UNIT-2
The Data Link Layer, Access Networks, and LANs

1. Data Link Layer Design Issues

The data link layer uses the services of the physical layer to send and receive bits over communication channels. It has a number of functions, including:

1. Providing a well-defined service interface to the network layer.
2. Dealing with transmission errors.
3. Regulating the flow of data so that slow receivers are not swamped by fast senders.

The data link layer takes the packets it gets from the network layer and encapsulates them into frames for transmission.

Each frame contains three parts: frame header, payload field for holding packets, and frame trailer.

Fig: Frame

Header: The header consists of control information whose role is to guide the whole frame to its correct destination.

Frame header includes Source and Destination address field,Physical Link Control field, Flow control field, and Congestion Control field etc.,

Trailer: Data-link Layer adds also a trailer at the end of each frame. The trailer is responsible for ensuring that frames are received intact or undamaged.

Services Provided to the Network Layer

The data-link layer is located between the physical and the network layers. The data link layer provides services to the network layer; it receives services from the physical layer.

The three major types of services offered by data link layer are:

1. Unacknowledged connectionless service.
2. Acknowledged connectionless service.
3. Acknowledged connection-oriented service.

1. Unacknowledged Connectionless Service

(a) In this type of service source machine sends frames to destination machine but the destination machine does not send any acknowledgement of these frames back to the source. Hence it is called unacknowledged service.

(b) There is no connection establishment between source and destination machine before data transfer or release after data transfer. Therefore, it is known as connectionless service.

(c) There is no error control *i.e.* if any frame is lost due to noise on the line; no attempt is made to recover it.

(d) This type of service is used when error rate is low.

(e) It is suitable for real time traffic such as speech.

2. Acknowledged Connectionless Service

(a) In this service, neither the connection is established before the data transfer nor is it released after the data transfer between source and destination.

(b) When the sender sends the data frames to destination, destination machine sends back the acknowledgement of these frames.

(c) This type of service provides additional reliability because source machine retransmit the frames if it does not receive the acknowledgement of these frames within the specified time.
(d) This service is useful over unreliable channels, such as wireless systems.

3. Acknowledged Connection - Oriented Service
(a) This service is the most sophisticated service provided by data link layer to network layer.
(b) It is connection-oriented. It means that connection is establishment between source & destination before any data is transferred.
(c) In this service, data transfer has three distinct phases: -
(i) Connection establishment
(ii) Actual data transfer
(iii) Connection release
(d) Here, each frame being transmitted from source to destination is given a specific number and is acknowledged by the destination machine.
(e) All the frames are received by destination in the same order in which they are sending by the source.

 Framing:A packet at the data-link layer is normally called a frame.The first service provided by the data-link layer is framing. The data-link layer at each node needs to encapsulate the datagram (packet received from the network layer) in a frame before sending it to the next node. We have shown that a frame may have both a header and a trailer. Different data-link layers have different formats for framing.

 Frame Size
Frames can be of fixed or variable size. In ***fixed-size framing,*** there is no need for defining the boundaries of the frames; the size itself can be used as a delimiter. In ***variable-size framing***, we need a way to define the end of one frame and the beginning of the next.

Two approaches were used for this purpose: a ***character-oriented*** approach and a ***bit-oriented*** approach.

- ***Character-Oriented Framing:*** In *character-oriented (or byte-oriented) framing,* data to be carried are 8-bit characters from a coding system such as ASCII.
The *header*, which normally carries the source and destination addresses and other control information, and the *trailer*, which carries error detection redundant bits.
To separate one frame from the next, an 8-bit (1-byte) **flag** is added at the beginning and the end of a frame. Figure shows the format of a frame in a character-oriented protocol.

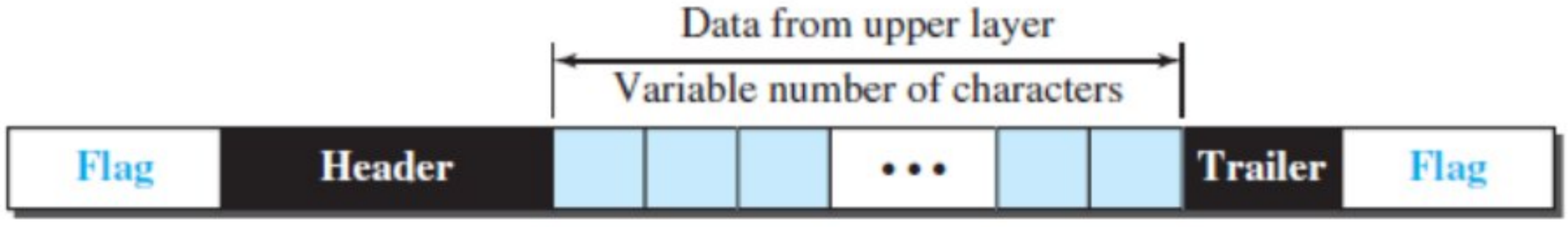


Fig:A frame in a character-oriented protocol

Character-oriented framing was popular when only text was exchanged by the data-link layers. The flag could be selected to be any character not used for text communication.
Now we send other types of information such as graphs, audio, and video; any character used for the flag could also be part of the information.

To solve this problem, a byte-stuffing strategy was added to character-oriented framing. In **byte stuffing** is the process of adding one extra byte whenever there is a flag or escape character in the text. The data section is stuffed with an extra byte. This byte is usually called the *escape character (ESC)* and has a predefined bit pattern.

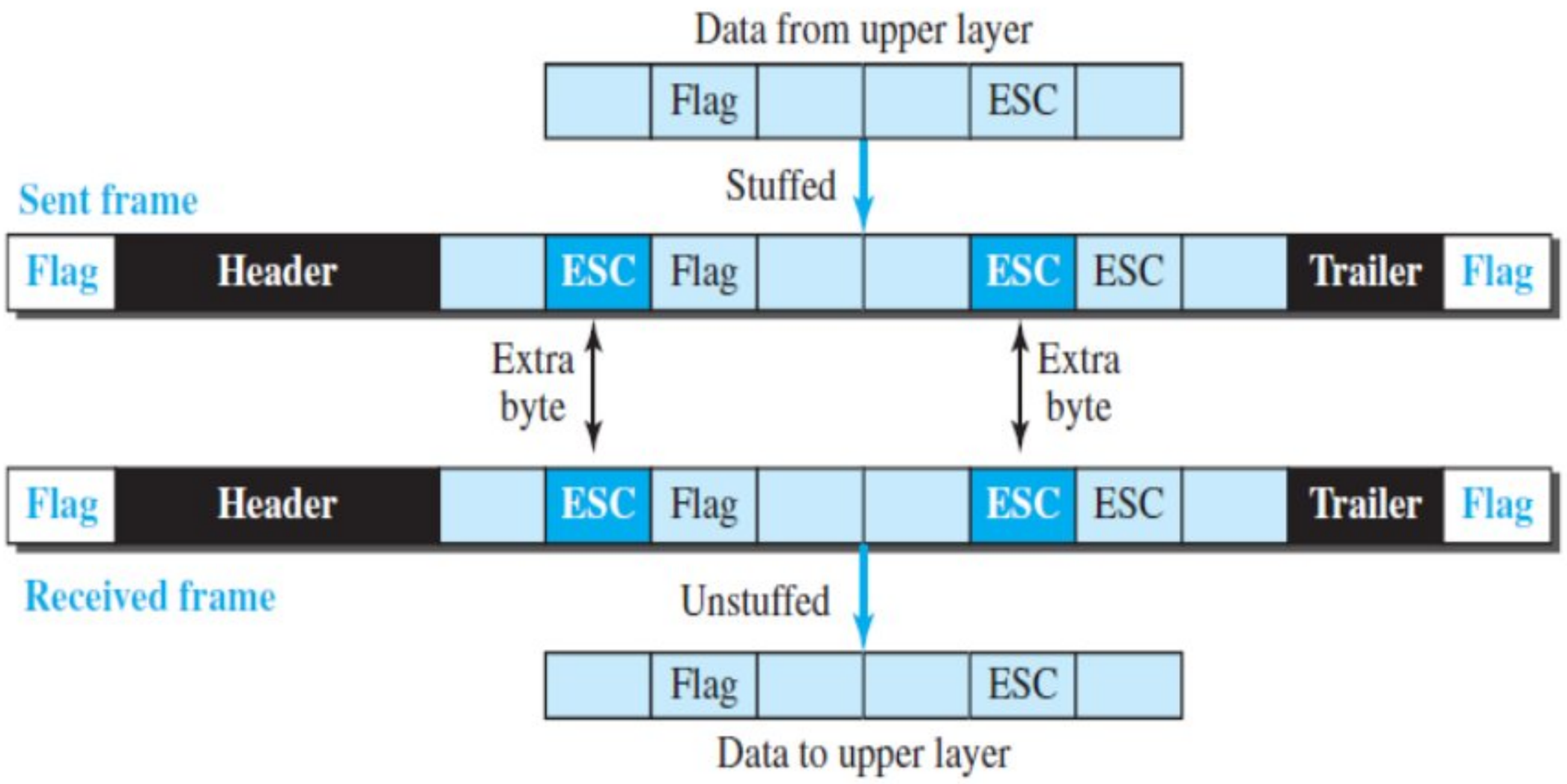


Fig:Byte stuffing and unstuffing

- ***Bit-Oriented Framing:*** In *bit-oriented framing,* the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on. However, in addition to headers (and possible trailers), we still need a delimiter to separate one frame from the other.

Most protocols use a special 8-bit pattern **flag, 01111110**, as the delimiter to define the beginning and the end of the frame, as shown in Figure.

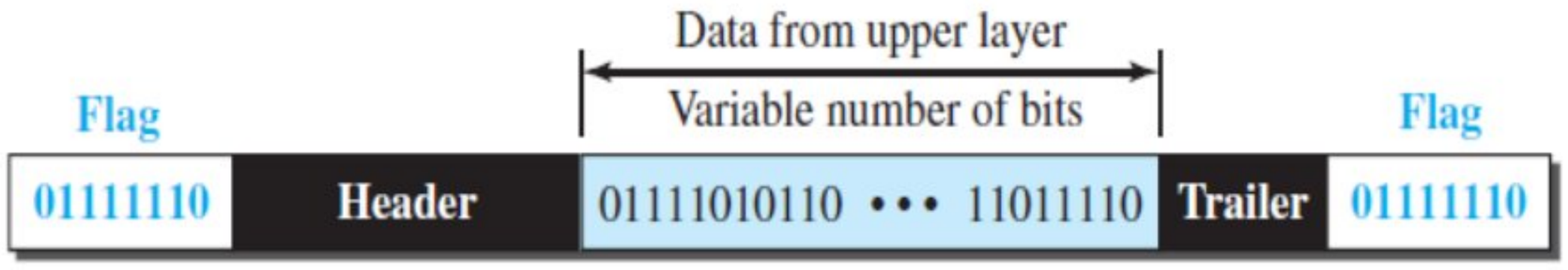


Fig: A frame in a bit-oriented protocol

If the flag pattern appears in the data, we need to somehow inform the receiver that this is not the end of the frame. We do this by stuffing 1 single bit to prevent the pattern from looking like a flag. The strategy is called **bit stuffing.** In bit stuffing, if 0 and five consecutive 1 bits are encountered, an extra 0 is added.

