ERROR DETECTION

ECE 416 – DIGITAL COMMUNICATION SYSTEMS

Friday, 05 March 2021

SYLLABUS

ECE 416 – DIGITAL COMMUNICATION SYSTEMS (3 Units)

Pre-requisites: ECE 328 - Principles of Communication Systems

Course Purpose:
To enable students understand the fundamental principles of digital transmission systems as used in fixed and mobile telephony, wired and wireless computer networks, data storage and digital broadcasting.

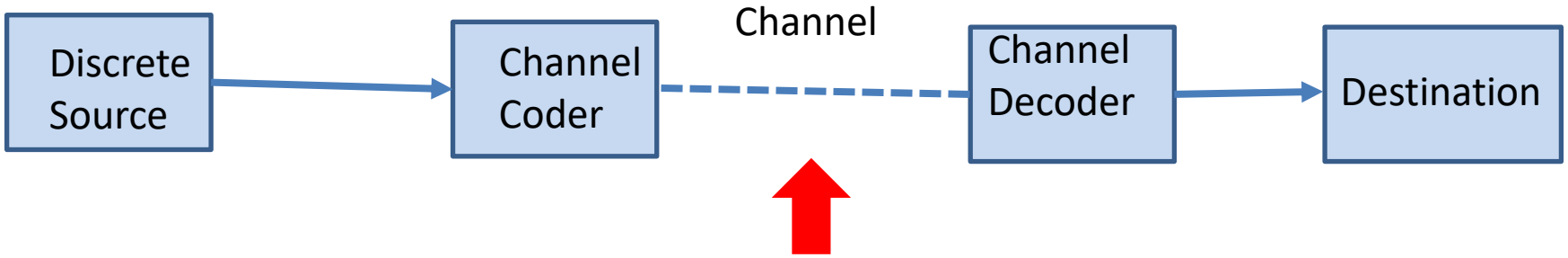
Expected Learning Outcomes:
At the end of the course, students will be able to:

(i) describe binary and duo binary pulse Amplitude Modulation (PAM);
(ii) design digital coding schemes;
(iii) derive error performance equations for digital modulation schemes(ASK,FSK,PSK,DPSK);
(iv) state strengths and weaknesses of M-ary PSK with QAM signaling schemes;
(v) design a basic digital communication systems.

Course Content:
Signal digitization: Pulse Amplitude Modulation (PAM), sampling theorems and sampling circuits, Pulse code modulation (PCM). Quantization and signal conditioning: Uniform and non-uniform quantization; companding methods; vocoders; signal-to- quantization noise ratio. Waveform coding: Pulse transmission, PCM, Pulse-shaping; Delta modulation; adaptive delta modulation; Differential Pulse Code Modulation (DPCM), M-ary encoding. Digital Modulation: Amplitude shift keying (ASK), Frequency Shift Keying (FSK), Phase Shift Keying (PSK), Quadrature Amplitude Modulation (QAM) and Differential Phase Shift Keying (DPSK). Signal recovery in ASK, FSK and PSK; Gaussian Minimum Shift Keying (GMSK); Performance comparison. Information theory: information sources, entropy, channel capacity; Source Coding; entropy coding. Error control: Error control coding techniques; Transmission errors; Error detection methods; intersymbol interference and the eye pattern; Linear block codes; Cyclic codes; convolution codes. Multiplexing: Frequency division multiplex (FDM), Time Division Multiplexing (TDM), plesiochronous digital hierarchy (PDH). Spread spectrum communication: Direct sequence and frequency hopping methods; synchronization, spreading codes and their generation. Data transmission: Local data transmission protocals (Ethernet, token ring); Modems; high Asymmetric Digital subscriber line (ADSL); Very–high Speed Digital subscriber line (VDSL), integrated services digital network (ISDN).

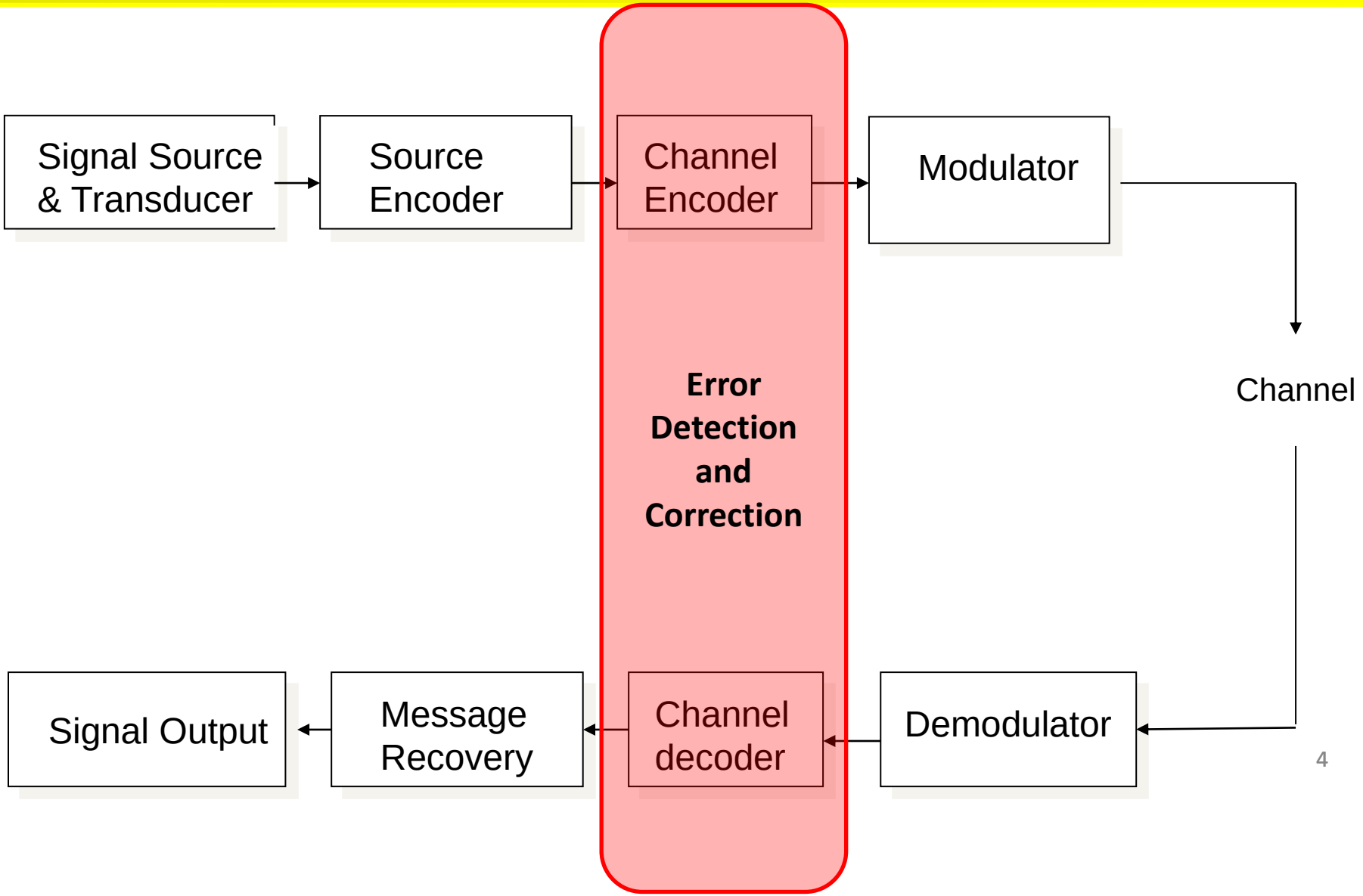
GENERATION AND DETECTION OF CODED SIGNALS

Error detection and correction or error control are techniques that enable reliable delivery of digital data over unreliable communication channels.



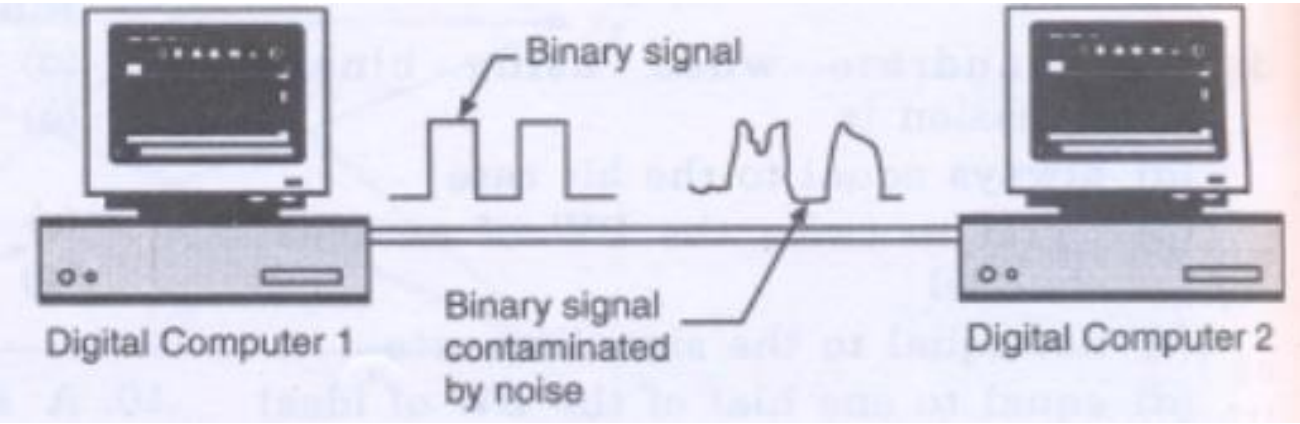
Noise/Interference:
leading to alteration and loss of blocks of data

ERROR DETECTION & CORRECTION IN A DIGITAL COMMUNICATION SYSTEM



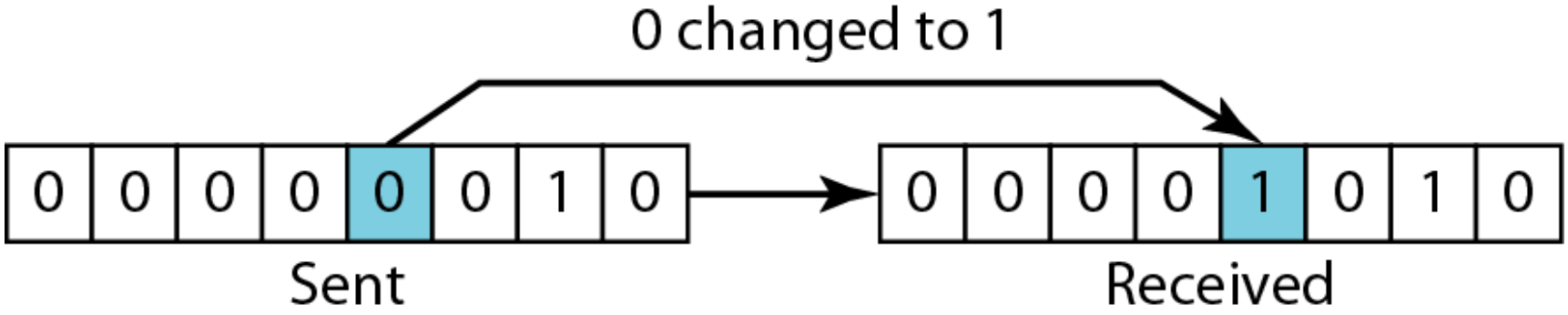
CLASSIFICATION OF ERROR CONTROL CODES

- Error control codes can be divided into two categories:
  1. **Error detecting codes:** Only detect errors but cannot correct
  2. **Error Correcting codes:** Can detect and correcting errors

