Parul® University

Computer Networks (203105255)

Prof. varsha Naregalkar and Prof. Bhaliya Nirali N,
Assistant Professors
Computer Science & Engineering

Parul®
University

DIGITAL LEARNING CONTENT

DATA LINK LAYER AND MEDIUM ACCESS SUB LAYER

Topics covered :

Error Detection and Error Correction

1-Fundamentals

2-Block coding

3-Hamming Distance

4-CRC

5-Sliding Window

6-Piggybacking

7-Random Access

8-Multiple Access Protocols:

- **Pure ALOHA**
- **Slotted ALOHA**
- **CSMA/CD**
- **CDMA/CA**

Parul®
University

DIGITAL LEARNING CONTENT

Topic-1

Error Detection and Error Correction

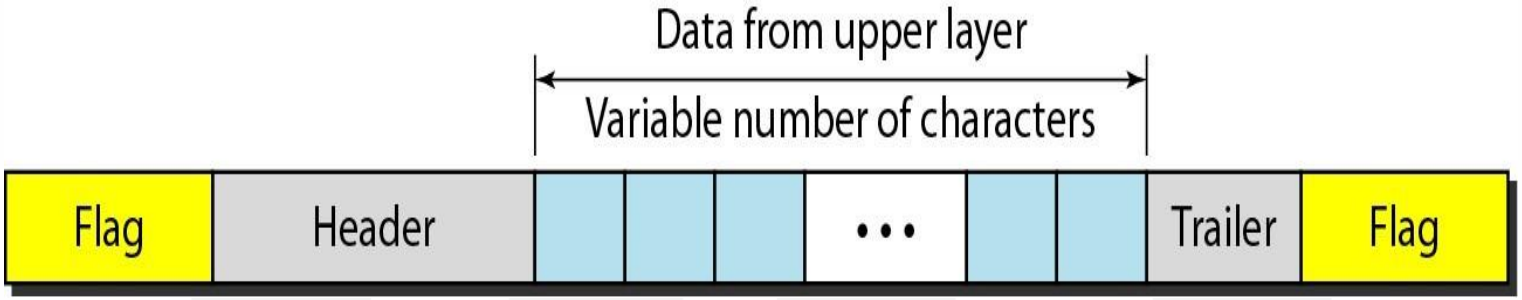
Parul® University

DIGITAL LEARNING CONTENT

About the Data Link Layer

- The data link layer needs to pack bits into frames, so that each frame is distinguishable from another. – For e.g. Our postal system practices a type of framing. The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter.
- In data link layer data unit is called, frames. At data link layer, data put in between header and trailer.
- Header consist of how data transmitted to next node into the network and trailer consist of error detecting and correcting code. Frame size is fixed or variable in length.
- At data Link Layer, Two type of protocols are used 1. byte oriented (character oriented protocol) 2. bit oriented protocol

Sample Frame of Data Link Layer…



- Flag is a fixed pattern indicating starting and ending of frame
- Header consist of source and destination MAC address(48-bit physical address) and control information regarding flow control.
- Data may be consider as byte wise(character)or bit wise depends on protocol
- Trailer consist of error detecting and error correcting codes to ensure correct delivery of data to next node

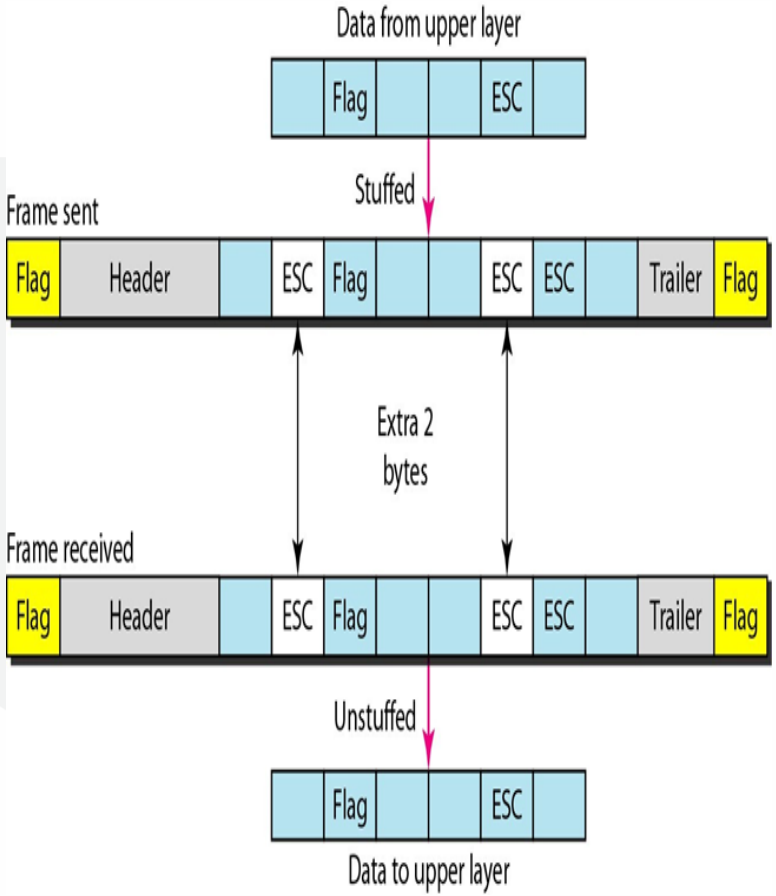
Image source : Google

Byte oriented (character oriented)protocol & Byte Stuffing.....

- In this protocol data consider as in terms of bytes...i.e. characters for e.g. we use ASCII code.

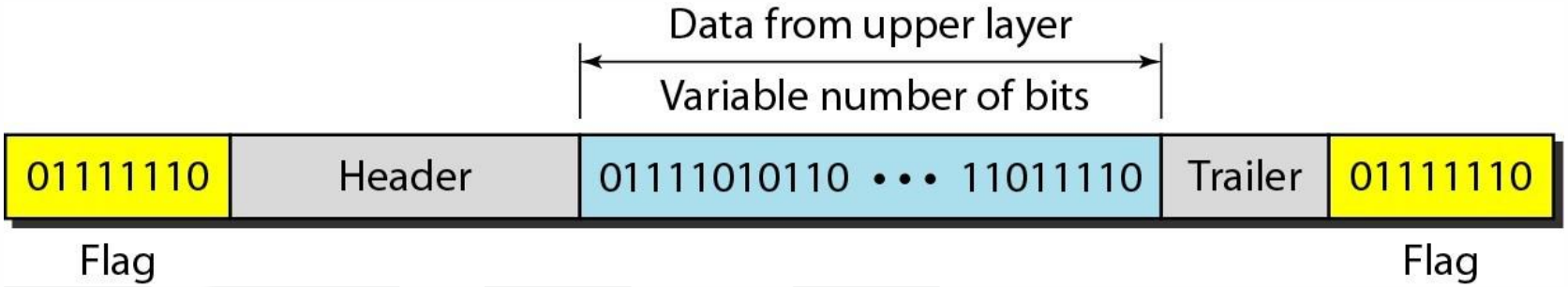
- In byte oriented, certain control pattern repeated into data, so to avoid that a extra byte is stuffed into data to avoid conflicts.

- In given image, flag is a pattern of group of bits (0111 1110) may appear into the data so for that an ESC – extra character stuffed into data.



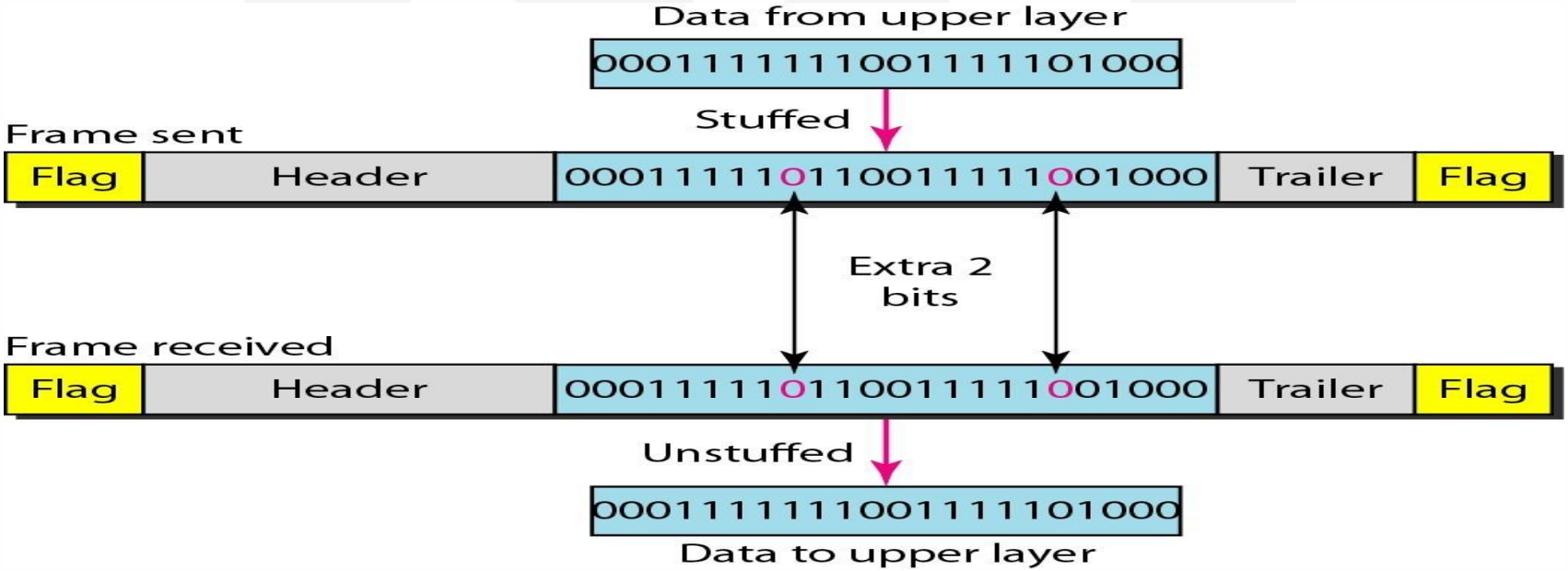
- Similarly, ESC pattern may appear into the data, then additional ESC is added.
- At the receiving end ESC-FLAG or ESC-ESC is received, then receiver end device can understand it as part of data not the control information.

•Bit Oriented Protocol & bit Stuffing….



•Here data consider as a stream of bits. Here also bit stuffing is required, for
e.g. whenever there is a group of bit like 0111 1110 is appeared is by mistake consider as a flag.
•So to avoid this, whenever there is a consecutive five 1's appear a bit-'0' is stuffed into data to avoid conflicts
at receiving end which is already show in figure.

- So at sender side, in data after consecutive five a bit ‘0’ inserted (stuffed).
- At Receiver side, in data whenever consecutive 1’s occurs, bit ‘0’ is removed.



Parul® University

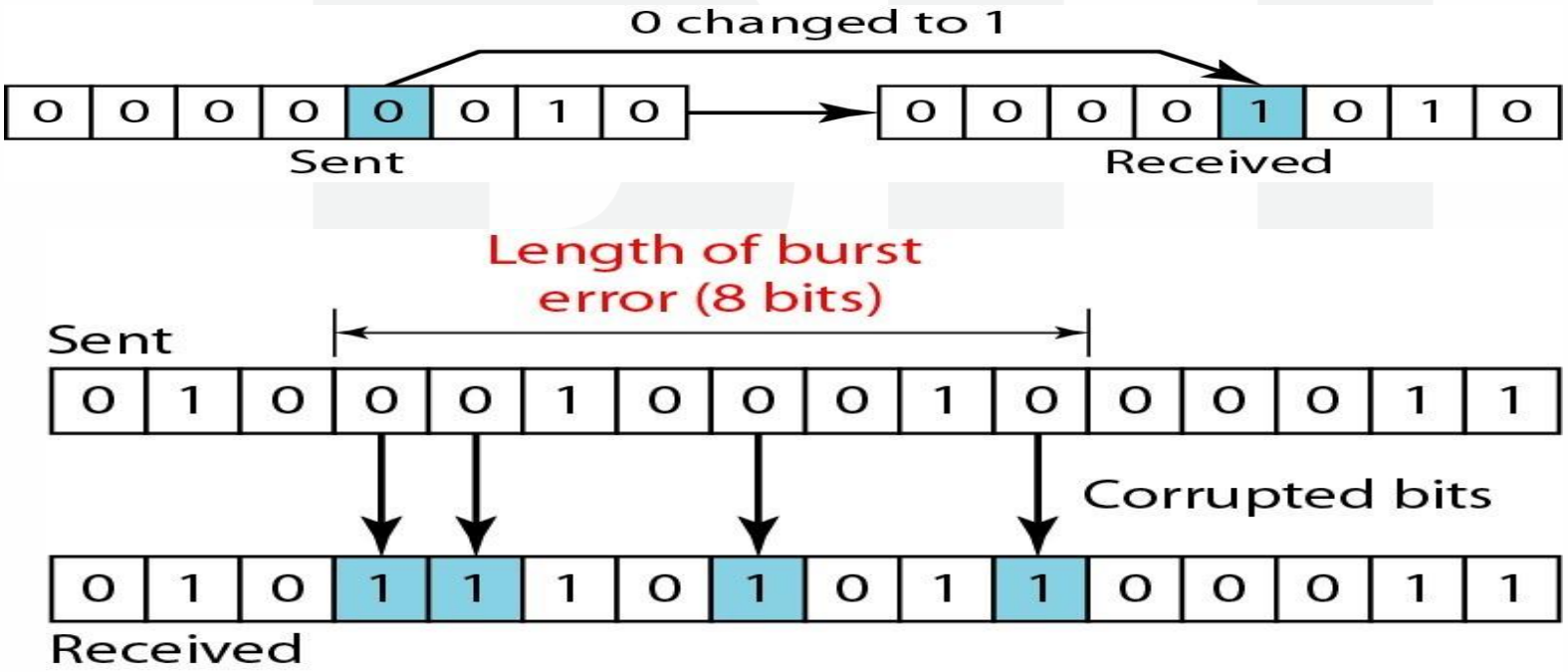
DIGITAL LEARNING CONTENT

Error Detection and Correction

- Data can be corrupted during transmission.
- Some applications require that errors be detected and corrected
- Topic to be discussed
  - –Types of Errors
  - –Redundancy
  - –Détection Versus Correction
  - –Forward Error Correction Versus Retransmission
  - –Coding