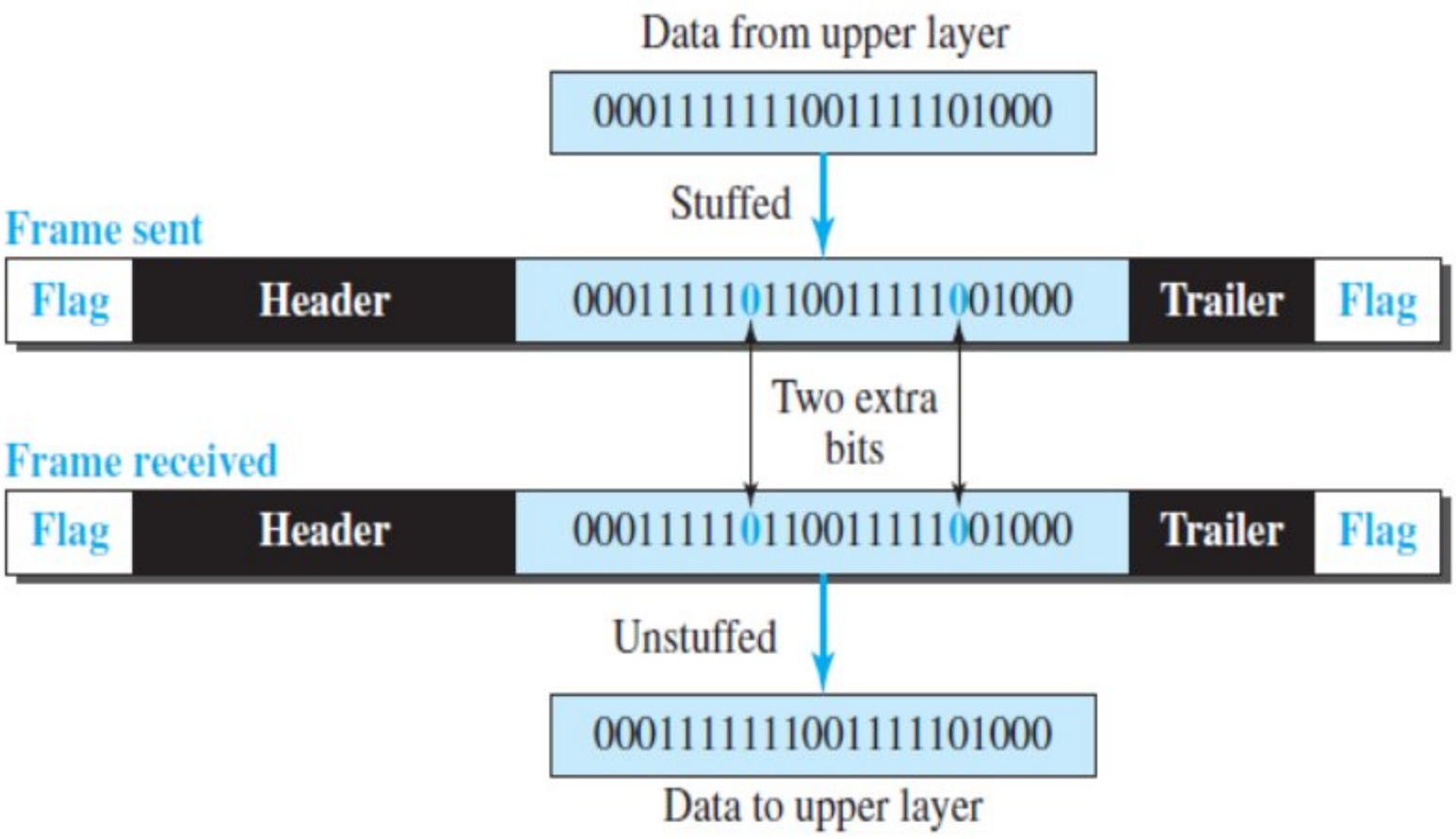


Fig: Bit stuffing and unstuffing

 Flow and Error Control: One of the responsibilities of the data-link control sublayer is flow and error control at the data-link layer.

Flow Control: Whenever an entity produces items and another entity consumes them, there should be a balance between production and consumption rates. If the items are produced faster than they can be consumed, the consumer can be overwhelmed and may need to discard some items. We need to prevent losing the data items at the consumer site.

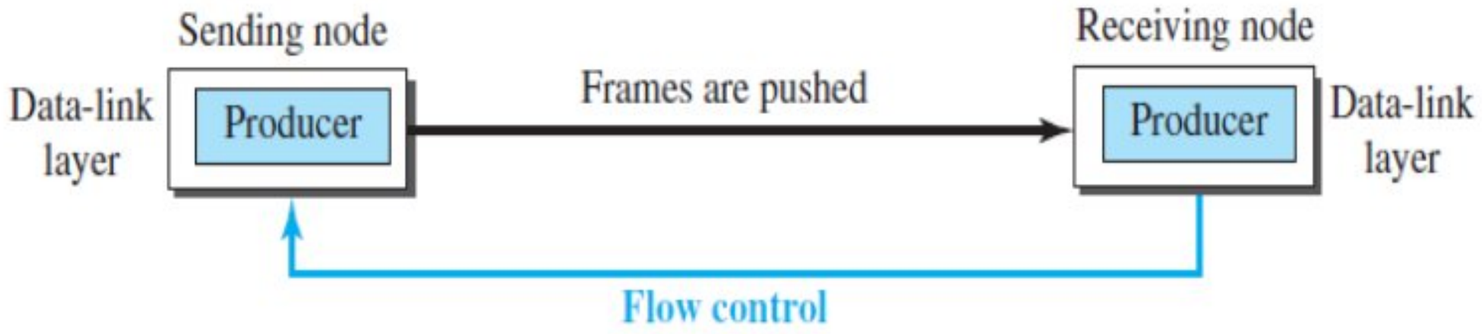


Fig: Flow control at the data-link layer

Buffers

Flow control can be implemented in several ways, one of the solutions is normally to use two *buffers*; one at the sending data-link layer and the other at the receiving data -link layer.

A buffer is a set of memory locations that can hold packets at the sender and receiver. The flow control communication can occur by sending signals from the consumer to the producer. When the buffer of the receiving data-link layer is full, it informs the sending data-link layer to stop pushing frames.

Error Control: we need to implement error control at the data-link layer to prevent the receiving node from delivering corrupted packets to its network layer. Error control at the data-link layer is normally very simple and implemented using one of the following two methods.

In both methods, a CRC is added to the frame header by the sender and checked by the receiver.

- In the first method, if the frame is corrupted, it is silently discarded; if it is not corrupted, the packet is delivered to the network layer. This method is used mostly in wired LANs such as Ethernet.

- In the second method, if the frame is corrupted, it is silently discarded; if it is not corrupted, an acknowledgment is sent (for the purpose of both flow and error control) to the sender.

2. Error Detection and Correction and Error-Detection and -Correction Techniques

Bit-level error detection and correction

Detecting and correcting the corruption of bits in a link-layer frame sent from one node to another physically connected neighboring node.
Data can be corrupted during transmission, some applications require that errors can be detected and corrected.

- **Redundancy:** The central concept in detecting or correcting errors is **redundancy**.

To be able to detect or correct errors, we need to send some extra bits with our data. These redundant bits are added by the sender and removed by the receiver. Their presence allows the receiver to detect or correct corrupted bits.

- **Detection versus Correction**

The correction of errors is more difficult than the detection. In **error detection**, we are only looking to see if any error has occurred. The answer is a simple yes or no. We are not even interested in the number of corrupted bits.
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 errors and the size of the message are important factors.

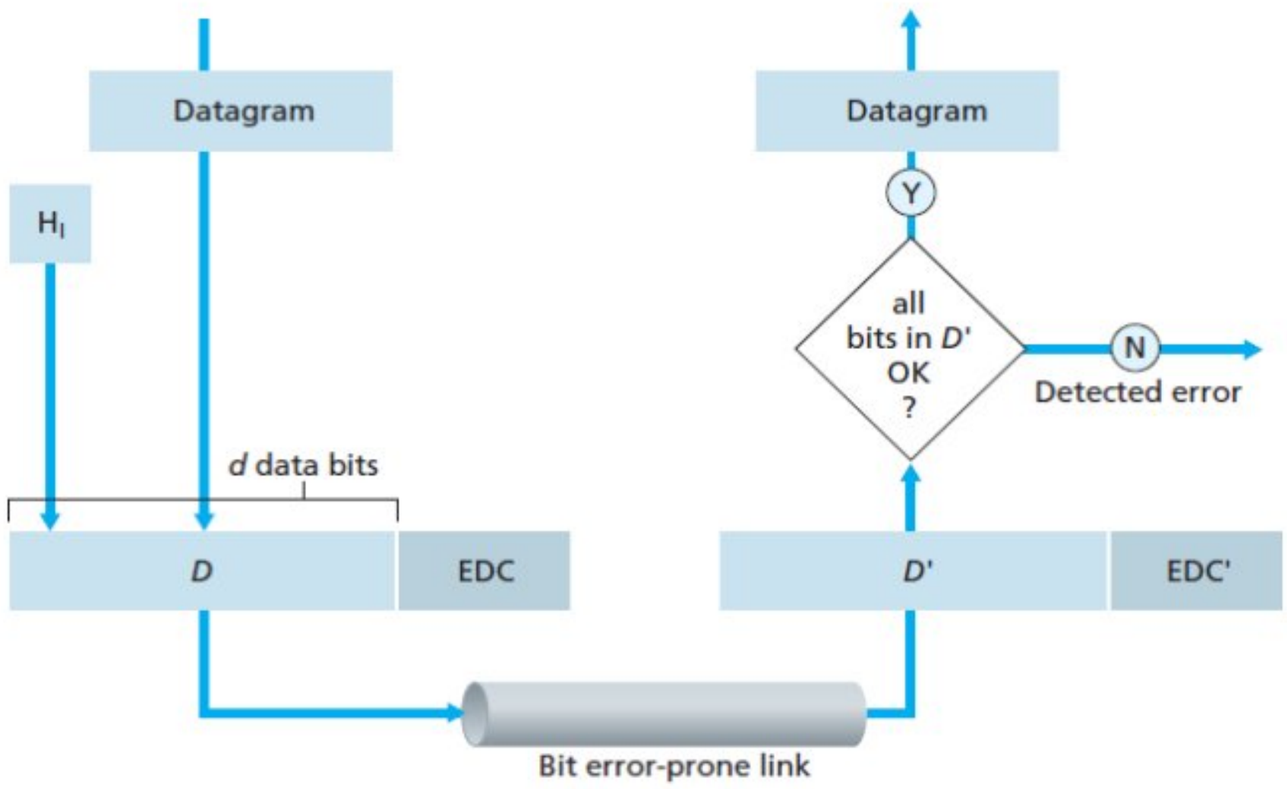


Fig: Error-detection and -correction scenario

We divide our message into blocks, each of *d* bits, called ***datawords***. We add *r* redundant bits to each block to make the length *n* = d+ *r*. The resulting *n*-bit blocks are called ***codewords***. How the extra *r* bits are chosen or calculated.

Error-Detection and -Correction Techniques

Parity Checks

The simplest form of error detection is the use of a single **parity bit**. Suppose that the information to be sent, *D* in Figure, has *d* bits. In an even parity scheme, the sender simply includes one additional bit and chooses its value such that the total number of 1s in the *d* + 1 bits (the original information plus a parity bit) is even.
For odd parity schemes, the parity bit value is chosen such that there is an odd number of 1s. Figure illustrates an even parity scheme, with the single parity bit being stored in a separate field.

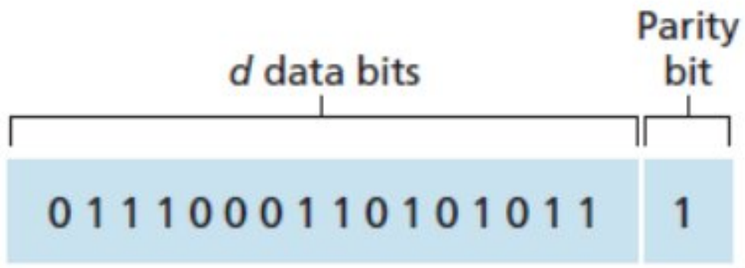


Fig: One-bit even parity

Receiver operation is also simple with a single parity bit. The receiver need only count the number of 1s in the received $d+1$ bits. If an odd number of 1- valued bits are found with an even parity scheme, the receiver knows that at least one bit error has occurred.

