Chapter 4

The Data Link Layer

Tran Thanh Dien, PhD
College of Information and communication Technology
Can Tho University

Contents

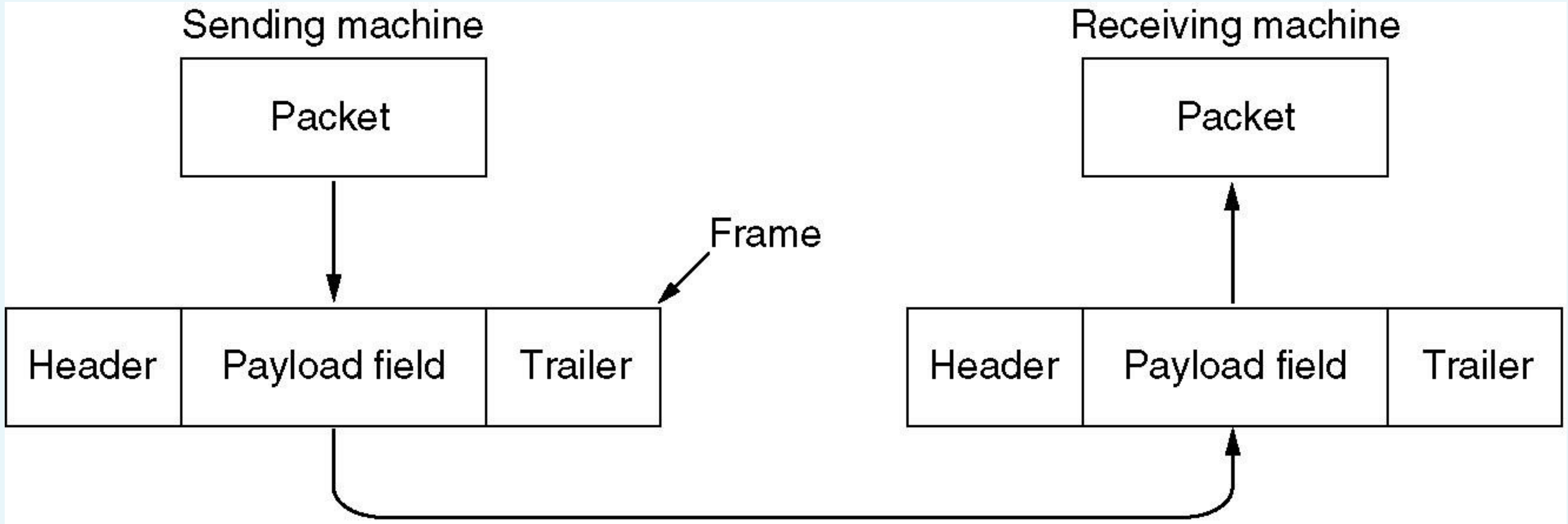
- Services Provided to the Network Layer
- Framing
- Error Control
- Flow Control

Functions of the Data Link Layer

- Provide service interface to the network layer
- Dealing with transmission errors
- Regulating data flow
  - Slow receivers not swamped by fast senders

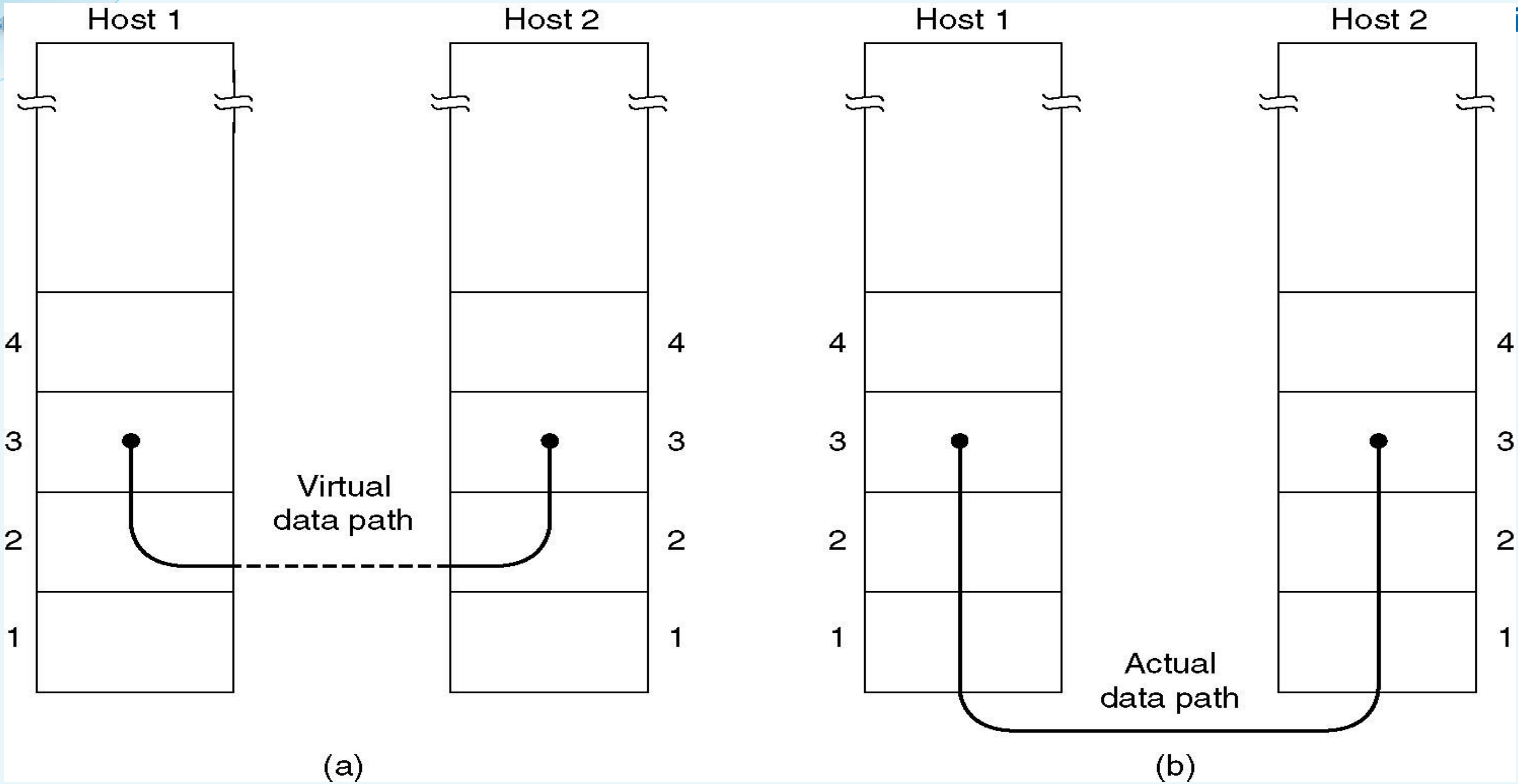
Functions of the Data Link Layer

- Providing services the Network layer
- Takes the packets from the network layer and encapsulates them into frames



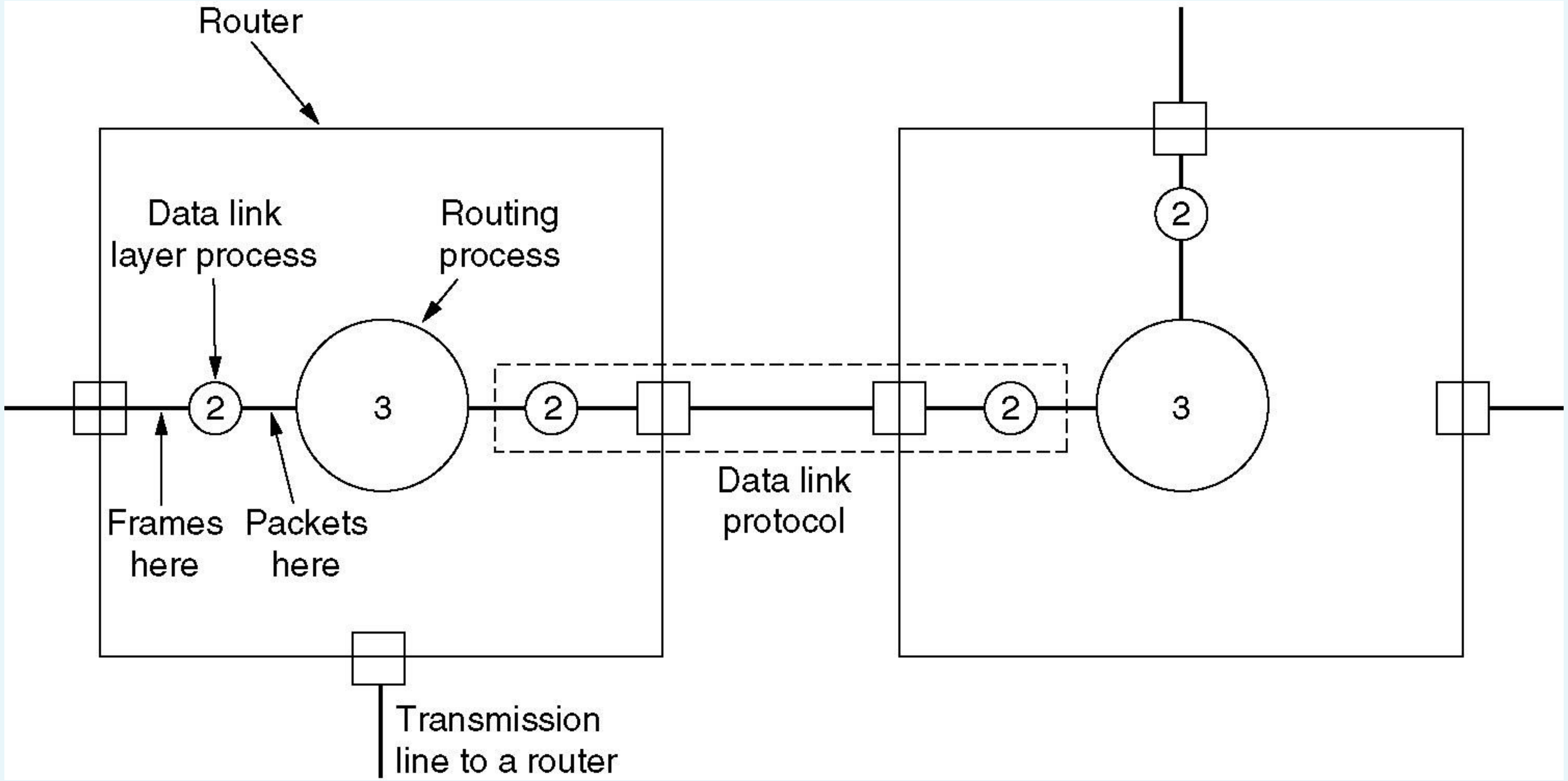
Relationship between packets and frames.

Services Provided to Network Layer



(a) Virtual communication (b) Actual communication.

Services Provided to Network Layer



Placement of the data link protocol.

Framing

- Bit stream received by the data link layer not guaranteed to be error free:
  - bits inversed
  - Number of bits received may be less than, equal to, or more than number of bits transmitted
- Data link layer:
  - Break up the bit stream into discrete frames
  - Compute a checksum and include the checksum in the sent frame
  - At the destination, the checksum is recomputed
  - If the computed checksum different from the one in the frame: an error occurred and takes steps to deal with it

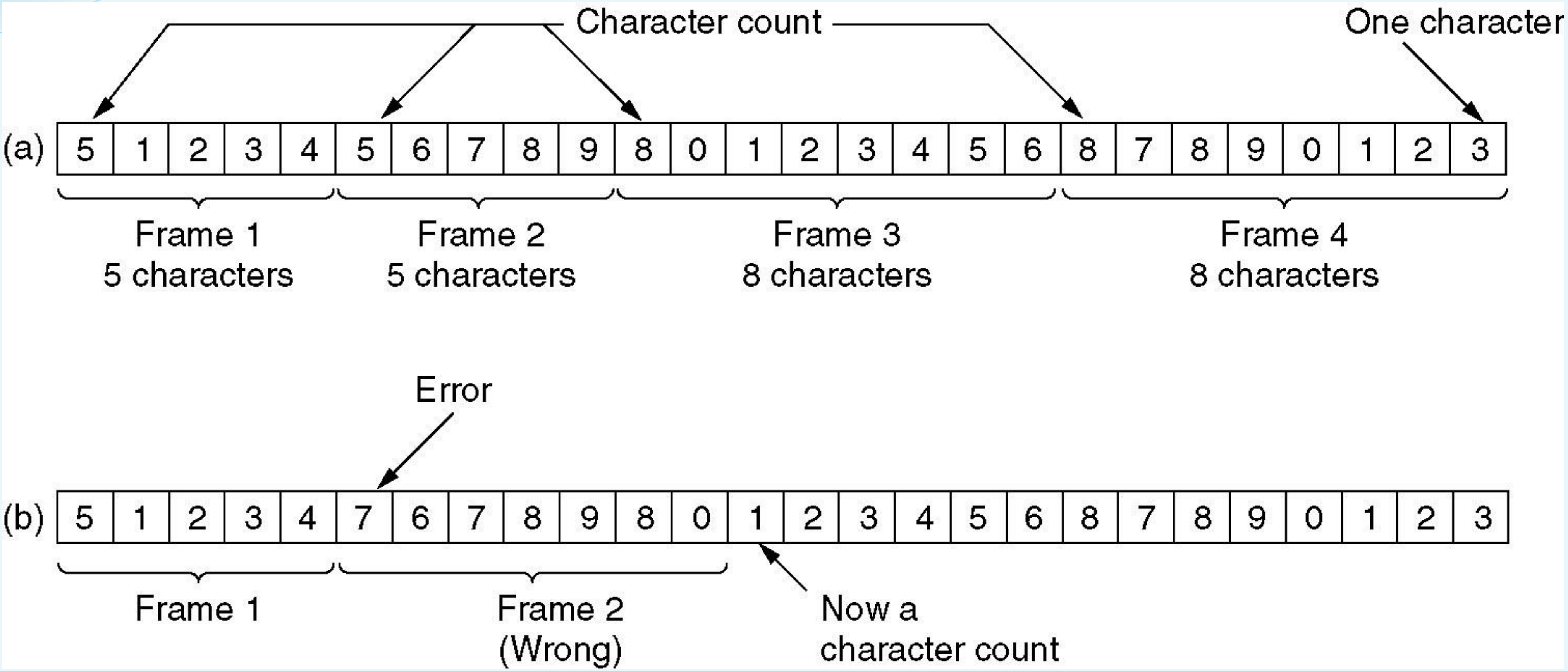
Framing

Framing methods

- Byte count
- Flag bytes with byte stuffing
- Flag bits with bit stuffing.

