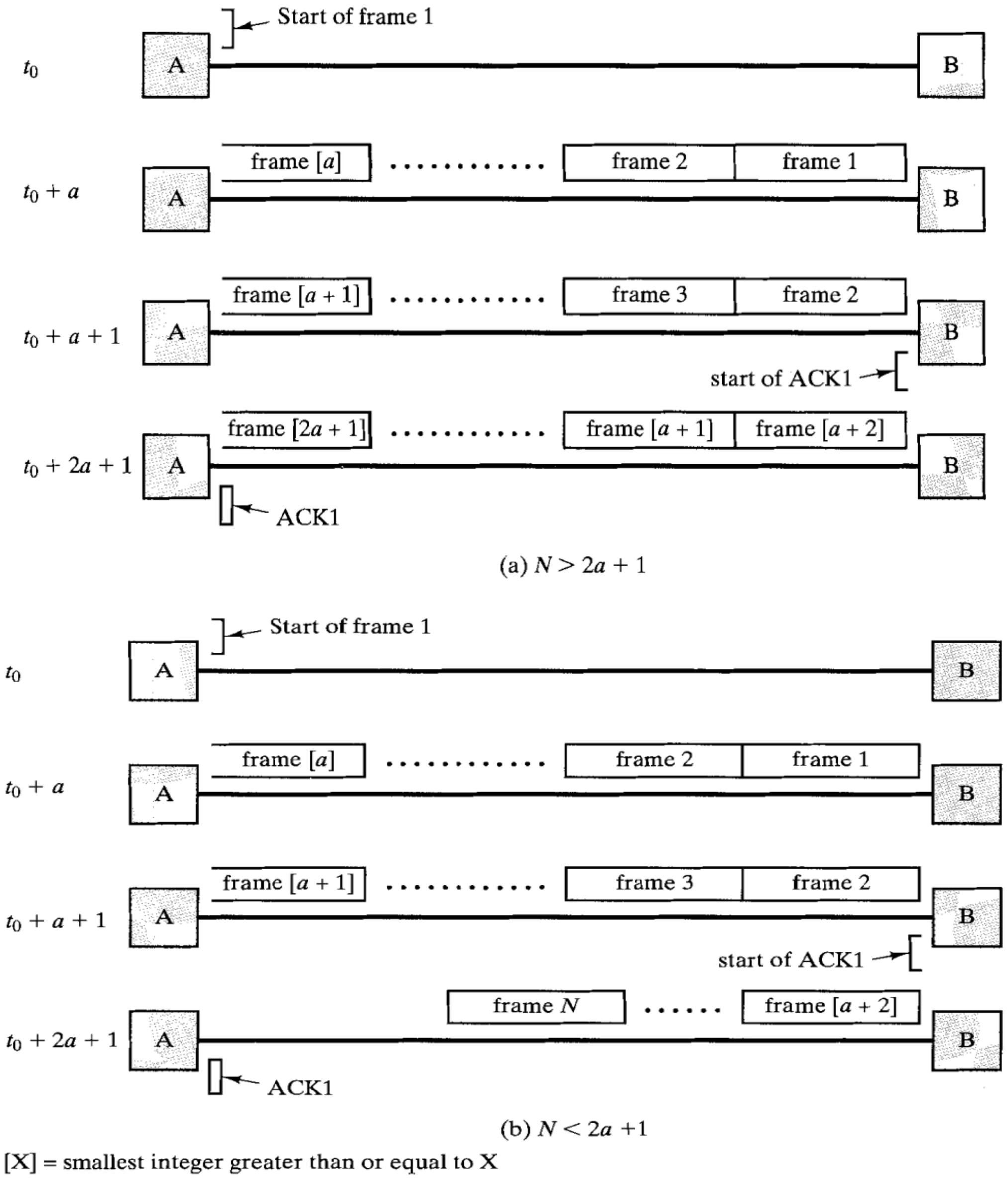
Flow Control

- Sliding Window Flow Control: Enhancements
  - Receiver can acknowledge frames without permitting further transmission (Receive Not Ready)
  - Must send a normal acknowledge to resume
  - If duplex, use piggybacking
  - If no data to send, use acknowledgement frame
  - If data but no acknowledgement to send, send last acknowledgement number again, or have ACK valid flag (TCP)