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

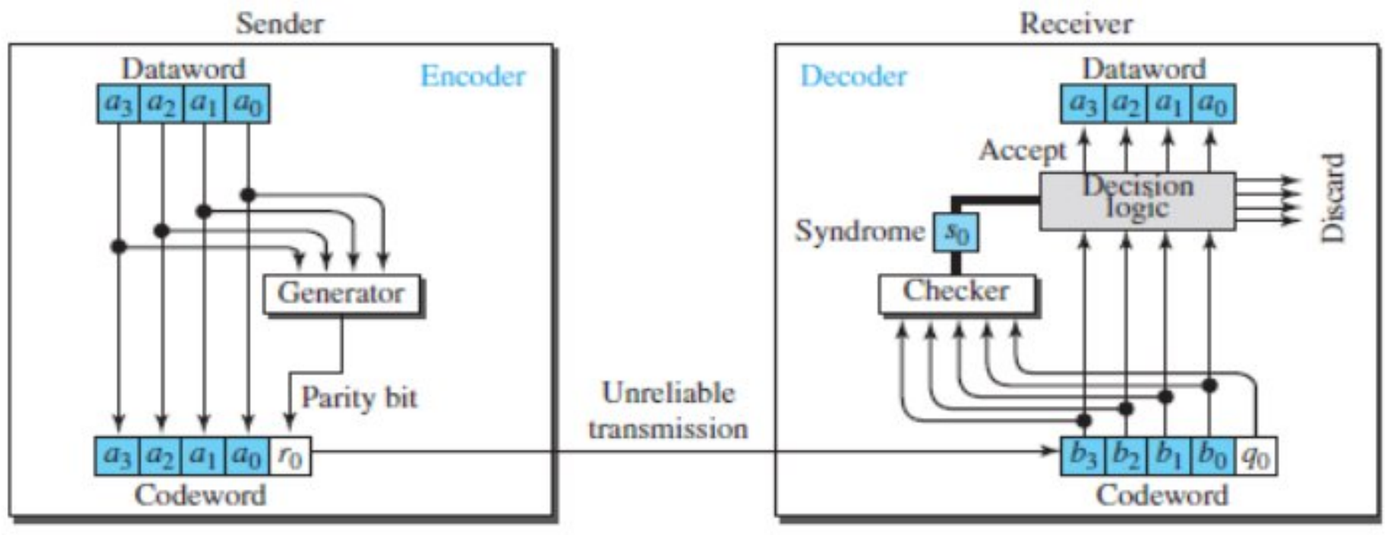
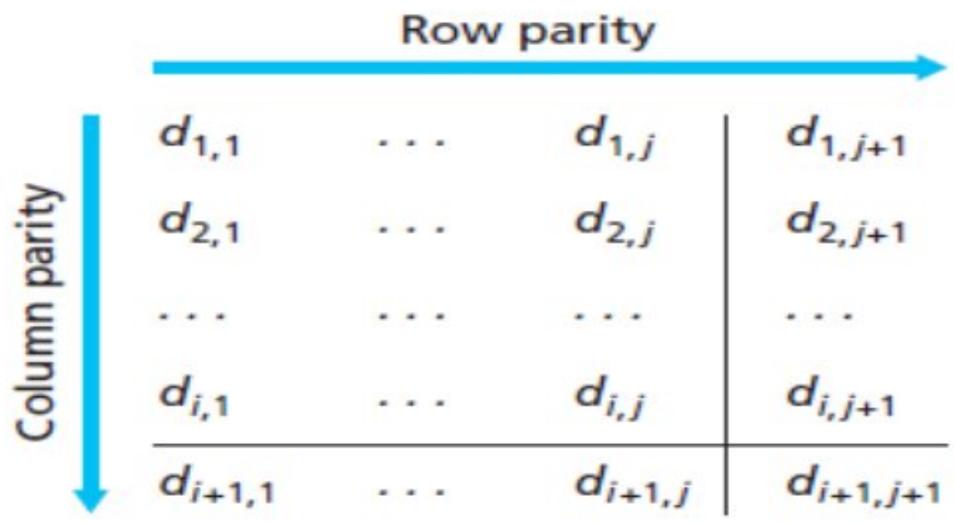


Fig: Encoder and decoder for simple parity-check code

Suppose now that a single bit error occurs in the original d bits of information. With this **two-dimensional parity** scheme, the parity of both the column and the row containing the flipped bit will be in error. The receiver can thus not only *detect* the fact that a single bit error has occurred, but can use the column and row indices of the column and row with parity errors to actually identify the bit that was corrupted and *correct* that error.



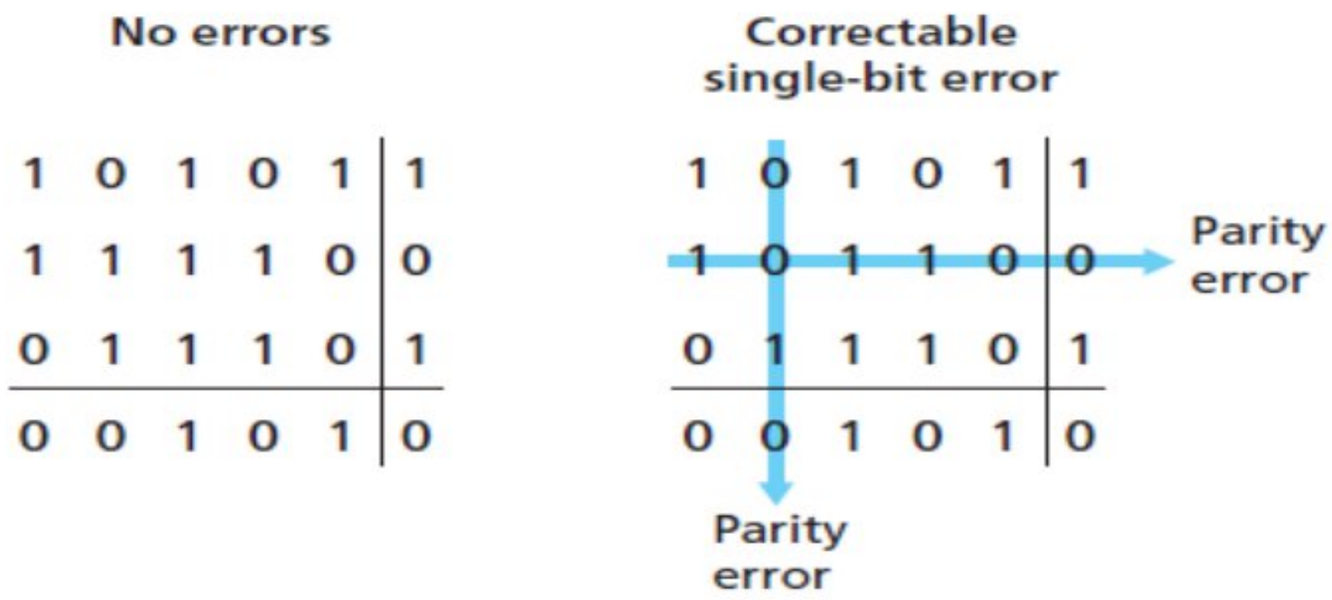


Fig: Two-dimensional even parity

ii) Cyclic Redundancy Check

The cyclic redundancy check (CRC), which is used in networks such as LANs and WANs.

CRC encoder and decoder:

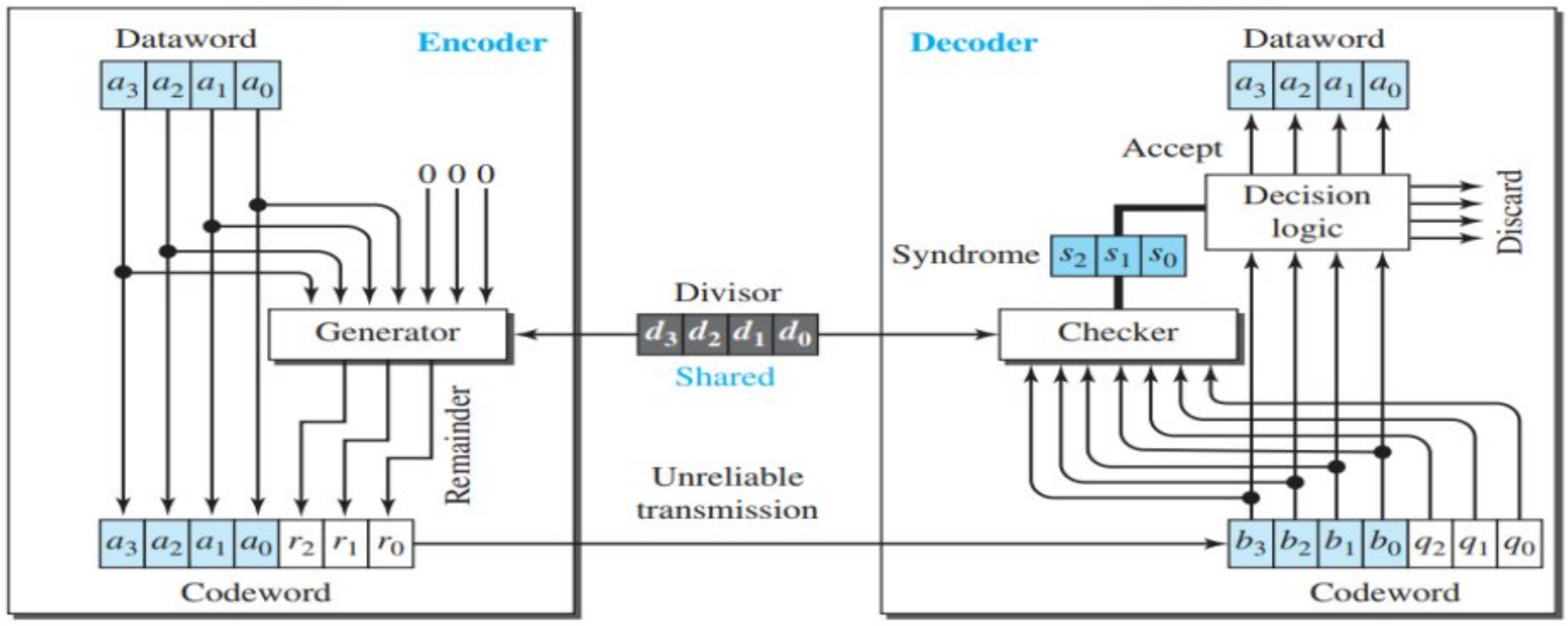


Fig: CRC encoder and decoder

CRC encoder

- In the **encoder**, the dataword has k bits (4 here); the codeword has n bits (7 here). The size of the dataword is increased by adding n − k (3 here) 0s to the right-hand side of the word.

- The n-bit result is providing to the generator. The generator uses a divisor of size n − k + 1 (4 here), predefined.

- The generator divides the increased dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder (r2r1r0) is appended to the dataword to create the codeword.

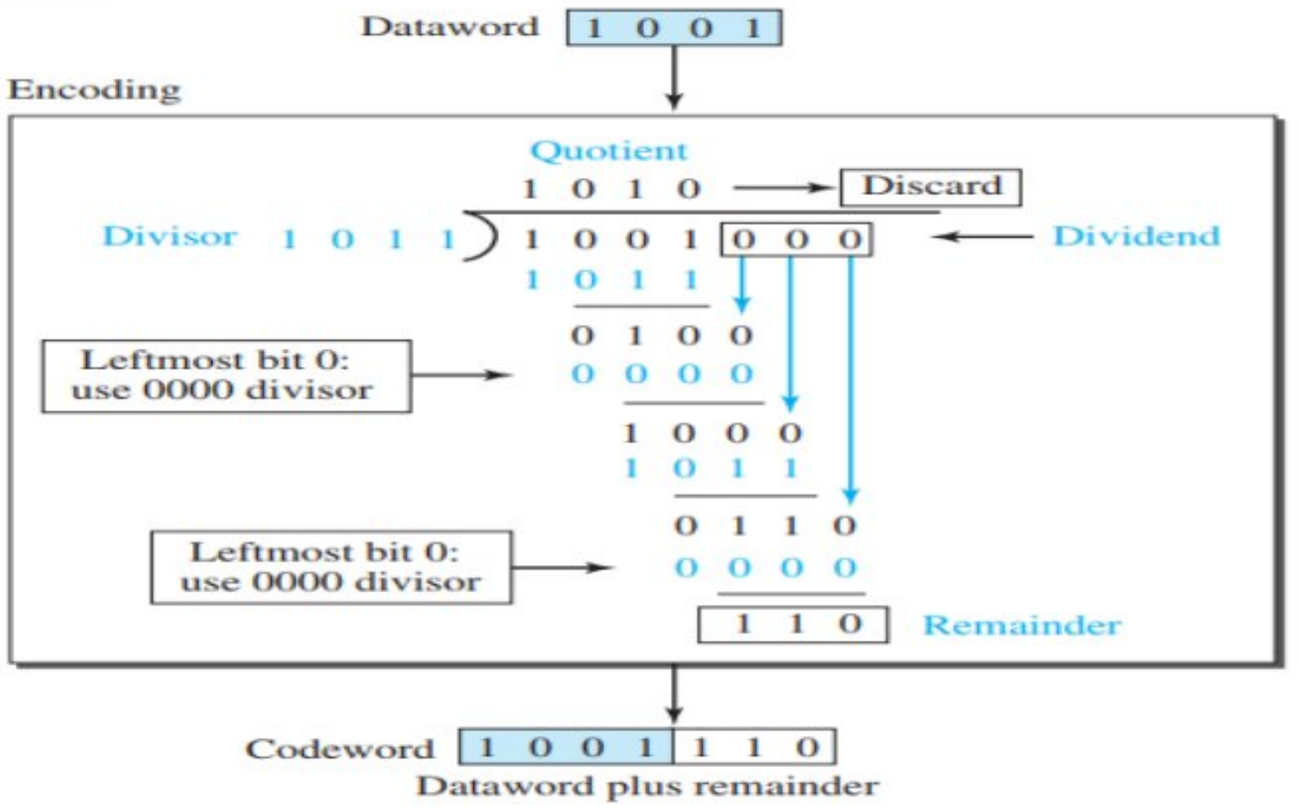
CRC decoder

- The decoder receives the codeword. A copy of all n bits is fed to the checker, which is a copy of the generator.