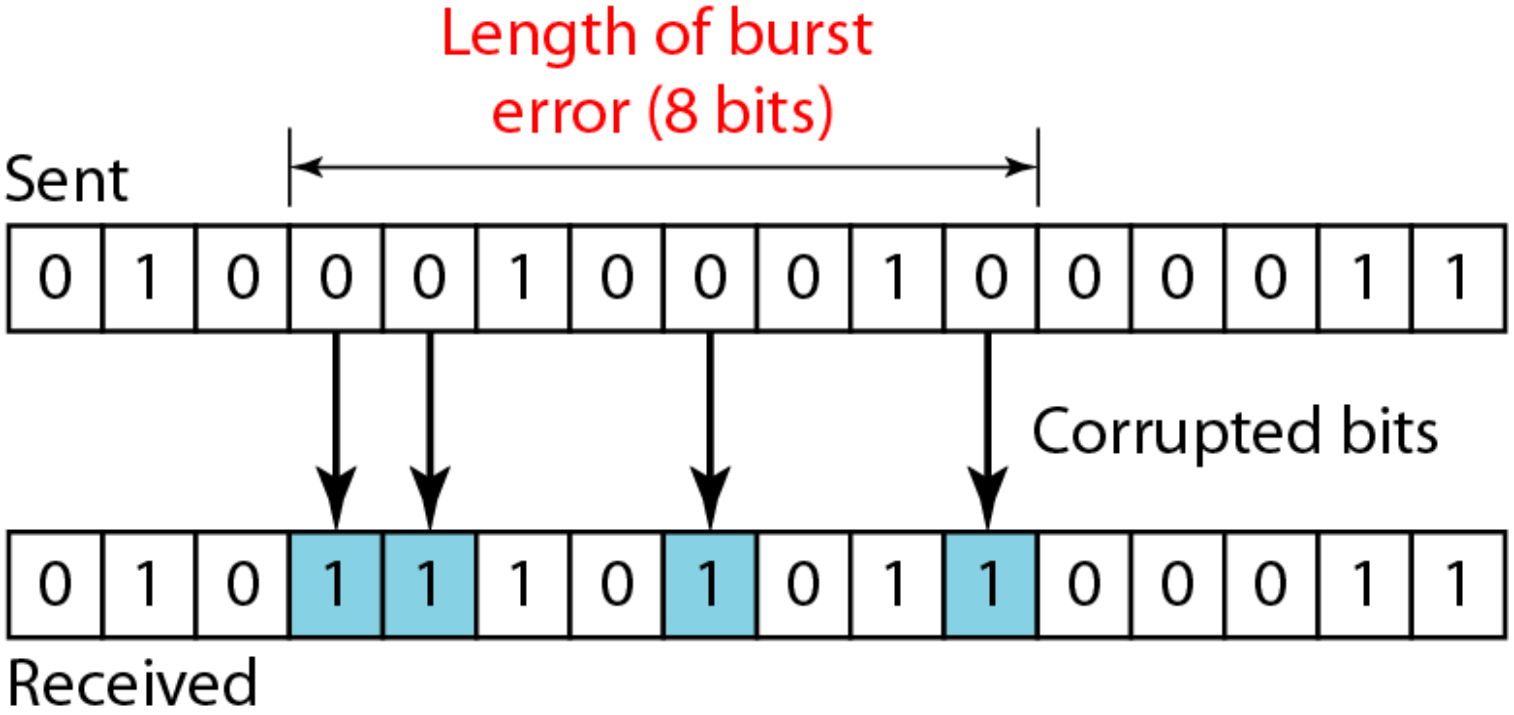


CLASSIFICATION OF ERRORS/01

(a) Single-bit error

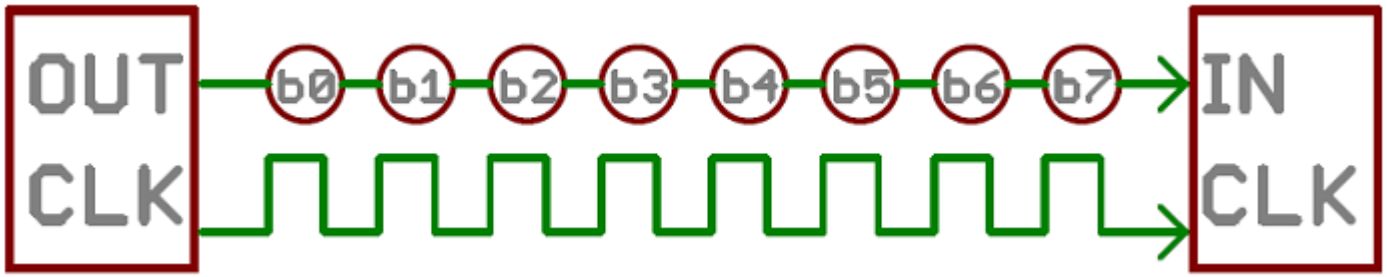


(b) Multiple bit /burst Error



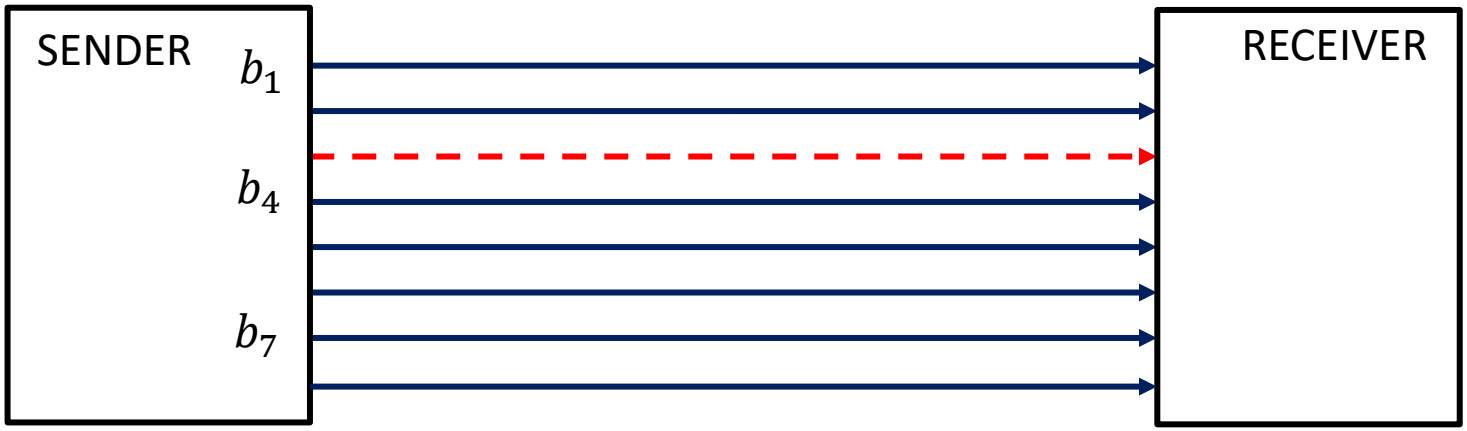
SINGLE-BIT ERRORS IN SERIAL COMMUNICATION

- Single-bit errors are the least likely type of error in serial data transmission.
- **Illustration**
- Suppose a sender sends data at 1 Mbps.
- This means that the bit duration is 1 micro sec.
- For a single bit error to occur the noise must have a duration of 1 micro sec which is very rare.



SINGLE-BIT ERRORS IN PARALLEL TRANSMISSION

- Single bit errors can occur if we are sending data using parallel transmission.
- **Illustration**
- Suppose 8 wires are used to send all 8 bits of 1 byte at the same time
- One of the wires is noisy, one bit can be corrupted in each byte.



BURST ERRORS

- The term burst error means that two or more bits in the data unit have changed.
- The burst error does not necessarily mean that errors occur in consecutive bits.
- The length of a burst is measured from the first corrupted bit to the last corrupted bit.
- Some bits in between may not be corrupted.

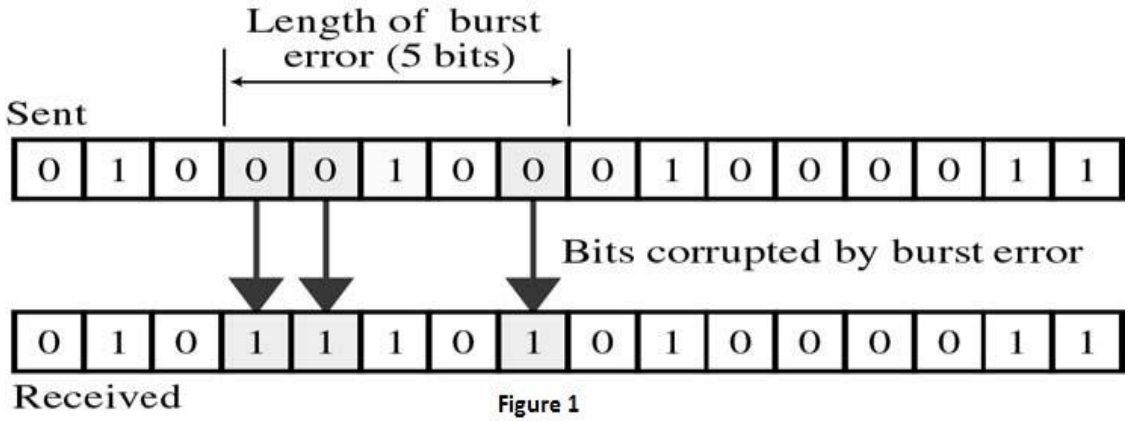
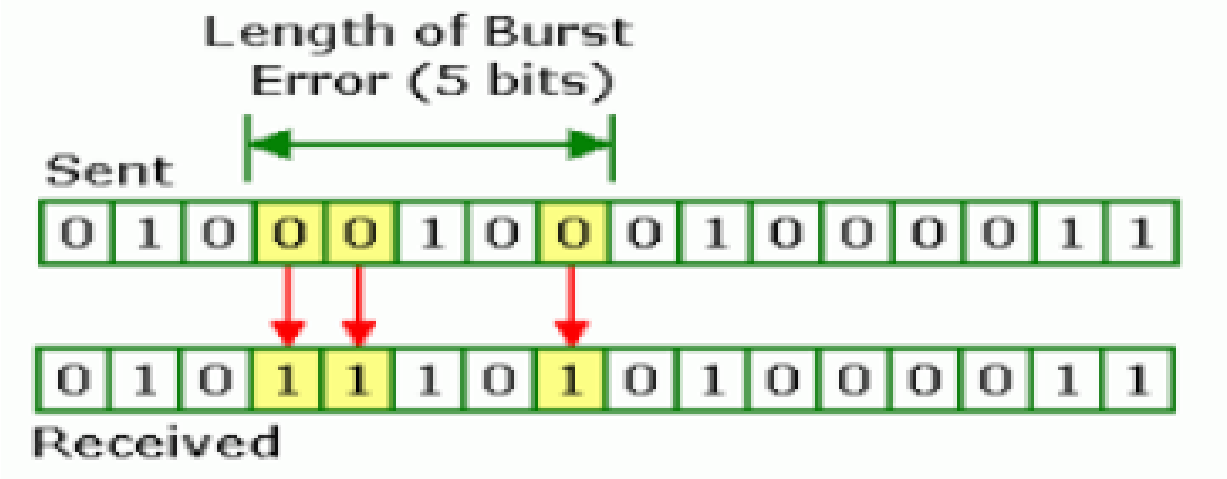