Framing

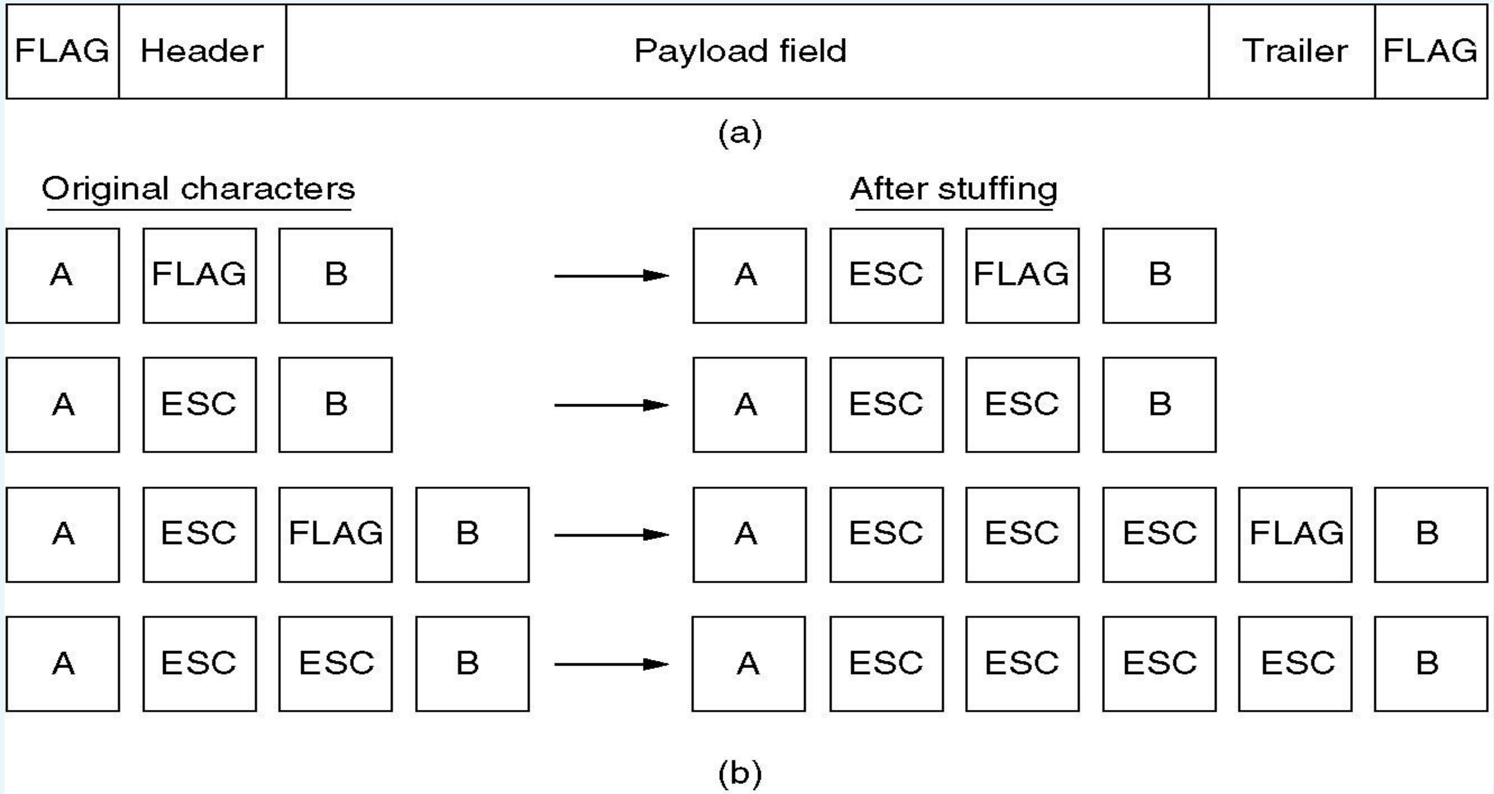
Framing methods: Character count



A character stream. (a) Without errors. (b) With one error.

Framing

Framing methods: Flag byes and byte stuffing

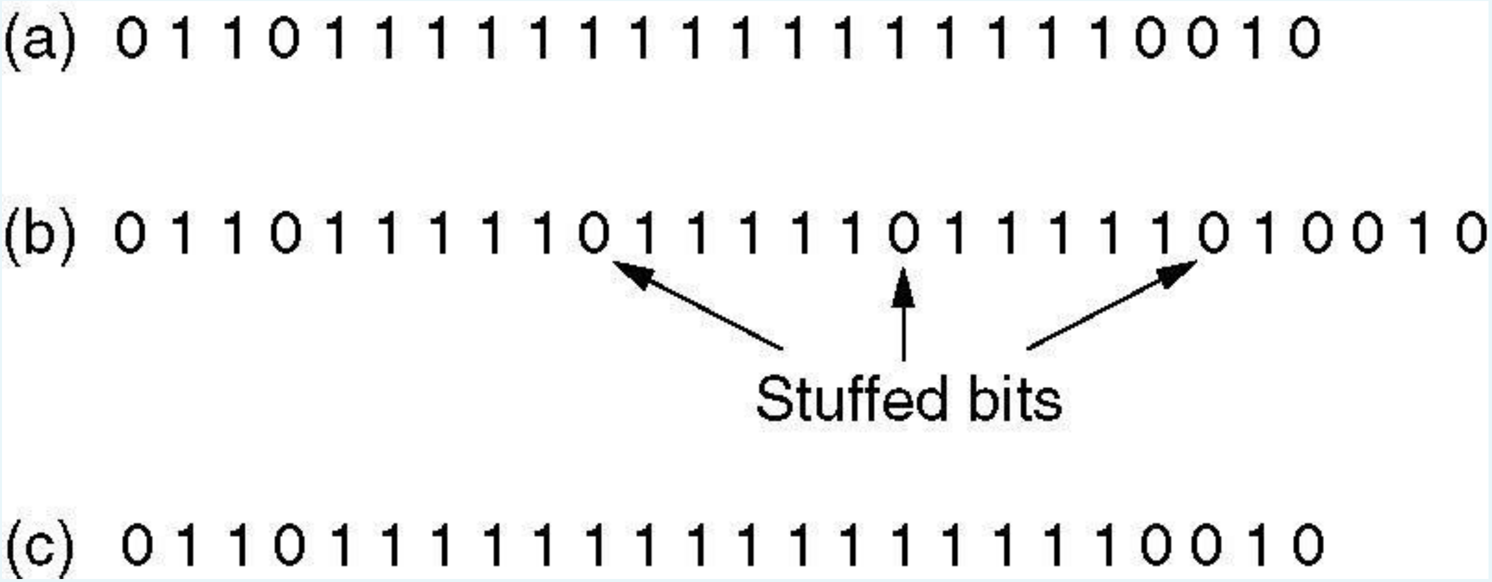


(a) A frame delimited by flag bytes.

(b) Four examples of byte sequences before and after stuffing.

Framing

Framing methods: Flag byes and byte stuffing



Bit stuffing

(a) The original data.

(b) The data as they appear on the line.

(c) The data as they are stored in receiver's memory after destuffing.

Error Control

Make sure all frames eventually delivered to the network layer at the destination and in the proper order

1) Receiver sending back special control frames bearing positive or negative acknowledgements about the incoming frames:

   Using acknowledgement frame

2) Avoiding the sender to be wait forever when acknowledge frame lost

   Using a timer and time-out for each sent frame

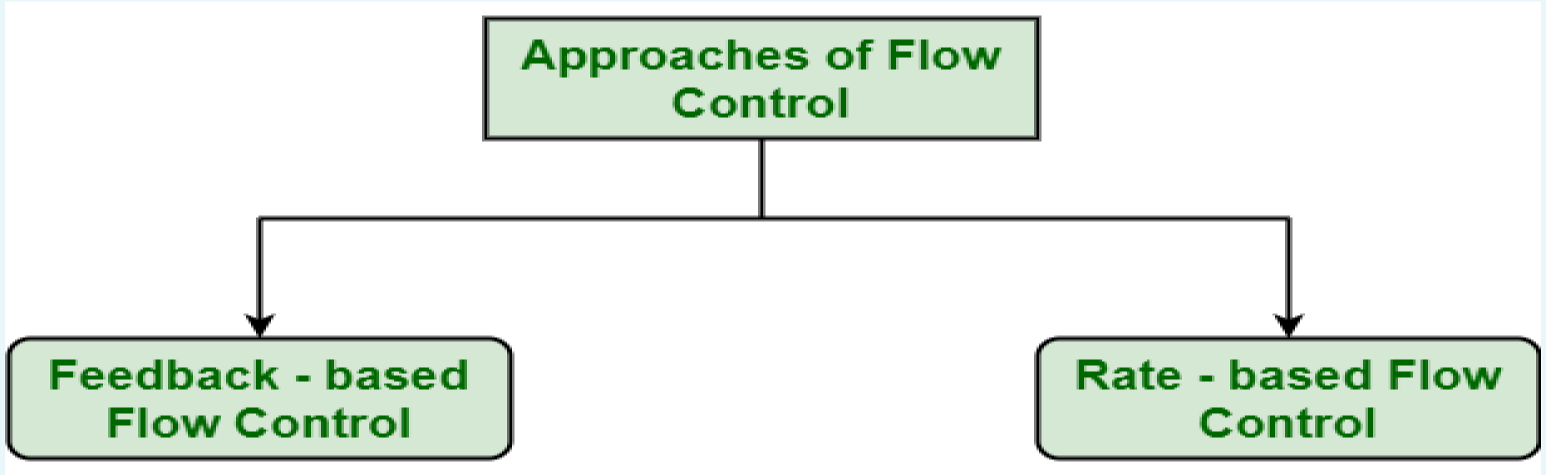
3) Avoiding frame duplication

   Assign each sent frame a sequence number

CANTHO UNIVERSITY

Flow Control

- Prevent a sender to transmit frames faster than the receiver can accept them
- Two approaches

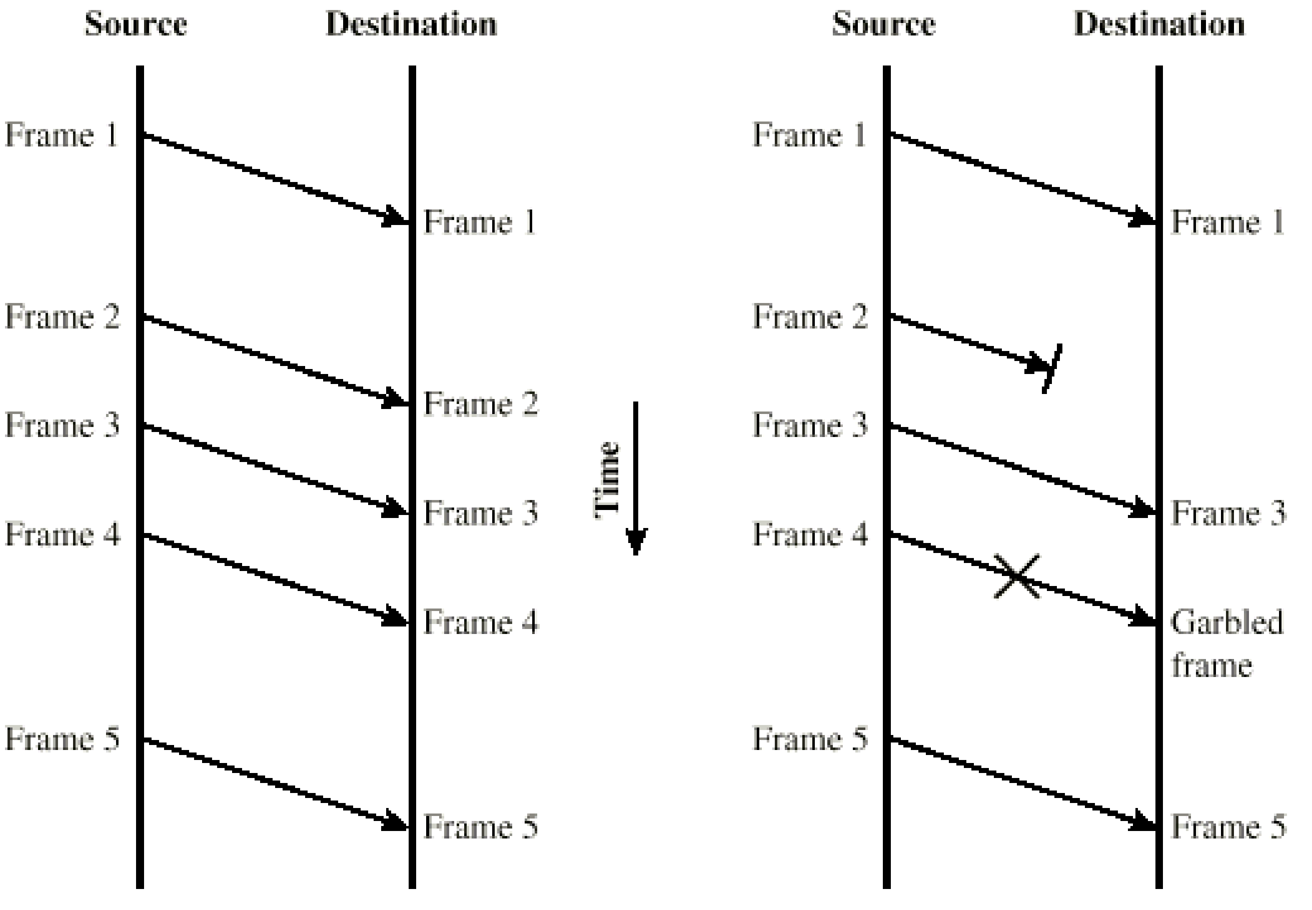


The receiver send back information to sender giving it permission to send more data or at least tell sender what the receiver doing

The protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver

Error Control

Transmission errors



(a) Error-free transmission

(b) Transmission with losses and errors

Error Control

Transmission errors

- Bit 1 becomes bit 0 and reversely
- Error rate

  $\forall\tau$ = Error bits / Transmitted bits

  $\forall\tau$: 10^{-5} to 10^{-8}

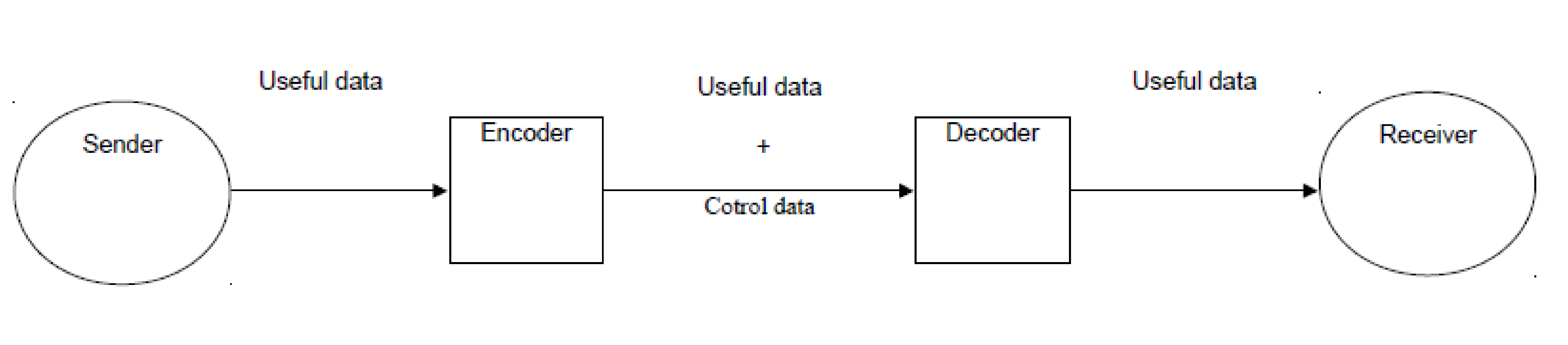
  88% : error of one bit

  10% : error of two adjacent bits

Error Control

Error Detection and Correction

- Error-Correcting Codes
- Error-Detecting Codes



- Sender: add control data into useful data for sending
- Receiver: decode control data to determine whether data has errors or not

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

Use of a Hamming code to correct burst errors.