- The remainder produced by the checker is a syndrome of n − k (3 here) bits, which is fed to the decision logic analyzer.

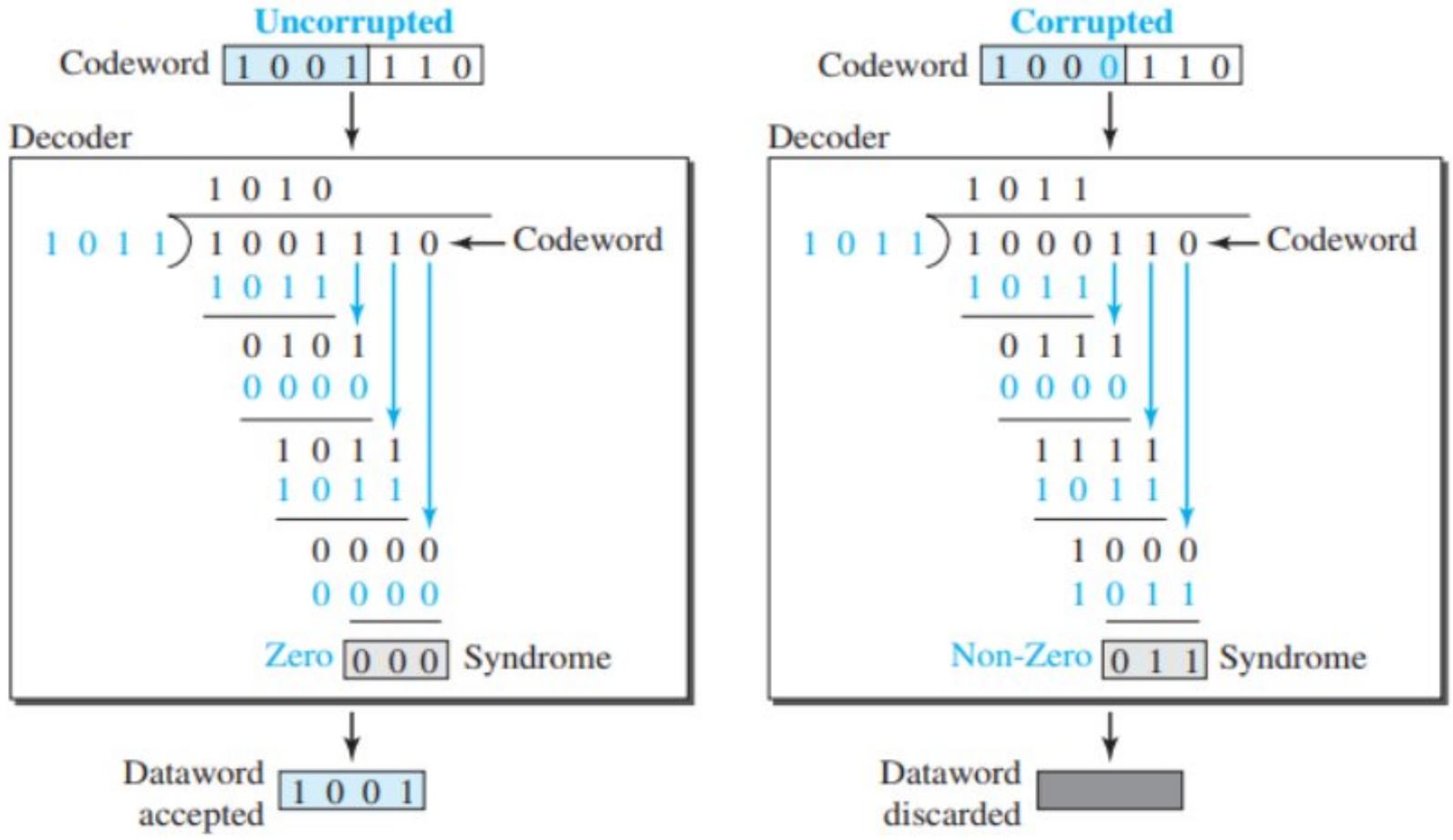
- The analyzer has a simple function. If the syndrome bits are all 0s, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).

Example:
Encoder



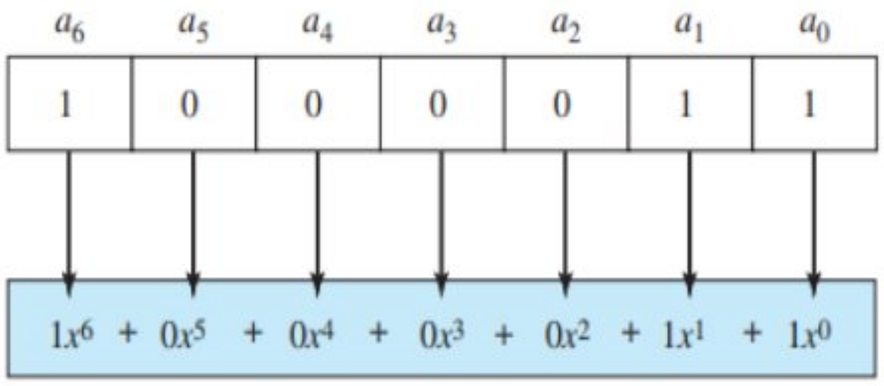
Note:
Multiply: AND
Subtract: XOR

Decoder:

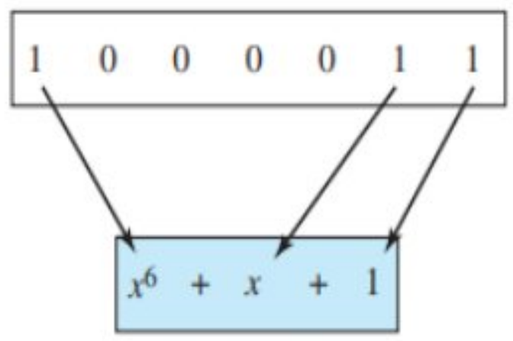


Polynomials: A better way to understand cyclic codes and how they can be analyzed is to represent them as polynomials. A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0 and 1. The power of each term shows the position of the bit; the coefficient shows the value of the bit.

Figure shows a binary pattern and its polynomial representation. In Figure a we show how to translate a binary pattern into a polynomial; in Figure b we show how the polynomial can be shortened by removing all terms with zero coefficients and replacing x^1 by x and x^0 by 1.



a. Binary pattern and polynomial

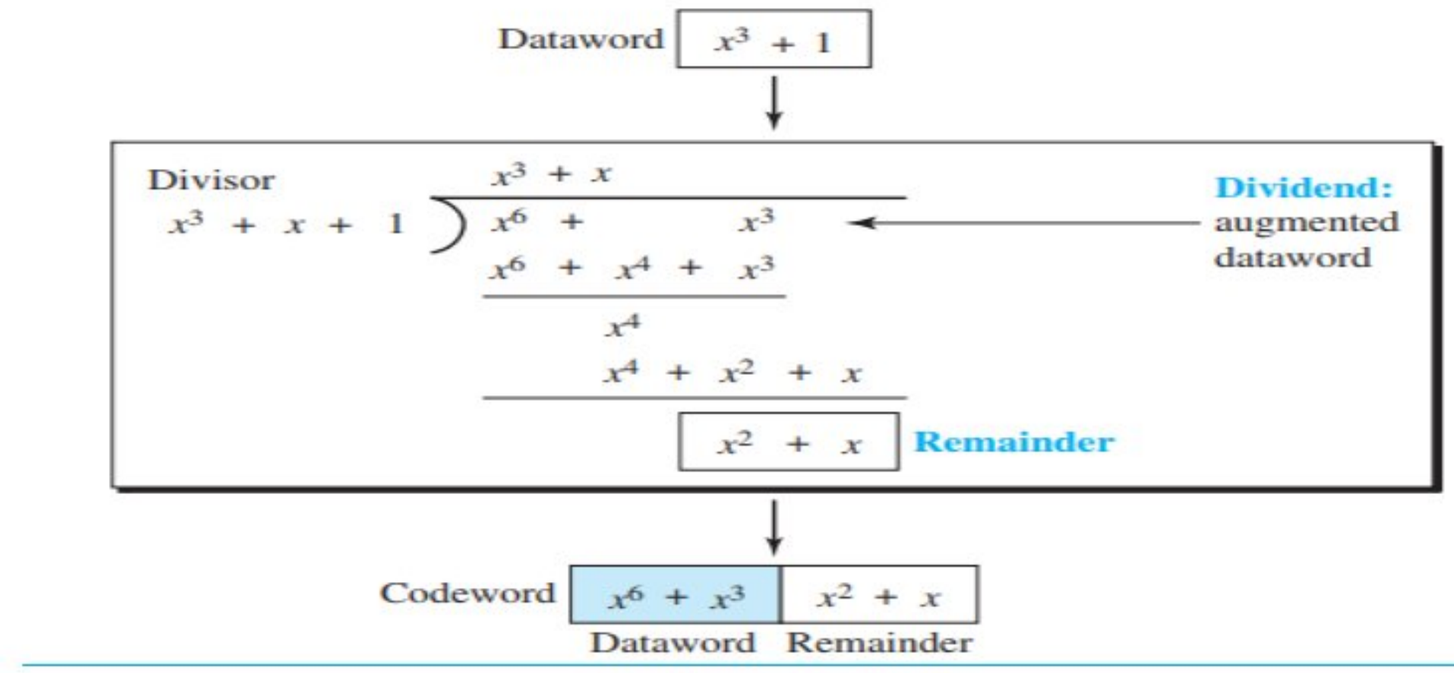


b. Short form

Fig: A polynomial to represent a binary word

Degree of a Polynomial The degree of a polynomial is the highest power in the polynomial. For example, the degree of the polynomial $x^6 + x + 1$ is 6.

Cyclic Code Encoder Using Polynomials: The dataword 1001 is represented as $x^3 + 1$. The divisor 1011 is represented as $x^3 + x + 1$. To find the augmented dataword, we have left-shifted the dataword 3 bits (multiplying by x^3). The result is $x^6 + x3$.



iii) Checksum

Checksum is an error-detecting technique that can be applied to a message of any length.

- In checksum error detection scheme, the data is divided into k segments each of m bits.
- In the **source,** 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 **destination**, 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.