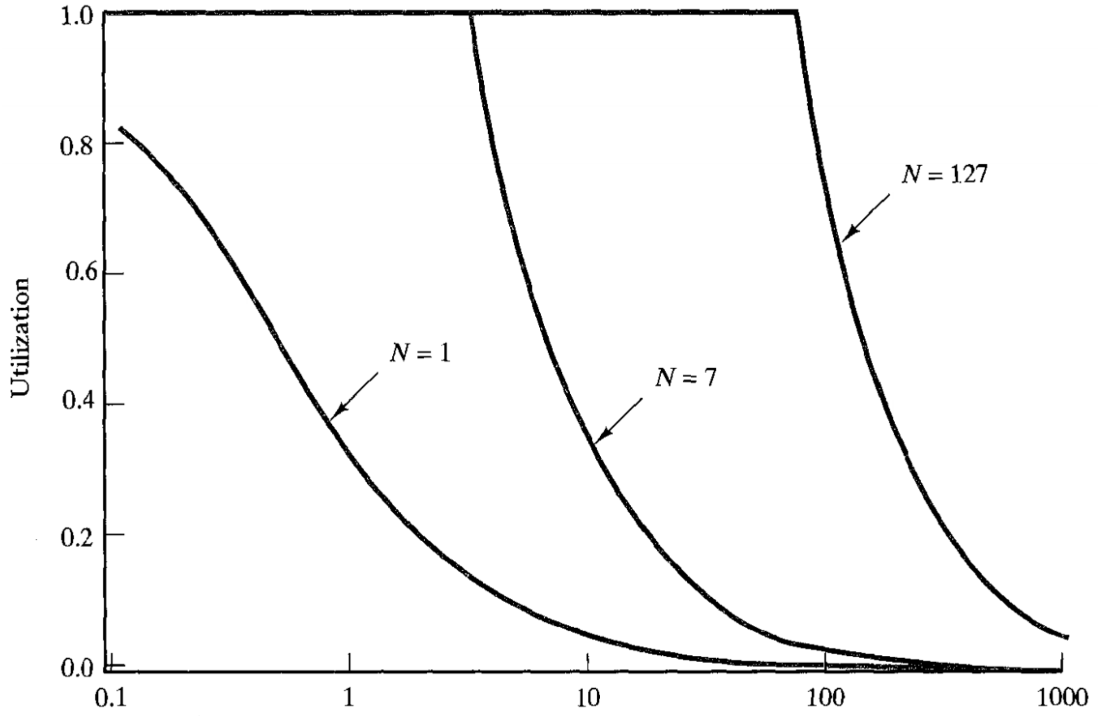


Example of Sliding Window

Performance Issue: Sliding Window Flow Control



Timing of a Sliding Window Protocol



Line Utilization as a Function of Window Size

- Window size:
  - N = 1: Stop-and-Wait Flow Control
  - N=7 (3-bits): Adequate for many applications
  - N = 127 (7-bits): Found in high-speed WANs

Error Control

- Two types of error
  - Lost frame: network fails to deliver a frame
  - Damaged frame: frame arrives, but some bits are in error

- Common Error Control Techniques
  - Error detection
  - Positive acknowledgement
  - Retransmission after time-out
  - Negative acknowledgement and retransmission

Automatic Repeat Request (ARQ)