Error Control

Error-Correcting Codes

Allow receivers to identify error data and correct error data

Hamming Code

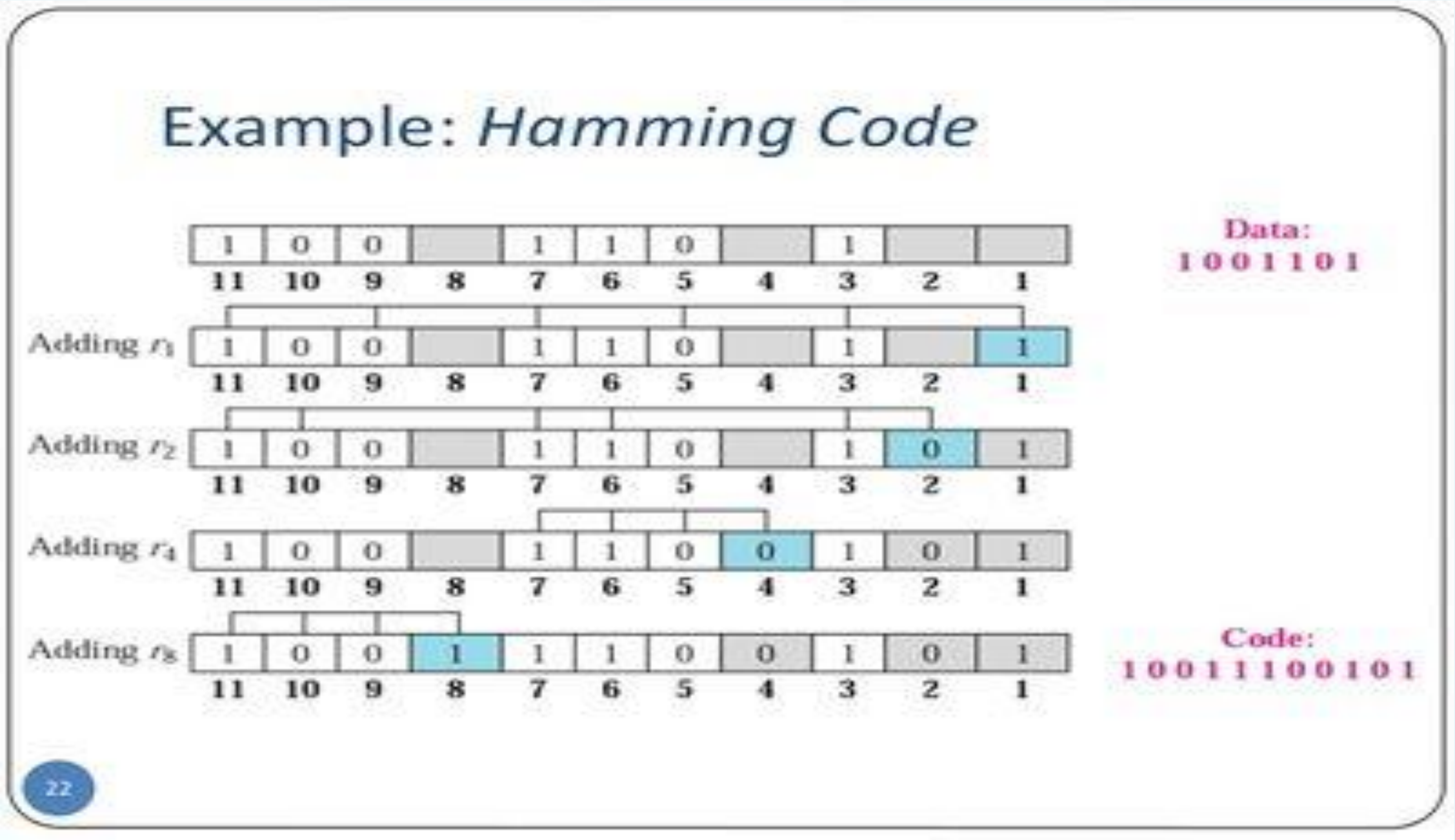
- For each integer m > 2
  - Code exists with m parity bits and $2^m - m - 1$
- Basically
  - Parity bit for odd bits
  - Parity bit for each two bits
  - Parity bit for each four bits
  - Parity bit for each eight bits
  - Parity bit for each group of bits that are power of 2
    - 1, 2, 4, 8, 16, 32, …

Calculating Hamming Code

1. Mark all bit positions that are powers of two parity bit
   1, 2, 4, 8, 16, 32, 64, etc
2. All other bit positions are for the data to be encoded
   3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.
3. Each parity bit calculates the parity for some of the bits in the code word
   The position of the parity bit determines the sequence of bits that it alternately checks and skips
   Position 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (1,3,5,7,9,11,13,15,...)
   Position 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (2,3,6,7,10,11,14,15,...)
   Position 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. (4,5,6,7,12,13,14,15,20,21,22,23,...)
   Position 8: check 8 bits, skip 8 bits, check 8 bits, skip 8 bits, etc.
   (8-15,24-31,40-47,...)
4. Set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even

Error Control

Error-Correcting Codes



Error Control

Error-Detecting Codes

- Allow receivers to determine whether received data has error or not
- If error is detected in received data, request for resending data
- Popular error-detecting codes
  - ✓ Parity checks
  - ✓ Check sum
  - ✓ Cyclic redundancy check

Error Control

Error-Detecting Codes: Parity Check

- xxxxxxx: Useful data need to be transmitted
- One parity bit is appended to useful data
- Transmitted stream of bits: xxxxxxxp
- Calculation of p
  - ✓ Event parity: xxxxxxxp consists of an event number of bits 1
  - ✓ Odd parity: xxxxxxxp consists of an odd number of bits 1

Detecting error in a stream of bits xxxxxxxp:

- In even parity check:
  - ✓ If there is an even number of bits 1→ data xxxxxxx are error-free
  - ✓ Else data xxxxxxx has error
- In odd parity check:
  - ✓ If there is an odd number of bits 1→ data xxxxxxx are error-free
  - ✓ Else data xxxxxxx has error

Error Control

Error-Detecting Codes: Parity Check

- Example: G = 1110001 are useful data
- Using even parity check:
  - ✓ p=0
  - ✓ Transmitted data: 1110001**0**
- Different cases of received data:
  - ✓ 11100010: 4 bits 1=> Error-free
  - ✓ 11000010: 3 bits 1=> Error
  - ✓ 11000110: 4 bits 1=> Error ???

Error Control

Error-Detecting Codes: Longitudinal Redundancy Check

								Parity bits
	1	0	1	1	0	1	1	**1**
Data bits →	1	1	0	1	0	1	1	**1**
	0	0	1	1	1	0	1	**0**
	1	1	1	1	0	0	0	**0**
	1	0	0	0	1	0	1	**1**
	0	1	0	1	1	1	1	**1**
LRC bits →	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**0**

Error Control

Error-Detecting Codes: Cyclic Redundancy Check

Different methods of implementation:

- Modulo 2,
- Polynomial
- Shift register
- Exclusive-or gate

Error Control

Error-Detecting Codes: CRC-Modulo 2

- M: Message of k bits need to send
- F : frame check sequence of r bits (control data appended to M to detect error on M)
- T =MF: the transmitted frame of (k + r) bits created by concatenating M and F, with r < k
- P (r+1 bits): a prior defined sequence
- F computed as follows:

1) Appending r bits 0 to the end of M, or multiply M with 2r
2) Using binary division to divide M*2r by P.
3) F is the remainder of the binary division

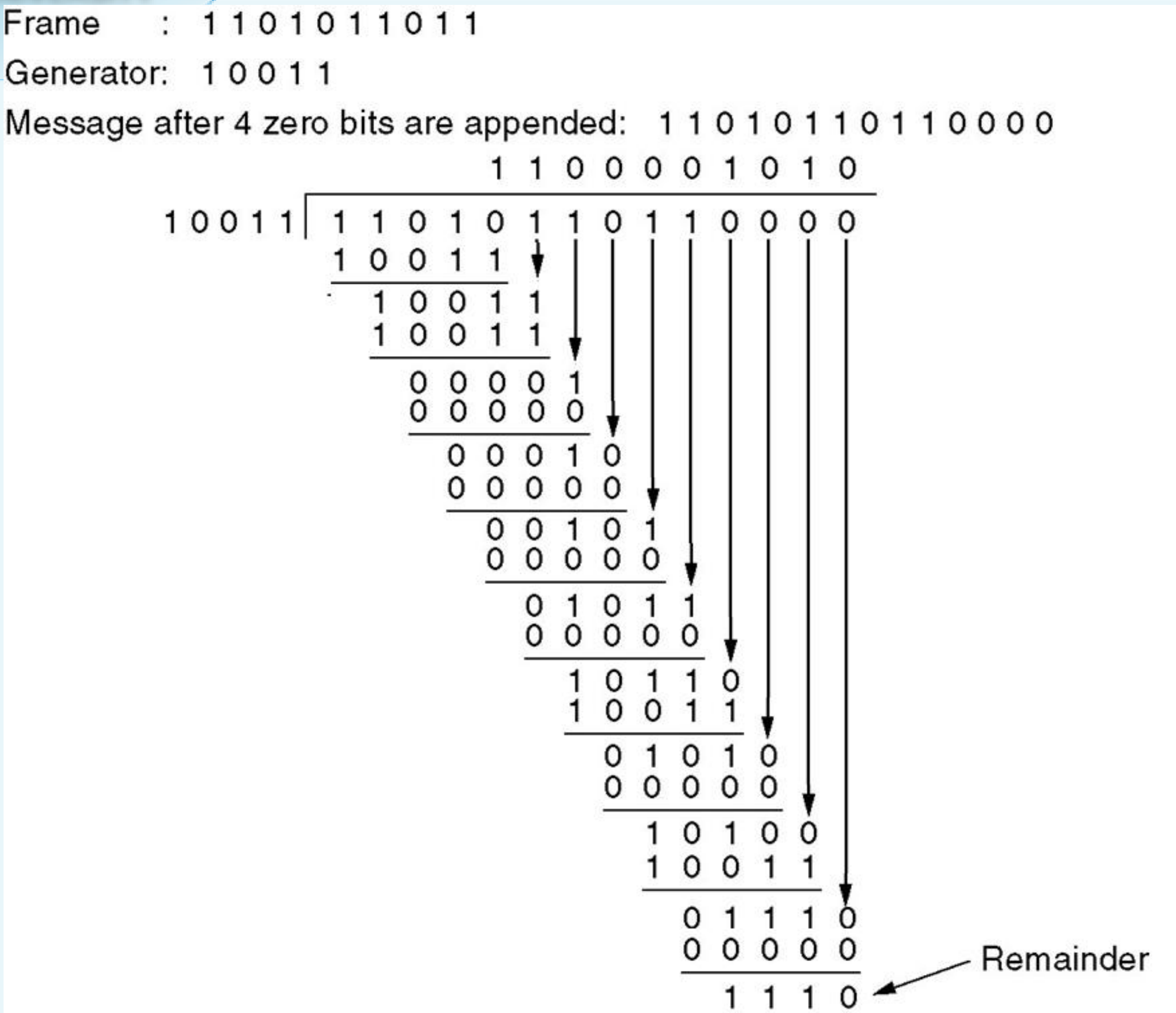
Error Control

Error-Detecting Codes: CRC-Modulo 2

- Appending F to M to create transmitted frame T
  - ✓ Note: P has to be longer than F one bit and the values of its most significant bit and the least significant must be 1
- Receiver applies binary division to divide T by P:
  - ✓ No remainder: Error-free, M extracted from (T – k) high order bits
  - ✓ Remainder existed: Transmission of T is error

Error Control

Error-Detecting Codes: CRC-Modulo 2



Transmitted frame
11010110111110

Error Control

Error-Detecting Codes: CRC-Modulo 2

- M = 1010001101 (k=10 bits)
- P = 110101 (r+1=6 bits)
- Steps to calculate FCS (r=5bits):

1) Calculate $M*2^5$ = 101000110100000.
2) Apply binary division to divide $M*2^5$ by P

   Get remainder F = **01110**

- Create transmitted frame

✓ $T = M*2^r + F$ = 1010001101**01110**

- Bài tập 1: Bên gởi va nhận đa thống nhất chọn P=110101

Cho biết các khung T nhận được sau có lỗi hay không?

a) **T=101.0001.1010.1110**

b) **T=101.0101.1010.1110**

- Supposing that M=110011 and P = 11001, then M and P are represented by two following polynomials:
  - $M(x) = x^5 + x^4 + x + 1$
  - $P(x) = x^4 + x^3 + 1$
- Algorithm for computing the CRC is represented as follows:

$$\frac{X^r M(X)}{P(X)} = Q(X) + \frac{F(X)}{P(X)}$$

$\Rightarrow T(X)=X^n M(X)+R(X)$

Error Control

Error-Detecting Codes: CRC-Polynomial

- Popular versions of P

  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 = $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + + X^8 + X^7 + X + X^4 + X^2 + X + 1$