Figure 1

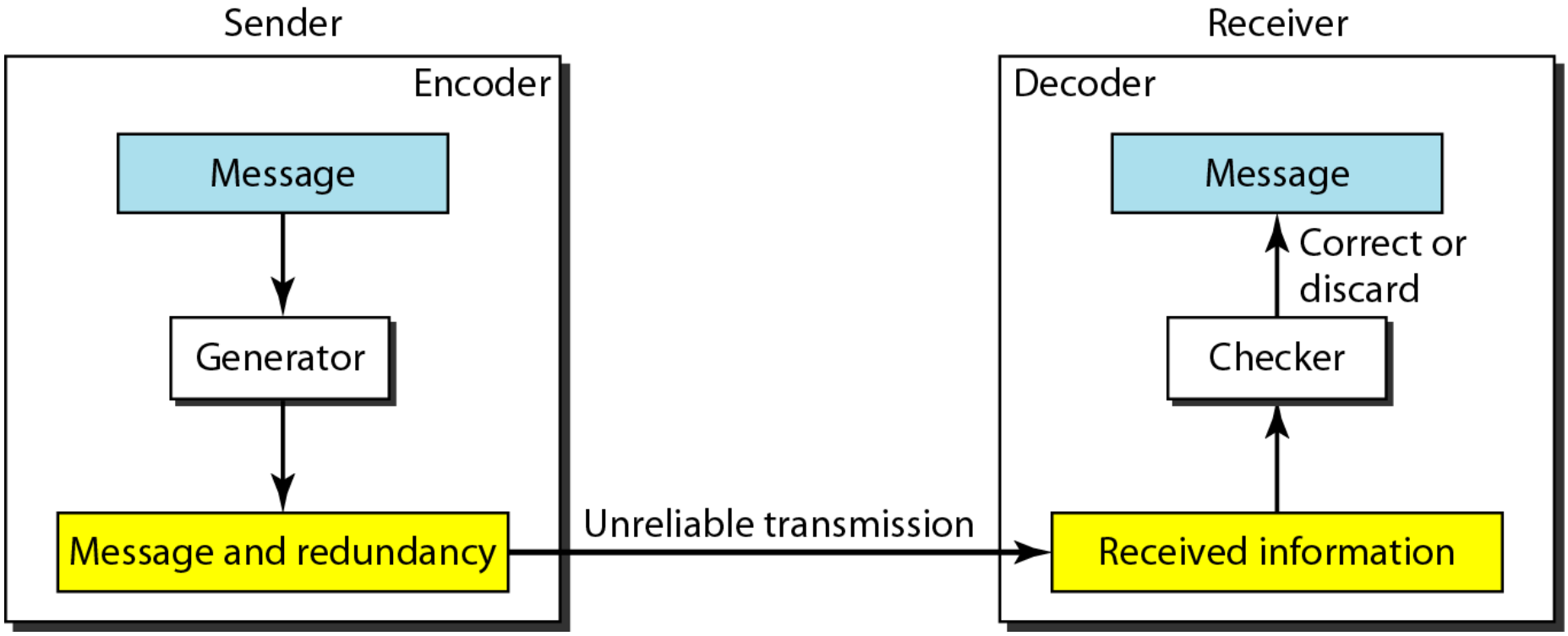
BURST ERRORS IN SERIAL COMMUNICATION

- Burst error is most likely to occur in serial transmission.
- The duration of noise is normally longer than the duration of one bit.
- Which means that when noise affects data, it affects a set of bits.
- The number of bits affected depends on the data rate and duration of noise.



ERROR DETECTION & CORRECTION

To detect or correct errors, we need to send the message data together with redundant data.



ERROR DETECTION AND CORRECTION

1. The most Common Error Detection Methods are:
   (a) Parity Checking
   (b) Checksum error detection
   (c) Cyclic Error Check (CRC)

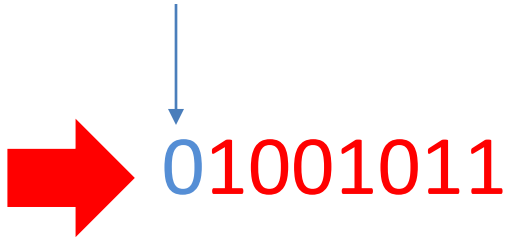
PARITY CHECKING

1. Parity checking is the simplest form of error detecting code.
2. Parity checking methods introduce an additional bit, called a parity bit which is added to each data word.
3. The additional bit is chosen so as to make the resulting word have even or odd parity.
   (a) Even parity is when the number of '1' bits in the word is even
   (b) Odd parity is when the number of '1' bits is odd.

EVEN PARITY

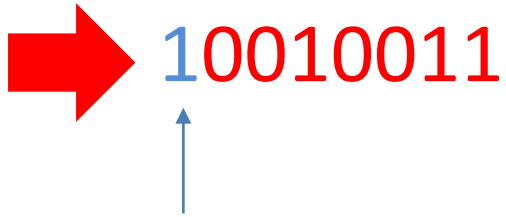
1. The number of ‘1’ bits in the data word is 4, i.e Even, therefore Parity bit is set to 0.

PARITY BIT	DATA WORD
0	1001011



2. The number of ‘1’ bits in the data word is 3, i.e Odd, therefore Parity bit is set to 1.

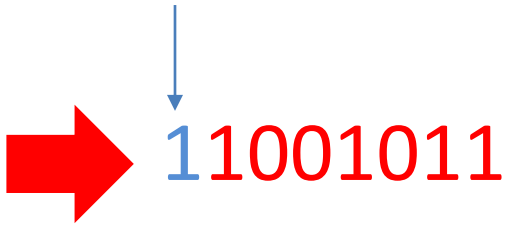
PARITY BIT	DATA WORD
1	0010011



ODD PARITY

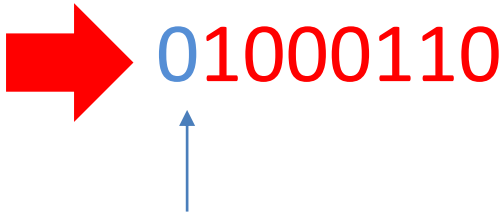
1. The number of ‘1’ bits in the data word is 4, i.e even, therefore Parity bit is set to 1.

PARITY BIT	DATA WORD
1	1001011



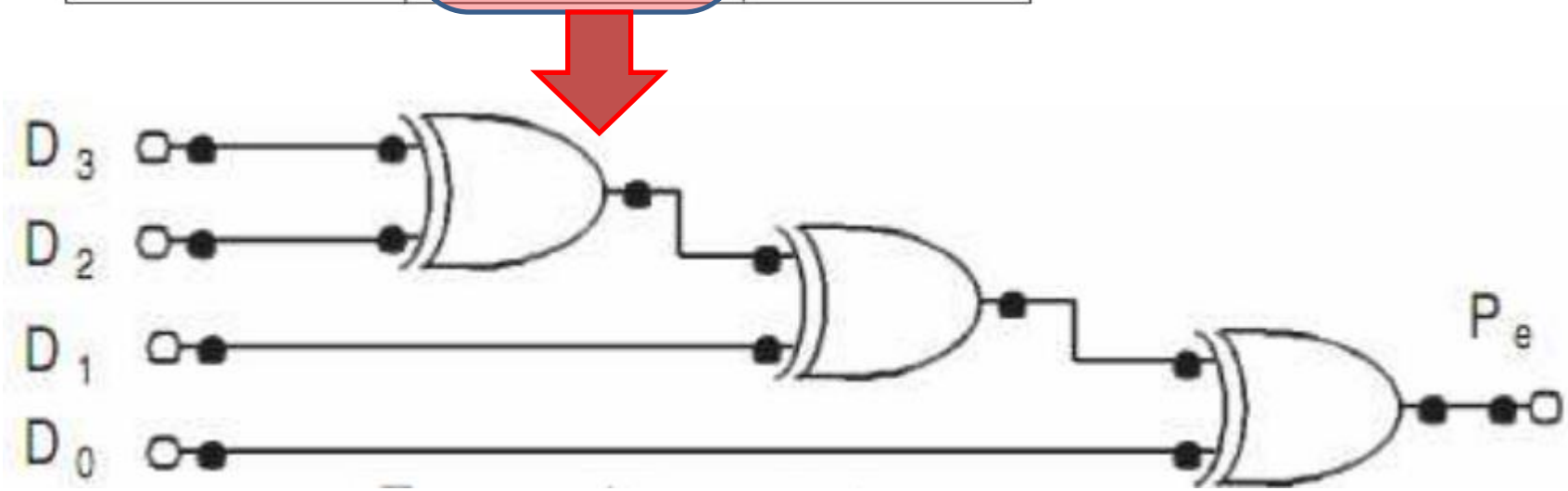
2. The number of ‘1’ bits in the data word is 3, i.e Odd, therefore Parity bit is set to 0.