- Effect of ARQ: turn a potentially unreliable data link into a reliable one

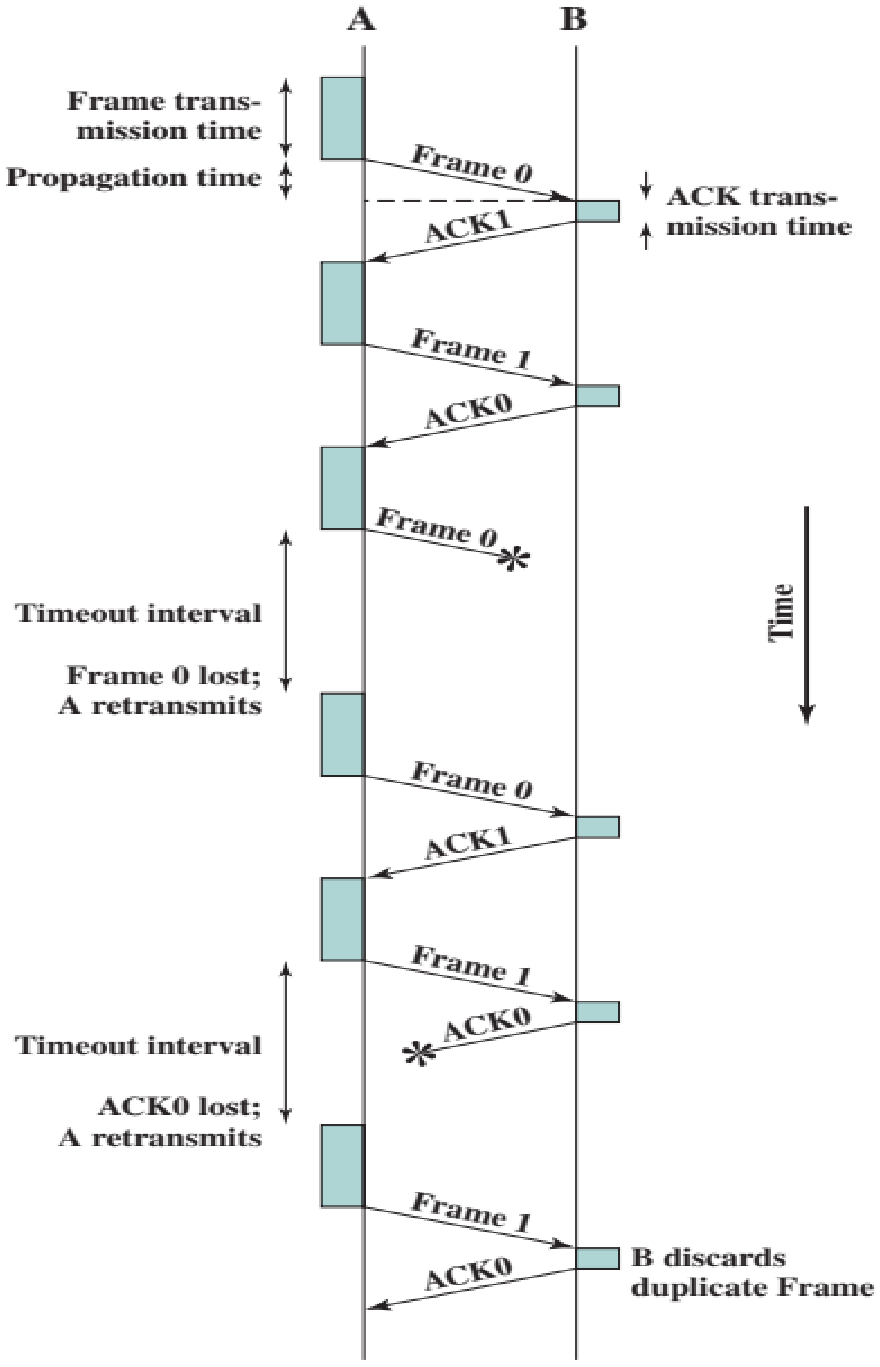
- Standardized versions of ARQ:
  - Stop-and-wait ARQ
  - Go-back-N ARQ
  - Selective-reject/Selective-retransmission ARQ

Stop-and-Wait ARQ

- Addresses two types of errors
  - Frame arrived at the destination is damaged
  - Acknowledgement is damaged in transit