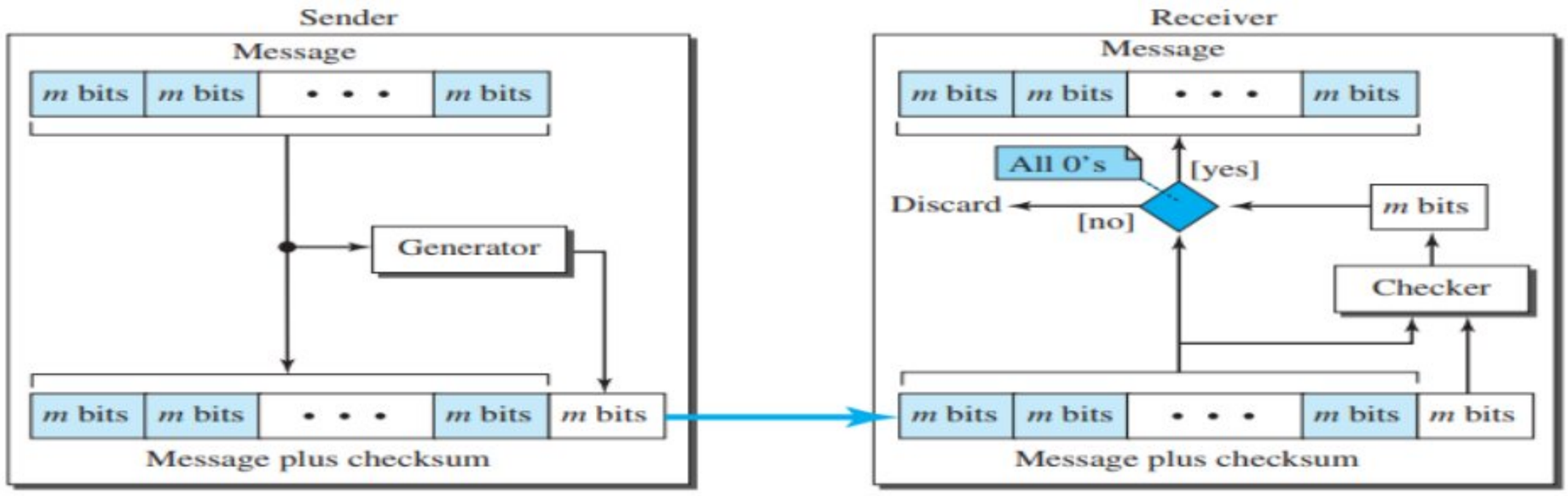
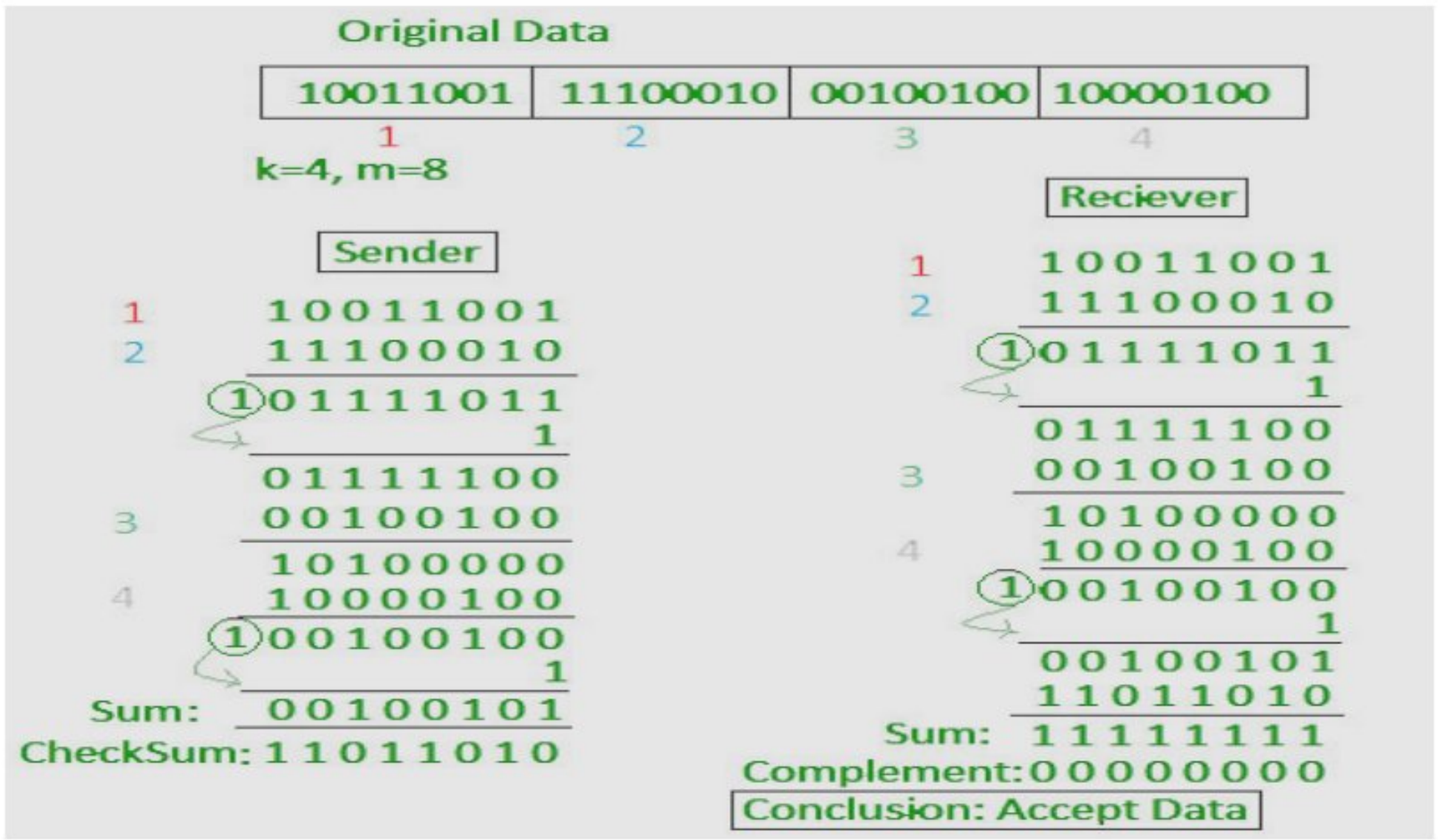


Fig: Checksum

Example:



iv) Forward Error Correction

Numbers of methods are used for error detection and retransmission. However, retransmission of corrupted and lost packets is not useful for real-time multimedia transmission because it creates an unacceptable delay in reproducing: we need to wait until the lost or corrupted packet is resent. We need to correct the error or reproduce the packet immediately.

Several schemes have been designed and used in these cases that are collectively referred to as forward error correction (FEC) techniques.

Hamming code example: The key to the Hamming Code is the use of extra parity bits to allow the identification of a single error. Create the code word as follows:

a) Mark all bit positions that are powers of two as parity bits. (Positions 1, 2, 4, 8, 16, 32, 64, etc.)

b) All other bit positions are for the data to be encoded. (Positions 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.)

c) 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.

- Parity bit 1 covers all the bits positions whose binary representation includes a 1 in the least significantposition (1, 3, 5, 7, 9, 11, etc).

- Parity bit 2 covers all the bits positions whose binary representation includes a 1 in the second position fromthe least significant bit (2, 3, 6, 7, 10, 11, etc).
- Parity bit 4 covers all the bits positions whose binary representation includes a 1 in the third position fromthe least significant bit (4–7, 12–15, 20–23, etc).
- Parity bit 8 covers all the bits positions whose binary representation includes a 1 in the fourth position fromthe least significant bit bits (8–15, 24–31, 40–47, etc).

d) 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.

Codeword: data word + additional bits (parity)

P4	D4	D3	D2	P3	D1	P2	P1

Example:

Data word: 1001, we check for even parity.

7	6	5	4	3	2	1
1	0	0	P3	1	P2	P1

- P1 bit is calculated using the bits positions: 1, 3, 5, and 7. To find the parity bit P1, we check for even parity. Since the total number of 1's in all the bit positions corresponding to P1 is an even number value (parity bit's value) = 0.
- P2 bit is calculated using the bits positions: 2, 3, 6, and 7. Since the total number of 1's in all the bit positions corresponding to P2 is an even number value (parity bit's value) = 0.
- P3 bit is calculated using the bits positions: 4 , 5, 6 and 7. Since the total number of 1's in all the bit positions corresponding to P2 is an even number value (parity bit's value) = 1.

Thus, the data transferred is:

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

If the data transfer perfectly, there is no error.

Error detection and correction: Suppose in the above example the 3rd bit is changed from 1 to 0 during data transmission, then it gives new parity values in the binary number:

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

P1 bit positions(1,3,5,7), even parity value=1
P2 bit positions(2,3,6,7), even parity value=1
P3 bit positions(4,5,6,7), even parity value=0

The bits give the binary number as 011 whose decimal representation is 3. Thus, the bit 3 contains an error. To correct the error the 3rd bit is changed from 0 to 1.

3. Data-Link Layer Protocols

The 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.

- Protocols for noiseless channel: Simplest, Stop-and-Wait
- Protocols for noisy channel: Stop-and-Wait ARQ, Go-Back-N ARQ, Selective Repeat ARQ.

Elementary Data Link Protocols (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***

- **Simplest Protocol:** Our first protocol, which we call the Simplest, is one that has no flow or error control. It is a unidirectional protocol in which data frames are traveling in only one direction-from the sender to receiver.

We assume that the receiver can immediately handle any frame it receives with a processing time. The data link layer of the receiver immediately removes the header from the frame and hands the data packet to its network layer, which can also accept the packet immediately.

Design:

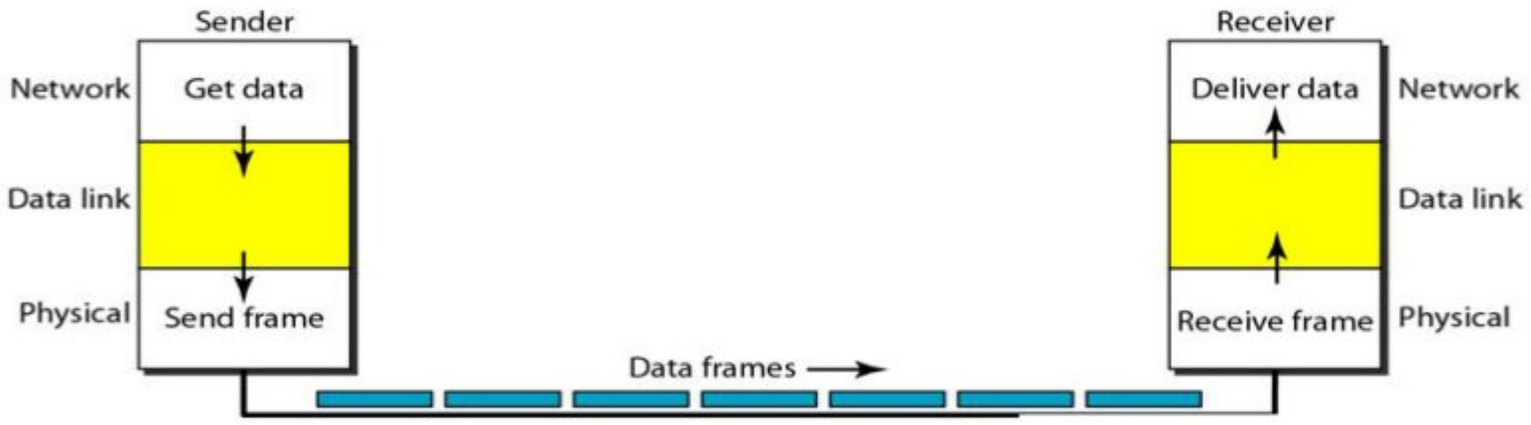
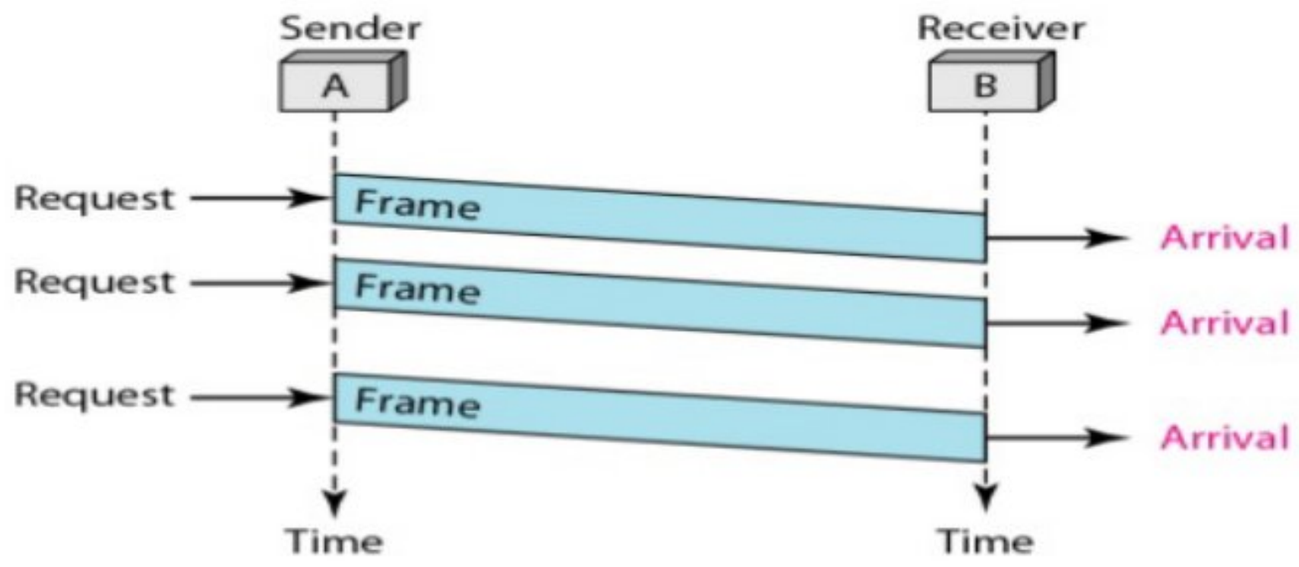


Fig: The design of the simplest protocol with no flow or error control

Flow diagram:



- **Stop-and-Wait:** The receiver does not have enough storage space, especially if it is receiving data from many sources. This may result in either the discarding of frames or denial of service.