PARITY BIT	DATA WORD
0	1000110



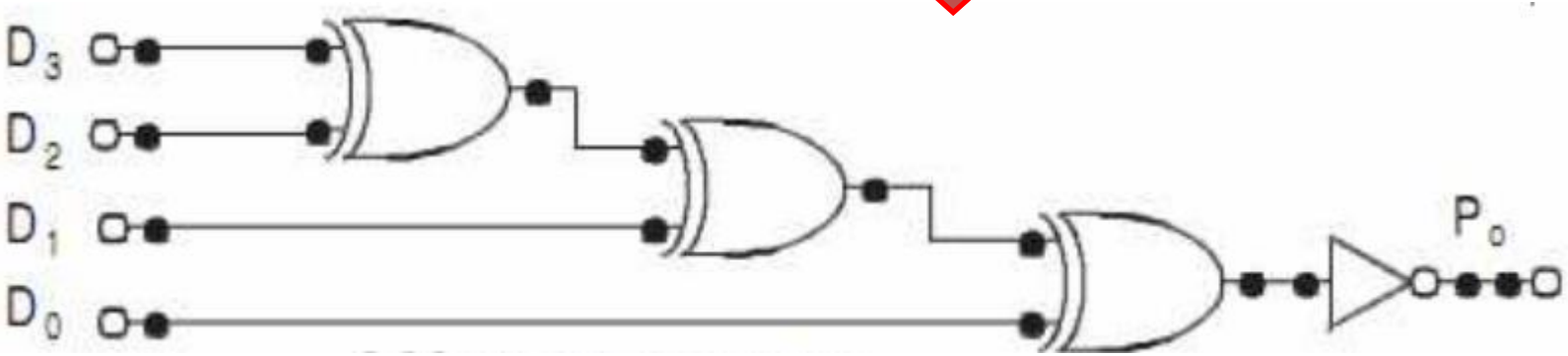
EVEN PARITY GENERATOR

Four bit Message $D_3D_2D_1D_0$	*Even Parity* (P_e)	*Odd Parity* (P_o)
0000	0	1
0001	1	0
0010	1	0
0011	0	1
0100	1	0
0101	0	1
0110	0	1
0111	1	0
1000	1	0
1001	0	1
1010	0	1
1011	1	0
1100	0	1
1101	1	0
1110	1	0
1111	0	1



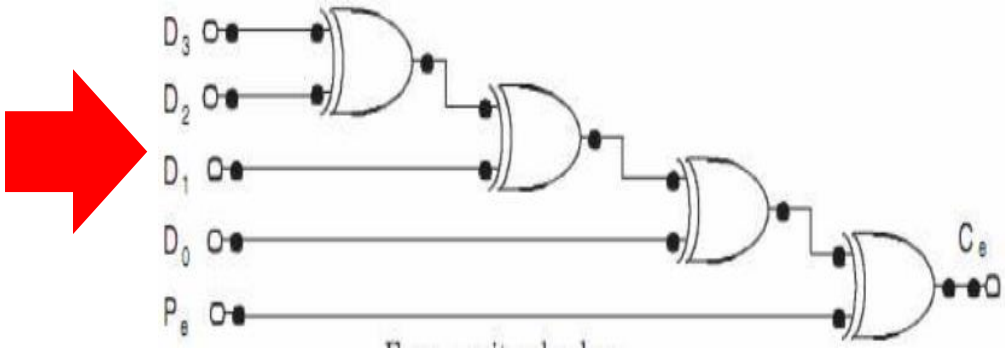
ODD PARITY GENERATOR

Four bit Message $D_3D_2D_1D_0$	*Even Parity* (P_e)	*Odd Parity* (P_o)
0000	0	1
0001	1	0
0010	1	0
0011	0	1
0100	1	0
0101	0	1
0110	0	1
0111	1	0
1000	1	0
1001	0	1
1010	0	1
1011	1	0
1100	0	1
1101	1	0
1110	1	0
1111	0	1



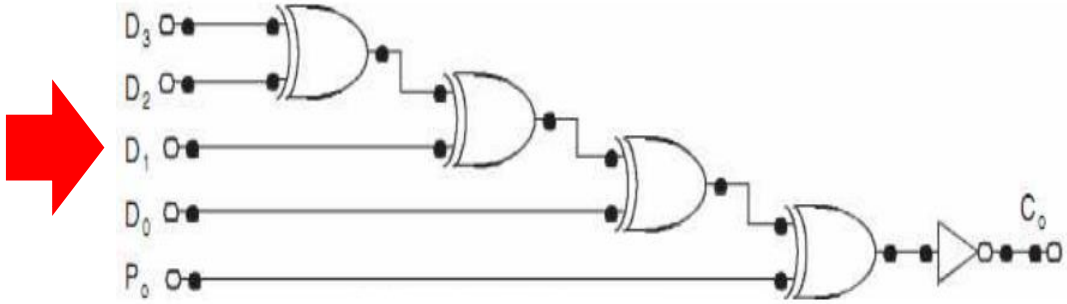
EVEN PARITY CHECKER

4-bit message $D_3D_2D_1D_0$	Even Parity (P_e)	Even Parity Checker (C_e)
0000	0	0
0001	1	0
0010	1	0
0011	0	0
0100	1	0
0101	0	0
0110	0	0
0111	1	0
1000	1	0
1001	0	0
1010	0	0
1011	1	0
1100	0	0
1101	1	0
1110	1	0
1111	0	0



ODD PARITY CHECKER

4-bit message $D_3D_2D_1D_0$	Odd Parity (P_0)	Odd Parity Checker (C_o)
0000	1	0
0001	0	0
0010	0	0
0011	1	0
0100	0	0
0101	1	0
0110	1	0
0111	0	0
1000	0	0
1001	1	0
1010	1	0
1011	0	0
1100	1	0
1101	0	0
1110	0	0
1111	1	0

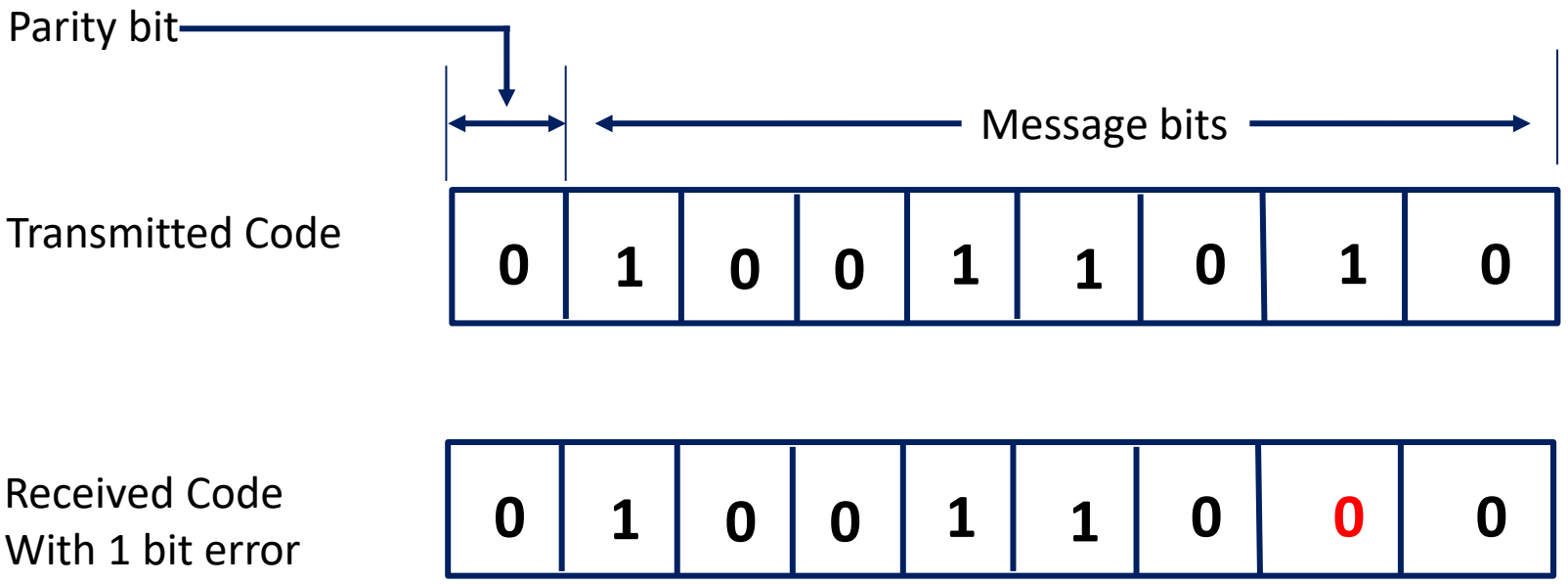


USE OF PARITY IN PCs

1. **Parity checking** is a rudimentary method of detecting simple, single-bit errors in a computer RAM.
2. It has been present in PCs since the original IBM PC in 1981.
3. It requires the use of **parity memory**, which provides an extra bit for every byte stored.
4. **When parity checking is enabled**, each time a byte is written to memory, a logic circuit called a parity generator/checker examines the byte and determines whether the data byte had an even or an odd number of ones.
5. **If it had an even number of ones**, the ninth (parity) bit is set to a one, otherwise it is set to a zero.

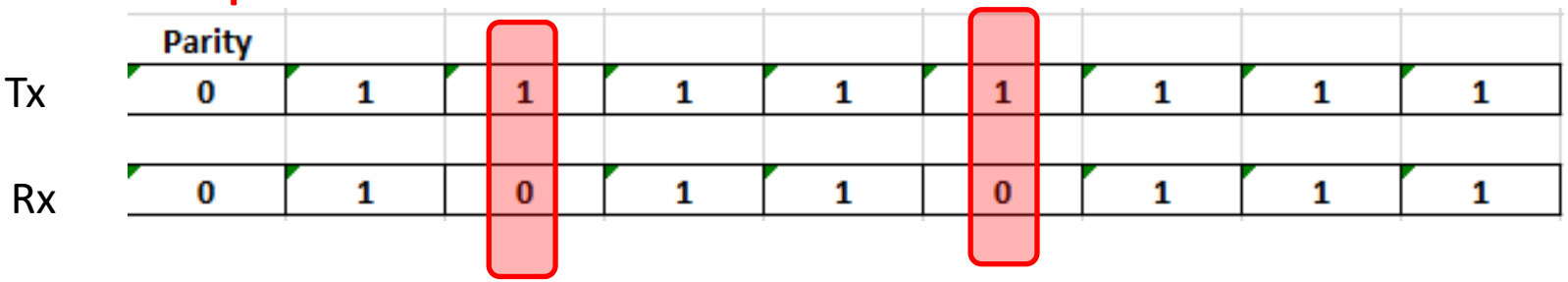
ERROR DETECTION

- Parity check at the receiver detects an error if the parity of the received word is different from the expected parity.