- Simple technique – easy to implement

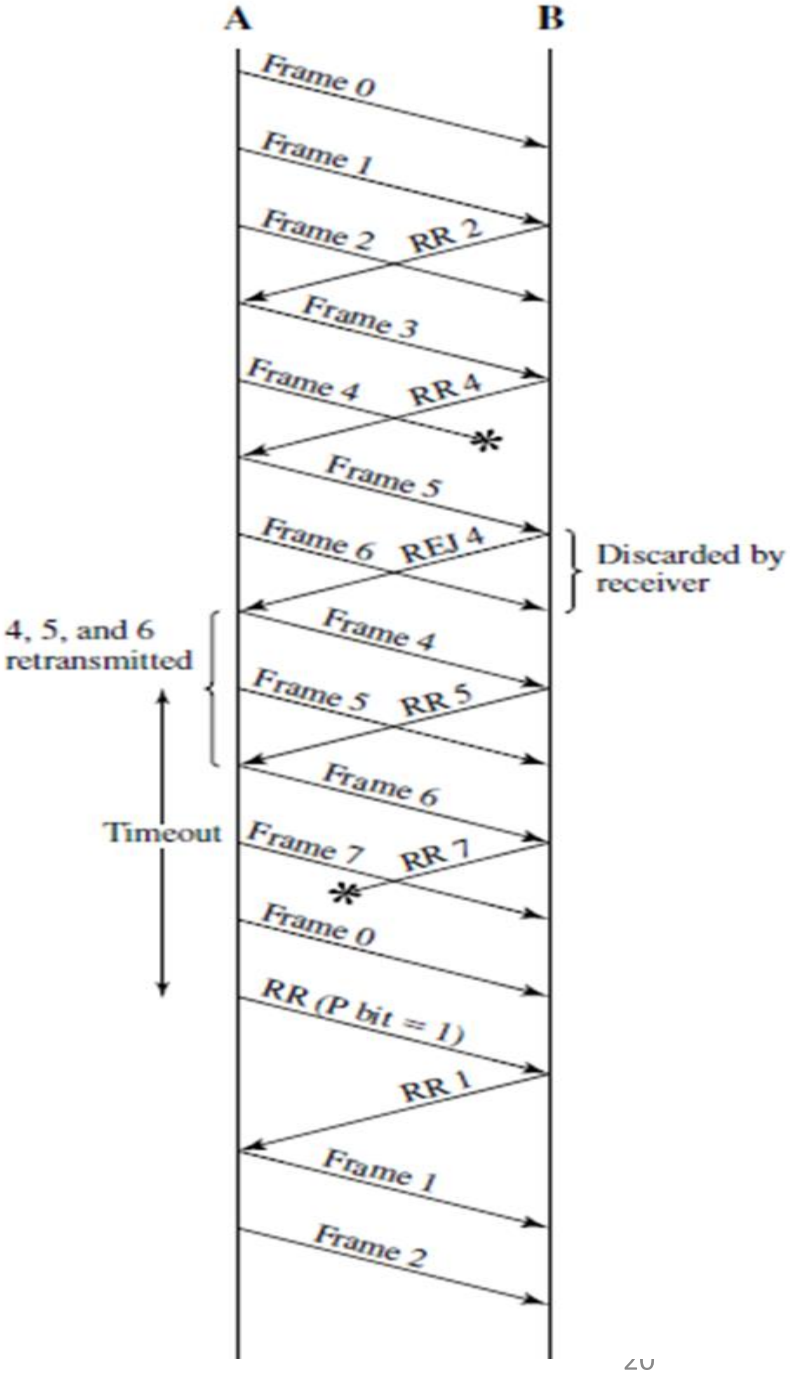
- Inefficiency in link utility



Stop-and-Wait ARQ

Go-back-N ARQ

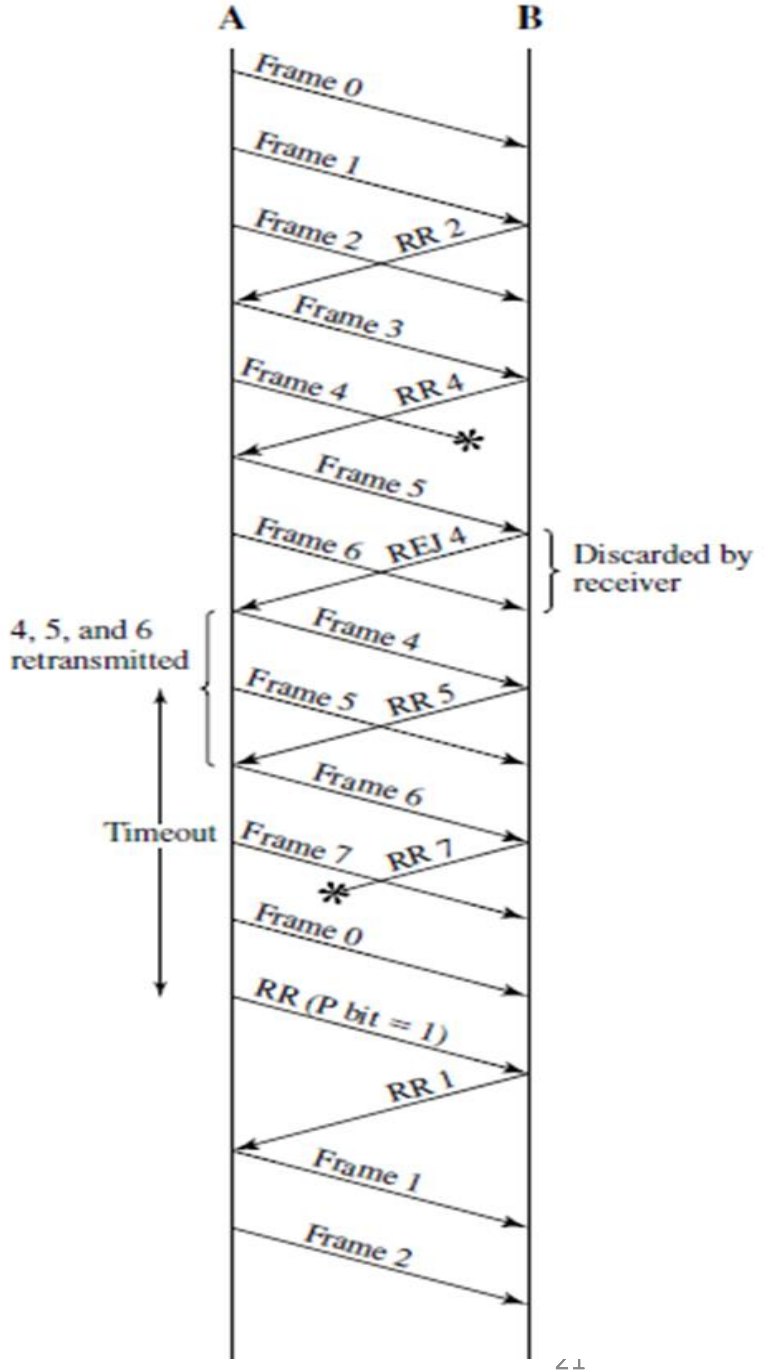
- Based on sliding window flow control
- Sliding window: tracks unacknowledged frames
- Transmitter: sends series of sequentially numbered frames modulo some maximum value
- Receiver
  - If no error, acknowledge incoming frames (RR = receive ready, or piggybacked acknowledgment)
  - If error, reply with negative acknowledgement (REJ = rejection)
  - Discards the frame and all the subsequent incoming ones
- Sender retransmits the frame in error plus all succeeding frames



Go-back-N ARQ

Go-back-N ARQ (Contd…)