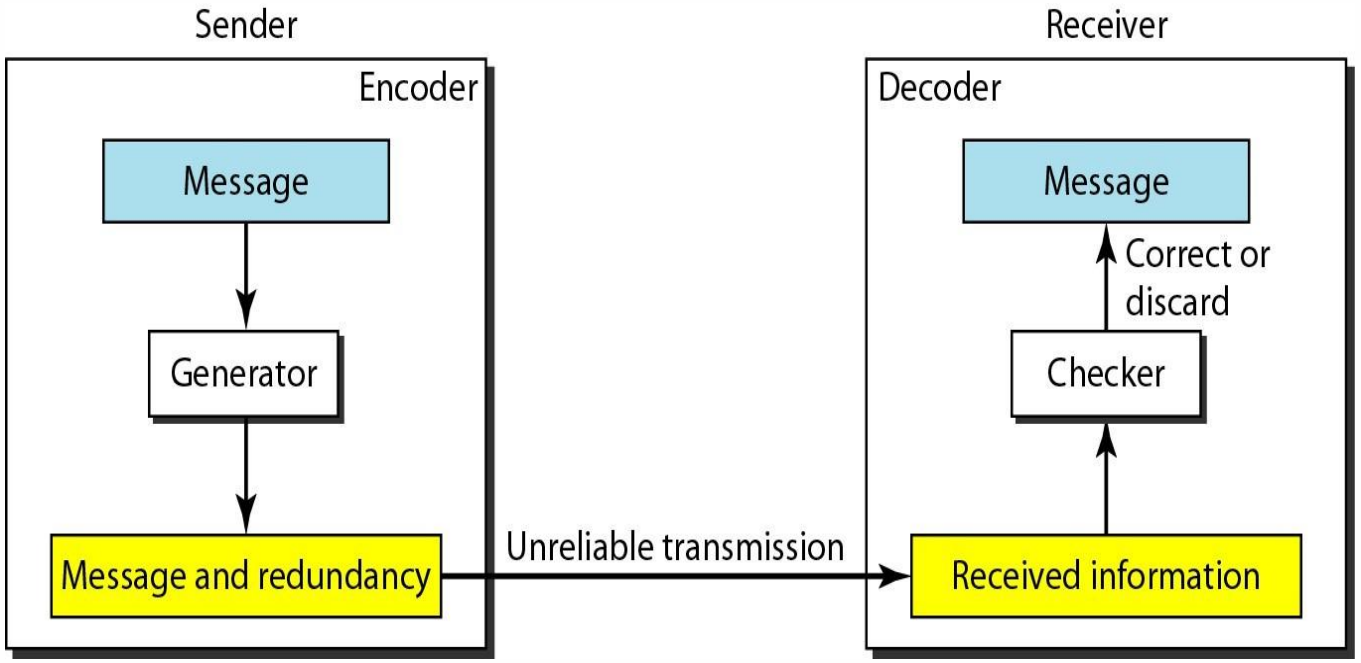
Types of Error

- In a single-bit error, only 1 bit in the data unit has changed (fig 1) .
  Burst error of length of 8 (fig 2)



Error Detection and Correction: Redundancy

To detect or correct errors, we need to send extra (redundant) bits with data



Detection Versus Correction :

–The correction of errors is more difficult than the detection.

–In error detection, we are looking only to see if any error has occurred. The answer is a simple yes or no. We are not even interested in the number of errors. A single-bit error is the same for us as a burst error.

–In error correction, we need to know the exact number of bits that are corrupted and more importantly, their location in the message. The number of the errors and the size of the message are important factors. If we need to correct one single error in an 8-bit data unit, we need to consider eight possible error locations; if we need to correct two errors in a data unit of the same size, we need to consider 28 possibilities

–So error correction is more complex than error detection.

Forward Error Correction Versus Retransmission

- There are two main methods of error correction.

1.Forward error correction is the process in which the receiver tries to guess the message by using redundant bits.

- For example. If we have to send data 0 0 1 1 then we send these data 000 000 111 111,(instead of 4 bits we send 12 bits, message replicated 3 times.
- Now receiver receive message like 001 000 111 111. Then here in first 000 change to 001. So here actual bit is 0 in 001 , two '0' and one '1'. So it consider this as 0. so actual message will be 0 0 1 1.
- FEC is used when burst error rate is high like wireless transmission medium.

2.In Retransmission, if the number of errors are less. Correction can be perform by retransmission technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Resending is repeated until a message arrives that the receiver believes is error-free.

Redundancy codes

•Redundancy is achieved through various coding schemes.

•The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits.

•The receiver checks the relationships between the two sets of bits to detect or correct the errors.

•We can divide coding schemes into two broad categories: block coding and convolution coding. These are two main methods of error correction.

Error Detection and Correction: Use of XORing

- Here to verify the correctness, a simple XOR operation is used.
- As shown in fig, when two bit are same i.e. 0-0 or 1-1 then XOR operation is gives zero. And if both bit are different then XOR is 1.

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

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

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

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

	1	0	1	1	0
$\oplus$	1	1	1	0	0
	0	1	0	1	0

c. Result of XORing two patterns

Continue.....

To verify the correctness, sender's data word & receiver's data word, a XOR operation is performed so, if you get all bits are zero then correctly received otherwise there is an error.

1100 sender codeword
1100 receiver codeword

1100 sender codeword
1000 receiver codeword

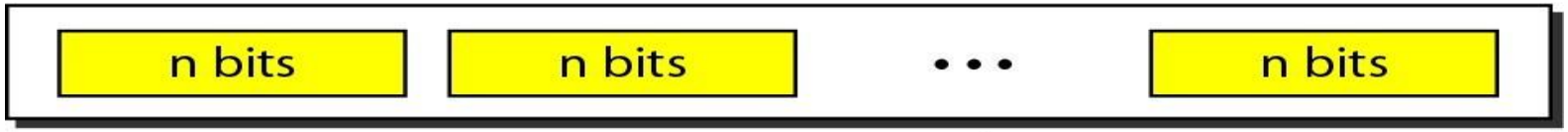
0000 result of XOR

0100 result of XOR (there is an error)

Error Detection and Correction: BLOCK CODING

- In block coding, we divide our message into blocks, each of k bits, called data words. We add r redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called code words
- In below example original message is divided into chunks of k-bit and each chunks of k-bit, there is a n bit code word is define.

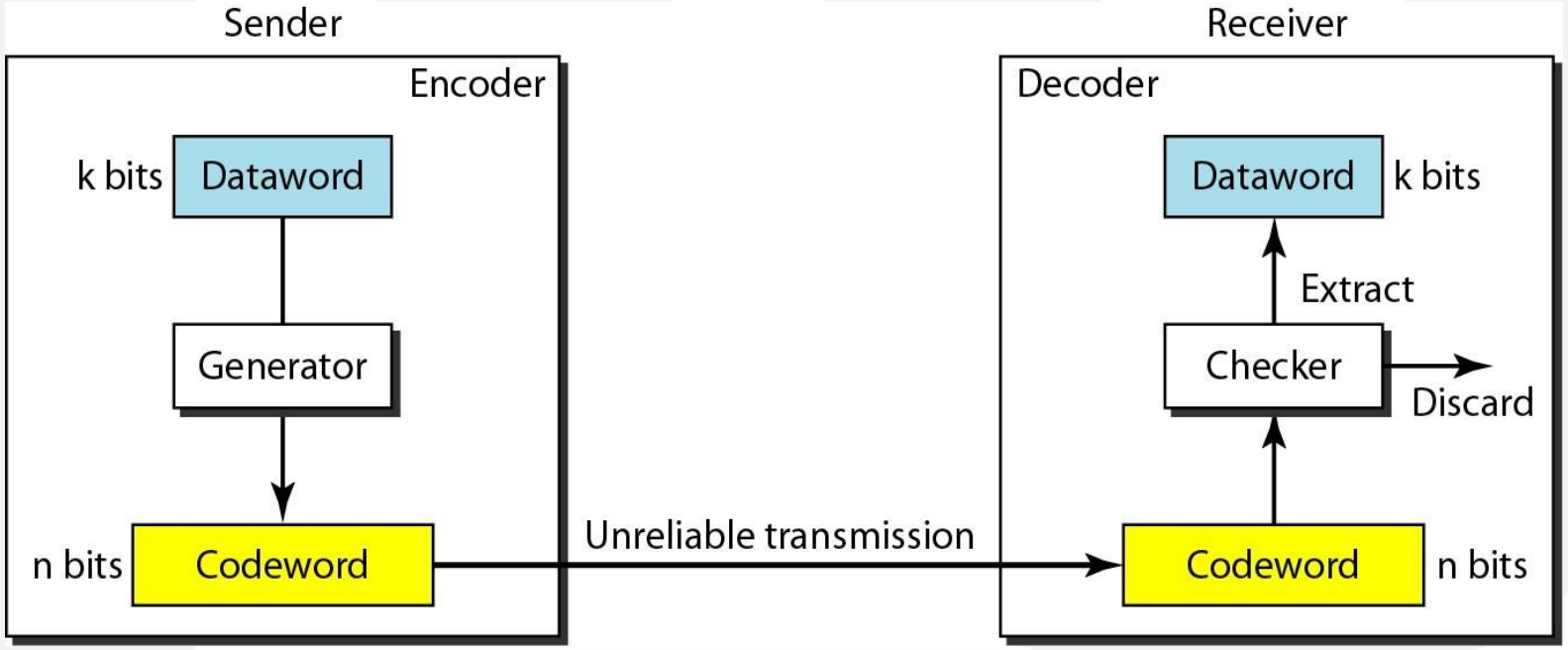
2^k Datawords, each of k bits



2^n Codewords, each of n bits (only 2^k of them are valid)

Error Detection and Correction: BLOCK CODING

- In n bit codeword there is a k bit is message and some extra bit for error correction/detection.
- Process of error detection for block coding.



Error Detection and Correction: BLOCK CODING

• Let us assume that k = 2 and n = 3. Table given below shows the list of data words and codewords. We will see how to derive a codeword from a data word

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

E rror Detection and Correction: BLOCK CODING

•Assume the sender encodes the data word 01 as 011 and sends it to the receiver. Consider the following cases:

1.The receiver receives 011. It is a valid codeword. The receiver extracts the data word 01 from it.

2.The codeword is corrupted during transmission, and 111 is received. This is not a valid codeword and is discarded.

3.The codeword is corrupted during transmission, and 000 is received. This is a valid codeword. The receiver incorrectly extracts the data word 00. Two corrupted bits have made the error undetectable.

•In above example, this codeword is capable of detecting single bit error only, but if more then two bits are change then it is unable to detect.

•So, An error-detecting code can detect only the types of errors for which it is designed; other types of errors may remain undetected

Error Detection and Correction: BLOCK CODING

Another Example:

- Let us add more redundant bits to Previous Example to see if the receiver can correct an error without knowing what was actually sent.
- Here, We add 3 redundant bits to the 2-bit data word to make 5-bit codewords.

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

Continue Example:

Dataword	Codeword
00	00000
01	01011
10	10101
11	11110

• New table, shows the datawords and codewords. Assume the dataword is 01. The sender creates the codeword 01011. The codeword is corrupted during transmission, and 01001 is received.

• First, the receiver finds that the received codeword is not in the table. This means an error has occurred. The receiver, assuming that there is only 1 bit corrupted, uses the following strategy to guess the correct dataword.

Continue…

1.Comparing the received codeword with the first codeword in the table (01001 versus 00000), the receiver decides that the first codeword is not the one that was sent because there are two different bits.

2.By the same reasoning, the original codeword cannot be the third or fourth one in the table.

3.The original codeword must be the second one in the table because this is the only one that differs from the received codeword by 1 bit. The receiver replaces 01001 with 01011 and consults the table to find the dataword 01.

•So here, at receiver end, able to correct single bit error.

Error Detection and Correction: Linear Block code.

LINEAR BLOCK CODES

•Almost all block codes used today belong to a subset called linear block codes

.

•A linear block code is a code in which the exclusive OR of two valid codewords creates another valid codeword.

•Simple Block Code : Parity Check Code

•A simple parity-check code is a single-bit error- detecting code in which n = k + 1 with dmin = 2.

Parul® University

DIGITAL LEARNING CONTENT

Error Detection and Correction: Parity Check code

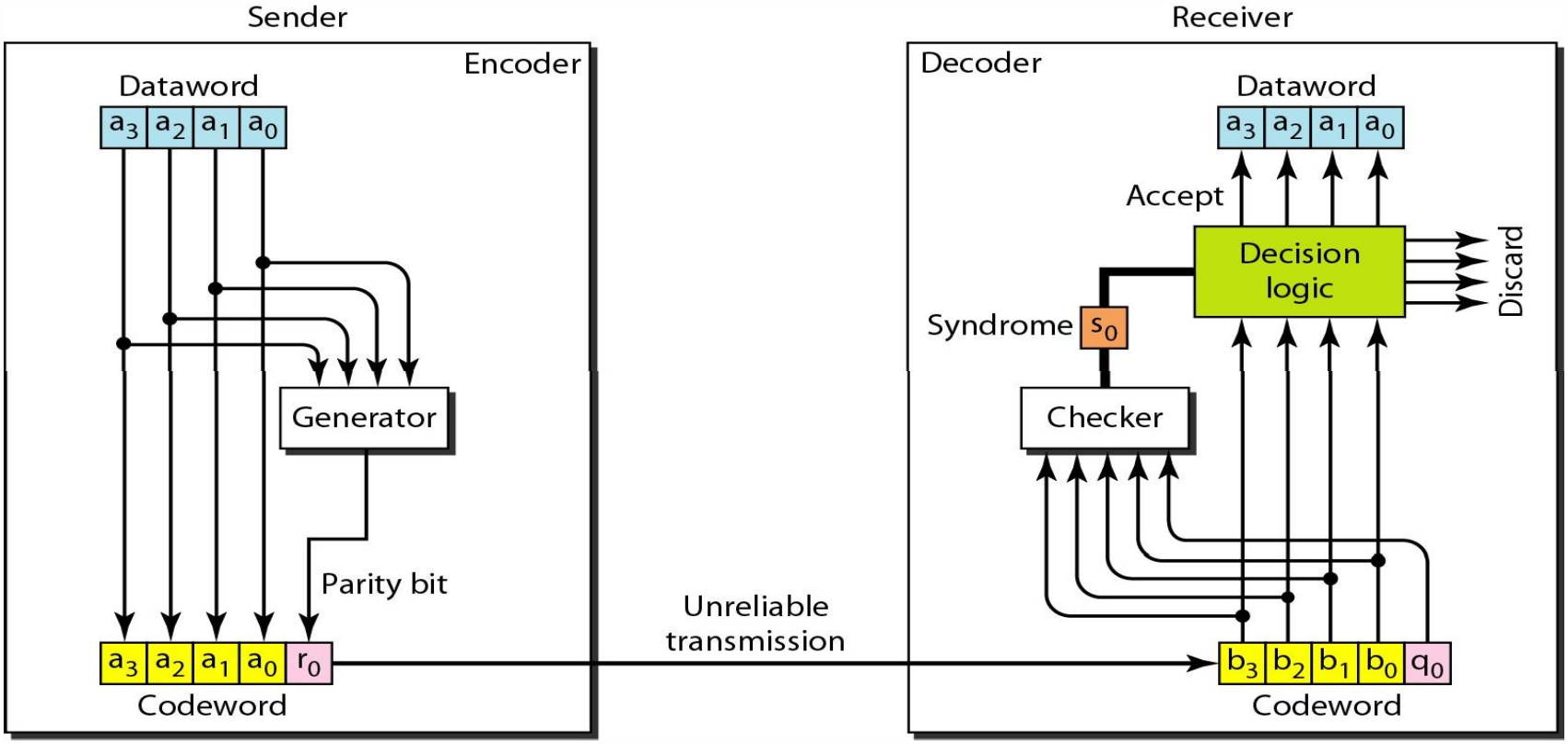


Fig. shows a possible structure of an encoder (at the sender) and a decoder

Error Detection and Correction: Parity Check code.