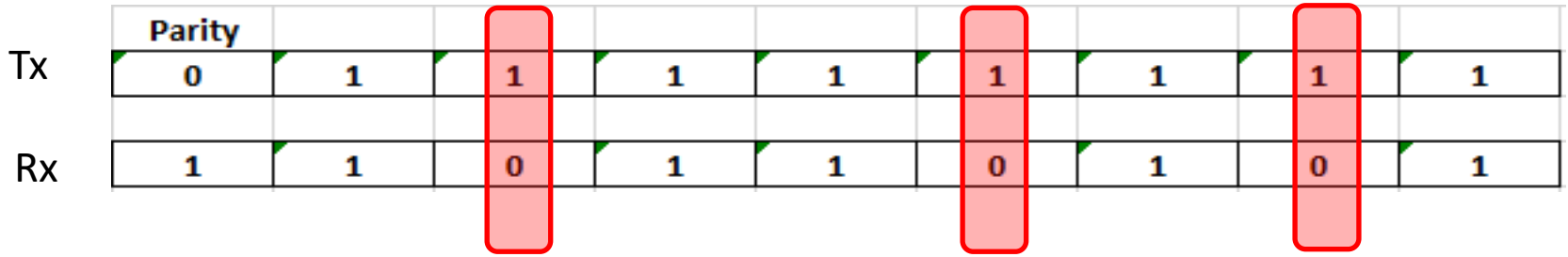


FAILURE OF EVEN PARITY CHECKING METHOD

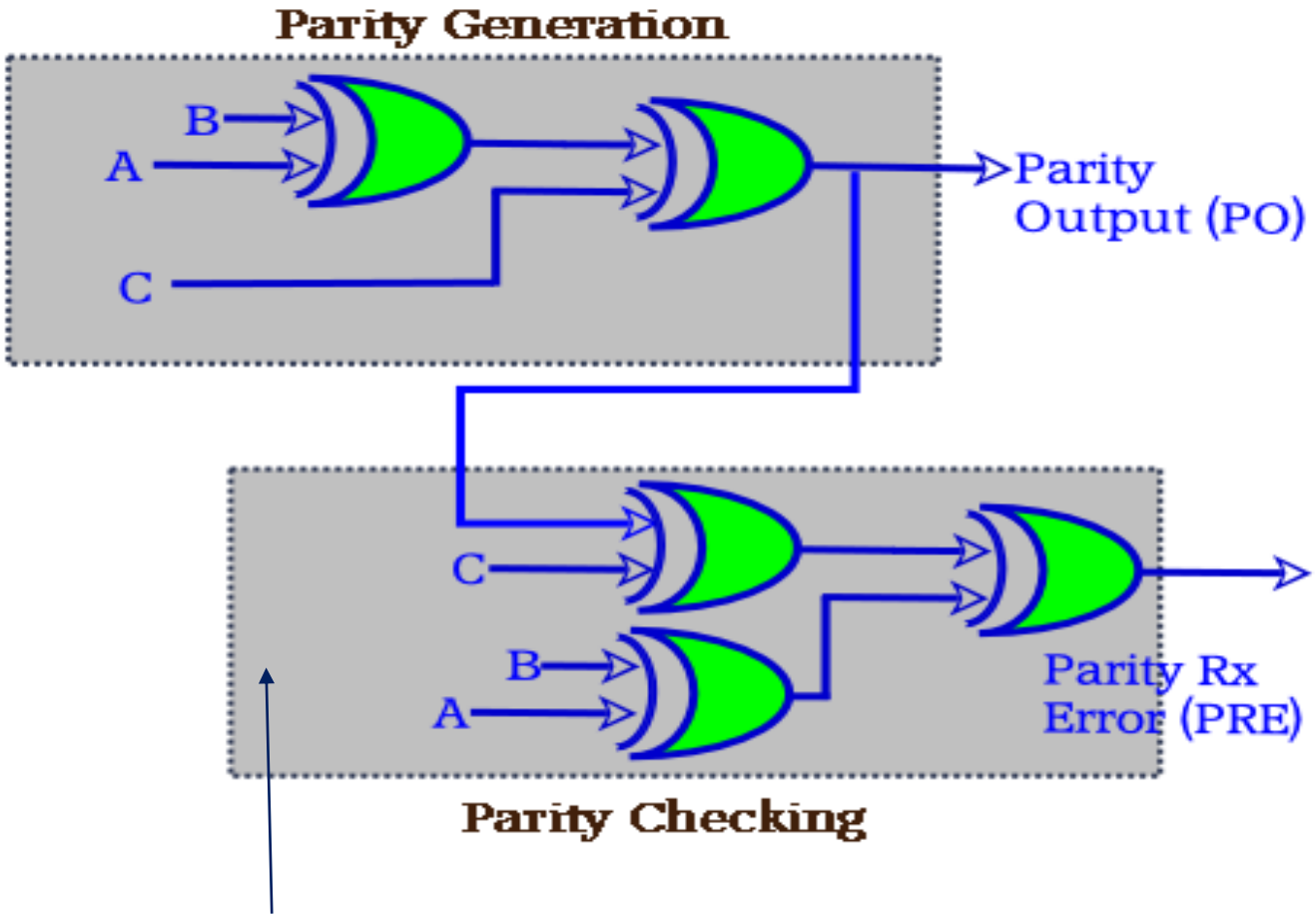
1. If the number of errors introduced in a transmitted word is two or any even number, then the parity does not change and the receiver will fail to detect the presence of errors.



2. If the number of errors introduced in a transmitted word is one or any odd number, then the parity changes and the receiver can detect the presence of errors but cannot correct them.

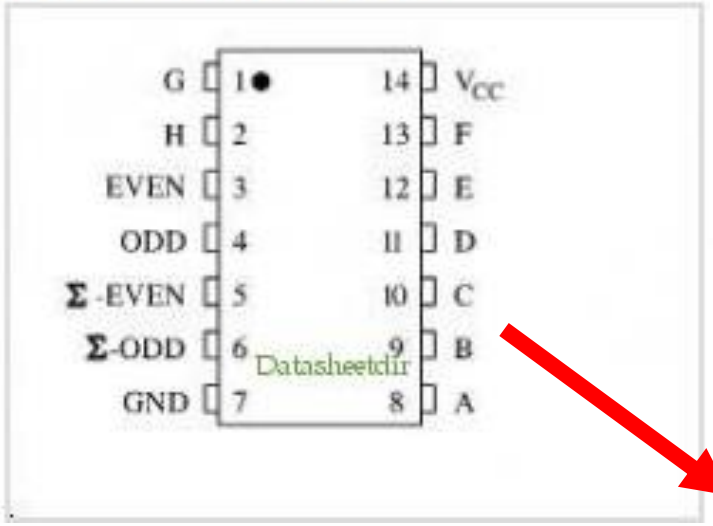


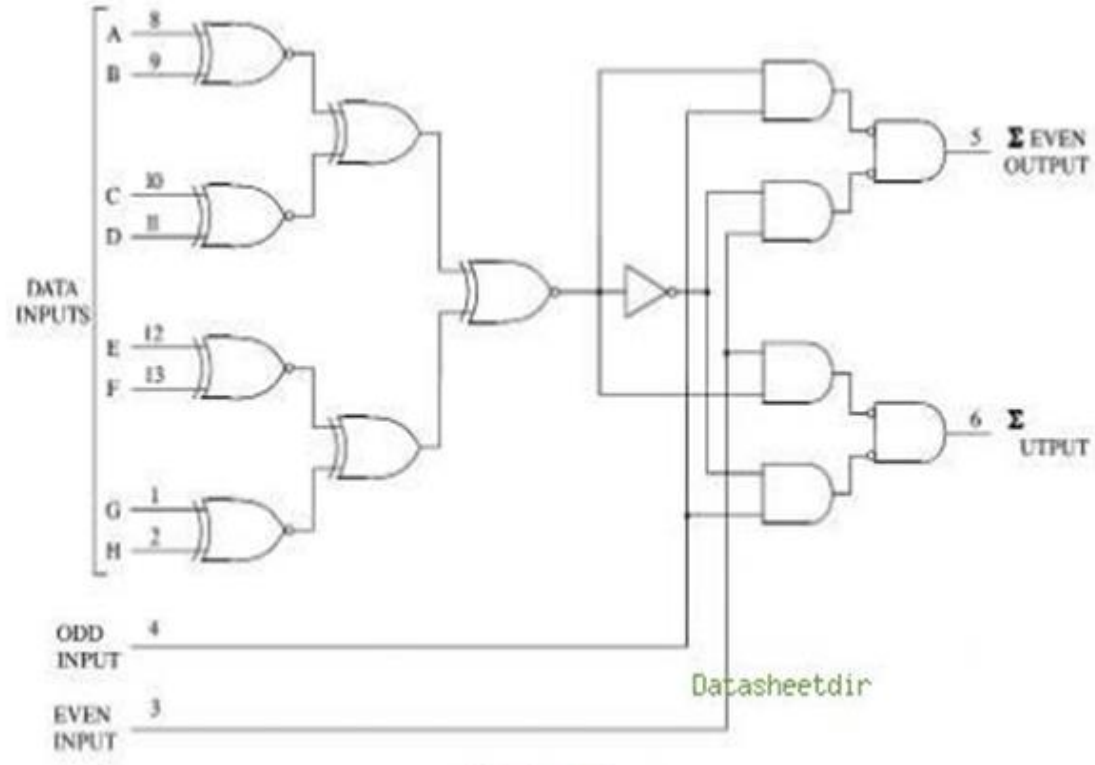
PARITY GENERATING & CHECKING CIRCUIT IN THE PC



IN74180 PARITY GEN/CHECKER INTEGRATED CIRCUIT

IN74180 Pinout, Pinouts





PIN 14 = V_{CC}
PIN 7 = GND

EXAMPLE 1: PARITY

1. Write the ASCII code of the word TALE using even parity.

SOLUTION (1)

Table ASCII -I

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Ch
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z

SOLUTION (2)

HEX	D		P	7	6	5	4	3	2	1	0
54	84	T	1	0	1	0	1	0	1	0	0
41	65	A	0	0	1	0	0	0	0	0	1
4C	76	L	1	0	1	0	0	1	1	0	0
45	69	E	1	0	1	0	0	0	1	0	1

CHECKSUM ERROR DETECTION

The checksum method works as follows:

1. **A checksum of the data** to be transmitted is computed and transmitted together with the data.
2. At the receiver, a checksum is separately computed and then compared with what was received.
3. If the two checksums are different, then an error condition is reported.