To prevent the receiver from becoming overwhelmed with frames, we somehow need to tell the sender to slow down. There must be feedback from the receiver to the sender.

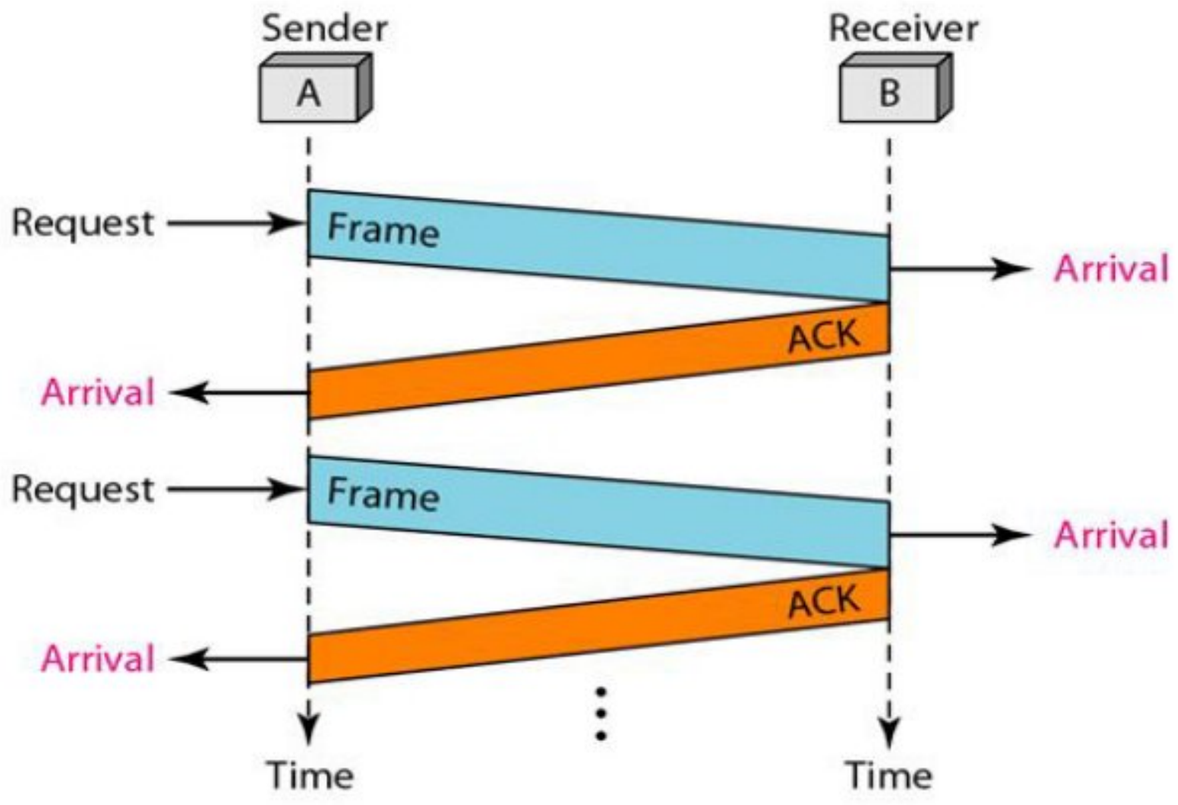
The protocol we discuss now is called the Stop-and-Wait Protocol because the sender sends one frame, stops until it receives confirmation from the receiver, and then sends the next frame.

We still have unidirectional communication for data frames, but auxiliary ACK frames (simple tokens of acknowledgment) travel from the other direction. We add flow control to protocol.

Design:

Fig: Design of Stop-and-Wait Protocol

Flow diagram:



Sliding Window Protocols (Noisy Channels): Although the Stop-and-Wait Protocol gives us an idea of how to add flow control to its predecessor, noiseless channels are nonexistent. We can ignore the error or we need to add error control to our protocols. We discuss three protocols in this section that use error control.

- **Stop-and-Wait Automatic Repeat Request:** the Stop-and-Wait Automatic Repeat Request (Stop-and-Wait ARQ), adds a simple error control mechanism to the Stop-and-Wait Protocol. Let us see how this protocol detects and corrects errors.

To detect and correct corrupted frames, we need to add redundancy bits to our data frame. When the frame arrives at the receiver site, it is checked and if it is corrupted, it is silently discarded.
The completed and lost frames need to be resent in this protocol. The sender keeps a copy of the sent frame. At the same time, it starts a timer. If the timer expires and there is no ACK for the sent frame, the frame is resent, the copy is held, and the timer is restarted.

Sequence Numbers: the protocol specifies that frames need to be numbered. This is done by using sequence numbers. A field is added to the data frame to hold the sequence number of that frame.
Acknowledgment Numbers Since the sequence numbers must be suitable for both data frames and ACK frames, we use this convention: The acknowledgment numbers always announce the sequence number of the next frame expected by the receiver. For example, if frame 0 has arrived safe and sound, the receiver sends an ACK frame with acknowledgment 1.

Design:

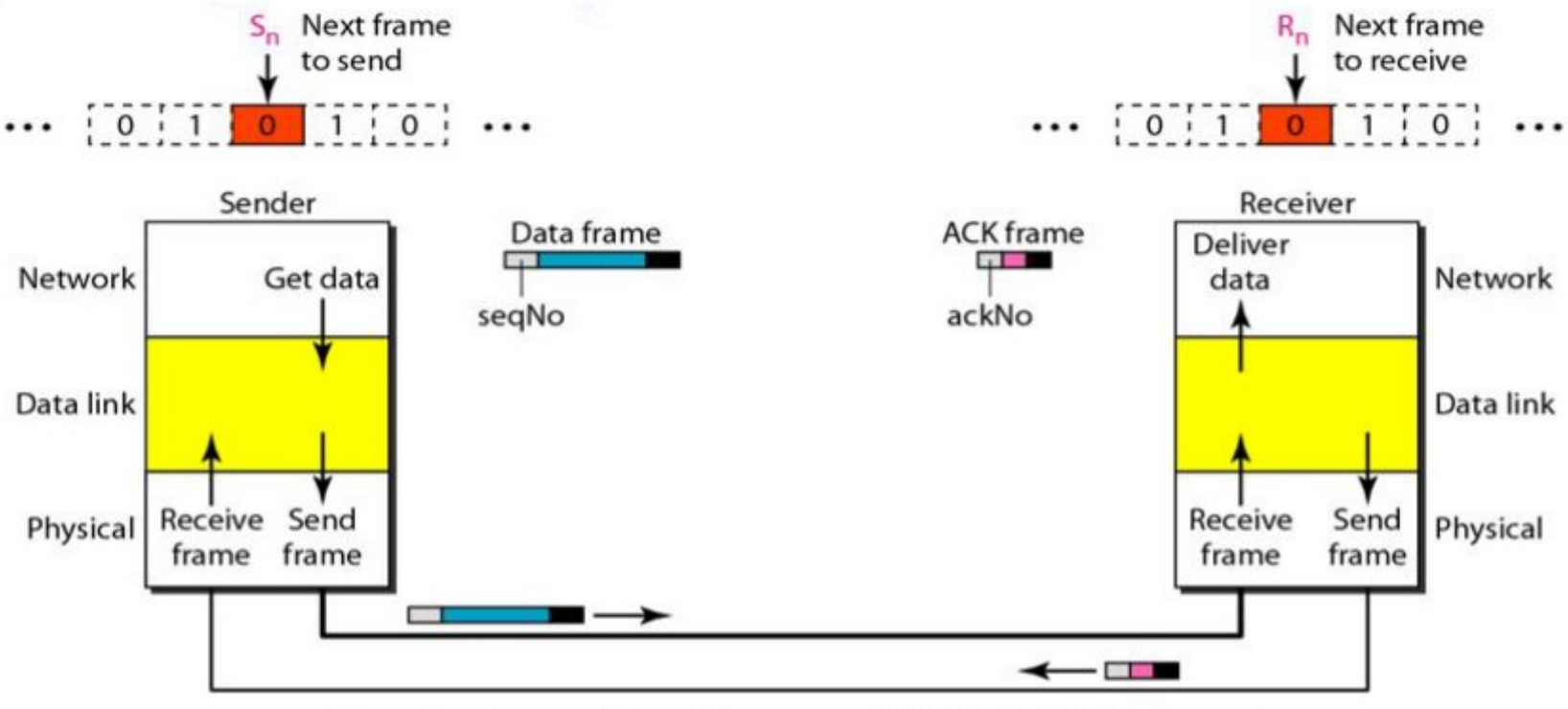
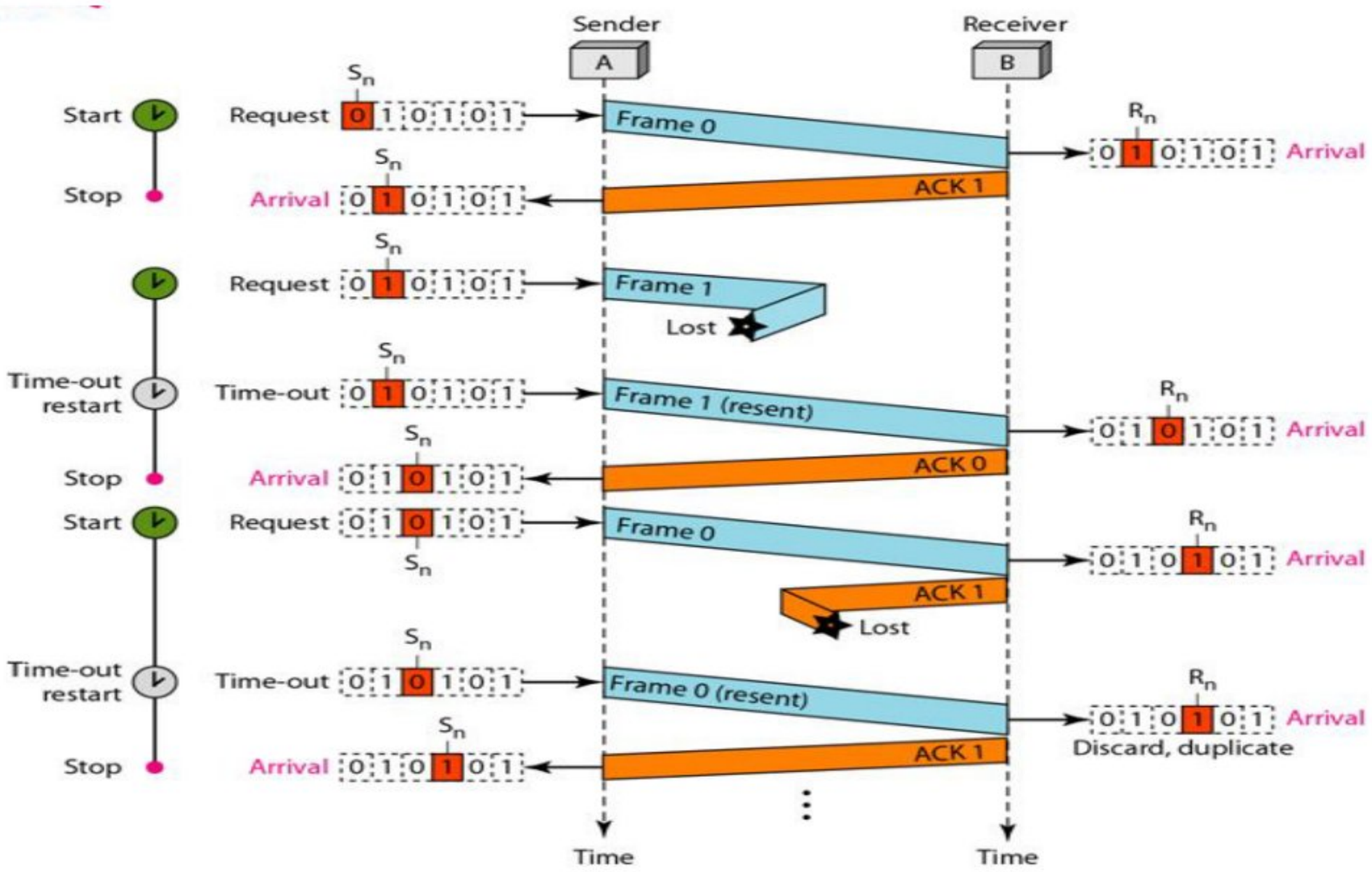