Simple parity-check code C(5, 4) : k=4bit and C=5bit, 1-bit is used as parity

Datawords	*Codewords*	*Datawords*	*Codewords*
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

Continue…

•If the number of 1s is even, the result is 0; if the number of 1s is odd, the result is 1. In both cases, the total number of 1s in the codeword is even. So here, codeword is made up of 4-bit message (K) and 1-bit parity total 5- bit code word shown in table.

•The sender sends the codeword which may be corrupted during transmission. The receiver receives a 5-bit word. The checker at the receiver does the same thing as the generator in the sender with one exception: The addition is done over all 5 bits. The result, which is called the syndrome, is just 1bit. The syndrome is 0 when the number of Is in the received codeword is even; otherwise, it is 1.

Error Detection and Correction: Parity Check code.

•Let us look at some transmission scenarios. Assume the sender sends the dataword 1011. The codeword created from this dataword is 10111, which is sent to the receiver. We examine five cases:

1.No error occurs; the received codeword is 10111. So, at the receiver end The syndrome is 0. The dataword 1011 is created.

2.One single-bit error changes a1 . The received codeword is 10011. The syndrome is 1.No dataword is created.

3.One single-bit error changes r0 . The received codeword is 10110. The syndrome is 1.No dataword is created.

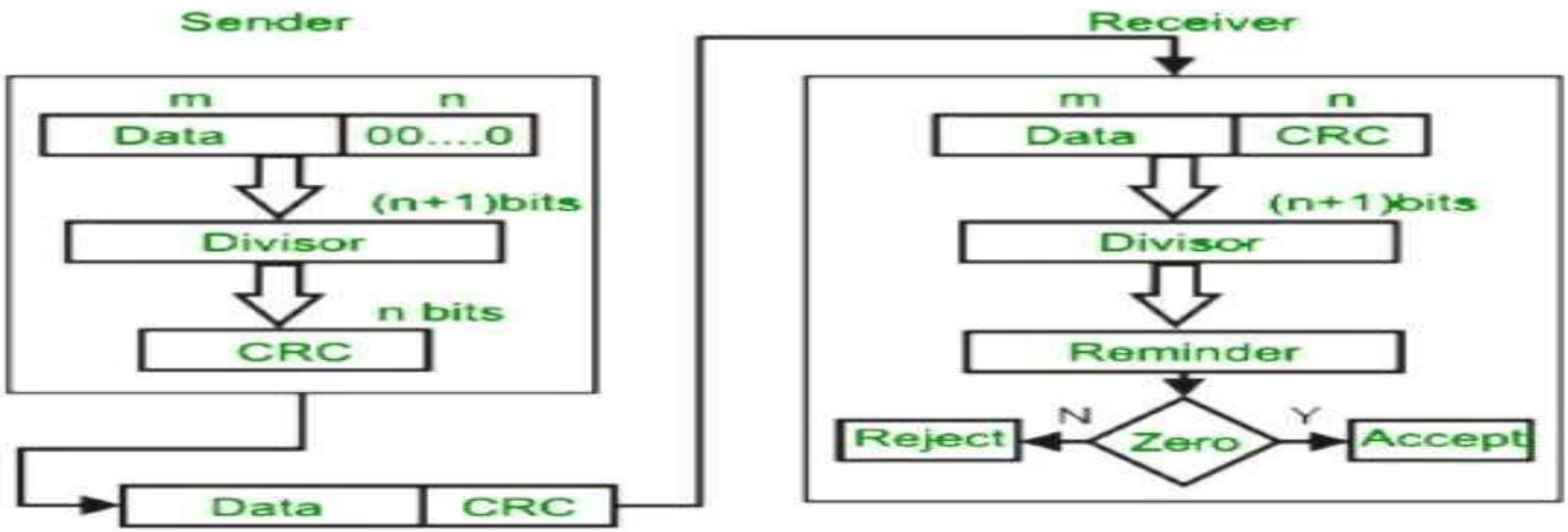
4.An error changes r0 and a second error changes a3 . The received codeword is 00110. The syndrome is 0. The dataword 0011 is created at the receiver. Note that here the dataword is wrongly created due to the syndrome value.

5.Three bits—a3, a2, and a1—are changed by errors. The received codeword is 01011. The syndrome is 1. The dataword is not created. This shows that the simple parity check, guaranteed to detect one single error, can also find any odd number of

Error Detection and Correction: CRC

•CRC or Cyclic Redundancy Check is a method of detecting accidental changes/errors in communication channel.

•CRC uses Generator Polynomial which is available on both sender and receiver side.



Error Detection and Correction: CRC Example

•An example generator polynomial is of the form like x3 + x + 1.

•This generator polynomial represents key 1011. Another example is x2 + 1 that represents key 101.

```
n : Number of bits in data to be sent
    from sender side.
k : Number of bits in the key obtained
    from generator polynomial.
```

Error Detection and Correction: CRC Example

•Sender Side (Generation of Encoded Data from Data and Generator Polynomial (or Key)):

1.The binary data is first augmented by adding k-1 zeros in the end of the data

2.Use modulo-2 binary division to divide binary data by the key and store remainder of division.

3.Append the remainder at the end of the data to form the encoded data and send the same

•Receiver Side (Check if there are errors introduced in transmission)

Perform modulo-2 division again and if remainder is 0, then there are no errors.

Error Detection and Correction: CRC Example

original message
1 0 1 0 0 0 0

@ means X-OR

Generator polynomial
x^3+1
$1.x^3+0.x^2+0.x^1+1.x^0$
CRC generator
1 0 0 1 4-bit

If CRC generator is of n bit then append (n-1) zeros in the end of original message

Sender

```
1001 | 1010000000
     @1001
      0011000000
       @1001
        01010000
         @1001
          0011000
           @1001
            01010
             @1001
              0011
```

Message to be transmitted

```
1010000000
        +0 1 1
----------
1010000011
```

```
1001 | 1010000011
     @1001
      0011000011
       @1001
        01010011
         @1001
          0011011
           @1001
            01001
             @1001
              0000
```

← Receiver

Zero means data is accepted

CRC Example-1 (No error in transmission):sender

Data word to be sent - 100100

Key - 1101 [Or generator polynomial $x^3 + x^2 + 1$]

Sender Side:

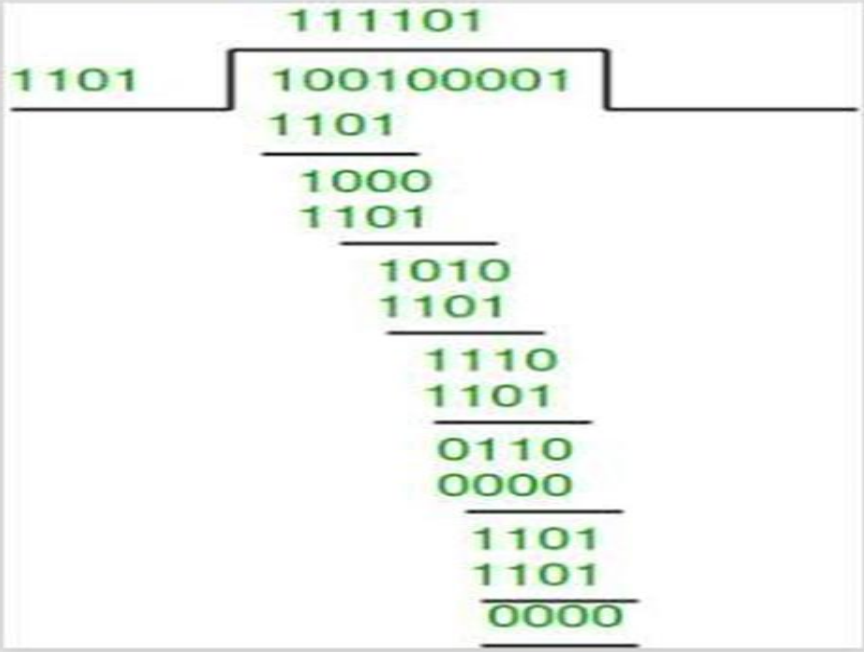
```
              111101
          ┌─────────
1101      │ 100100000
          │ 1101
          │ ----
          │  1000
          │  1101
          │  ----
          │   1010
          │   1101
          │   ----
          │    1110
          │    1101
          │    ----
          │    0110
          │    0000
          │    ----
          │     1100
          │     1101
          │     ----
          │      001
          └─────────
```

Therefore, the remainder is 001 and hence the encoded data sent is 100100001.

CRC Example-1 (No error in transmission): receiver

Receiver Side:

Code word received at the receiver side 100100001



Therefore, the remainder is all zeros. Hence, the data received has no error.

CRC Example-2 (Error in transmission):sender

Data word to be sent - 100100
Key - 1101

Sender Side:

```
            111101
        ┌──────────
1101    │ 100100000
        │ 1101
        │ ----
        │  1000
        │  1101
        │  ----
        │   1010
        │   1101
        │   ----
        │    1110
        │    1101
        │    ----
        │    0110
        │    0000
        │    ----
        │     1100
        │     1101
        │     ----
        │      001
        └──────────
```

Therefore, the remainder is 001 and hence the code word sent is 100100001.

CRC Example-2 (Error in transmission):receiver

Receiver Side
Let there be error in transmission media
Code word received at the receiver side - 100000001

```
          111010
      ┌──────────
1101  │ 100000001
      │ 1101
      │ ----
      │  1010
      │  1101
      │  ----
      │   1110
      │   1101
      │   ----
      │    0110
      │    0000
      │    ----
      │     1100
      │     1101
      │     ----
      │      0011
      │      0000
      │      ----
      │       011
      └──────────
```

Since the remainder is not all zeroes, the error is detected at the receiver side.

Error Detection and Correction: Checksum

•In checksum error detection scheme, the data is divided into k segments each of m bits.

•In the sender's end the segments are added using 1's complement arithmetic to get the sum. The sum is complemented to get the checksum.

•The checksum segment is sent along with the data segments.

•At the receiver's end, all received segments are added using 1's complement arithmetic to get the sum. The sum is complemented.

•If the result is zero, the received data is accepted; otherwise discarded.

Error Detection and Correction: Checksum Example

Original Data

10011001	11100010	00100100	10000100
1	2	3	4

k=4, m=8

Sender

```
1   10011001
2   11100010
    --------
   (1)01111011
          1
    --------
    01111100
3   00100100
    --------
    10100000
4   10000100
    --------
   (1)00100100
          1
    --------
Sum: 00100101
CheckSum: 11011010
```

Reciever

```
1   10011001
2   11100010
    --------
   (1)01111011
          1
    --------
    01111100
3   00100100
    --------
    10100000
4   10000100
    --------
   (1)00100100
          1
    --------
    00100101
    11011010
    --------
Sum: 11111111
Complement: 00000000
```

Conclusion: Accept Data

Error Detection and Correction: Hamming code.

- Hamming code can be applied to data units of any length.
- It is used to detect & correct single bit errors.

- Hamming Code Structure

All bit positions that are power of 2 are marked as Parity bits (1,2,4,8,16…..)
Remaining bits are for data

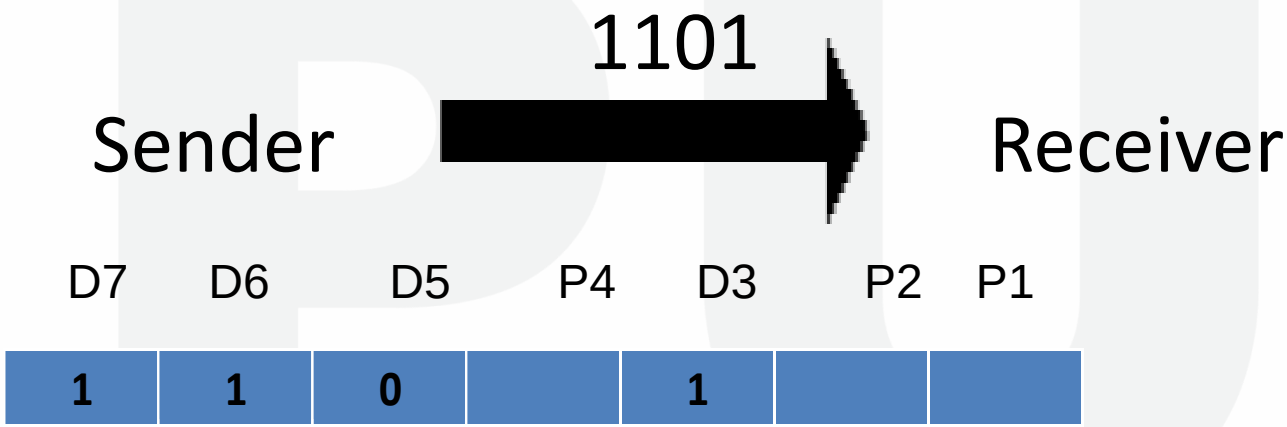
7 Bits

D7	D6	D5	P4	D3	P2	P1

Continue…

Determine the value of Parity Bits

Rules:- The value of parity bit is determined by the sequence of bits that is alternatively checks and skips.



Error Detection and Correction: Hamming code.