Why use Checksum?

- Errors usually occur in bursts hence making the parity method ineffective.
- In such cases the checksum method is more effective.

ERROR CORRECTING CODES

ECE 416 – DIGITAL COMMUNICATION

Friday, 05 March 2021

ERROR DETECTION USING VERTICAL & LONGITUDINAL REDUNDANCY CHECKS

CHAR	C	O	M	P	U	T	E	R	LRC
b_0	1	1	1	0	1	0	1	0	1
b_1	1	1	0	0	0	0	0	1	1
b_2	0	1	1	0	1	1	1	0	1
b_3	0	1	1	0	1	0	0	0	0
b_4	0	0	0	1	1	1	0	1	0
b_5	0	0	0	0	0	0	0	0	0
b_6	1	1	1	1	1	1	1	1	0
VRC	1	1	0	0	0	1	1	1	1

Table ASCII -I

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Ch
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z

IDENTIFICATION OF ERRORS IN LRC & VRC

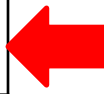
1. A single error in any row will result in an incorrect LRC. Similarly, a single error in any column will result in an incorrect VRC.

CHAR	C	O	M	P	U	T	E	R	LRC
b_0	1	1	1	0	1	0	1	0	1
b_1	1	1	0	0	0	0	0	1	1
b_2	0	1	1	0	1	1	1	0	1
b_3	0	1	1	0	~~1~~/0	0	0	0	~~0~~/1
b_4	0	0	0	1	1	1	0	1	0
b_5	0	0	0	0	0	0	0	0	0
b_6	1	1	1	1	1	1	1	1	0
VRC	1	1	0	0	~~0~~/1	1	1	1	1

IDENTIFICATION OF ERRORS IN LRC & VRC

- The bit at the intersection of the incorrect VRC and LRC can then be identified as the incorrect bit.

CHAR	C	O	M	P	U	T	E	R	LRC
b_0	1	1	1	0	1	0	1	0	1
b_1	1	1	0	0	0	0	0	1	1
b_2	0	1	1	0	1	1	1	0	1
b_3	0	1	1	0	0/1	0	0	0	1
b_4	0	0	0	1	1	1	0	1	0
b_5	0	0	0	0	0	0	0	0	0
b_6	1	1	1	1	1	1	1	1	0
VRC	1	1	0	0	1	1	1	1	1



IDENTIFICATION OF ERRORS IN LRC & VRC

- Unfortunately, multiple errors in rows and columns can only be detected but not corrected.

CHAR	C	O	M	P	U	T	E	R	LRC
b_0	1	1	1	0	1	0	1	0	1
b_1	1	1	0	0	~~0~~/1	0	0	1	1/0
b_2	0	1	1	0	1	1	1	0	1
b_3	0	1	1	0	~~1~~/0	0	0	0	~~0~~/1
b_4	0	0	0	1	1	1	0	1	0
b_5	0	0	0	0	0	0	0	0	0
b_6	1	1	1	1	1	1	1	1	0
VRC	1	1	0	0	~~0~~/1/1	1	1	1	1

EXAMPLE: VRC AND LRC

- The following bit streams are encoded using VRC, LRC and even parity. Locate and detect the error if present.

11000011	**11110011**	**10110010**	**00001010**
00101010	**00101011**	**10100011**	**01001011**
11100001			

SOLUTION

11000011	11110011	10110010	00001010
00101010	00101011	10100011	01001011
11100001			

Bit in Error

	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	LRC
b_1	1	1	1	0	0	0	1	0	1/?
b_2	1	1	0	0	0	0	0	1	1
b_3	0	1	1	0	1	1	1	0	1
b_4	0	1	1	0	0	0	0	0	0
b_5	0	0	0	1	1	1	0	1	0
b_6	0	0	0	0	0	0	0	0	0
b_7	1	1	1	1	1	1	1	1	0
VRC	1	1	0	0	0/?	1	1	1	1

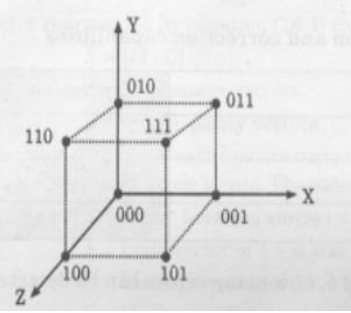
DEFINITIONS RELATING TO ERROR CODES

1. **Code Word** refers to an n bit block of bits containing message bits, parity or redundant bits.

2. **Code rate** is the ratio of the number of message bits (k) to the number of bits in the code word.

$$Coderate(r) = \frac{NumberMessageBits(k)}{NumberOfBits(n)}$$

3. **Code vectors** refers to the visualization of an n-bit word in m-dimensional space.



S. No.	Code word		
	$x_2 = Z$	$x_1 = Y$	$x_0 = X$
0	0	0	0
1	0	0	1
2	0	1	0
⋮	⋮	⋮	⋮
7	1	1	1

TYPES OF ERROR CORRECTION

There are basically two types of error control:

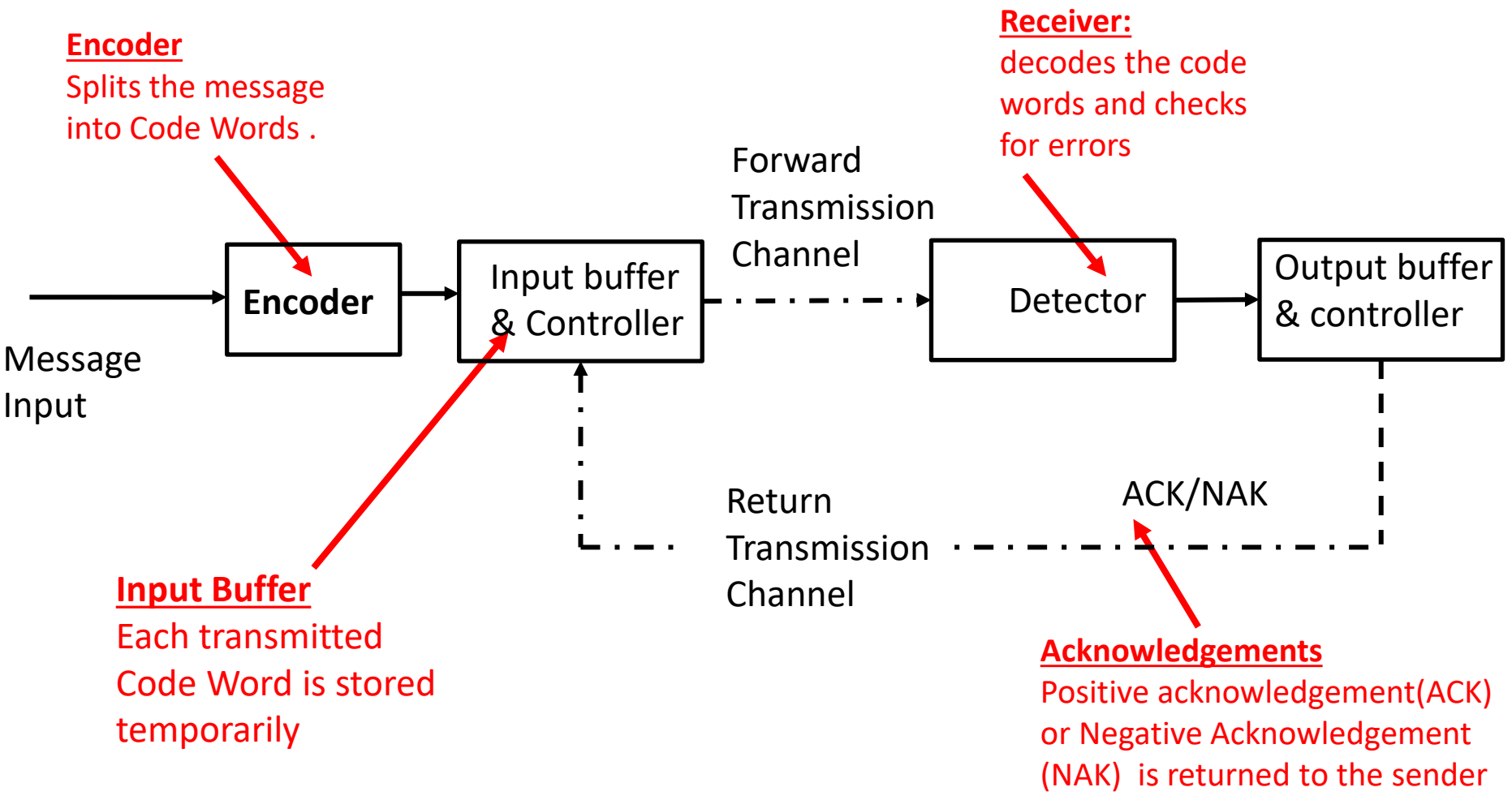
(a) Automatic Request for Retransmission (ARQ): The receiver requests for retransmission of complete or part of the message. This requires a feedback channel.

(b) Forward Error Correction (FEC): No feedback mechanisms exist.

AUTOMATIC REPEAT REQUEST (ARQ)