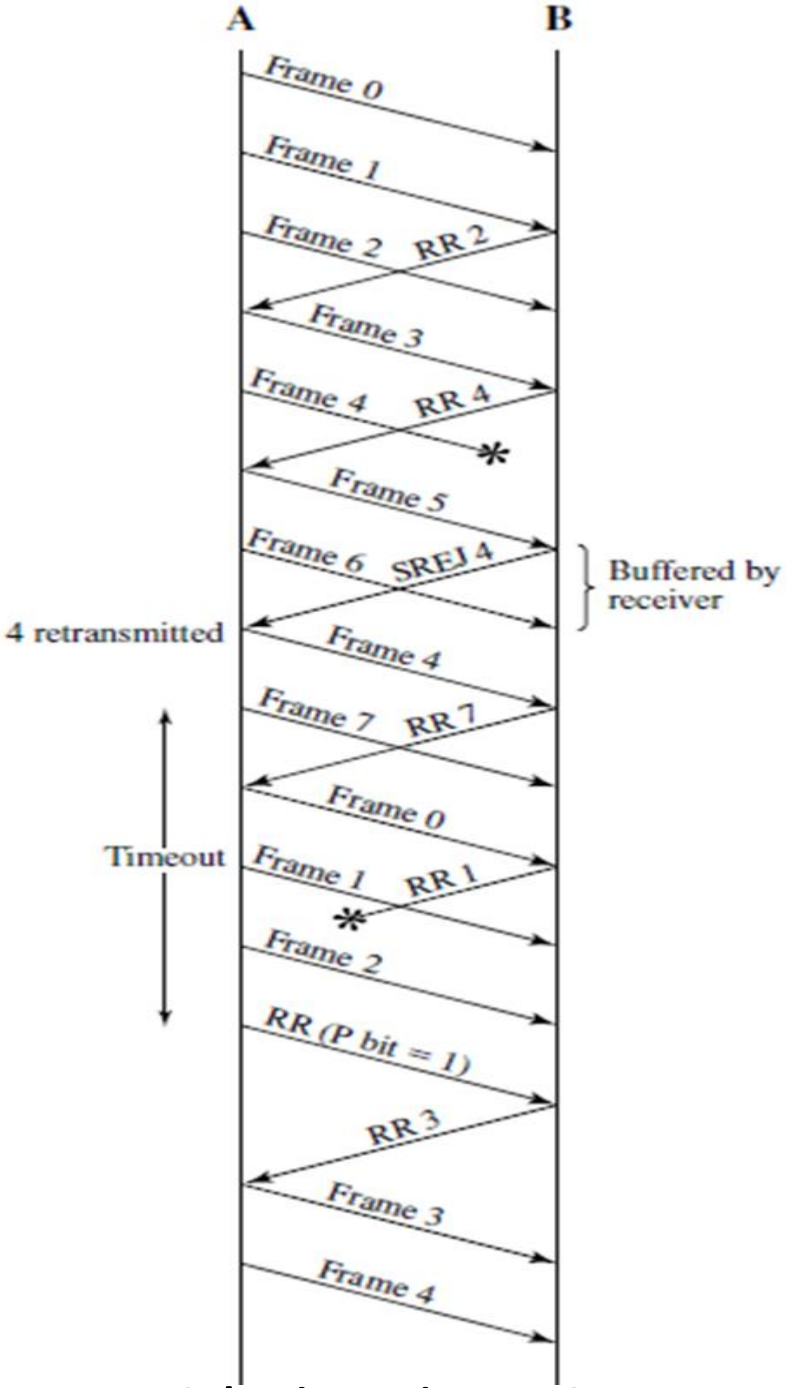
- Damaged frame: *B* detects error in frame *i ;* sends rejection-*i*
  - *A* receives rejection-*i*
  - Two scenarios
    - *A* retransmits frame *i* and all subsequent frames within a reasonable period of time
    - *A* does not respond back soon – sends an *RR* with *P* bit set to 1 if timer expires - Receiver acknowledges by sending an *RR* indicating the next frame that it expects (frame *i*) – A retransmits frame *i*
- Damaged RR: B receives frame *i* and sends *RR (i + 1)*, suffers an error in transit.
  - Two scenarios
    - Before *A*'s timer expires, *it* receives a subsequent *RR* to a subsequent frame arriving.
    - *A*'s timer expires - sets the *P*-bit timer - *B* fails to respond to the *RR*, or if its response suffers an error in transit - *A*'s *P*-bit timer will expire – It issues a new *RR* by restarting the *P*-bit timer – this is attempted for predefined number of times - *A* fails to obtain an acknowledgment - it initiates a reset procedure.
- Damaged REJ: Same as second case for damaged frame



Go-back-N ARQ

Selective-Reject/Selective Retransmission ARQ

- Only frames retransmitted are those that receive a negative acknowledgment (SREJ)
- Minimizes the amount of retransmission
  - More efficient than Go-back-N ARQ
- More overhead on transmitter and receiver end
- Receiver
  - Needs to maintain buffer large enough to save *post-SREJ* frames until the frame in error is retransmitted
  - Logic for reinserting the frame in proper sequence
- Transmitter
  - Logic to be able to send frames out of sequence
- For a *k*-bit sequence number field, which provides a sequence number range of 2^k, the maximum window size is limited to 2^{k-1}



Selective-Reject ARQ