For P1:- Check 1 bit, skip 1 bit, check 1 bit, skip 1 bit (1,3,5,7)

For P2:- Check 2 bits, skip 2 bits, check 2 bits, skip 2 bits (2,3,6,7)

For P4:- Check 4 bits, skip 4 bits, check 4 bits, skip 4 bits (4,5,6,7)

P1 D3 D5 D7	P2 D3 D6 D7	P4 D5 D6 D7
P1 1 0 1	P2 1 1 1	P4 0 1 1
P1=0(Even Parity)	P2=1(Odd Parity)	P4=0(Even Parity)

This message is send to receiver.

D7	D6	D5	P4	D3	P2	P1
1	1	0	0	1	1	0

Error Detection and Correction: Hamming code.

- Detecting Error

Consider 7 bit Hamming Code
D7 D6 D5 P4 D3 P2 P1
At receiving end bits are (1,3,5,7) , (2,3,6,7) and (4,5,6,7) are checked for every parity.
if P1 = 0, P2 = 0, P4 = 0 then No Error
If any one parity bit is 1 then there is an error.

- Correcting Error

An error is located by forming a 3 bit no. out of 3 parity checks

P4	P2	P1

Continue…

Finding P1:- we check parity of (P1, D3, D5, D7)
If it is odd no. of 1's then error exists. P1= 1
If it is even no. of 1's then no error. P1 =0

Similarly find out for P2 and P4.
After that we have found the error word, convert into decimal value.
Then we invert the incorrect bit to obtain the correct word.

Example

•A 7bit Hamming Code is received as 1011011. Assume even parity and state whether the received code is correct or wrong. If wrong locate the bit in error.

Error Detection and Correction: Hamming Code

Solution

Received HC:-

D7	D6	D5	P4	D3	P2	P1
1	0	1	1	0	1	1

1)Detecting Errors:-

Step 1:- Analyzing Parity bits P1-> 1,3,5,7

We have	P1	D3	D5	D7
	1	0	1	1

Odd Parity So error exists

We put P1 = 1

Error Detection and Correction: Hamming Code

Step 2 :- Calculate P2.

→ Analysing bits :- 2, 3, 6, 7

P2	D3	D6	D7	
1	0	0	1	even parity
				No error

P2 = 0

Step 3 :- Calculate P4

→ Analysing bits :- 4, 5, 6, 7

P4	D5	D6	D7	
1	1	0	1	odd parity
				error exists } P4 = 1

P1 & P4 are not equal to 0 so, received code is wrong.

* Error Correcting

error word :- E =

P4	P2	P1
1	0	1

decimal value = 5 ⇒ used to locate error location which shows that the 5th bit is in error.

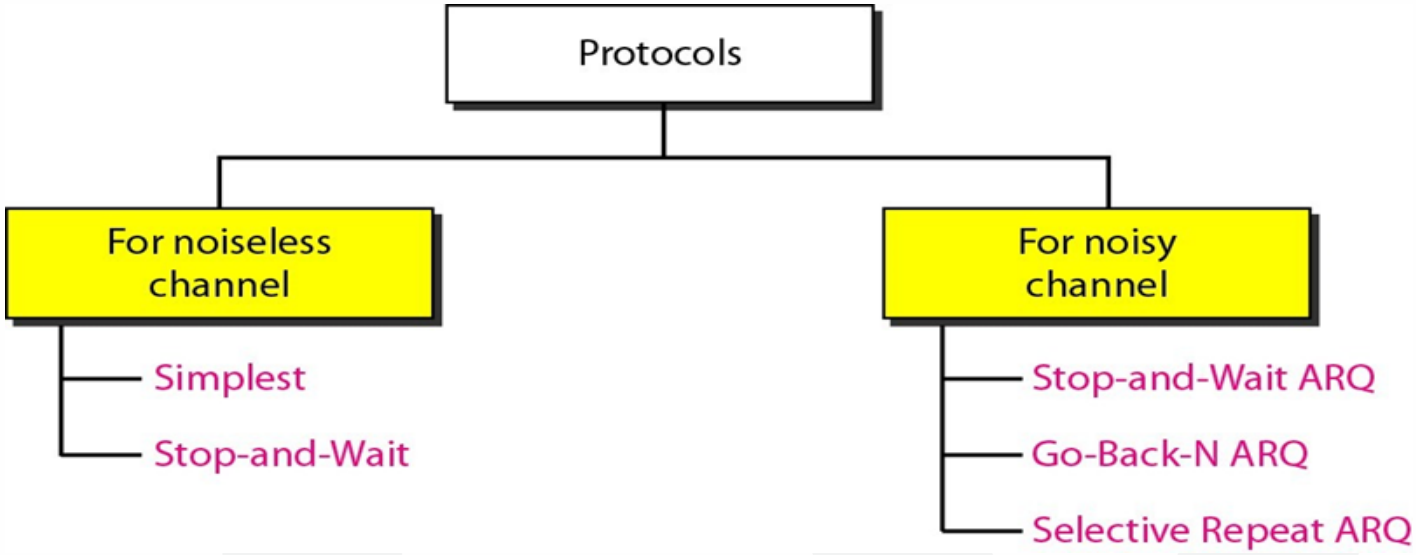
So, we write the correct word by simply inverting the 5th bit. ⇒ 1 to convert to 0

correct bit ⇒ 1001011

FLOW AND ERROR CONTROL

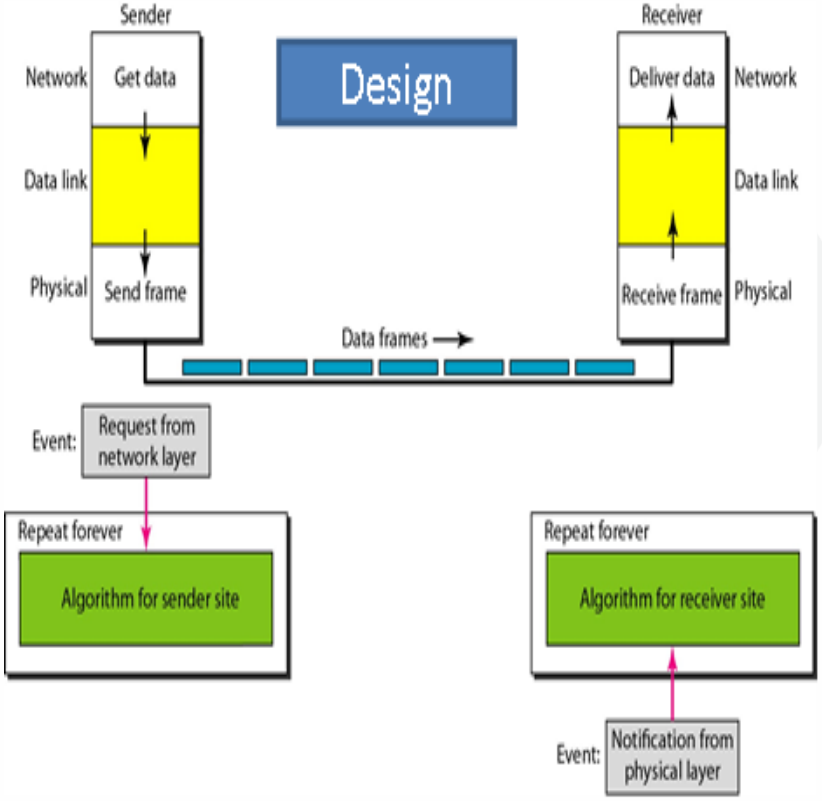
• The most important responsibilities of the data link layer are flow control and error control. Collectively, these functions are known as data link control.
• Flow control refers to a set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.
• Error control in the data link layer is based on automatic repeat request, which is the retransmission of data
• Now let us see how the data link layer can combine framing, flow control, and error control to achieve the delivery of data from one node to another. The protocols are normally implemented in software by using one of the common programming languages. To make our discussions language-free, we have written in pseudocode a version of each protocol that concentrates mostly on the procedure instead of delving into the details of language rules.

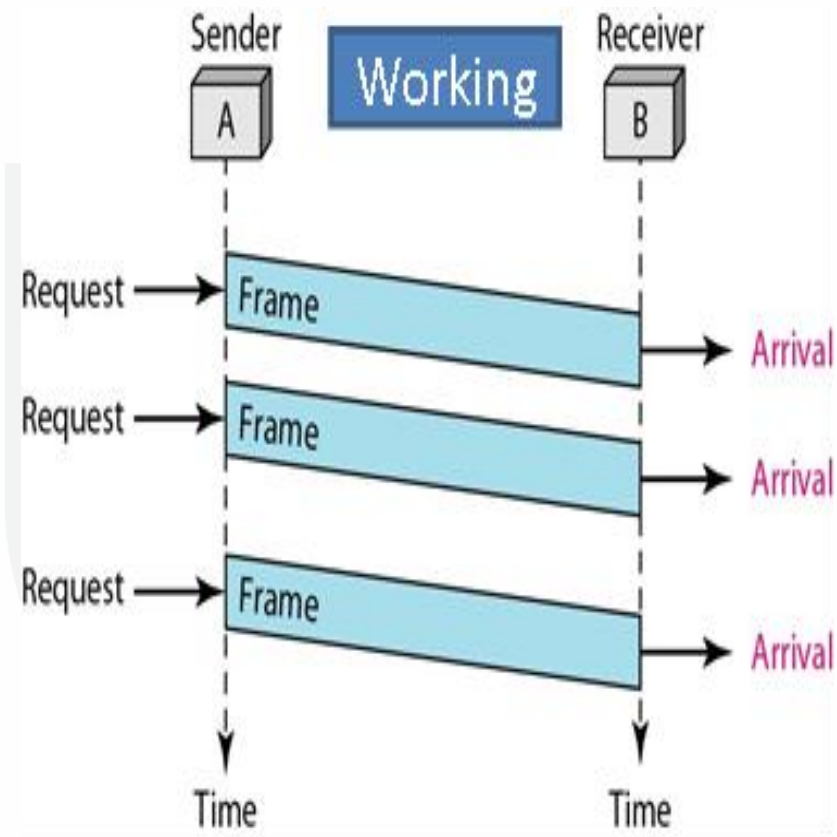
Flow Control Protocol for Data Link Layer



NOISELESS CHANNELS : Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted. We introduce two protocols for this type of channel. Simplest & Stop- and-Wait Protocol

NoiseLess Chanel : Simplex Protocol.....





NoiseLess Chanel : Simplex Protocol

•Design of This protocol is simple : At sender End, when ever a Request From Network Layer arrives, data receives from network layer convert into frames and give to physical layer, physical layer send frames bit by bit on transmission media.

•At Receiver End, Physical layer inform to network layer that, data is received.. then Data link layer receive the data bit by bit from physical layer and organize this data into frame

•In Above Figure, shows an example of communication using this protocol. It is very simple. The sender sends a sequence of frames without even thinking about the receiver. To send three frames, three events occur at the sender site and three events at the receiver site. Note that the data frames are shown by tilted boxes; the height of the box defines the transmission time difference between the first bit and the last bit in the frame.

NOISY CHANNELS: Three Protocols

• Here, For noisy channels, We discuss three protocols in this section that use error control.

1.Stop-and-Wait Automatic Repeat Request

2.Go-Back-N Automatic Repeat Request

3.Selective Repeat Automatic Repeat Request

NOISY CHANNELS: Stop-and-Wait (ARQ)

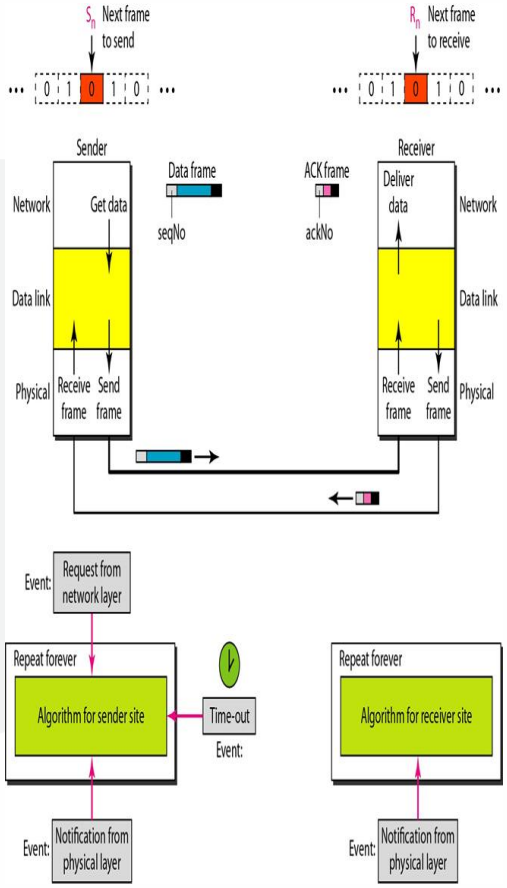
- Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.

- In Stop-and-Wait ARQ, we use sequence numbers to number the frames. The sequence numbers are based on modulo-2 arithmetic.

- In Stop-and-Wait ARQ, the acknowledgment number always announces in modulo-2 arithmetic the sequence number of the next frame expected.

NOISY CHANNELS: Stop-and-Wait (ARQ)

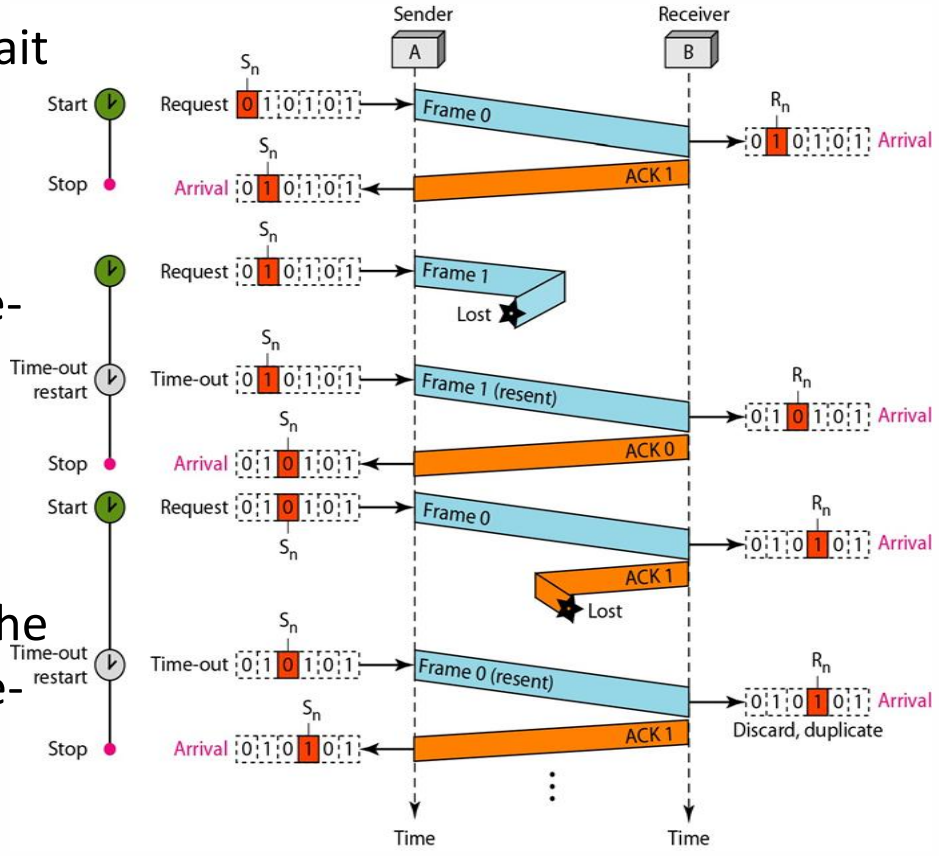
- Here At Sender end, data received from Network Layer, converted into Frame, Each Frame define Seq. No. and one frame given to physical layer, physical layer transmit data bit by bit on transmission media.
- At other end, receiving end, physical layer receive data bit by bit and inform to data link layer, data link layer organize bit into frame and error detecting code calculated for data and verify. If trailer code and calculated code match then data link layer send data ACK frame to physical layer and then physical layer send ack frame to sender.
- At sender end ack is received within time limit then next frame is prepared for transmission with next seq. no otherwise current frame is transmitted with same seq no. again.