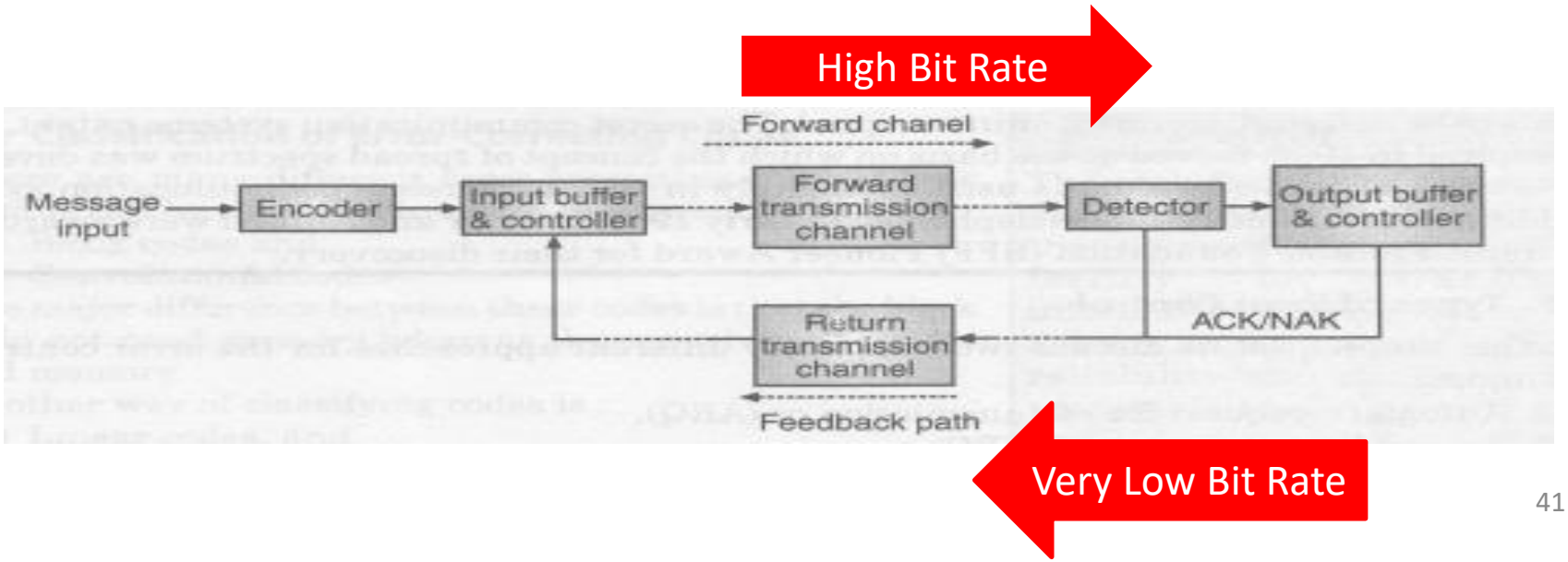
1. Automatic Repeat reQuest (ARQ) uses acknowledgements and timeouts to achieve reliable data transmission over an unreliable channel.
2. If the sender does not receive an acknowledgment before the timeout, it usually re-transmits the frame/packet until the sender receives an acknowledgment or it exceeds a predefined number of re-transmissions.

WORKING PRINCIPLE OF ARQ SYSTEMS



PROBABILITY OF ERROR ON RETURN PATH

1. The Bit rate for the return transmission is usually low compared with forward transmission since it transmits only ACK or NAK.
2. The probability of error is very small and is therefore negligible.



TYPES OF ARQ SYTEMS

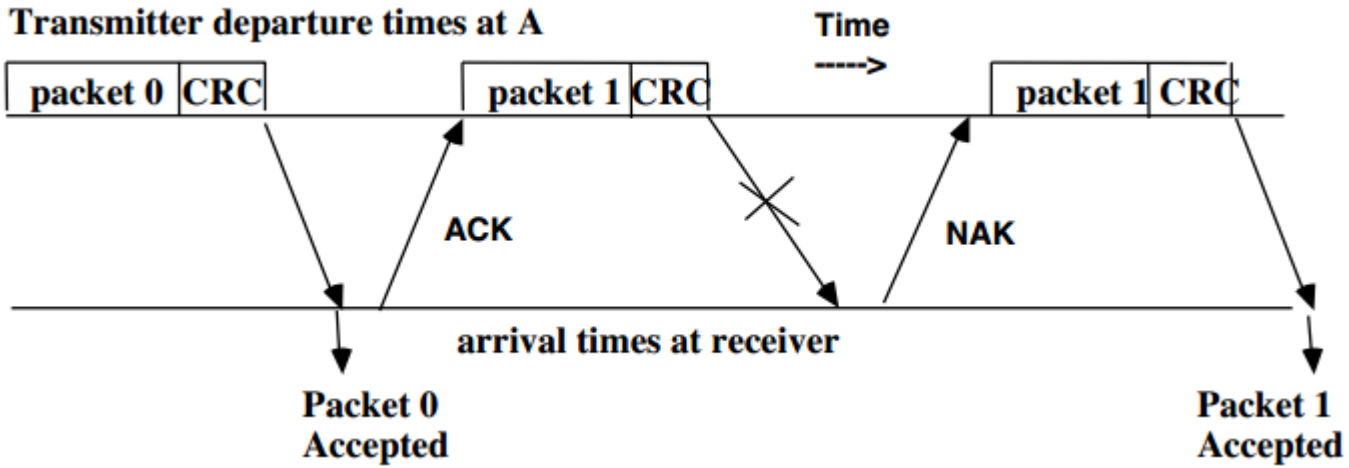
There are three types of ARQ Systems:

(a) Stop-and-Wait ARQ

(b) Go Back N ARQ

(c) Selective ARQ

STOP-AND-WAIT ARQ (1)



Problem:

Lost Packets

– Sender will wait forever for an acknowledgement

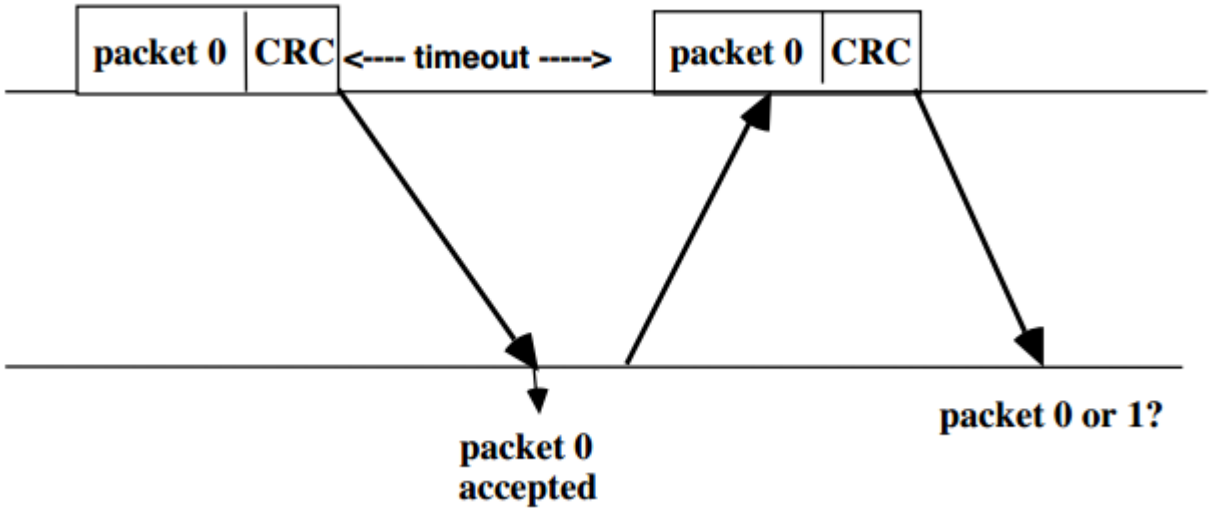
– Packet may be lost due to framing errors

Solution: Use time-out

– Sender retransmits the packet after a timeout

STOP-AND-WAIT ARQ (2)

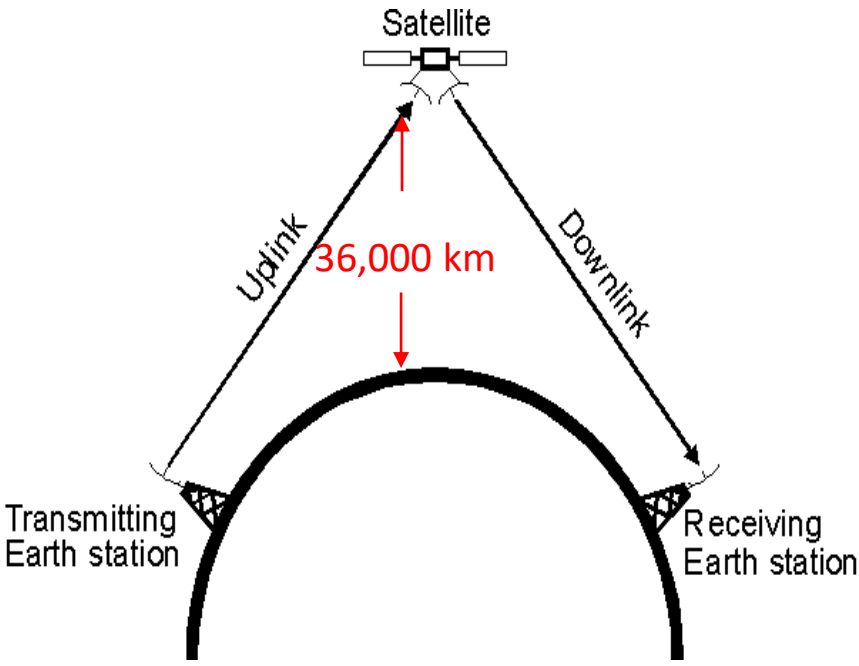
The Use of Timeouts For Lost Packets Requires Sequence Numbers



1. **Problem:** Unless packets are numbered, the receiver cannot tell which packet it received after timeout.
2. **Solution:** Use packet numbers (sequence numbers)

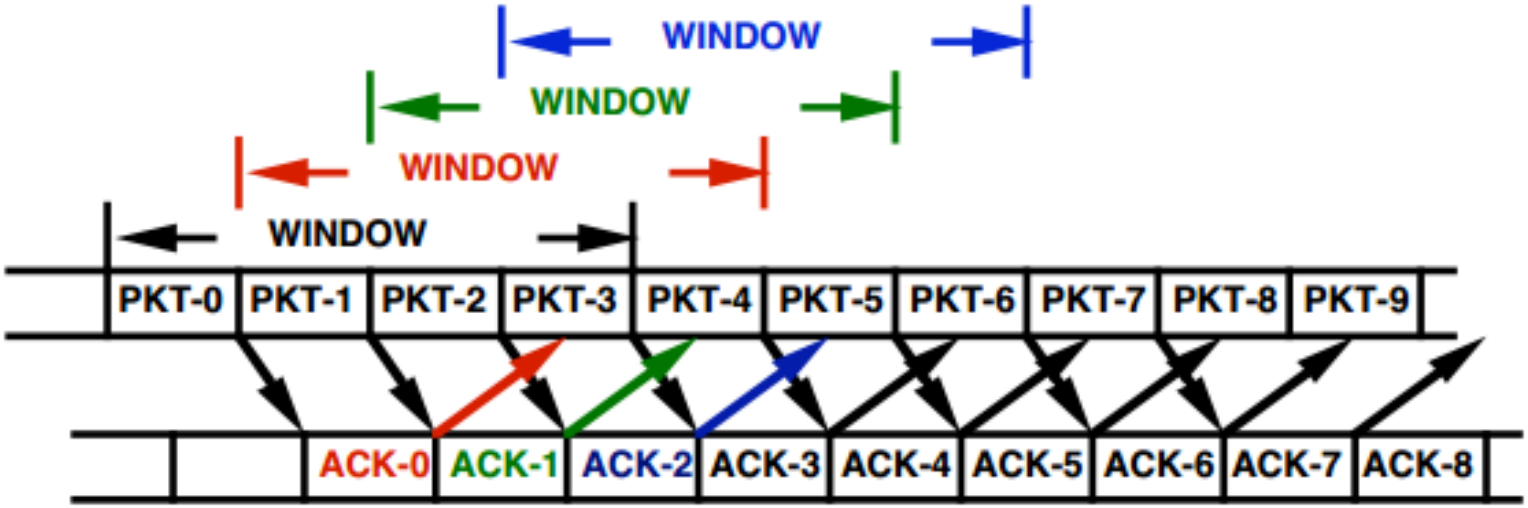
PROBLEMS OF STOP-AND-WAIT

1. Stop and Wait is inefficient when propagation delay is larger than the packet transmission time
2. Example is satellite communication where a round trip can be as high as 0.5 seconds.