- Example

M=1010001101, P=110101 => r=5

$M(X) = X^9 + X^7 + X^3 + X^2 + 1$

$X^5 M(X) = X^{14} + X^{12} + X^8 + X^7 + X^5$

$P(X) = X^5 + X^4 + X^2 + 1$

Compute

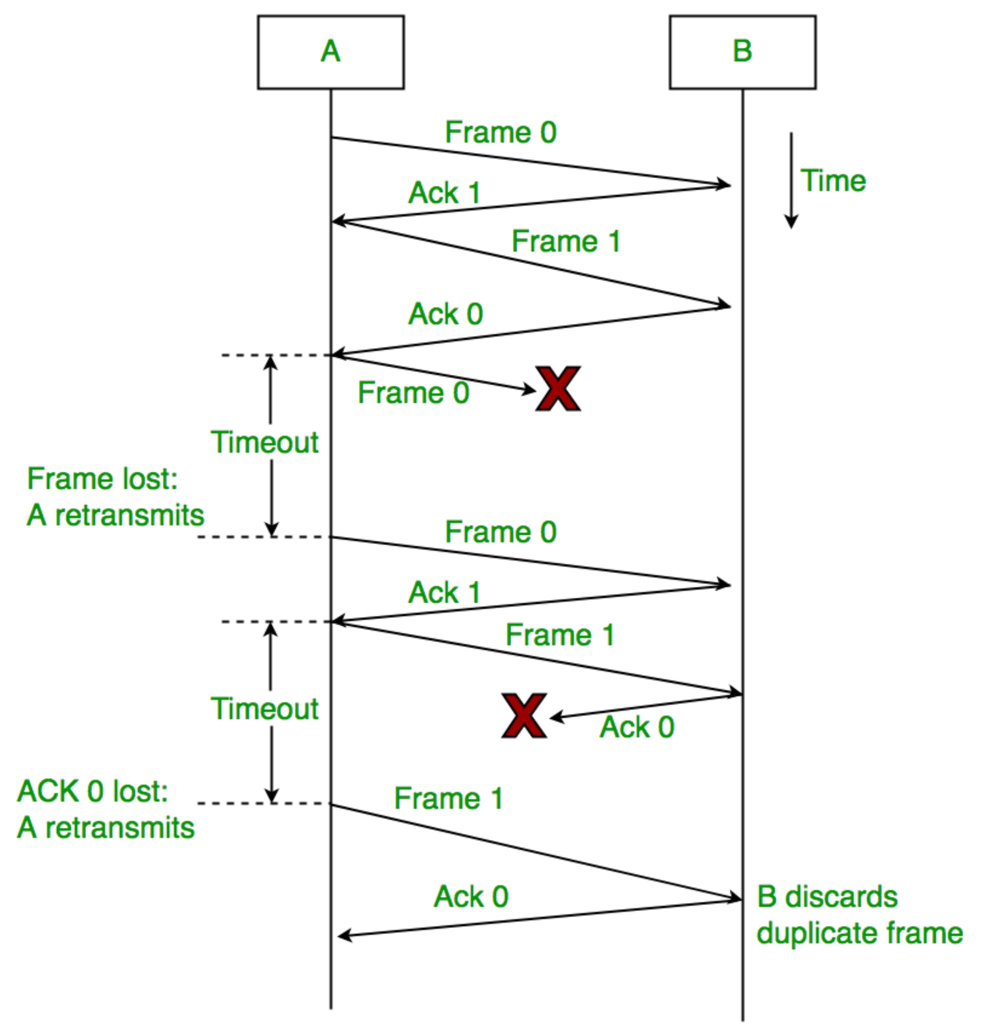
$\frac{X^r M(X)}{P(X)} = Q(X) + \frac{F(X)}{P(X)}$ => $F(X) = X^3 + X^2 + X^1$ Or F = 01110

=> Final transmission data: T = 101000110101110

Error Control

Stop and Wait protocol

- Sender doesn't know whether the frame was transmitted successfully.
  - ✓ Solution: Use acknowledgement frame
- Acknowledgement frame could be lost.
  - ✓ Solutions:
    - o Timer.
    - o Time-out
    - o Resend
- Receiver receives duplicate frames
  - ✓ Solution: Assign a sequence number for each frame



Error Control

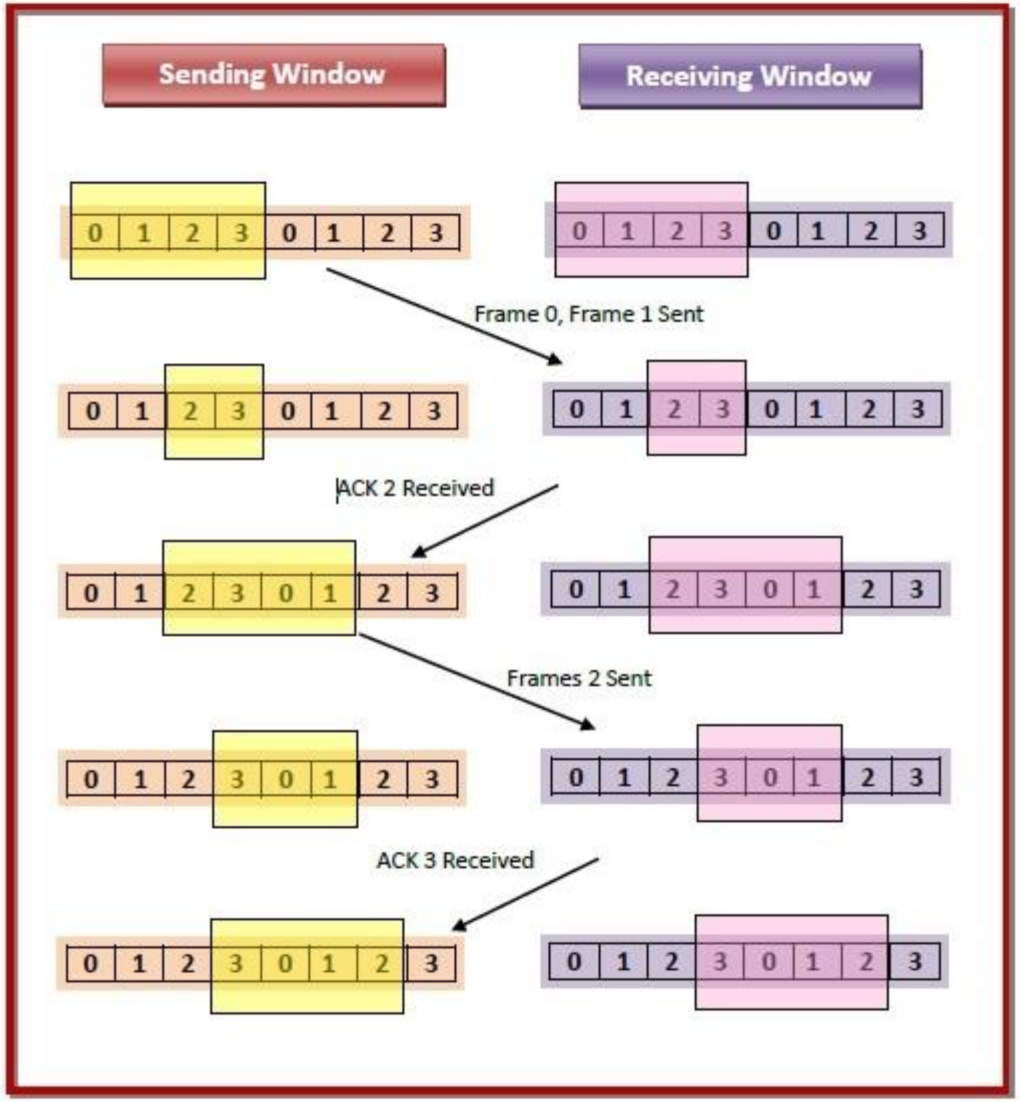
Duplex data transmission

- Stop and Wait: Simplex data transmission
- Need to achieve duplex data transmission to exploit maximum channel capacity
- Principle for archiving duplex data transmission:
  - ✓ Define frame types: DATA, ACK, NACK
  - ✓ Using **piggyback** technology

Error Control

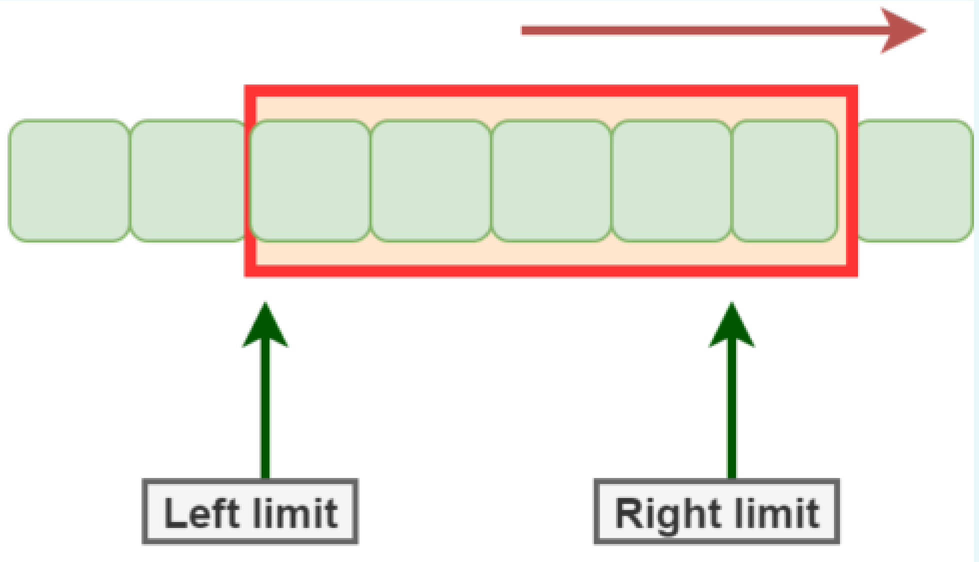
Sliding Window Protocols

- Sliding window protocol allows a sender to send multiple frames before waiting for acknowledgement
- Sending Windows: used by sender to monitor sent frames that acknowledgement frames are being waited for
- Receiving Windows: used by receiver to monitor the frames that are permitted to receive



Error Control

Sliding Window Protocols

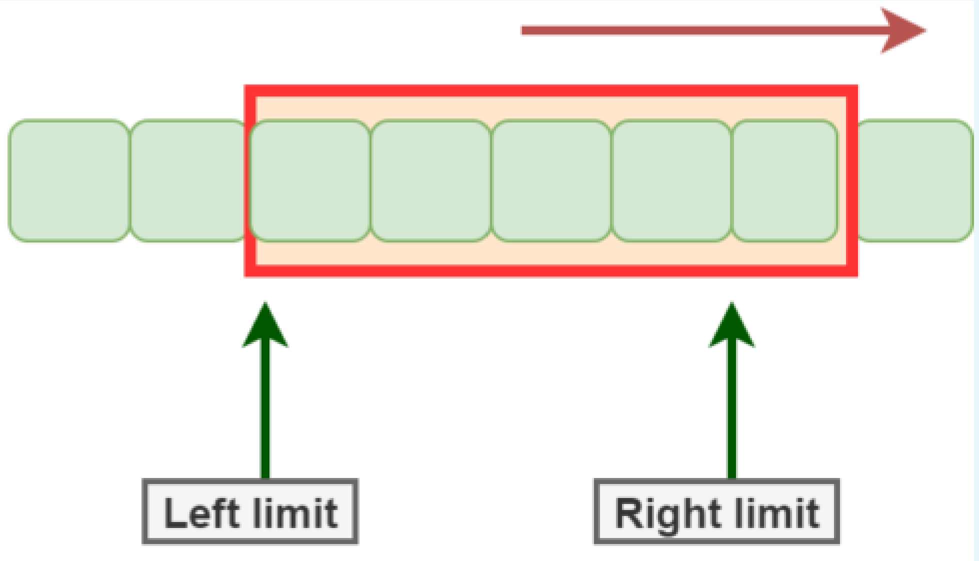


- Two leaves:
  - ✓ Front and back
  - ✓ Move the same direction
  - ✓ The size of the windows from the back to the front
- For sending window:
  - ✓ Potions inside the window represent the sequence numbers of frames sent and waiting for acknowledgements
  - ✓ Potions outside the window represent the sequence numbers of frames can send.
  - ✓ The size of the window not larger than the maximum size of the window

Error Control

Sliding Window Protocols

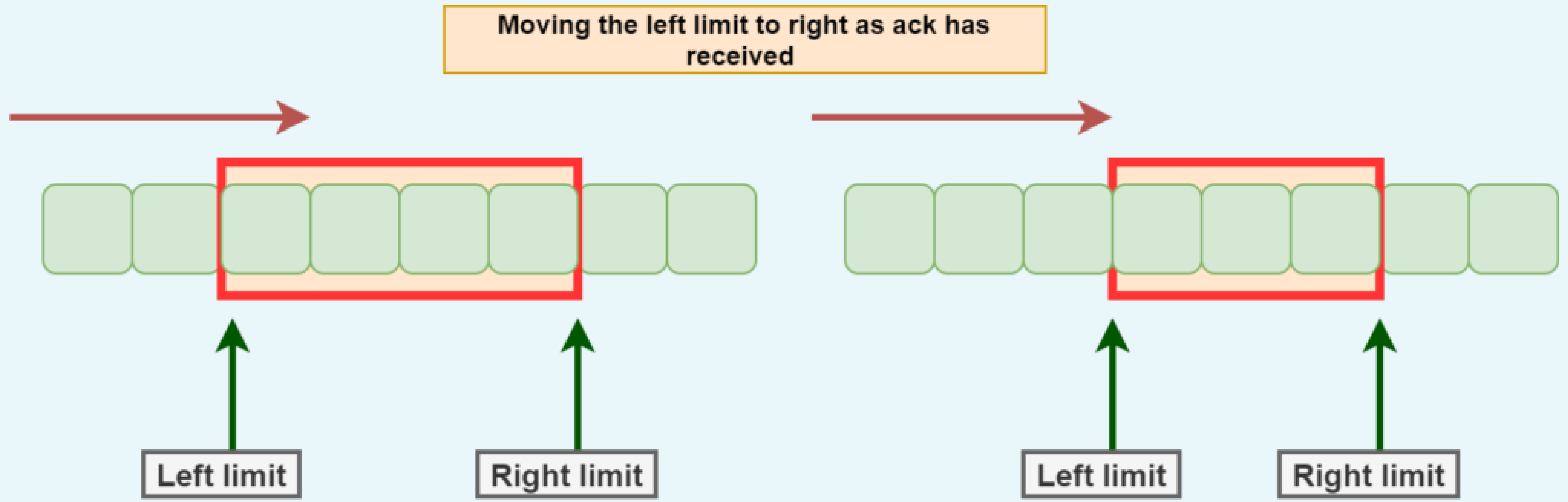
- For receiving window:
  - ✓ the positions inside its sliding window represent the sequence numbers of frame the receiver permitted to receive



- Maximum size of a sliding window represents the size of buffer for storing temporarily received frames before processing them
- Supposing that the receiver has a buffer of 4 frame sizes, then the maximum size of the sliding window will be 4.