NOISY CHANNELS: Stop-and-Wait (ARQ)

Figure shows an example of Stop- and-Wait ARQ

- Frame 0 is sent and acknowledged.
- Frame 1 is lost and resent after the time-out. The resent frame 1 is acknowledged and the timer stops.
- Frame 0 is sent and acknowledged, but the acknowledgment is lost.
- The sender has no idea if the frame or the acknowledgment is lost, so after the time-out, it resends frame 0, which is acknowledged.



Effectiveness of Stop and Wait protocol

- Assume that, in a Stop-and-Wait ARQ system, the bandwidth of the channel (Transmission Line) is 1 Mbps, and 1 bit takes 20 ms to make a round trip. What is the bandwidth-delay product? If the sender has data frames consist only 1000 bits in length, what is the utilization percentage of the link?
- The bandwidth-delay product is $(1 \times 10^6) \times (20 \times 10^{-3}) = 20{,}000 \text{ bits}$
- The system can send 20,000 bits during the time it takes for the data to go from the sender to the receiver and then back again. However, the system sends only 1000 bits. We can say that the link utilization is only 1000/20,000, or 5 percent. For this reason, for a link with a high bandwidth or long delay, the use of Stop- and-Wait ARQ wastes the capacity of the link.
- So stop-and-wait protocol is not suitable for a network which has higher bandwidth or high round trip time.

NOISY CHANNELS: Sliding Window Protocol

- Sliding window protocols are data link layer protocols for reliable and sequential delivery of data frames.
- The sliding window is also used in Transmission Control Protocol.
- In this protocol, multiple frames can be sent by a sender at a time before receiving an acknowledgment from the receiver.
- The term sliding window refers to the imaginary boxes to hold frames.
- Sliding window method is also known as windowing.

NOISY CHANNELS: Sliding Window Protocol

Working Principle

• In these protocols, the sender has a buffer called the sending window and the receiver has buffer called the receiving window.

• The sequence numbers are numbered as modulo-n. For example, if the sending window size is 4, then the sequence numbers will be 0, 1, 2, 3, 0, 1, 2, 3, 0, 1, and so on. The number of bits in the sequence number is 2 to generate the binary sequence 00, 01, 10, 11.

• The size of the receiving window is the maximum number of frames that the receiver can accept at a time. It determines the maximum number of frames that the sender can send before receiving acknowledgment.

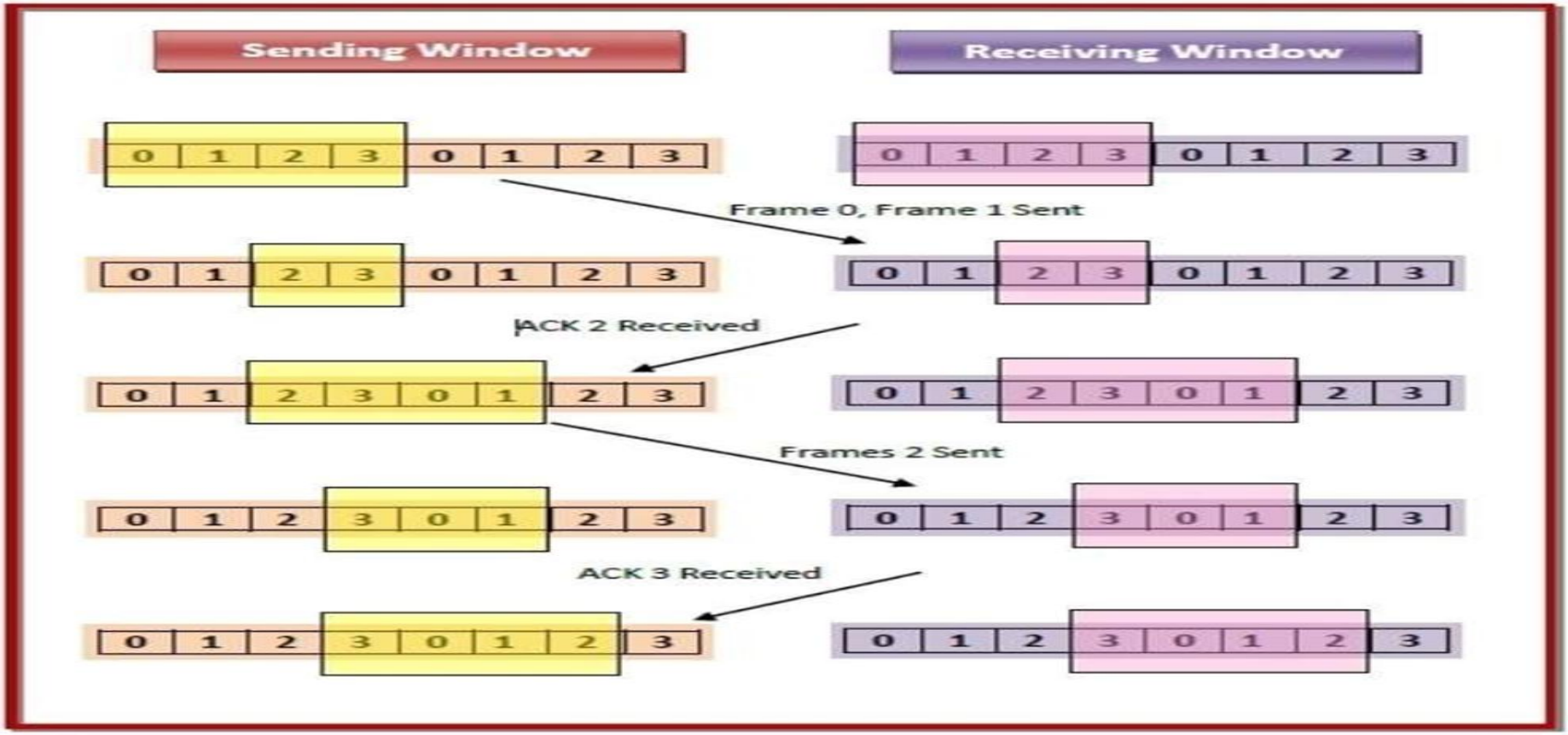
NOISY CHANNELS: Sliding Window Protocol

Example

Suppose that we have sender window and receiver window each of size 4. So the sequence numbering of both the windows will be 0,1,2,3,0,1,2 and so on.

A diagram shows the positions of the windows after sending the frames and receiving acknowledgments

NOISY CHANNELS: Sliding Window Protocol



NOISY CHANNELS: Sliding Window Protocol

Types of Sliding Window Protocols

The Sliding Window ARQ (Automatic Repeat reQuest) protocols are of two categories −

Sliding Window Protocol Efficiency

* Sliding window Protocol.

Window size = 1 + 2a

T_t sec ——— 1 sec. Packet

1 sec ——— $\frac{1}{T_t}$ Packet

$T_t + 2T_p$ ——— $\frac{T_t + 2T_p}{T_t}$ Packet

$= \frac{T_t}{T_t} + \frac{2T_p}{T_t}$ Packet

$= 1 + 2\frac{T_p}{T_t}$ Packet

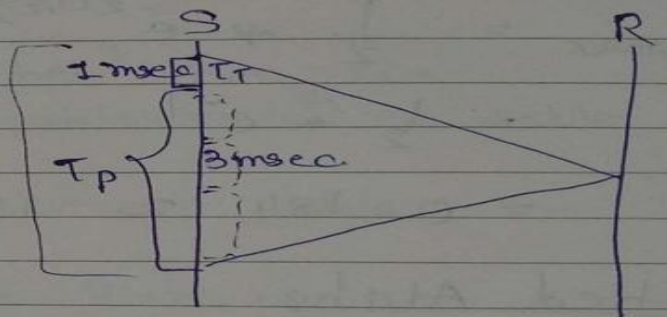
$= 1 + 2a$

ex → $T_t = 1$ msec

$T_p = 1.5$ msec.

efficiency $\eta = \frac{1}{1 + 2(1.5)}$

$\eta = \frac{1}{4}$



Sliding Window : Go Back N (ARQ)

- Go-Back-N ARQ protocol is also known as Go-Back-N Automatic Repeat Request. It is a data link layer protocol that uses a sliding window method. In this, if any frame is corrupted or lost, all subsequent frames have to be sent again.[2]

- The size of the sender window is N in this protocol. For example, Go-Back-8, the size of the sender window, will be 8. The receiver window size is always 1.

- If the receiver receives a corrupted frame, it cancels it. The receiver does not accept a corrupted frame. When the timer expires, the sender sends the correct frame again.[2]

Sliding Window : Go Back N (ARQ)

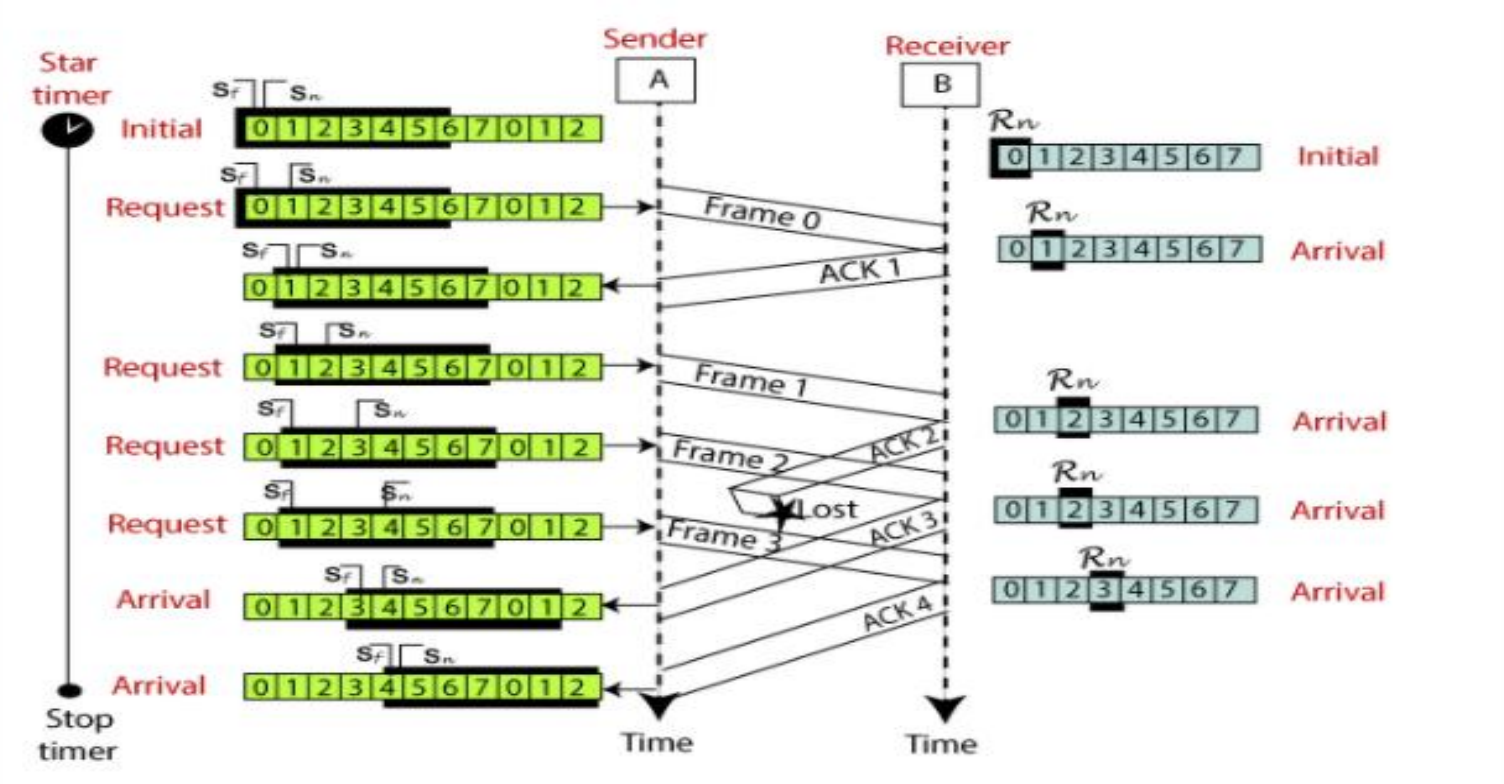


Fig. 2:3 example of Go-Back-N ARQ [2]

Sliding Window : Selective Repeat ARQ

- Selective Repeat ARQ is also known as the Selective Repeat Automatic Repeat Request.
- It is a data link layer protocol that uses a sliding window method.
- The Go-back-N ARQ protocol works well if it has fewer errors. But if there is a lot of error in the frame, lots of bandwidth loss in sending the frames again. So, we use the Selective Repeat ARQ protocol.
- In this protocol, the size of the sender window is always equal to the size of the receiver window. The size of the sliding window is always greater than 1.

Sliding Window : Selective Repeat ARQ

- If the receiver receives a corrupt frame, it does not directly discard it.
- It sends a negative acknowledgment to the sender.
- The sender sends that frame again as soon as on the receiving negative acknowledgment.
- There is no waiting for any time-out to send that frame.