GO BACK N ARQ

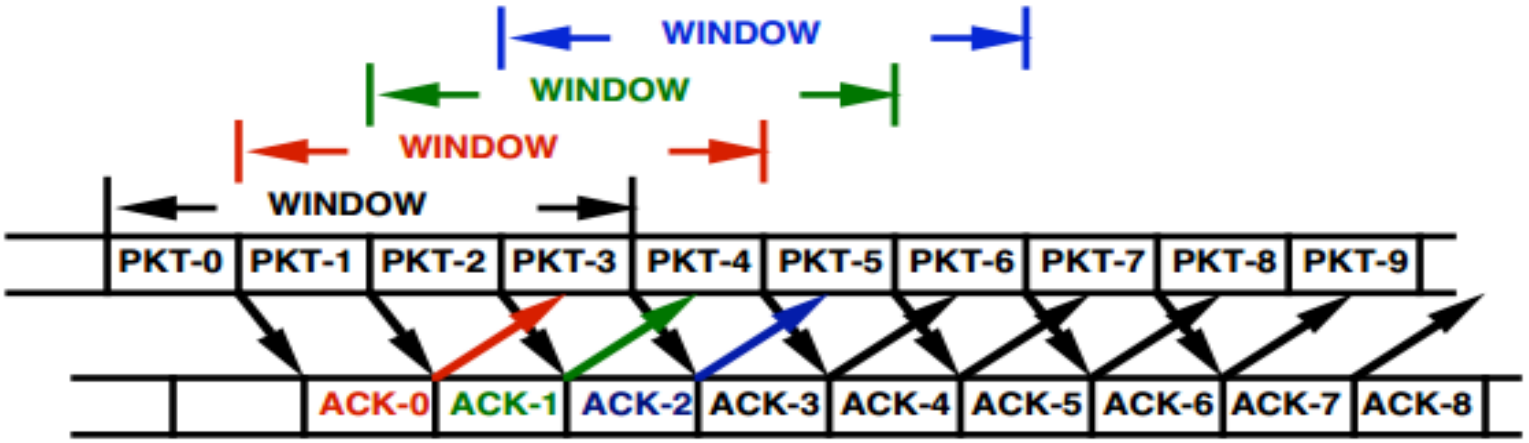
1. **Go Back N** allows the transmission of new packets before earlier ones are acknowledged.
2. It uses a window mechanism where the sender can send packets that are within a “window” (range) of packets
3. The window advances as acknowledgements for earlier packets are received



FEATURES OF GO BACK N ARQ

Window size = N

1. Sender cannot send packet i+N until it has received the ACK for packet i
2. Receiver operates just like in Stop and Wait, i.e it cannot accept packet out of sequence



CONDITIONS FOR GO BACK N TO FUNCTION CORRECTLY

Go Back N is guaranteed to work correctly, if

1) System is correctly initialized.
2) There are no failures in detecting errors.
3) Packets travel in First Come First Serve (FCFS) order.
4) There is Positive probability of correct reception
5) Transmitter occasionally resends (e.g., upon timeout).
6) Receiver occasionally sends RN

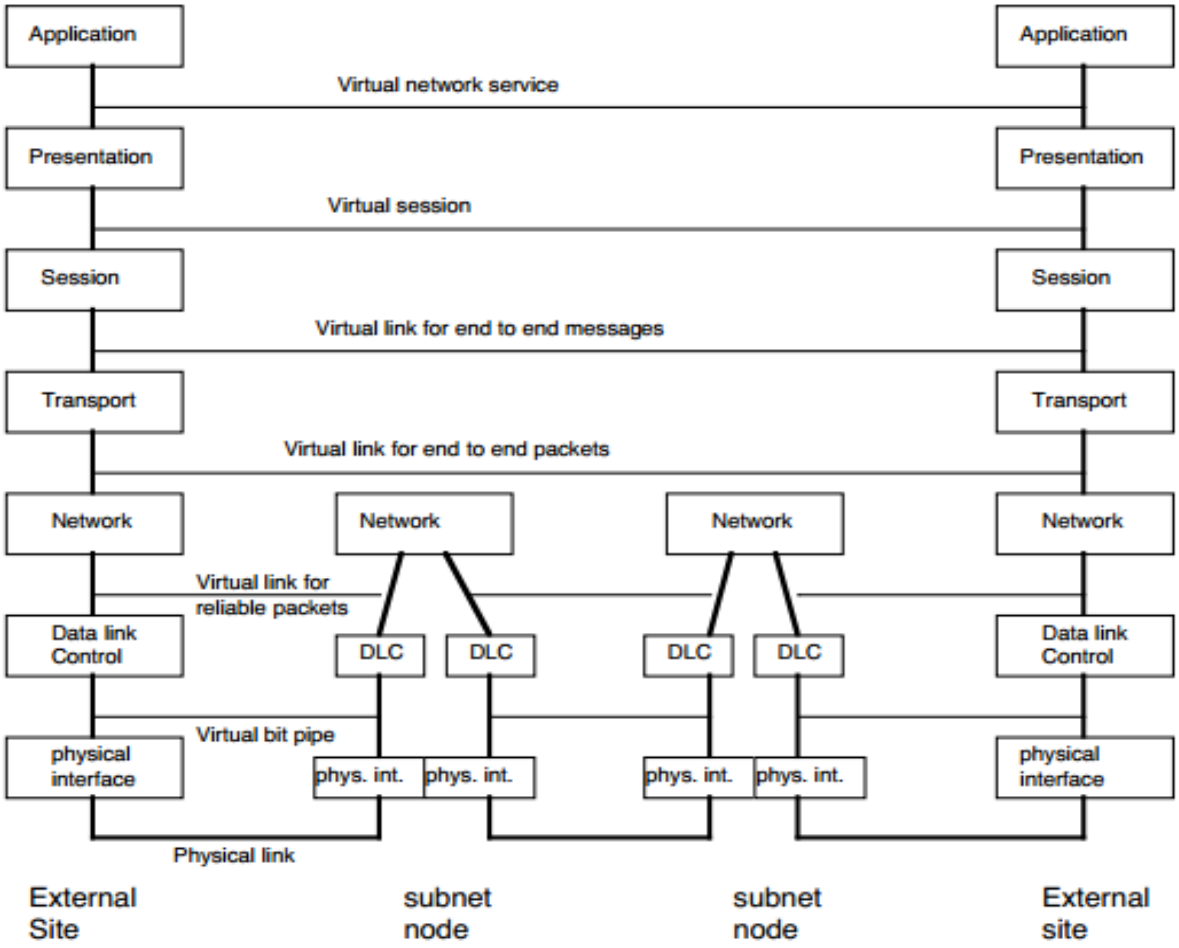
PROBLEM OF GO BACK N ARQ

1. The major problem with Go Back N is the need to re-send the entire window when an error occurs.
2. Further, the receiver can only accept packets in order in which they were transmitted.

SELECTIVE REPEAT PROTOCOL (SRP)

1. Selective Repeat attempts to retransmit only those packets that are actually lost (due to errors).
2. For it to work correctly,
   a) Receiver must be able to accept packets out of sequence.
   b) The receiver must be able to buffer some packets.

APPLICATIONS IN COMPUTER NETWORKING



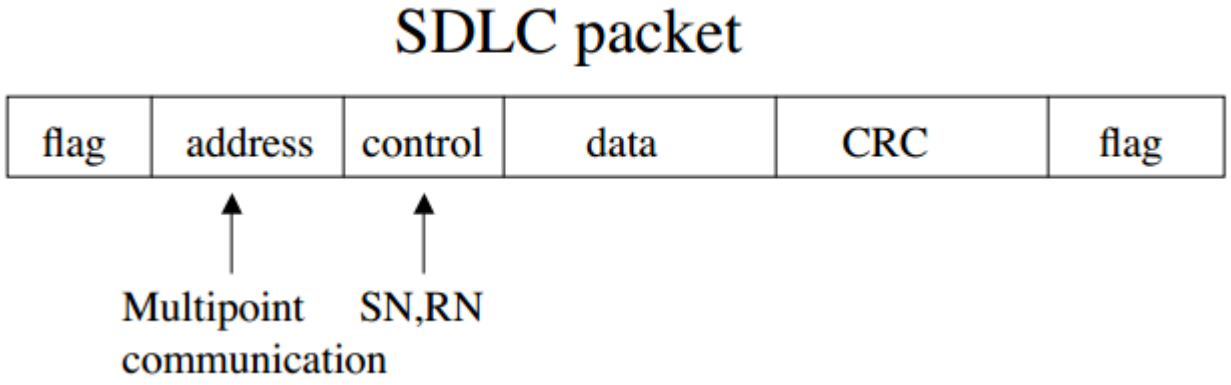
FEC/ARQ

ARQ

ARQ

FEC coding

STANDARD DATA LINK CONTROL STRUCTURES

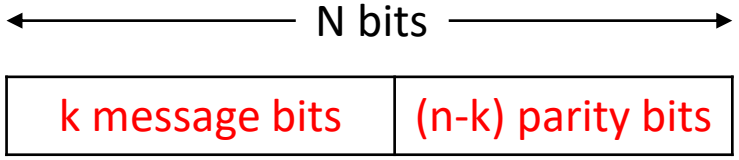


1. **HDLC**(High-level Data Link Control), **LAPB** (Link Access Procedure - Balanced), and **SDLC** are almost the same
   - HDLC/ SDLC developed by IBM for IBM SNA networks
   - LAPB developed for X.25 networks
2. They all use bit oriented framing with flag = 01111110
3. They all use a 16-bit CRC for error detection
4. They all use Go Back N ARQ with N = 7 or 127 (optional)

CLASSIFICATION OF ERROR CORRECTING CODES

There are generally two type of error correcting codes:

1. **Block Codes:** The Channel Coder generates accepts **n** successive **k** blocks and at the end of each block it adds **n-k** parity bits.



2. **Convolution Codes:** The Channel Coder generates code words through **discrete-time convolution of input sequence and the impulse response of the encoder.**