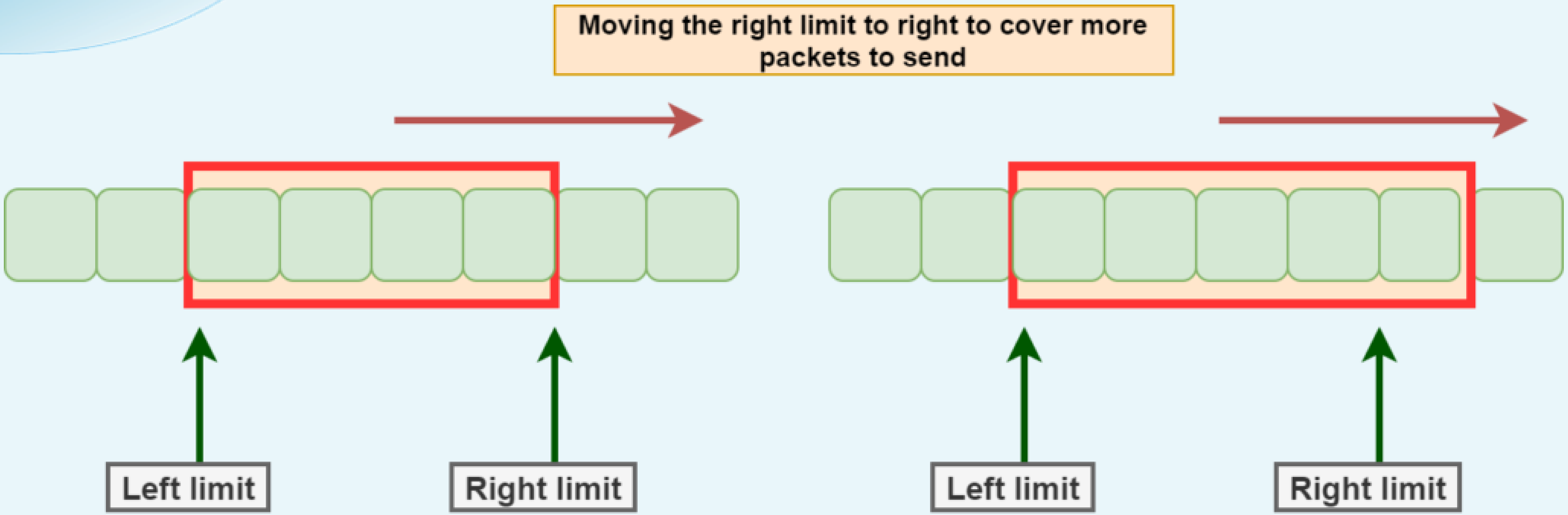
Error Control

Sliding Window Protocols



Error Control

Sliding Window Protocols



- The smallest size is 0
- The largest size is 2^k-1: k is number of bits for sequence number

Error Control

Sliding Window Protocols

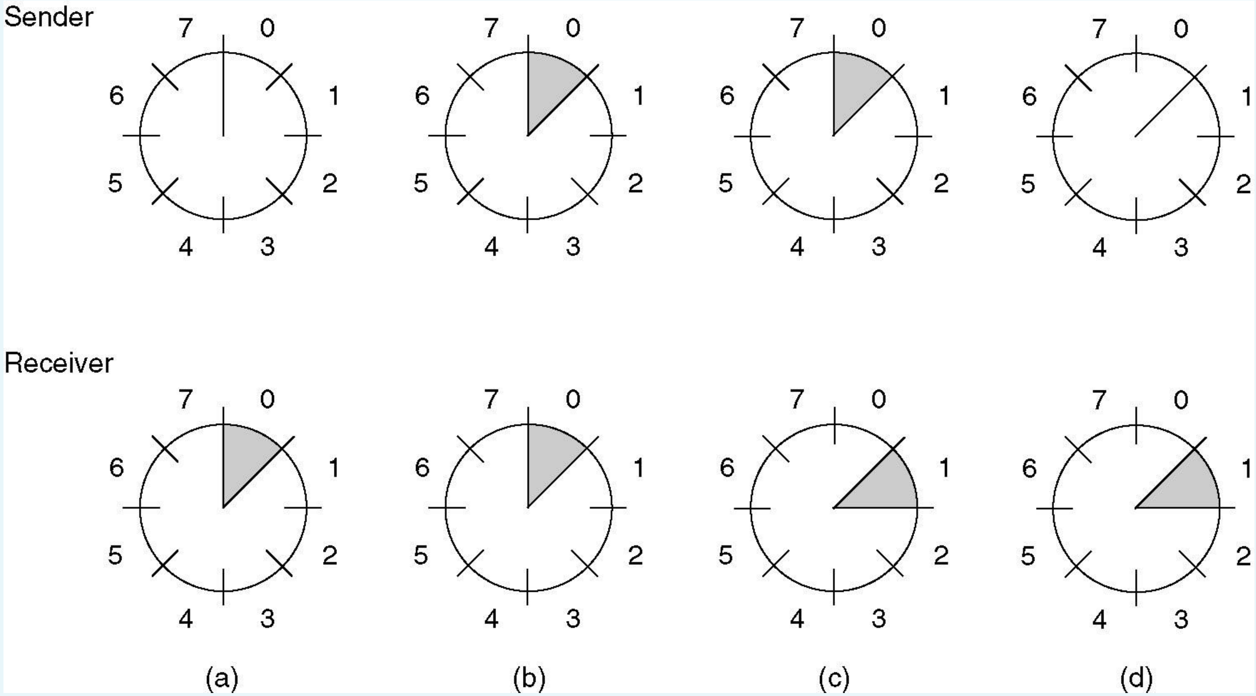
A sliding window of size 1, with a 3-bit sequence number.

(a) Initially.

(b) After the first frame has been sent.

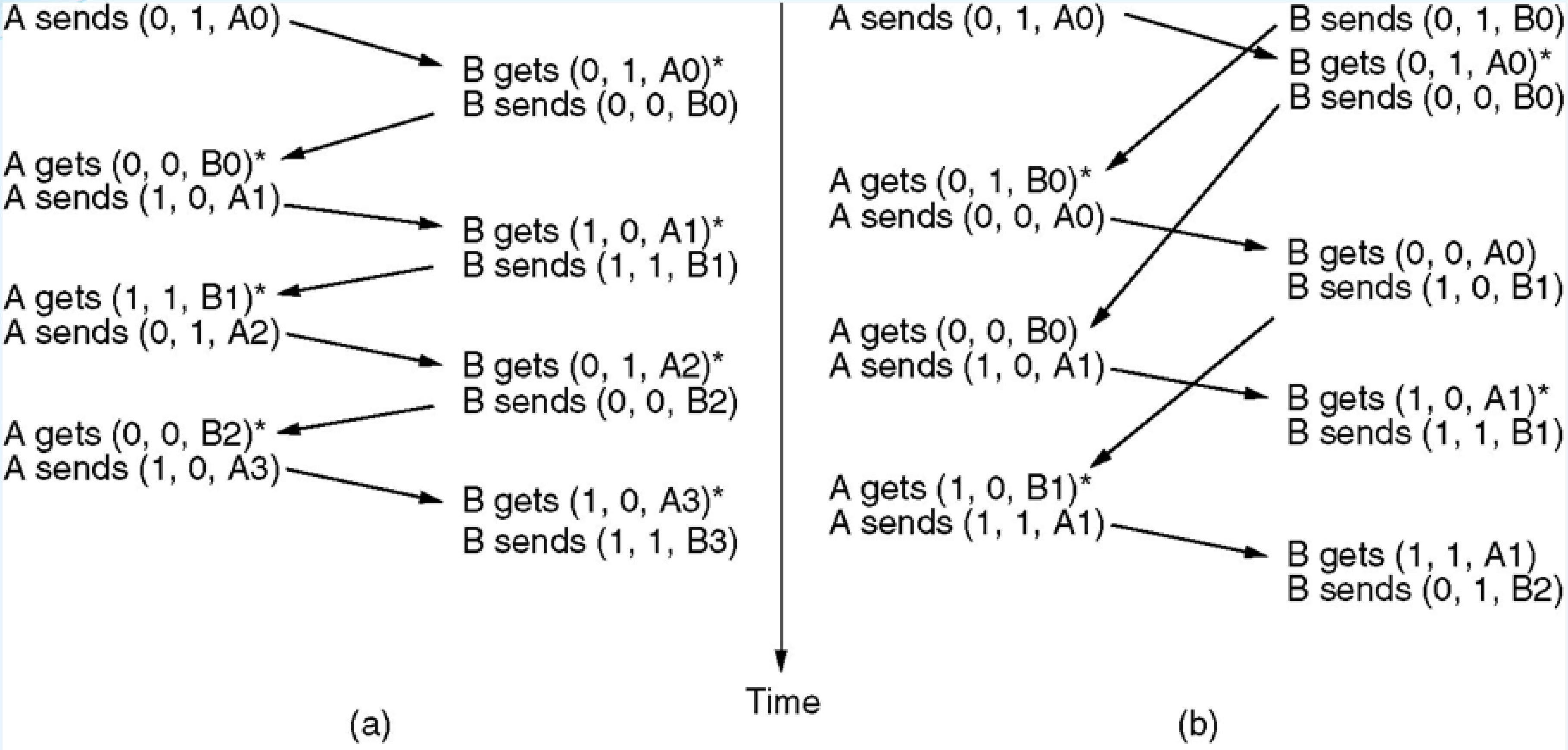
(c) After the first frame has been received.

(d) After the first acknowledgement has been received.



Error Control

Sliding Window Protocols

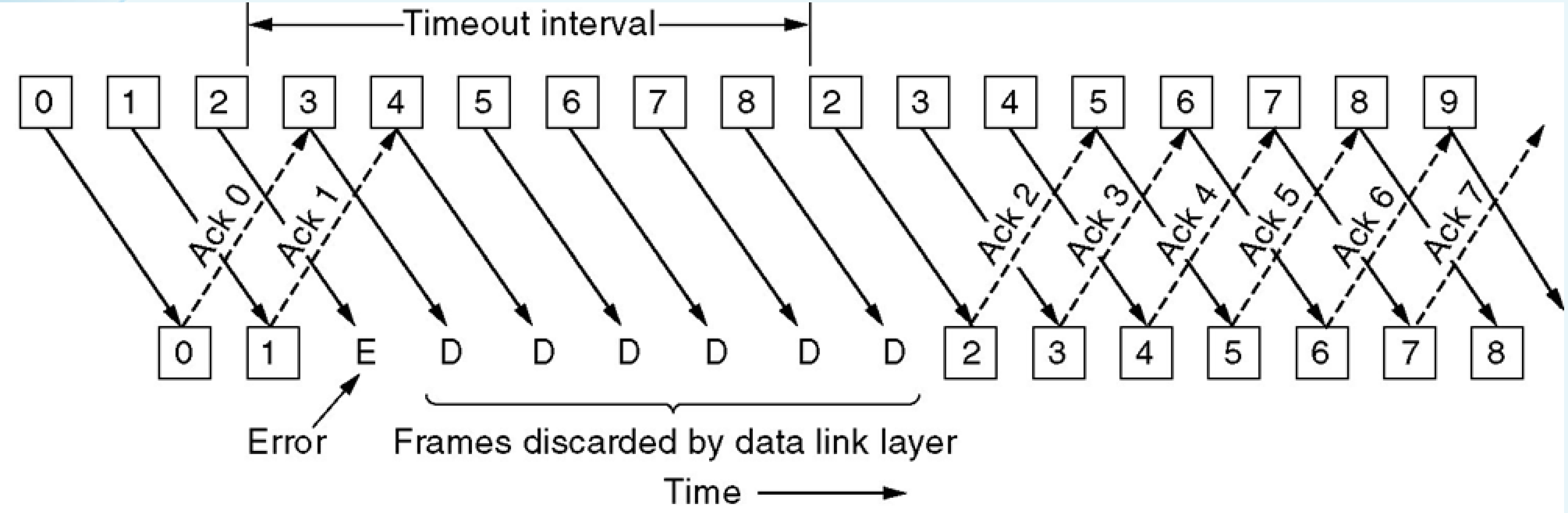


The notation is (seq, ack, packet number).

An asterisk indicates where a network layer accepts a packet.

Error Control

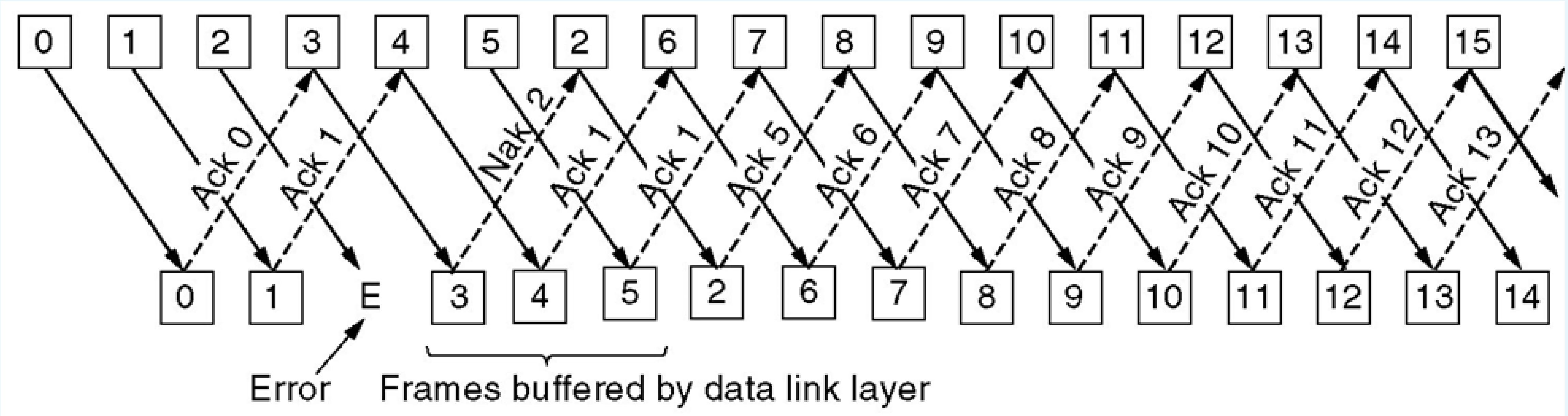
Sliding Window Protocols: Go Back N



- When a receiver receives an error frame it discards the frame.
- Because there is no acknowledgement frame for the error frame:
  - ✓ A time-out event for this error frame
  - ✓ The sender resends the error frame and all the following frames

Error Control

Sliding Window Protocols: Selective Repeat



Error Control

Sliding Window Protocols: Maximum sliding window size

- 3 bits sequence number
  - ✓ The frames numbered from 0-7
  - ✓ The maximum sliding window is 7

Sender

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Sent and waiting for Acknowledgement for frames 0,1,2,3,4,5,6

Receiver

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Ready receiving frames 0,1,2,3,4,5,6

Error Control

Sliding Window Protocols: Maximum sliding window size

- 3 bits sequence number
  - ✓ The frames numbered from 0-7
  - ✓ The maximum sliding window is 7

Sender

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Sent and waiting for Acknowledgement for frames 0,1,2,3,4,5,6

Receiver

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Frames 0,1,2,3,4,5,6 received
Checking Error

Error Control

Sliding Window Protocols: Maximum sliding window size

- 3 bits sequence number
  - ✓ The frames numbered from 0-7
  - ✓ The maximum sliding window is 7

Sender

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Sent and waiting for Acknowledgement for frames 0,1,2,3,4,5,6

1) Frames 0,1,2,3,4,5,6 error-free
2) Sending ACK for these frames
3) Move the receiving windows for accept new frames

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Receiver

Error Control

Sliding Window Protocols: Maximum sliding window size

- 3 bits sequence number
  - ✓ The frames numbered from 0-7
  - ✓ The maximum sliding window is 7

Sender

0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1

Sent and waiting for Acknowledgement for frames 0,1,2,3,4,5,6

Transmission error

X

ACK frame not reaches sender

1) Frames 0,1,2,3,4,5,6 error-free
2) Sending ACK for these frames
3) Move the receiving windows for accept new frames

0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1

Receiver

Error Control

Sliding Window Protocols: Maximum sliding window size

- 3 bits sequence number
  - ✓ The frames numbered from 0-7
  - ✓ The maximum sliding window is 7

Sender

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Time-out for frame 0

Resend frame 0

Waiting ACK for frames 0-6

Ready receiving Frames 7,0,1,2,3,4,5

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Receiver

Error Control

Sliding Window Protocols: Maximum sliding window size

- 3 bits sequence number
  - ✓ The frames numbered from 0-7
  - ✓ The maximum sliding window is 7