Sliding Window : Go Back N (ARQ)

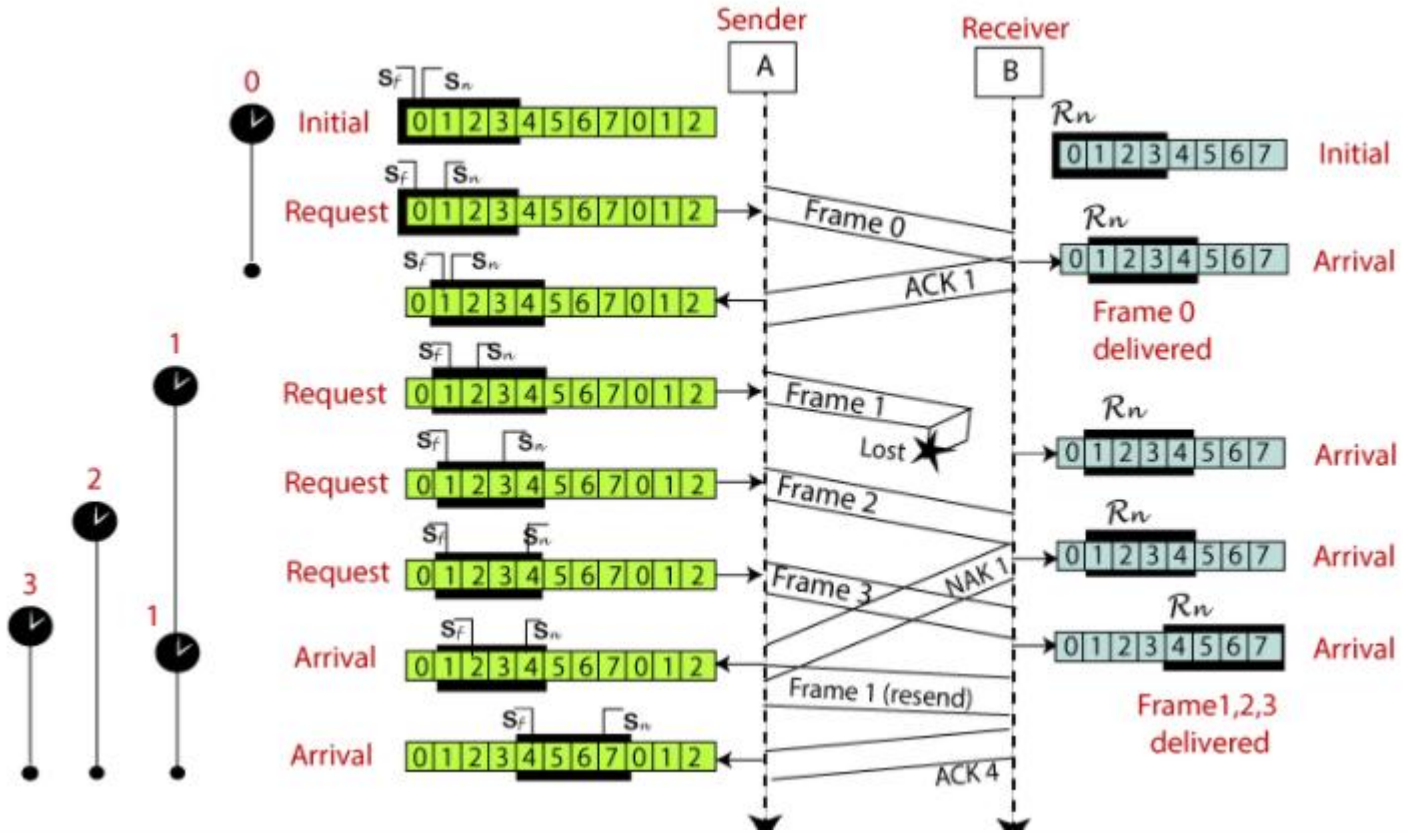


Fig. 2:4 example of the Selective Repeat ARQ protocol [2]

Parul®
University

DIGITAL LEARNING CONTENT

Piggybacking

Piggybacking

- For bidirectional data flow, the control information (ACK, NAK) can be transmitted along with data frames to improve the efficiency. This technique is called as Piggybacking.

Working Principle:

- Suppose that there are two communication stations X and Y. The data frames transmitted have an acknowledgment field, ack field that is of a few bits length. Additionally, there are frames for sending acknowledgments, ACK frames. The purpose is to minimize the ACK frames.[3]

- The three principles governing piggybacking when the station X wants to communicate with station Y are:

Piggybacking

- If station X has both data and acknowledgment to send, it sends a data frame with the ack field containing the sequence number of the frame to be acknowledged.[3]
- If station X has only an acknowledgment to send, it waits for a finite period of time to see whether a data frame is available to be sent. If a data frame becomes available, then it piggybacks the acknowledgment with it. Otherwise, it sends an ACK frame.[3]
- If station X has only a data frame to send, it adds the last acknowledgment with it. The station Y discards all duplicate acknowledgments. Alternatively, station X may send the data frame with the ack field containing a bit combination denoting no acknowledgment.

Example of Piggybacking:

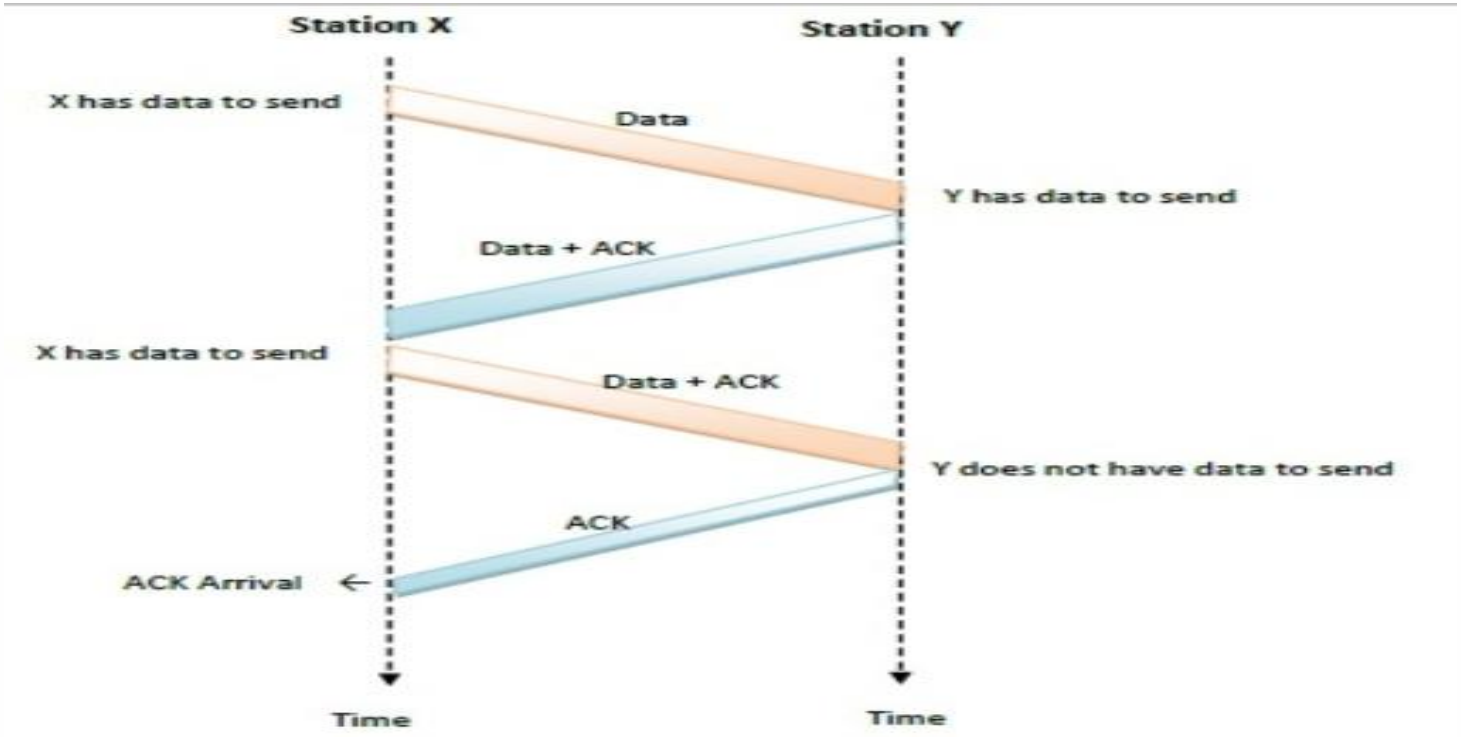


Fig. 2:5 example of piggybacking [3]

Parul®
University

DIGITAL LEARNING CONTENT

Multiple Access protocols

Multiple Access Protocols

- The Data Link Layer is responsible for transmission of data between two nodes. Its main functions are:

    1. Data Link Control
    2. Multiple Access Control

Data Link Control

- The data link control is responsible for reliable transmission of message over transmission channel by using techniques like framing, error control and flow control.

Multiple Access Control

- If there is a dedicated link between the sender and the receiver then data link control layer is sufficient.
- However if there is no dedicated link present then multiple stations can access the channel simultaneously.
- Hence multiple access protocols are required to decrease collision and avoid crosstalk.
- For example, in a classroom full of students, when a teacher asks a question and all the students (or stations) start answering simultaneously (send data at same time) then a lot of chaos is created(data overlap or data lost) then it is the job of the teacher (multiple access protocols) to manage the students and make them answer one at a time.[5]

Multiple Access Control

- Thus, protocols are required for sharing data on non dedicated channels. Multiple access protocols can be subdivided further as:

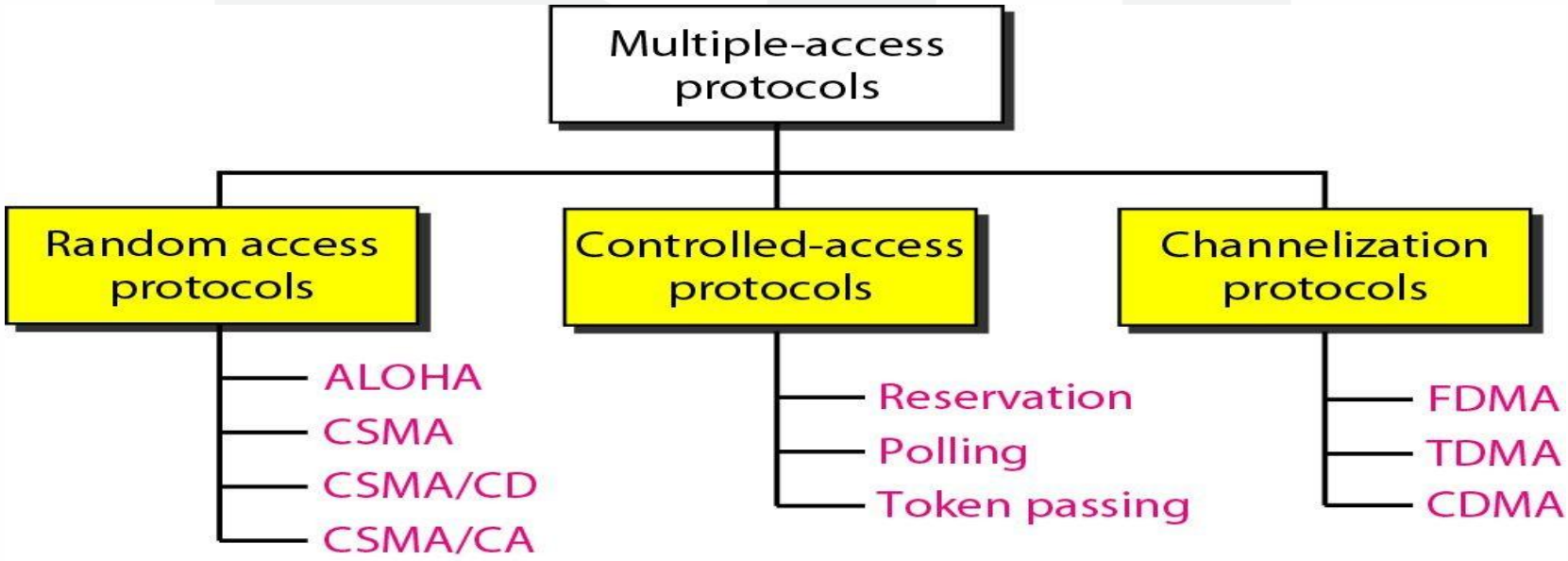


Fig. 2:6 Types of Multiple Access protocols [5]

Multiple Access Control

There are two types of network links:

- A **point-to-point link** consists of a single sender at one end of the link and a single receiver at the other end of the link.

- A **broadcast link**, can have multiple sending and receiving nodes all connected to the same, single, shared broadcast channel.
- The term broadcast is used here because when any one node transmits a frame, the channel broadcasts the frame and each of the other nodes receives a copy.

Random Access Protocols:

- In this, all stations have same superiority that is no station has more priority than another station. Any station can send data depending on medium's state(idle or busy).