Sender

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Time-out for frame 0

Resend frame 0

Waiting ACK for frames 0-6

Ready receiving Frames 7,0,1,2,3,4,5

Frame 0 arrives:

- ✓ It is in the receive window
- ✓ Receive frame 0
- ✓ Duplicate frame 0

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Receiver

Error Control

Sliding Window Protocols: Maximum sliding window size

- The new receiving windows must not overlap the old
- Maximum size of sliding window just half of the frame sequence number
- Example:
  - If 3 bits used for frame sequence number then Maximum size of sliding window is $2^3/2=4$
  - If 4 bits used for frame sequence number then Maximum size of sliding window is $2^4/2=8$

Error Control

Sliding Window Protocols: Maximum sliding window size

- 3 bits sequence number
  - ✓ The frames numbered from 0-7
  - ✓ The maximum sliding window is 4

Sender

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Sent and waiting for Acknowledgement for frames 0,1,2,3

Receiver

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Ready receiving frames 0,1,2,3

Error Control

Sliding Window Protocols: Maximum sliding window size

- 3 bits sequence number
  - ✓ The frames numbered from 0-7
  - ✓ The maximum sliding window is 4

Sender

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Sent and waiting for Acknowledgement for frames 0,1,2,3

Receiver

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Frames 0,1,2,3 received

Checking Error

Error Control

Sliding Window Protocols: Maximum sliding window size

- 3 bits sequence number
  - ✓ The frames numbered from 0-7
  - ✓ The maximum sliding window is 4

1) Frames 0,1,2,3 error-free
2) Sending ACK for these frames
3) Move the receiving windows for accept new frames 4,5,6,7

Sender

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Sent and waiting for Acknowledgement for frames 0,1,2,3

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Receiver

Error Control

Sliding Window Protocols: Maximum sliding window size

- 3 bits sequence number
  - ✓ The frames numbered from 0-7
  - ✓ The maximum sliding window is 4

Sender

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Sent and waiting for Acknowledgement for frames 0,1,2,3

Transmission error

X

ACK frame not reaches sender

1) Frames 0,1,2,3 error-free
2) Sending ACK for these frames
3) Move the receiving windows for accept new frames 4,5,6,7

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Receiver

Error Control

Sliding Window Protocols: Maximum sliding window size

- 3 bits sequence number
  - ✓ The frames numbered from 0-7
  - ✓ The maximum sliding window is 4

Sender

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

Time-out for frame 0

Resend frame 0

Waiting ACK for frames 0-3

Ready receiving Frames 4,5,5,6

0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1

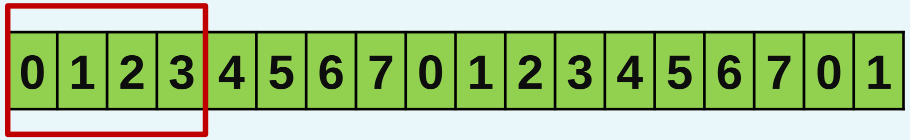
Receiver

Error Control

Sliding Window Protocols: Maximum sliding window size

- 3 bits sequence number
  - ✓ The frames numbered from 0-7
  - ✓ The maximum sliding window is

Sender



Ready receiving Frames 4,5,6,7

Frame 0 arrives:

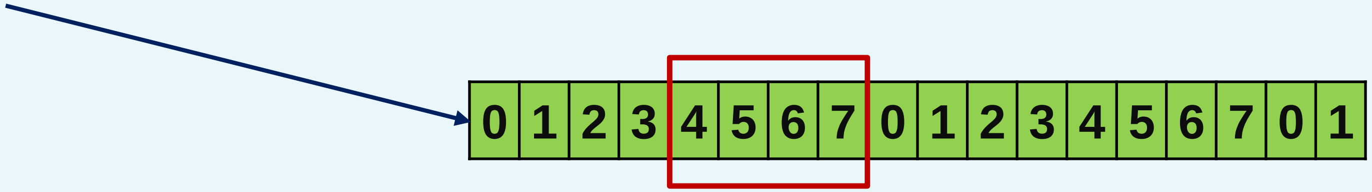
✓ It is not in the receive window

✓ Discard it

Time-out for frame 0

Resend frame 0

Waiting ACK for frames 0-3



Receiver

Error Control

Sliding Window Protocols: the size off the buffer

- Size of buffer is just equal to the maximum size of receiving window.
- Example: If 3 bits used for the sequence number of frames (from 0 to 7):
  - ✓ The maximum size of receiving window is $2^3/2 = 4$
  - ✓ The size of the buffer is also 4

Error Control

Sliding Window Protocols: Time to send acknowledgement frames

- Piggy-back: Attach acknowledgement into data frame
- In case the receiver has no data to send back to the sender
  - ✓ Start a timer for each received frame
  - ✓ Time-out event occurs, send an acknowledgement frame

Example Data Link Protocols

High Level Data Link Control

- A transmission protocol used at the data link layer (layer 2) of the OSI model
- HDLC protocol embeds information in a data frame that allows devices to control data flow and correct errors.
- HDLC is an ISO standard developed from the Synchronous Data Link Control (SDLC) standard proposed by IBM in the 1970’s.
- HDLC is a bit oriented protocol that supports both half-duplex and full-duplex communication over point to point & multipoint link.

Example Data Link Protocols

High Level Data Link Control: Station Types

- For any **HDLC communications session**, one station is designated primary and the other secondary
- Primary station
  - ✓ Control connection links
  - ✓ Send frames as commands
  - ✓ Maintain many logical links to secondary station
- Secondary station
  - ✓ Controlled by primary station
  - ✓ Send frames as responses to correspondent commands
- Combined station
  - ✓ Play both roles of Primary station and Secondary station
  - ✓ Can send commands or responses

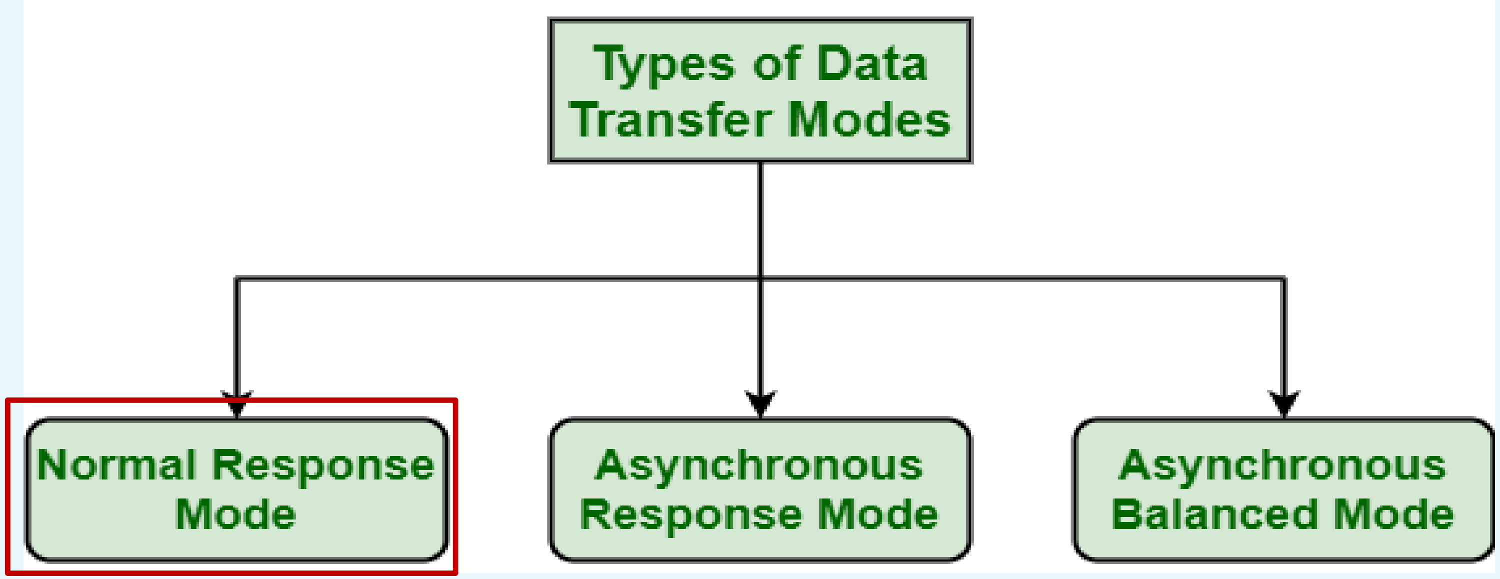
Example Data Link Protocols

High Level Data Link Control: Link Configuration

- Unbalanced configuration
  - ✓ One Primary station and one or many secondary stations
  - ✓ Support full duplex and half duplex
- Balanced configuration
  - ✓ Two combined stations
  - ✓ Support full duplex and half duplex

Example Data Link Protocols

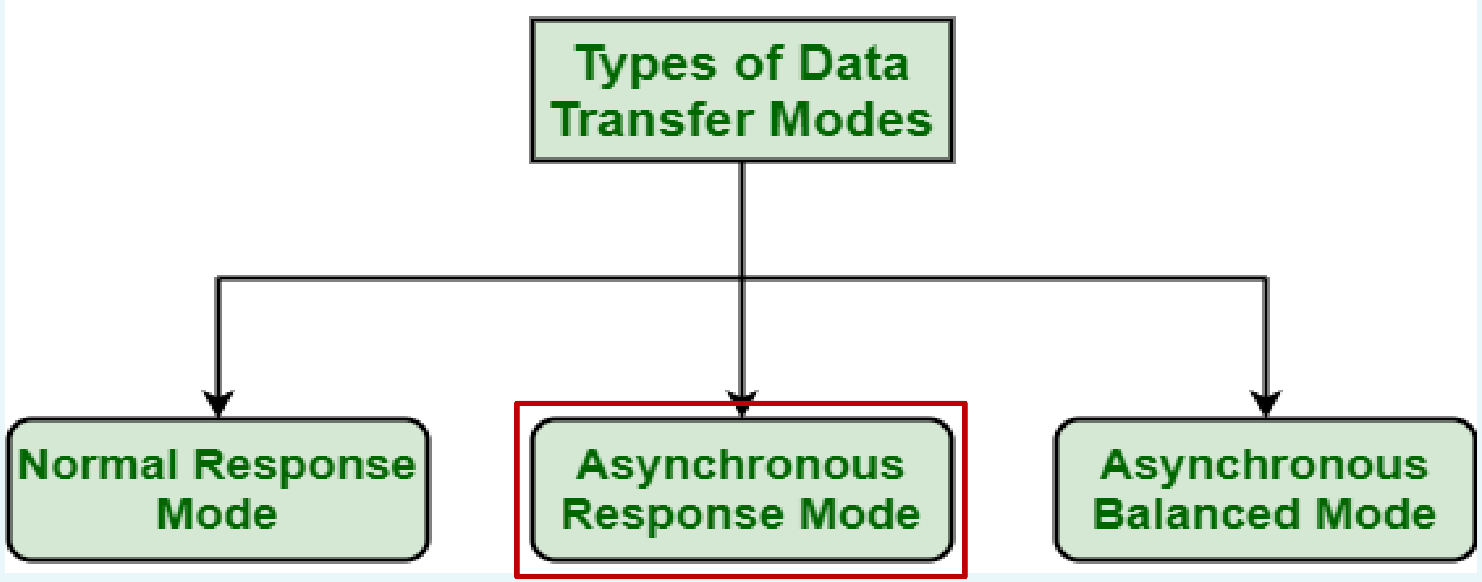
High Level Data Link Control: Transfer Modes



- Use unbalanced configuration
- Primary station initializes a data transmission to a secondary station
- A secondary station can only transfer data in form of the responses to the requests of primary station

Example Data Link Protocols

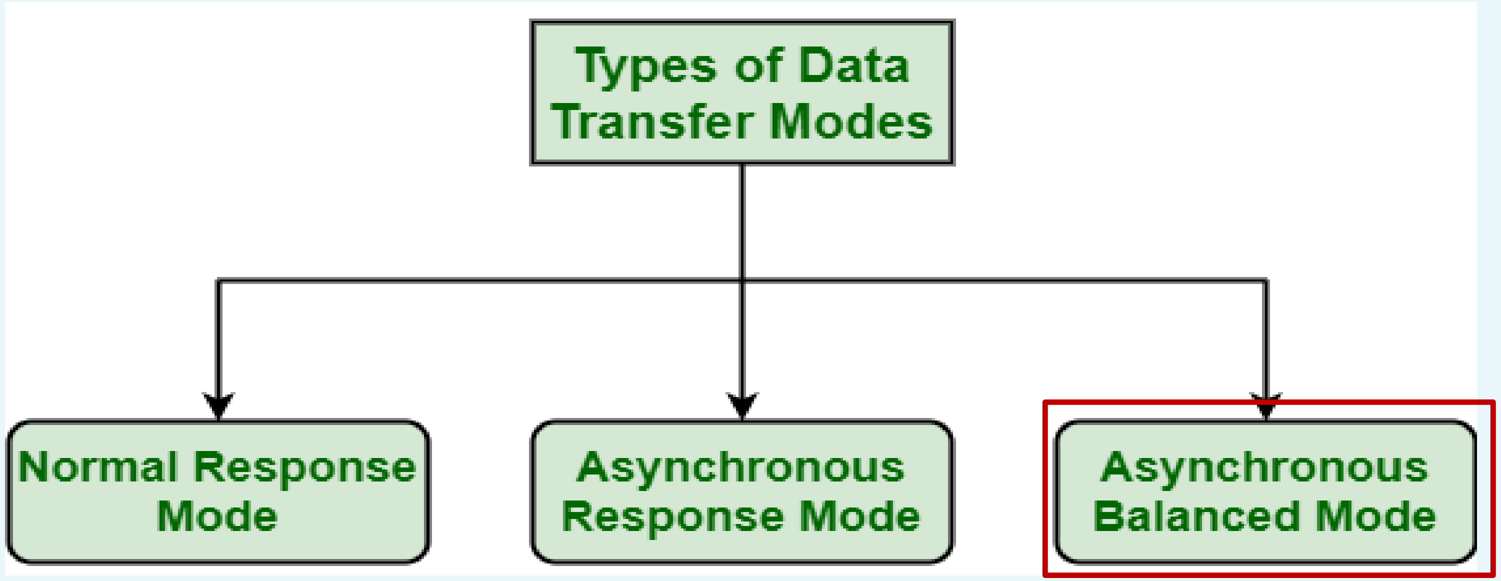
High Level Data Link Control: Transfer Modes



- Use balanced configuration
- All stations can initialize a data transmission without a permission from a primary station
- Used popularly

Example Data Link Protocols

High Level Data Link Control: Transfer Modes



- Use unbalanced configuration
- Secondary can initialize a data transmission without a permission from a primary
- Primary maintains connections
- Used rarely

Example Data Link Protocols

High Level Data Link Control: Frame structure

Bits	8	8	8	≥ 0	16	8
	0 1 1 1 1 1 1 0	Address	Control	Data	Checksum	0 1 1 1 1 1 1 0

- Synchronous transmission
- One frame structure for both data and control

Example Data Link Protocols

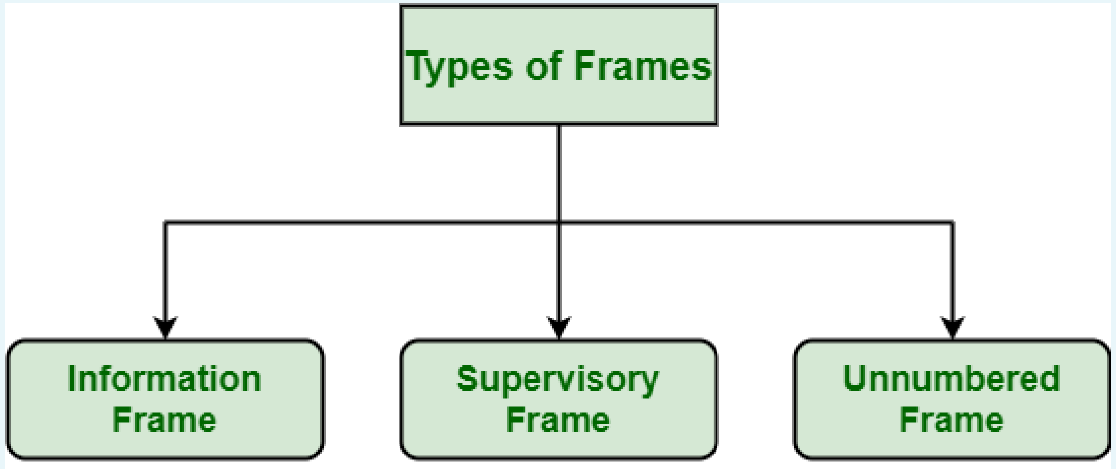
High Level Data Link Control: Frame structure

Bits	8	8	8	≥ 0	16	8
	0 1 1 1 1 1 1 0	Address	Control	Data	Checksum	0 1 1 1 1 1 1 0

- One frame structure for both data and control
- **Flag (8 bit):** 01111110 , uses bit stuffing technique
- **Address (8 bit):** Address of secondary station permitted to transfer or receive frames
- **Control (8bit):** Type of frames
- **Data (128-1024 bytes):** Data to transfer
- **FCS (Frame Check Sequence- 16 bit):** CRC-CCITT = $X^{16} + X^{12} + X^{5} +1$

Example Data Link Protocols

High Level Data Link Control: Types of Frames



	1	2	3	4	5	6	7	8
I: Information	0	N (S)			P/F	N (R)		
S: Supervisory	1	0	S		P/F	N (R)		
U: Unnumbered	1	1	M		P/F	M		

N (S): Send Sequence Number

S: Supervisory Function Bits

N (R): Receive Sequence Number

M: Unnumbered Function Bits

P/F: Poll/Final Bit

Example Data Link Protocols

High Level Data Link Control: Types of Frames

	1	2	3	4	5	6	7	8
I: Information	0	N (S)			P/F	N (R)		
S: Supervisory	1	0	S		P/F	N (R)		
U: Unnumbered	1	1	M		P/F	M		

N (S): Send Sequence Number

S: Supervisory Function Bits

N (R): Receive Sequence Number

M: Unnumbered Function Bits

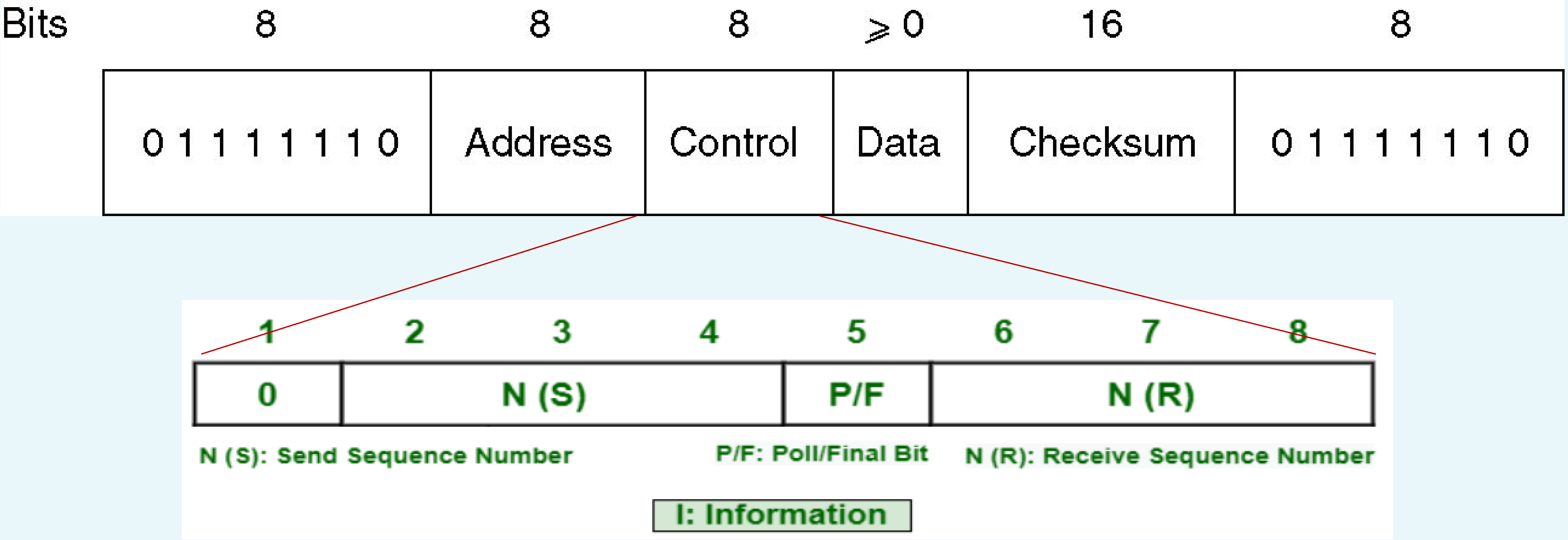
P/F: Poll/Final Bit

Poll/Final Bit:

- Frame sent by a primary station to secondary: Poll
- Frame sent by secondary to a primary: Final

Example Data Link Protocols

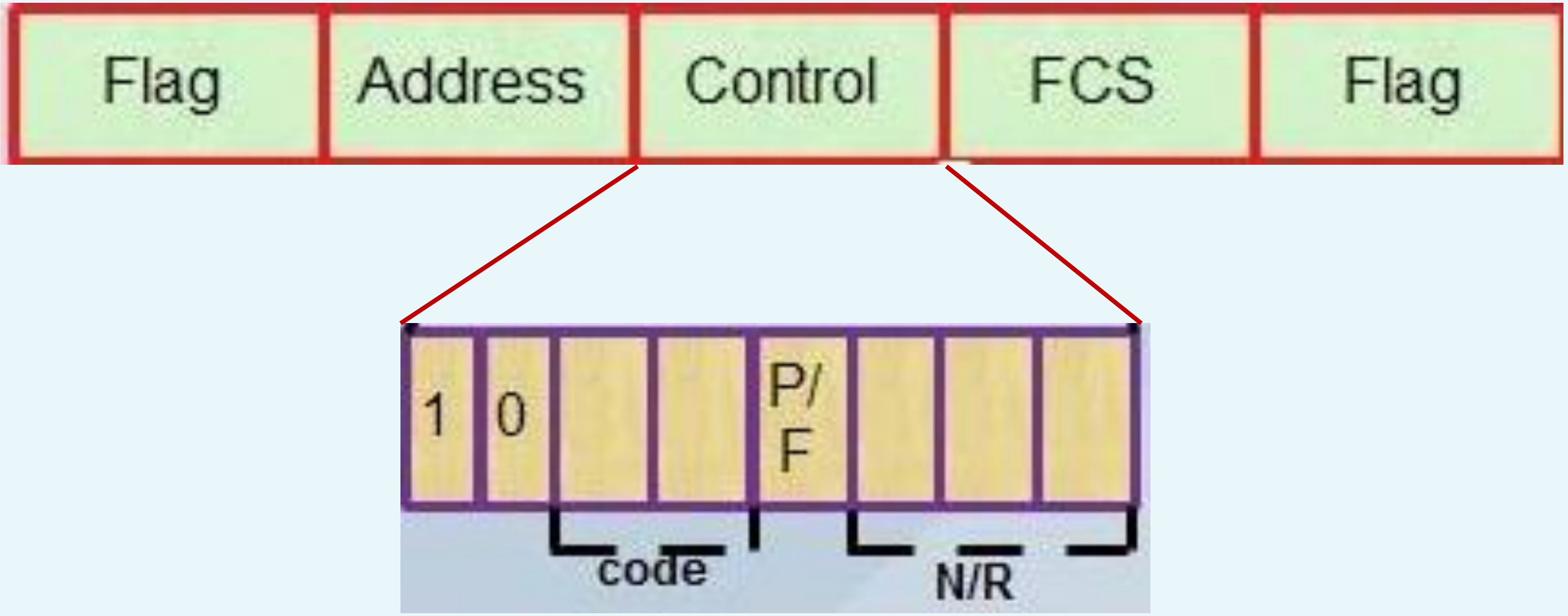
High Level Data Link Control: Types of Frames



- I-frames carry user's data and acknowledgement about user's data.

Example Data Link Protocols

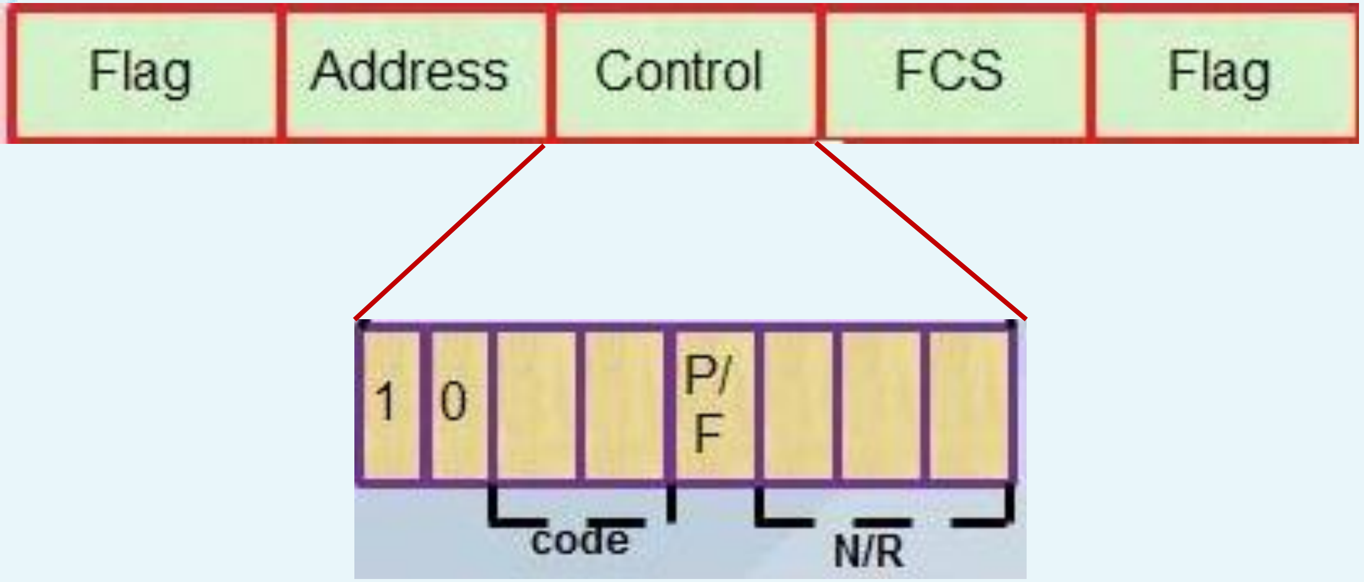
High Level Data Link Control: Types of Frames



- S-frame carries control information, primarily data link layer flow and error controls
- There is no N(S) field in control field of S-frame as S-frames do not transmit data.

Example Data Link Protocols

High Level Data Link Control: Types of Frames



Types of S-frame

Code	Command
00	RR Receive Ready
01	REJ Reject
10	RNR Receive Not Ready
11	SREJ Selective Reject

- RR: Acknowledge frames when no I-frames availab1e to piggyback the ACK
- REJ: Send a NAK when error has occurred.
- RNR: not ready to receive frame N(R)
- SREJ: request to retransmit the frame indicated in the N(R) subfield.

Example Data Link Protocols

High Level Data Link Control: Types of Frames

Bits	8	8	8	⩾ 0	16	8
	0 1 1 1 1 1 1 0	Address	Control	Data	Checksum	0 1 1 1 1 1 1 0

Used to establish connections

U: Unnumbered

1	2	3	4	5	6	7	8
1	1	M		P/F	M		

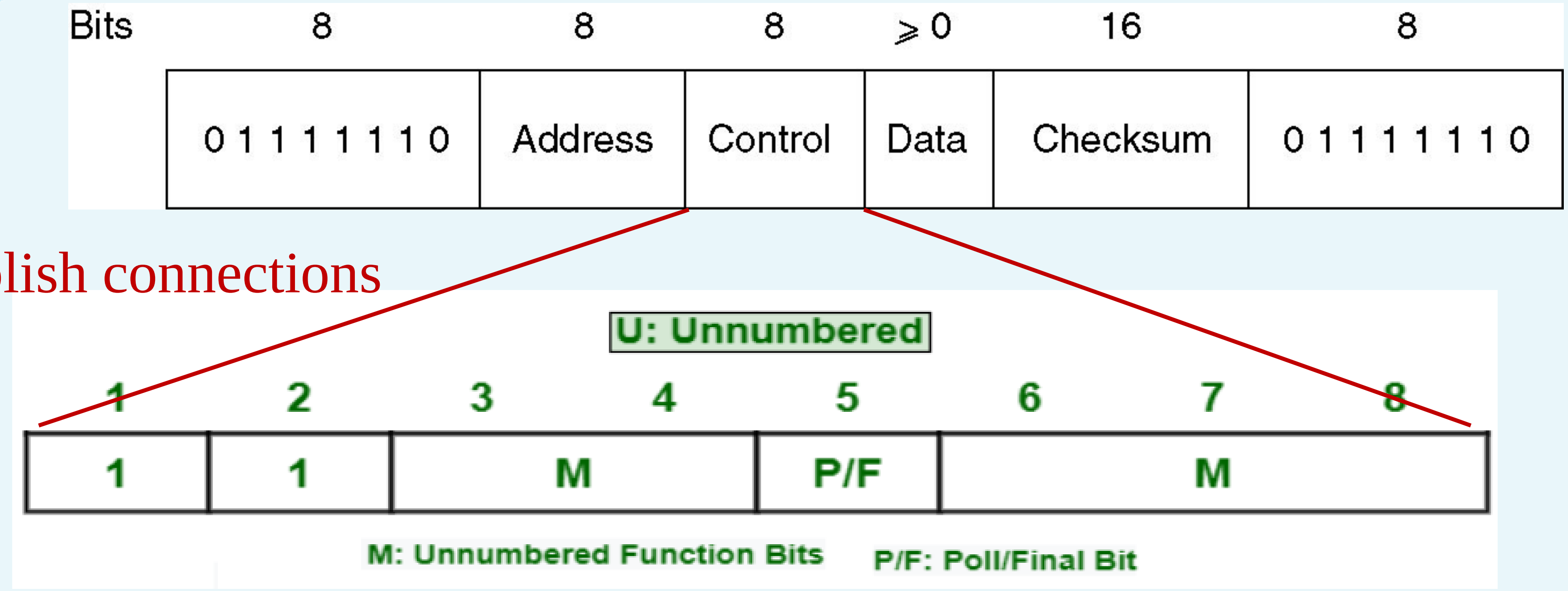
M: Unnumbered Function Bits P/F: Poll/Final Bit