- It has two features:
  1.There is no fixed time for sending data
  2.There is no fixed sequence of stations sending data

- Examples of random access MAC (Medium Access Control) protocols
  1.Pure ALOHA
  2.Slotted ALOHA
  3.CSMA, CSMA/CD, CSMA/CA

Pure ALOHA Protocol

- Frames are transmitted at completely arbitrary times.
- The throughput of the Pure ALOHA is maximized when the frames are of uniform length.
- The formula to calculate the throughput of the Pure ALOHA is [5]

  S-=G*e-2G
- The throughput is maximum when G=1/2 which is 18% of the total transmitted data frames.[5]

Pure ALOHA Protocol

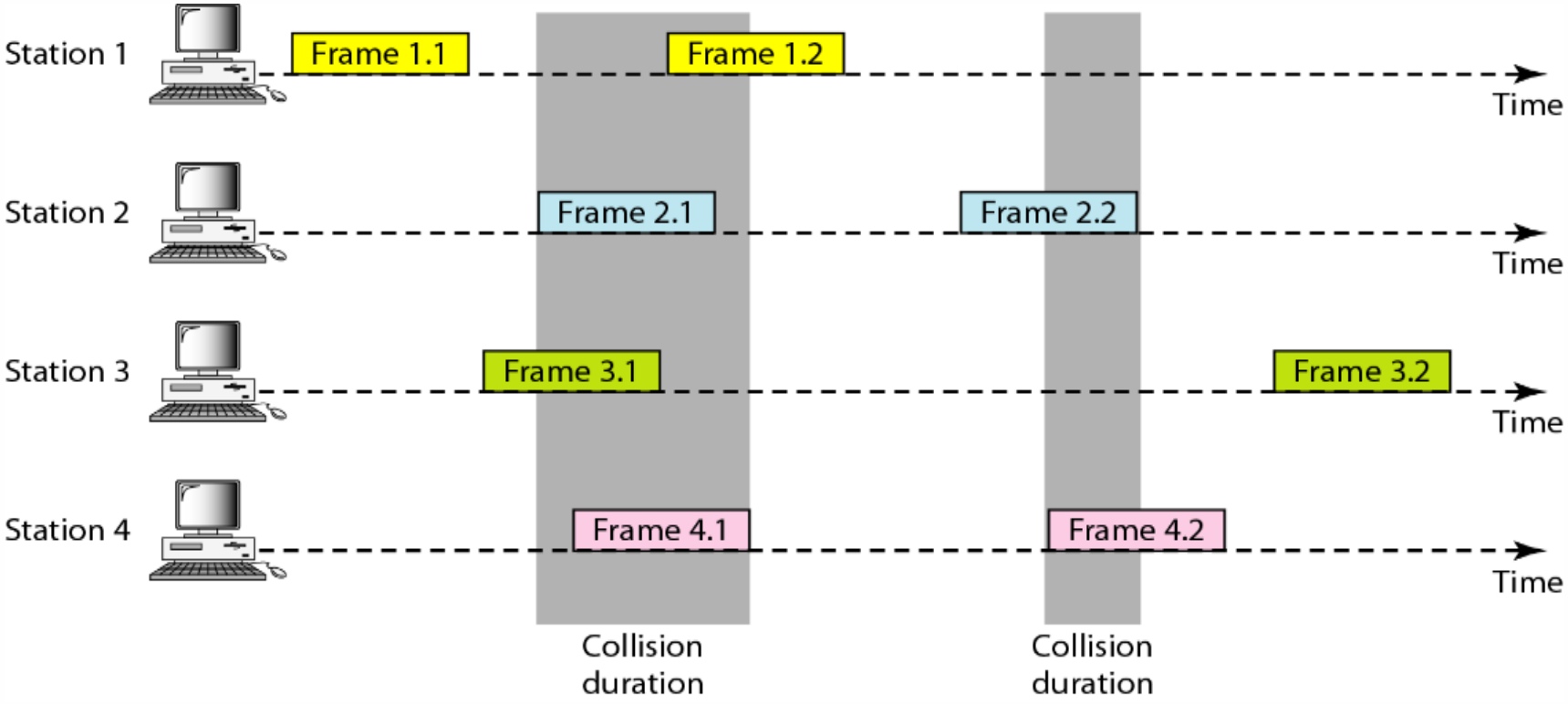


Fig. 2:7 Process of Pure ALOHA Protocol [5]

Pure ALOHA Protocol

- It allows users to transmit whenever they have data to be sent.
- Senders wait to see if a collision occurred (after whole message has been sent).
- If collision occurs, each station involved waits a random amount of time then tries again.
- Systems in which multiple users share a common channel in a way that can lead to conflicts are widely known as contention systems.
- Whenever two frames try to occupy the channel at the same time, there will be a collision and both will be garbled.
- If the first bit of a new frame overlaps with just the last bit of a frame almost finished, both frames will be totally destroyed and both will have to be retransmitted later.

Pure ALOHA Protocol

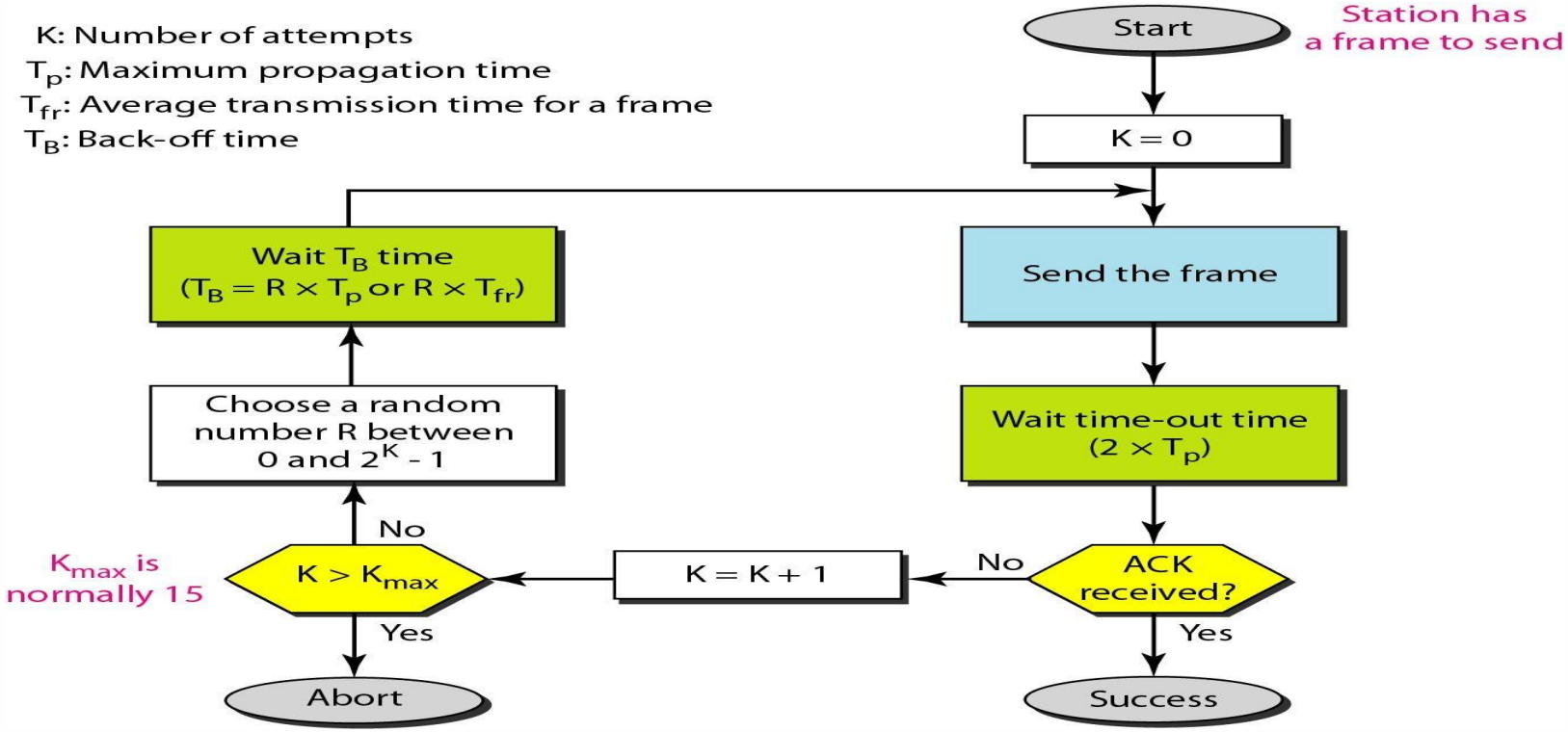


Fig. 2:8 Flowchart of Pure ALOHA Protocol [5]

Vulnerable time for pure ALOHA protocol

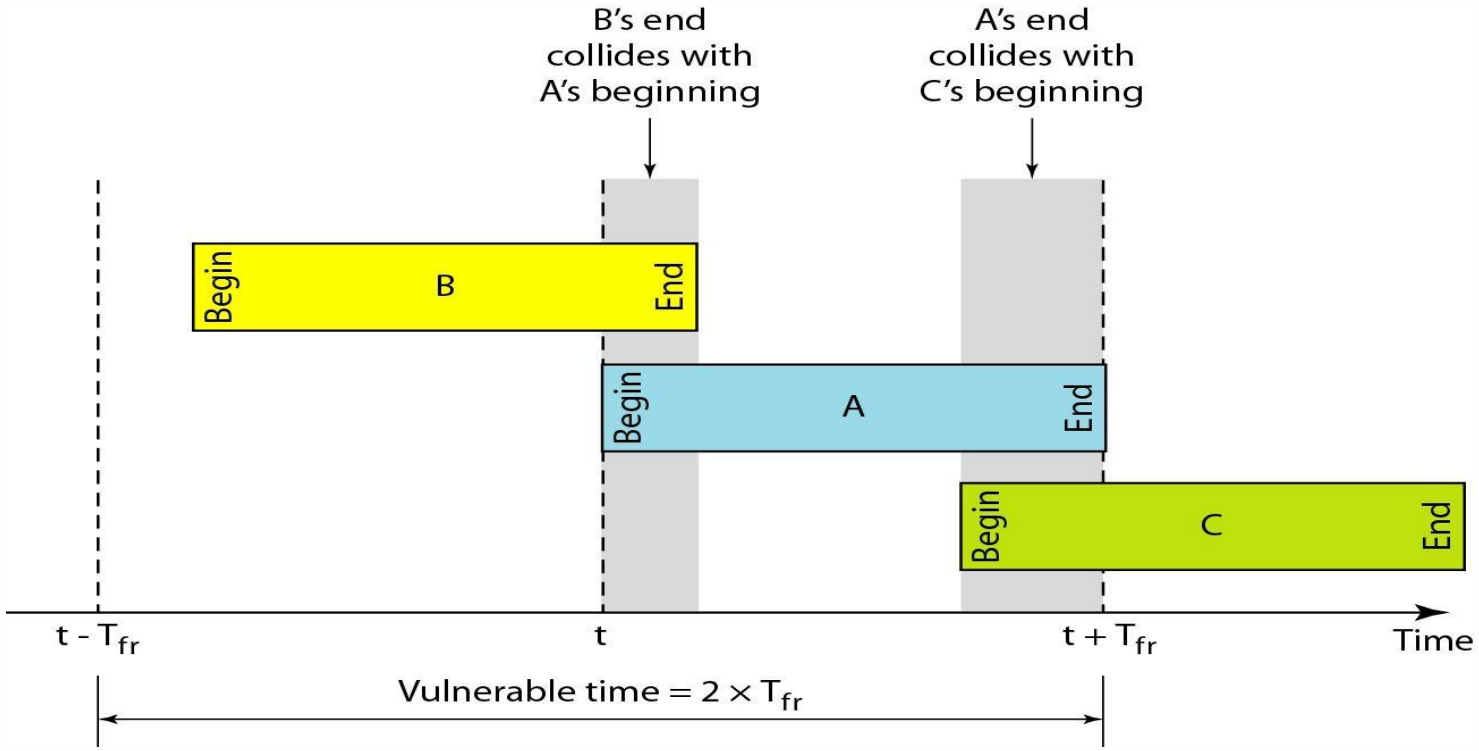


Fig. 2:9 Vulnerable time for Pure ALOHA Protocol [5]

Slotted Aloha Protocol

- It was invented to improve the efficiency of pure ALOHA as chances of collision in pure ALOHA are very high.
- The time of the shared channel is divided into discrete intervals called slots.
- The stations can send a frame only at the beginning of the slot and only one frame is sent in each slot.
- If any station is not able to place the frame onto the channel at the beginning of the slot then the station has to wait until the beginning of the next time slot.

Slotted Aloha Protocol

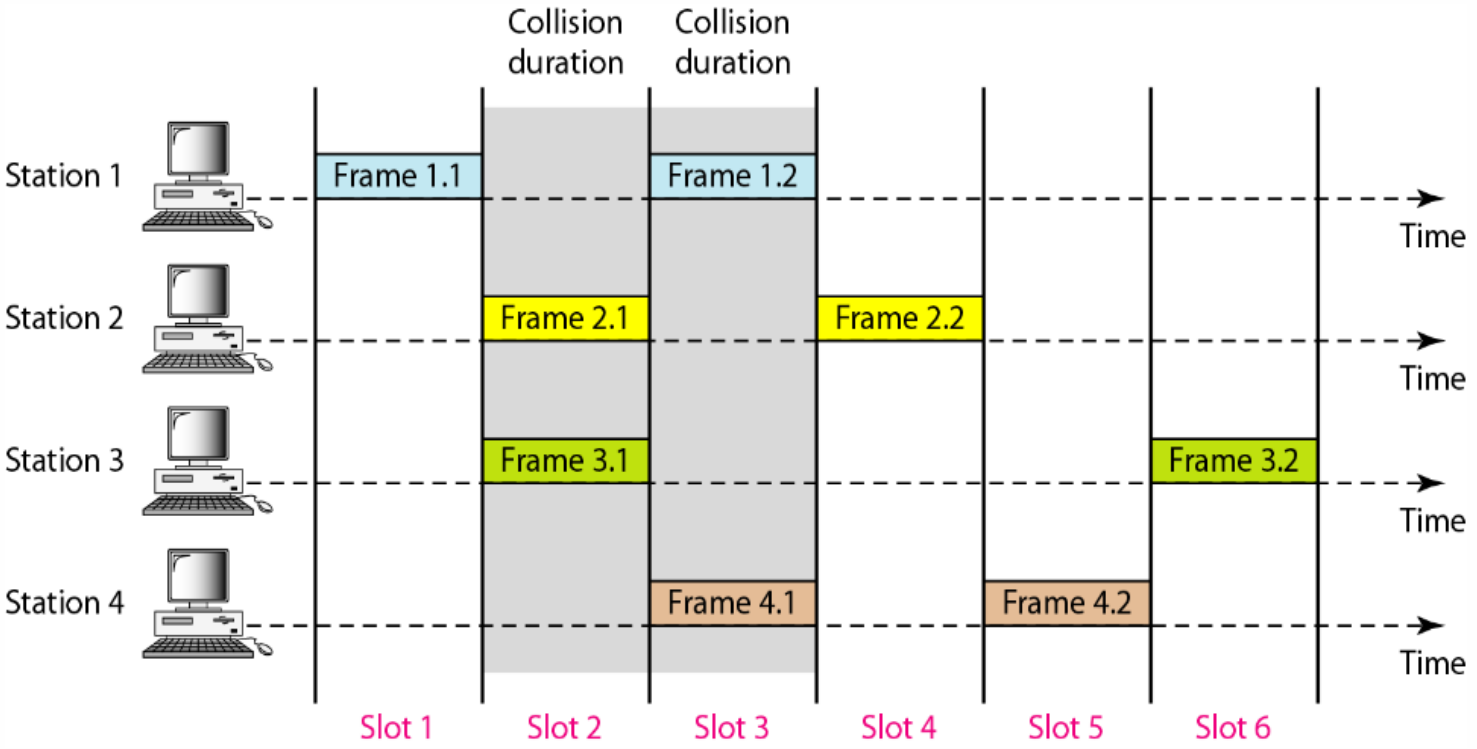


Fig. 2:10 Process of Pure ALOHA Protocol [5]