- 1111P100: Setup connection in Balanced mode (Set Asynchronous Balanced Mode)
- 1100P001: setup connection in Normal response mode(Set Normal Response Mode)
- 1111P000: setup connection in Asynchronous response mode (Set Asynchronous Response Mode)

Example Data Link Protocols

High Level Data Link Control: Types of Frames

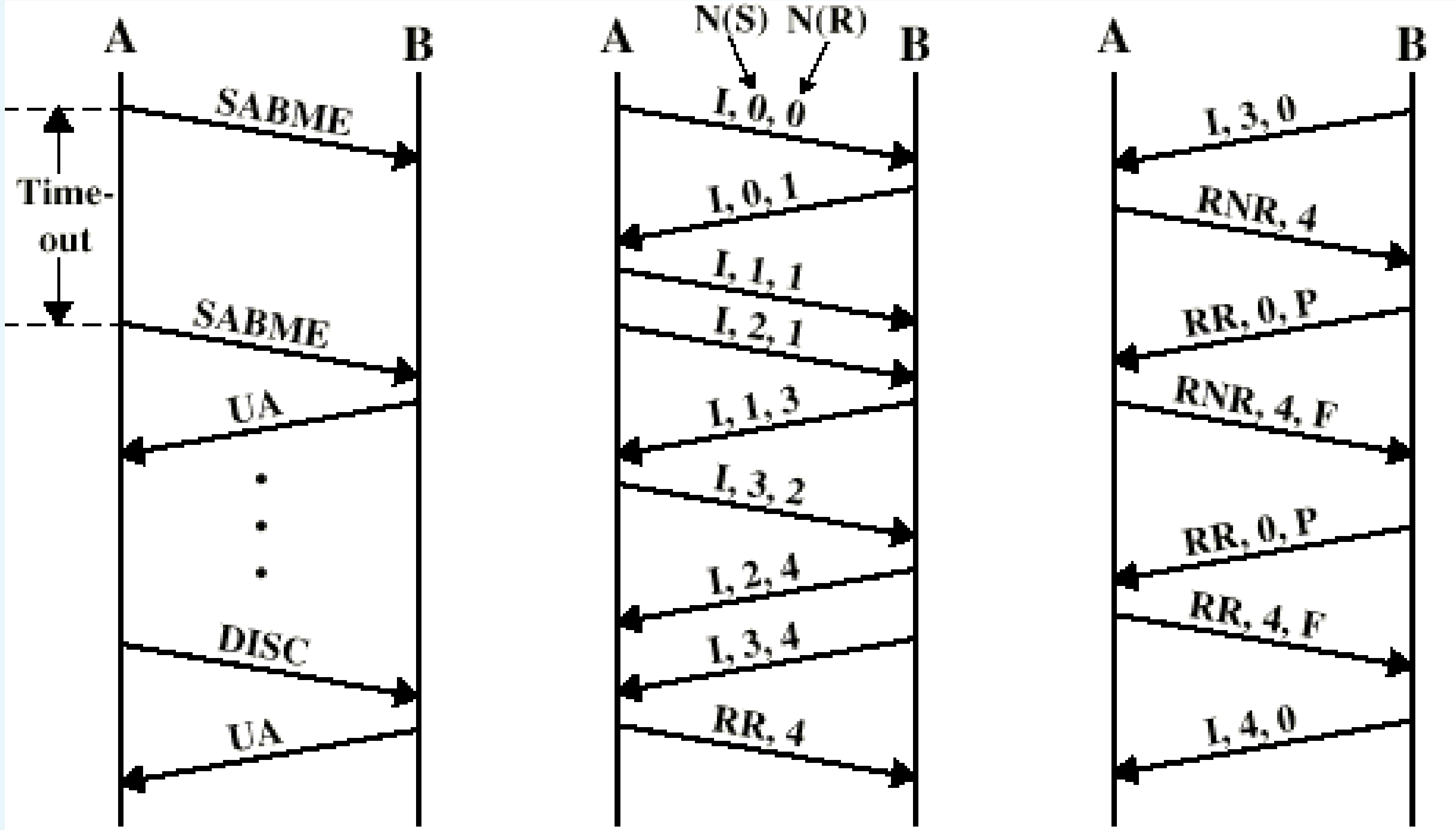
Used to establish connections



- 1100P010: requests to release a connection (DISC - Disconnect)
- 1100F110: Accepted all above U commands (UA - Unnumbered Acknowledgment)
- 1100F001: doesn't accept a well-received command (CMDR/FRMR - Command Reject/Frame Reject)

Example Data Link Protocols

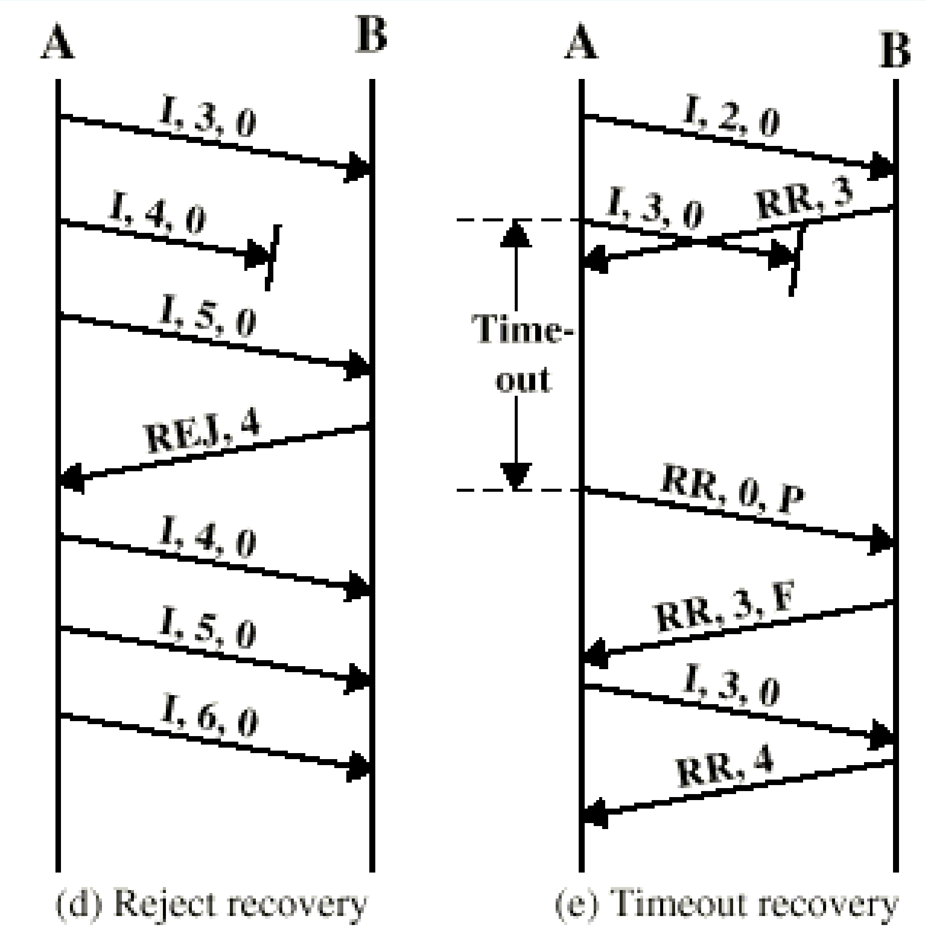
High Level Data Link Control: Scenarios



(a) Link setup and disconnect (b) Two-way data exchange (c) Busy condition

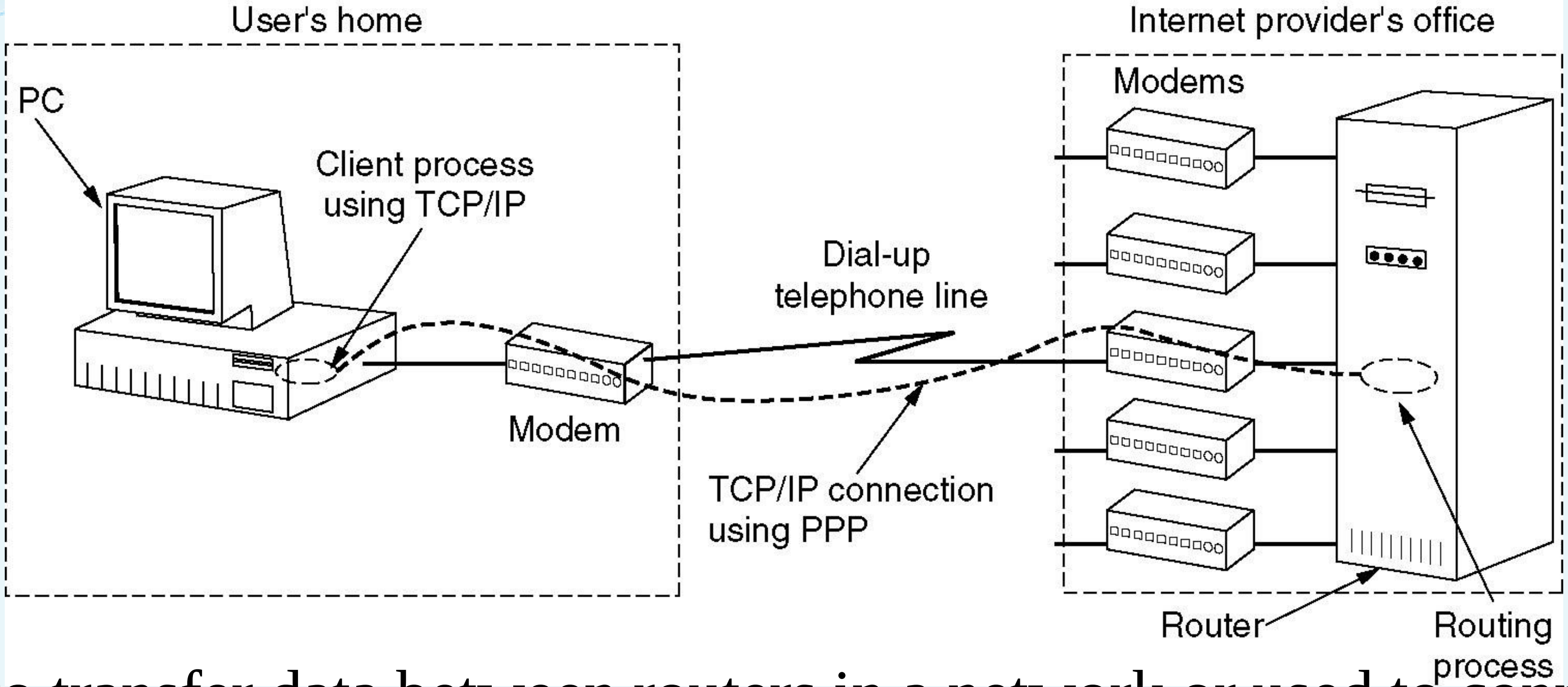
Example Data Link Protocols

High Level Data Link Control: Scenarios



Example Data Link Protocols

PPP – Point to Point Protocol

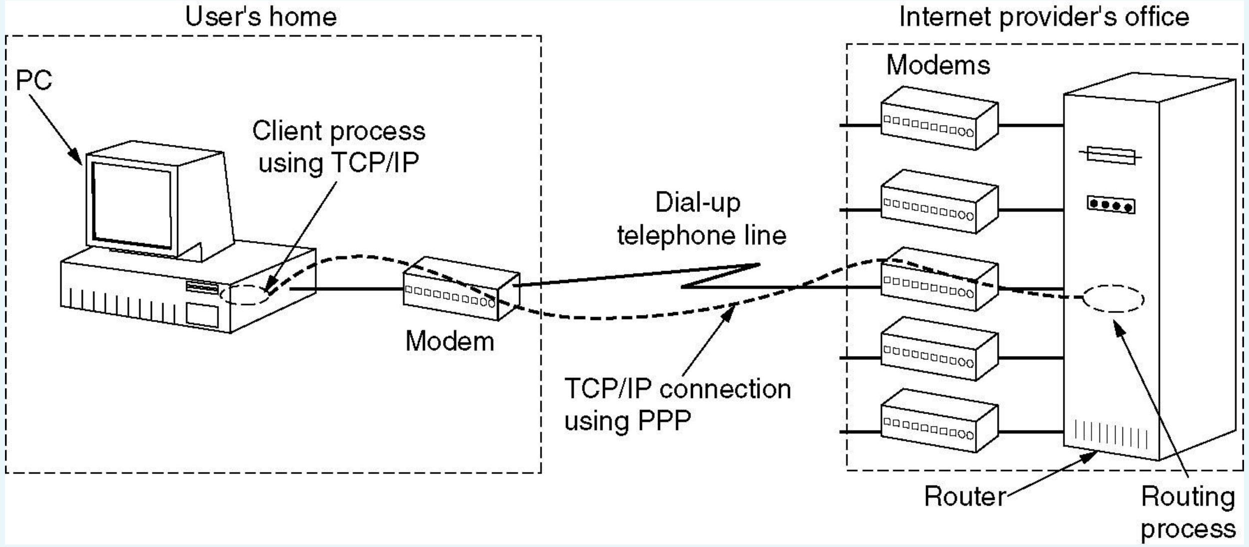


- Used to transfer data between routers in a network or used to connect user's home machine to a network of a Internet Service Provider (ISP)
- LCP (Link Control Protocol).
- NCP (Network Control Protocol): choose network protocol used by network layer

Example Data Link Protocols

PPP – Point to Point Protocol

- LCP (Link Control Protocol):
  - ✓ Establishing the link between two devices
  - ✓ Maintaining this established link
  - ✓ Configuring this link
  - ✓ Terminating this link after the transfer



- NCP (Network Control Protocol): negotiating the parameters and facilities for the network layer. Examples:
  - ✓ **Internet Protocol Control Protocol (IPCP)**: establishes and configures Internet Protocol (IP) over a PPP link.
  - ✓ **IPv6 Control Protocol (IPV6CP):** configures the IPv6 addresses

Example Data Link Protocols

PPP – Point to Point Protocol

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

- **Address field**: always 11111111 (the broadcast address *i.e.* all the stations accept this frame)
- **Control field**: Uses the format of the U-frame (unnumbered) in HDLC
- **Protocol field**: specify the kind of packet in the data field
- **Data field**: It carries user data or other information.
- **FCS field**: The frame checks sequence.