Vulnerable time for Slotted Aloha Protocol

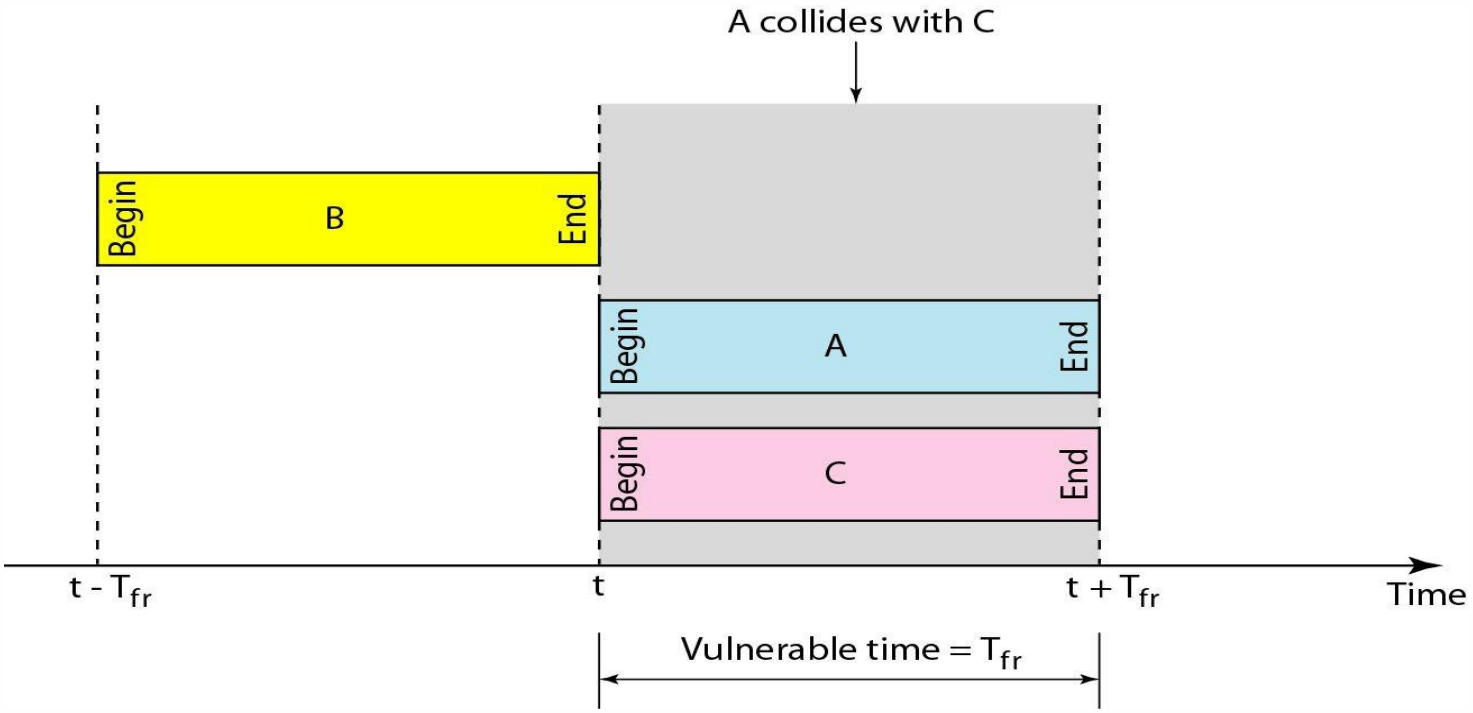


Fig. 2:11 Vulnerable time for Slotted ALOHA Protocol [5]

Slotted Aloha Protocol

- If any station is not able to place the frame onto the channel at the beginning of the slot then the station has to wait until the beginning of the next time slot.
- The formula to calculate the throughput of the Slotted ALOHA is

$$S=G*e-G$$

- The throughput is maximum when G=1 which is 37% of the total transmitted data frames.

- 37% of the time slot is empty, 37% successes and 26% collision.

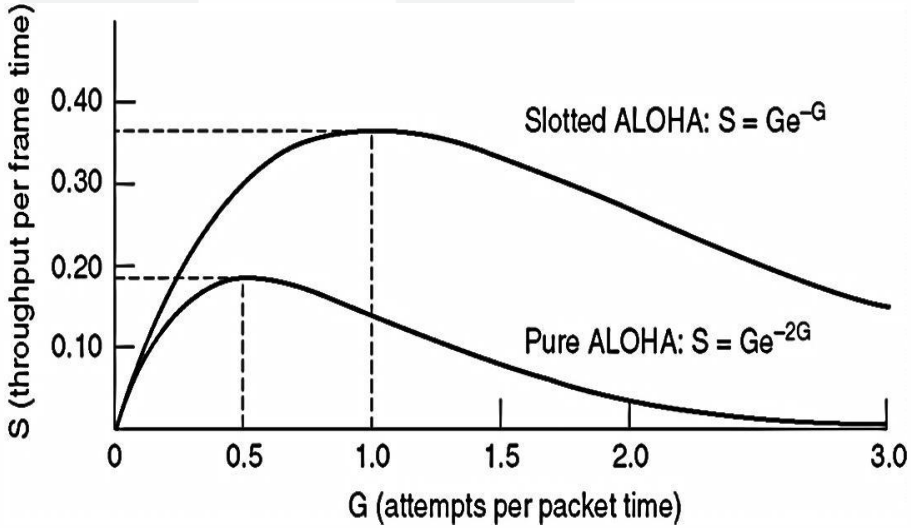


Fig. 2:12 Vulnerable time for Slotted ALOHA Protocol [5]

CSMA

- It was invented to improve the efficiency of pure ALOHA as chances of collision in pure ALOHA are very high.
- The time of the shared channel is divided into discrete intervals called slots.
- The stations can send a frame only at the beginning of the slot and only one frame is sent in each slot.
- If any station is not able to place the frame onto the channel at the beginning of the slot then the station has to wait until the beginning of the next time slot.

CSMA/CD (CSMA with Collision Detection)

- If two stations sense the channel to be idle and begin transmitting simultaneously, they will both detect the collision almost immediately.
- Rather than finish transmitting, they should abruptly stop transmitting as soon as the collision is detected.
- Quickly terminating damaged frames saves time and bandwidth.
- This protocol, known as CSMA/CD (CSMA with Collision Detection) is widely used on LANs in the MAC sub layer.

Flow diagram for the CSMA/CD

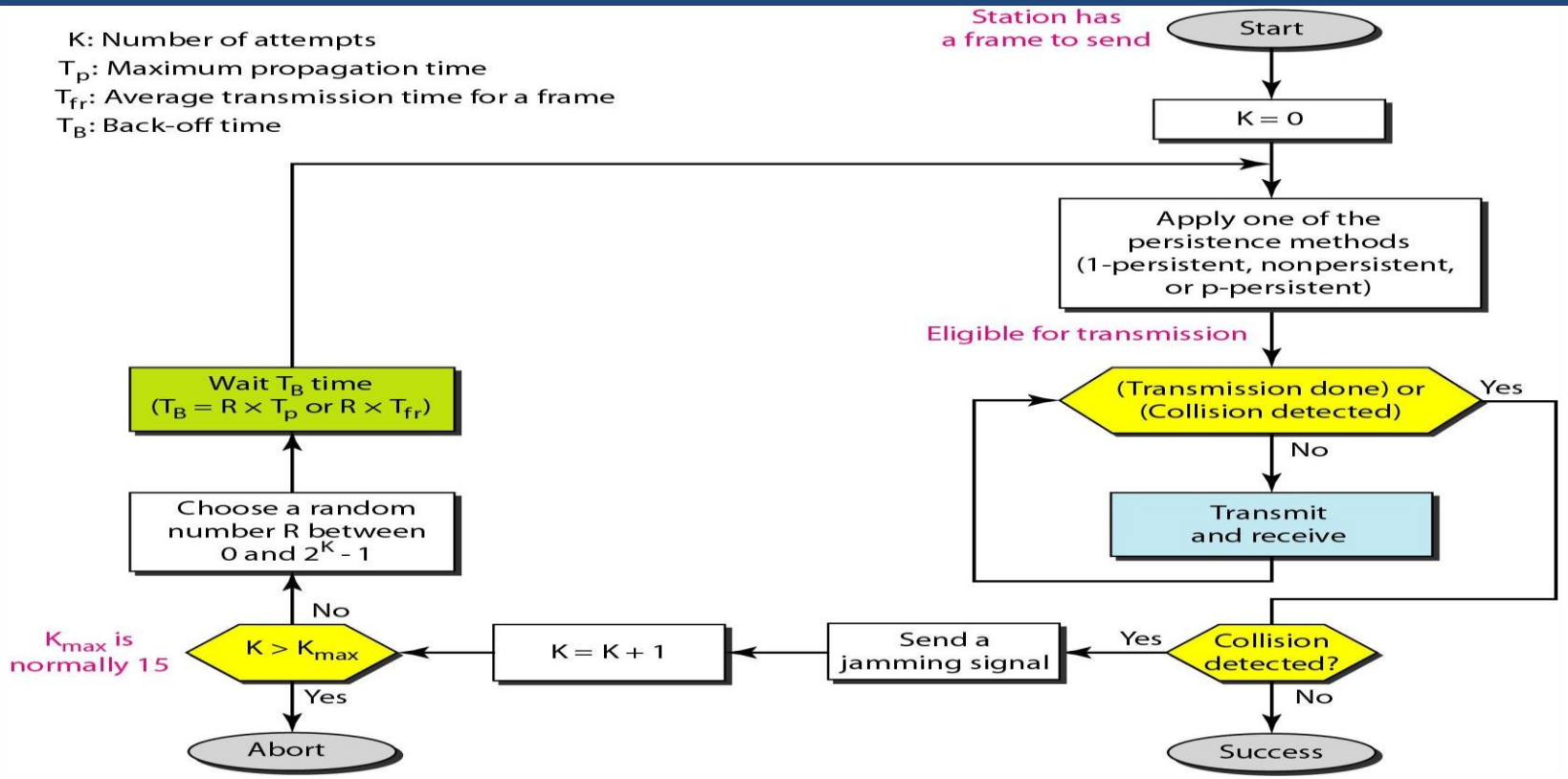


Fig. 2:13 Flow diagram for CSMA/CD[5]

CSMA/CA

- The basic idea behind CSMA/CD is that a station needs to be able to receive while transmitting to detect a collision
- When there is no collision, the station receives one signal: its own signal. When there is a collision, the station receives two signals: its own signal and the signal transmitted by a second station
- In wired network it is easy to detect collision due to significant energy, while in wireless the case is difficult
- In wireless channel collision needs to be avoided rather than detection

Flow diagram for CSMA/CA

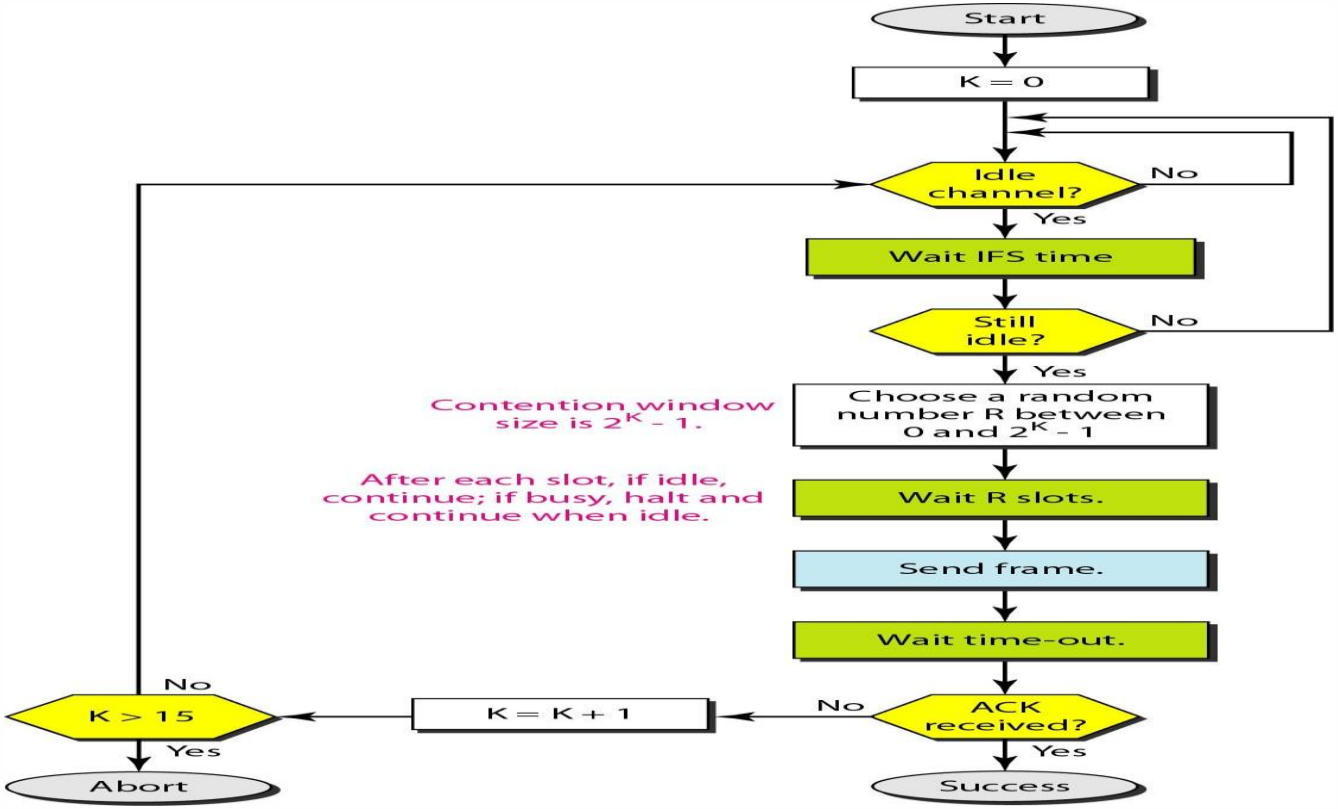


Fig. 2:14 Flow diagram for CSMA/CA[5]

References

1. https://www.tutorialspoint.com/sliding-window-protocol
2. https://www.javatpoint.com/sliding-window-protocol
3. https://www.tutorialspoint.com/what-is-piggybacking-in-networking
4. "Data and Computer Communication" by William Stallings
5. "Data Communication and Networking" by Behrouz A Forouzan
6. " Computer Networks" by Andrew S Tanenbaum

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in