Fig: Design of the Stop-and-Wait ARQ Protocol

Flow diagram:



- **Go-Back-N Automatic Repeat Request:** To improve the efficiency of transmission (filling the pipe), multiple frames must be in transition while waiting for acknowledgment.

The first is called Go-Back-N Automatic Repeat Request protocol we can send several frames before receiving acknowledgments; we keep a copy of these frames until the acknowledgments arrive.

Sequence Numbers: Frames from a sending station are numbered sequentially. In the Go-Back-N Protocol, the sequence numbers are modulo 2^m, where m is the size of the sequence number field in bits.

Sliding Window: In this protocol, the sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, the sender and receiver need to deal with only part of the possible sequence numbers.

The send window the maximum size of the window is 2m.The receive window makes sure that the correct data frames are received and that the correct acknowledgments are sent. The size of the receive window is always 1.

Design:

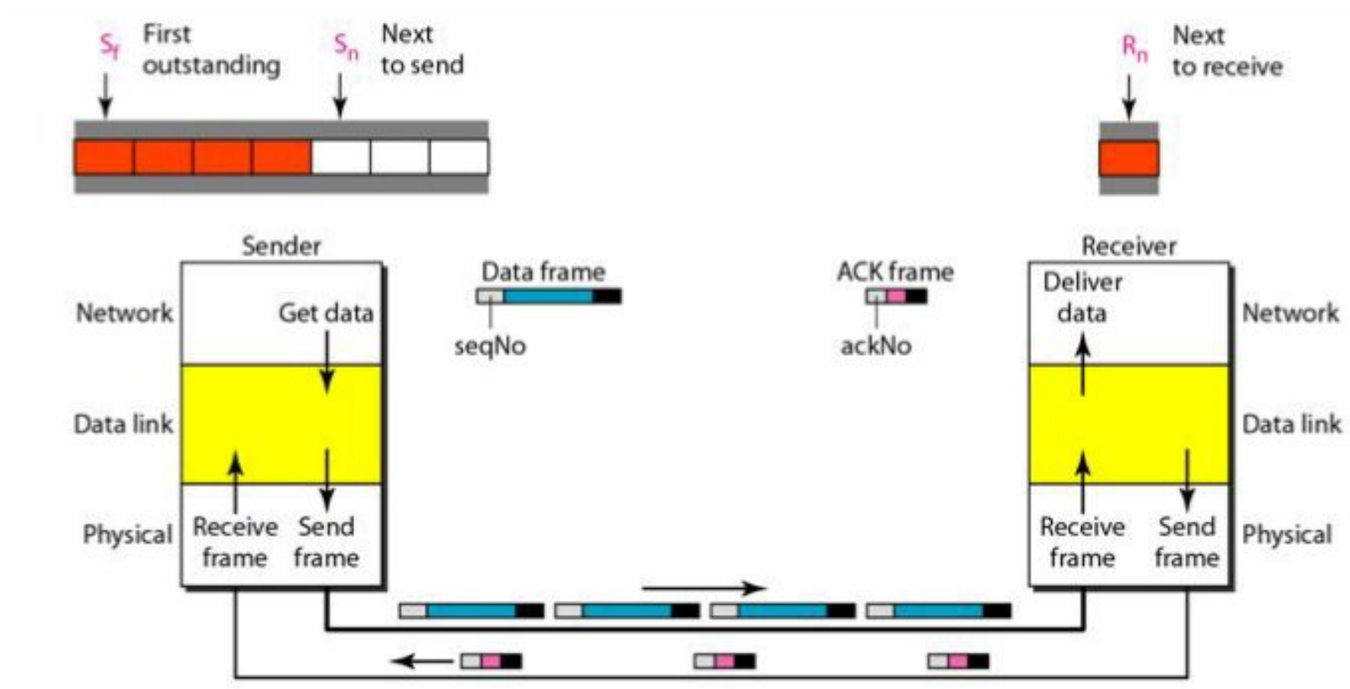
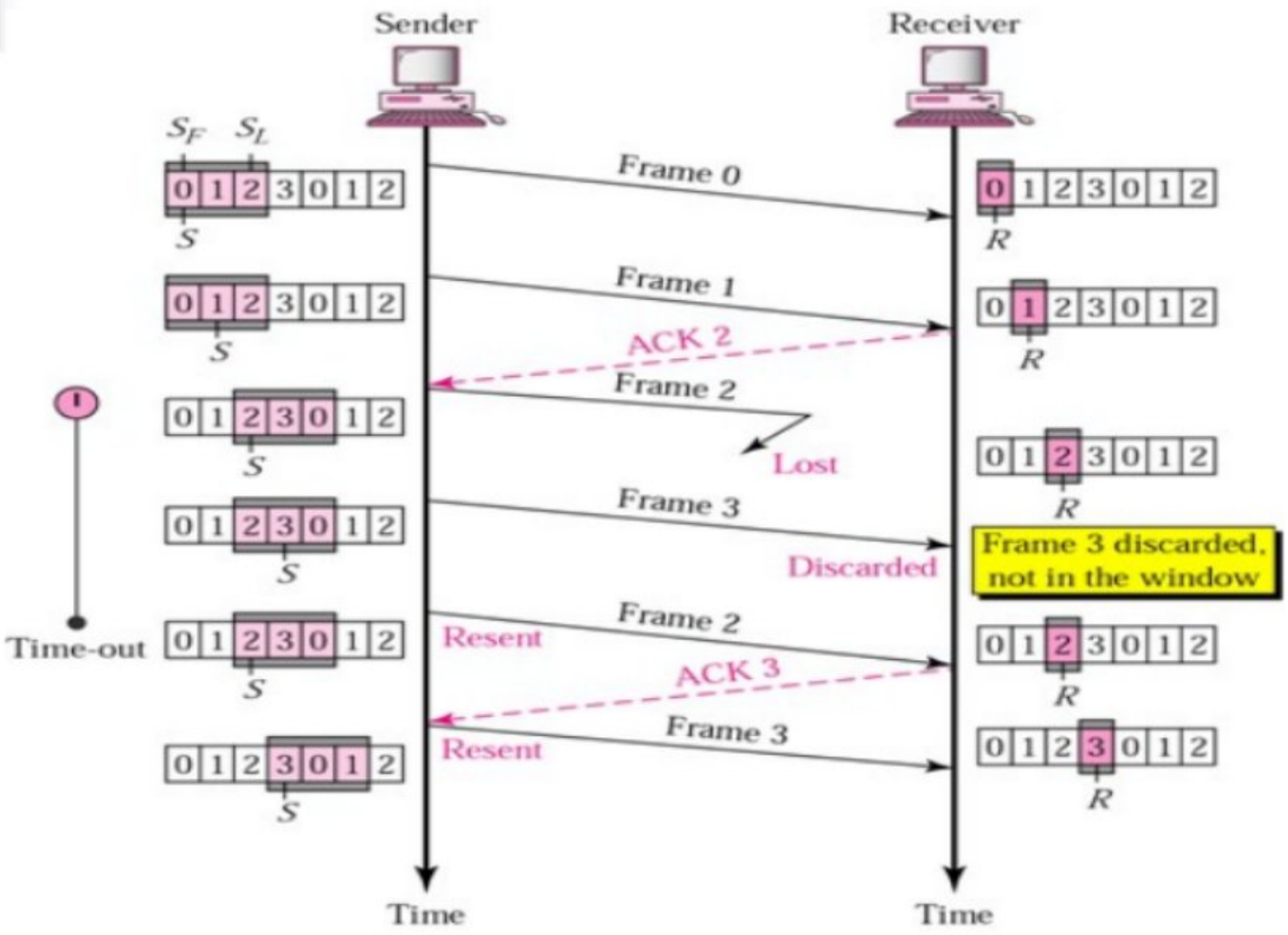


Fig: Design of Go-Back-NARQ

Flow diagram:



- **Selective Repeat Automatic Repeat Request:** Go-Back-N ARQ simplifies the process at the receiver site. The receiver keeps track of only one variable, and there is no need to buffer out-of-order frames; they are simply discarded. However, this protocol is very inefficient for a noisy link. In a noisy link a frame has a higher probability of damage, which means the resending of multiple frames.

For noisy links, there is another mechanism that does not resend N frames when just one frame is damaged; only the damaged frame is resent. This mechanism is called Selective Repeat ARQ.

Windows: The Selective Repeat Protocol also uses two windows: a send window and a receive window. The receive window is the same size as the send window.

Design:

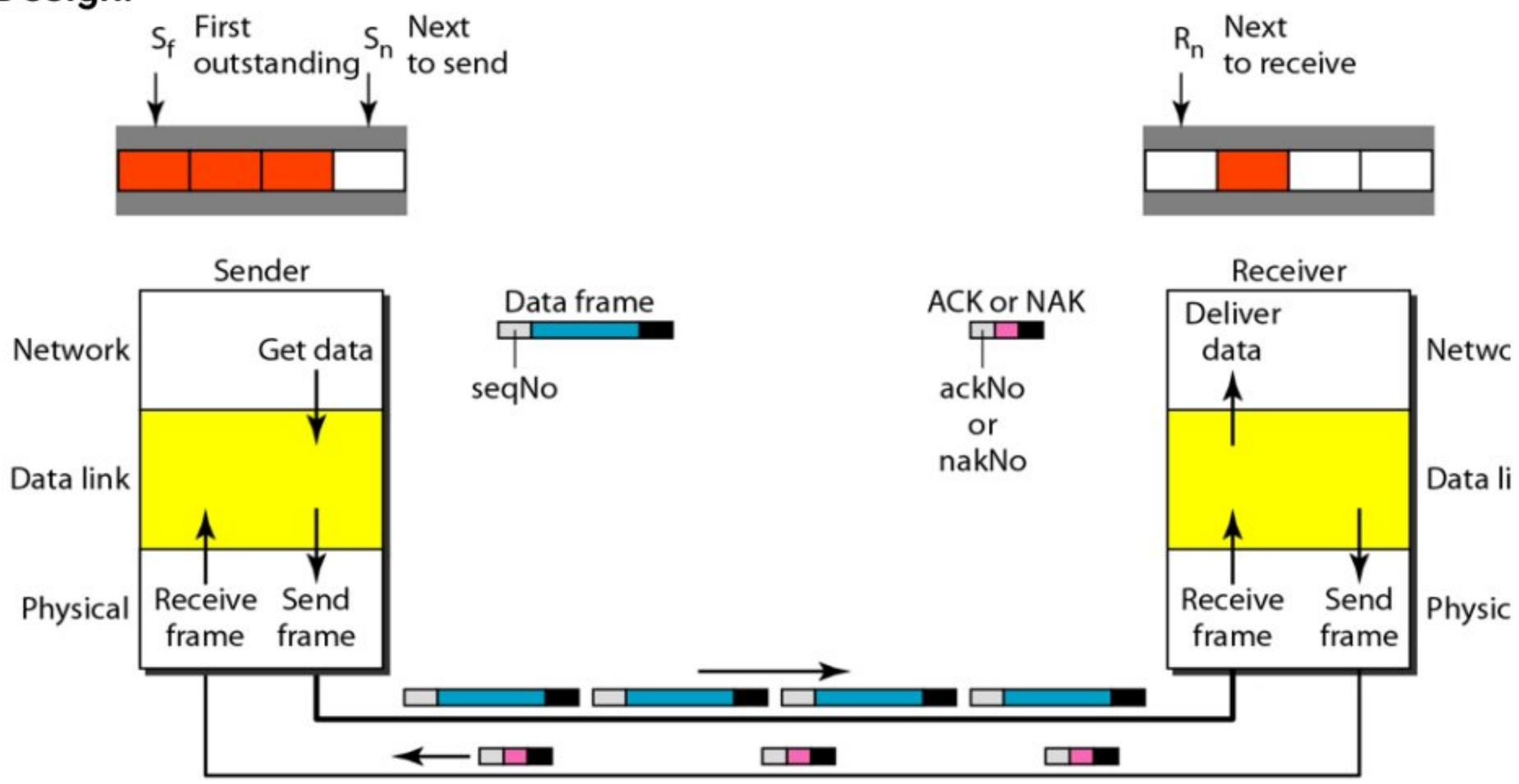
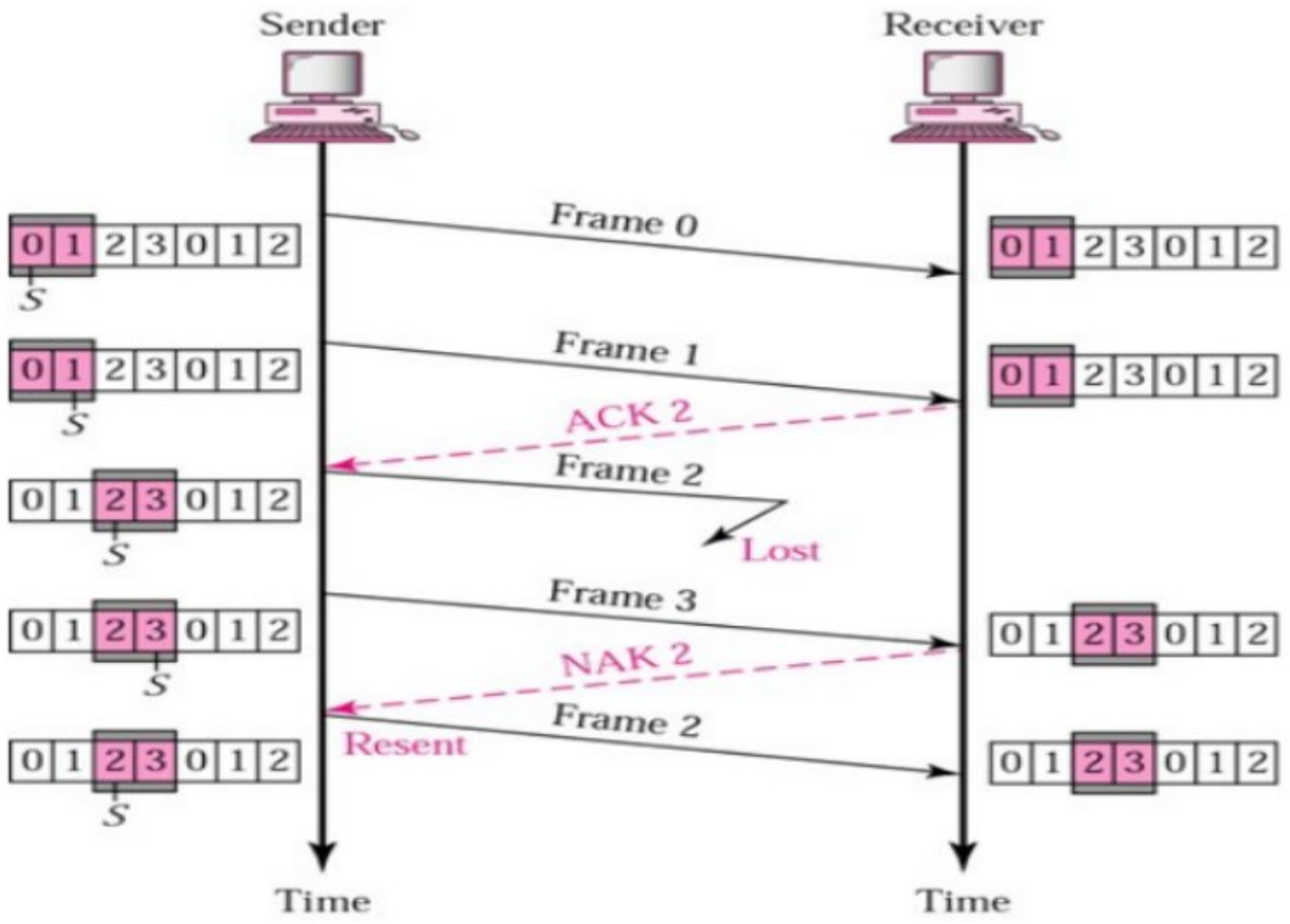


Fig: Design of Selective Repeat ARQ

Flow diagram:



4. Multiple Access Links and Protocols

When nodes or stations are connected and use a common link, called a *multipoint* or *broadcast link,* we need a multiple-access protocol to coordinate access to the link. The problem of controlling the access to the medium is similar to the rules of speaking in an assembly.

Many protocols have been devised to handle access to a shared link. All of these protocols belong to a sublayer in the data-link layer called *media access control (MAC).* We categorize them into three groups, as shown in Figure.

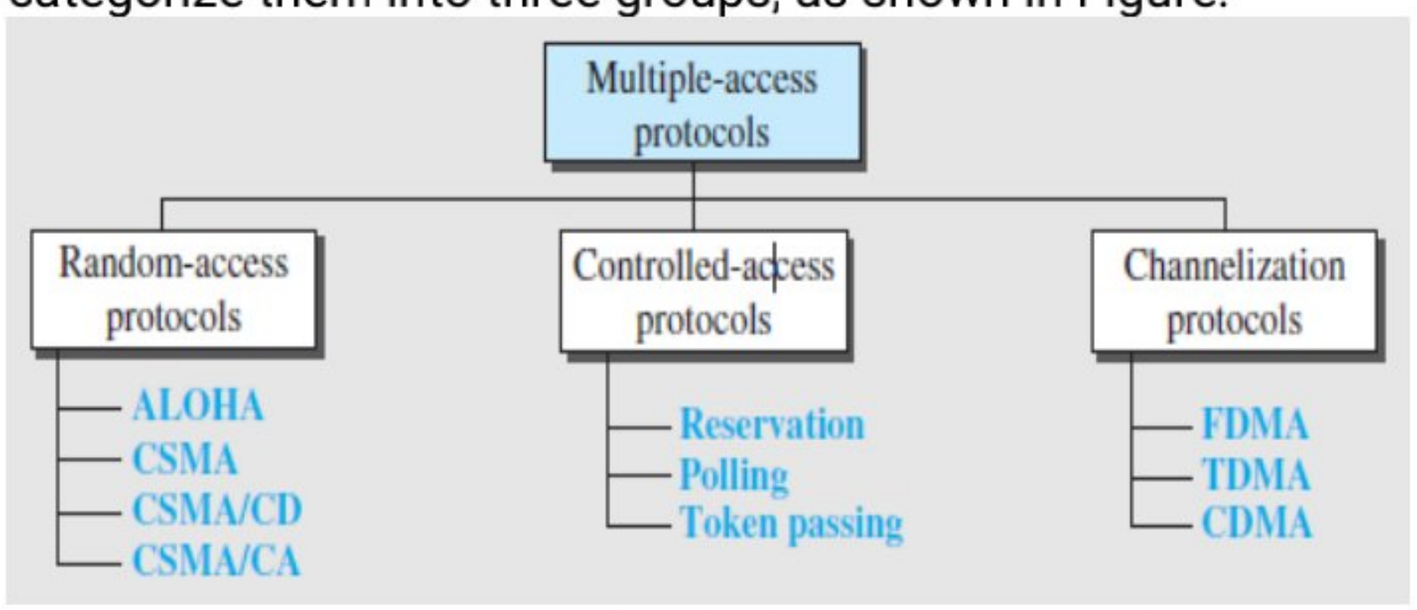


Fig: Taxonomy of multiple-access protocols

i) Random Access: In **random-access** or **contention** methods, no station is superior to another station and none is assigned control over another. At each instance, a station that has data to send uses a procedure defined by the protocol to make a decision on whether or not to send. This decision depends on the state of the medium (idle or busy).

In a random-access method, each station has the right to the medium without being controlled by any other station. However, if more than one station tries to send, there is an access conflict—***collision***—and the frames will be either destroyed or modified.

The random-access methods: **ALOHA, CSMA, CSMA/CD, CSMA/CA**

- **ALOHA**

ALOHA, the earliest random access method, was developed at the University of Hawaii in early 1970. The medium is shared between the stations. When a station sends data, another station may attempt to do so at the same time. The data from the two stations collide and become distorted.

- ***Pure ALOHA***

The original ALOHA protocol is called ***pure ALOHA.*** This is a simple but well-designed protocol.
The idea is that each station sends a frame whenever it has a frame to send (multiple access). However, since there is only one channel to share, there is the possibility of collision between frames from different stations. Figure shows an example of frame collisions in pure ALOHA.

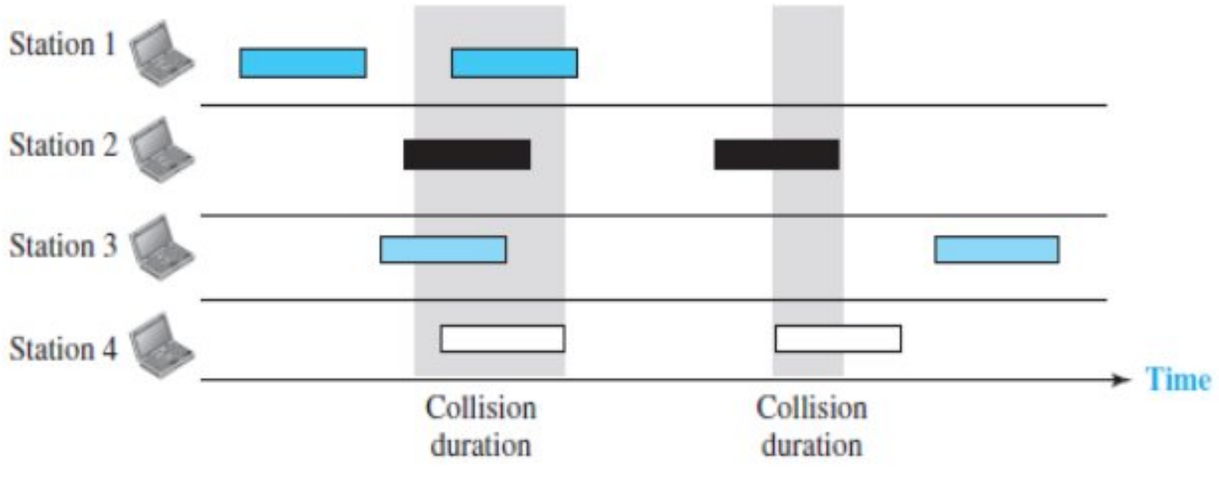


Fig: Frames in a pure ALOHA network

There are four stations (unrealistic assumption) that contend with one another for access to the shared channel. The figure shows that each station sends two frames; there are a total of eight frames on the shared medium. Some of these frames collide because multiple frames are in contention for the shared channel. Figure shows that only two frames survive: one frame from station 1 and one frame from station 3.

Pure ALOHA has a method to prevent congesting the channel with retransmitted frames. After a maximum number of retransmissions attempts *Kmax,* a station must give up and try later.

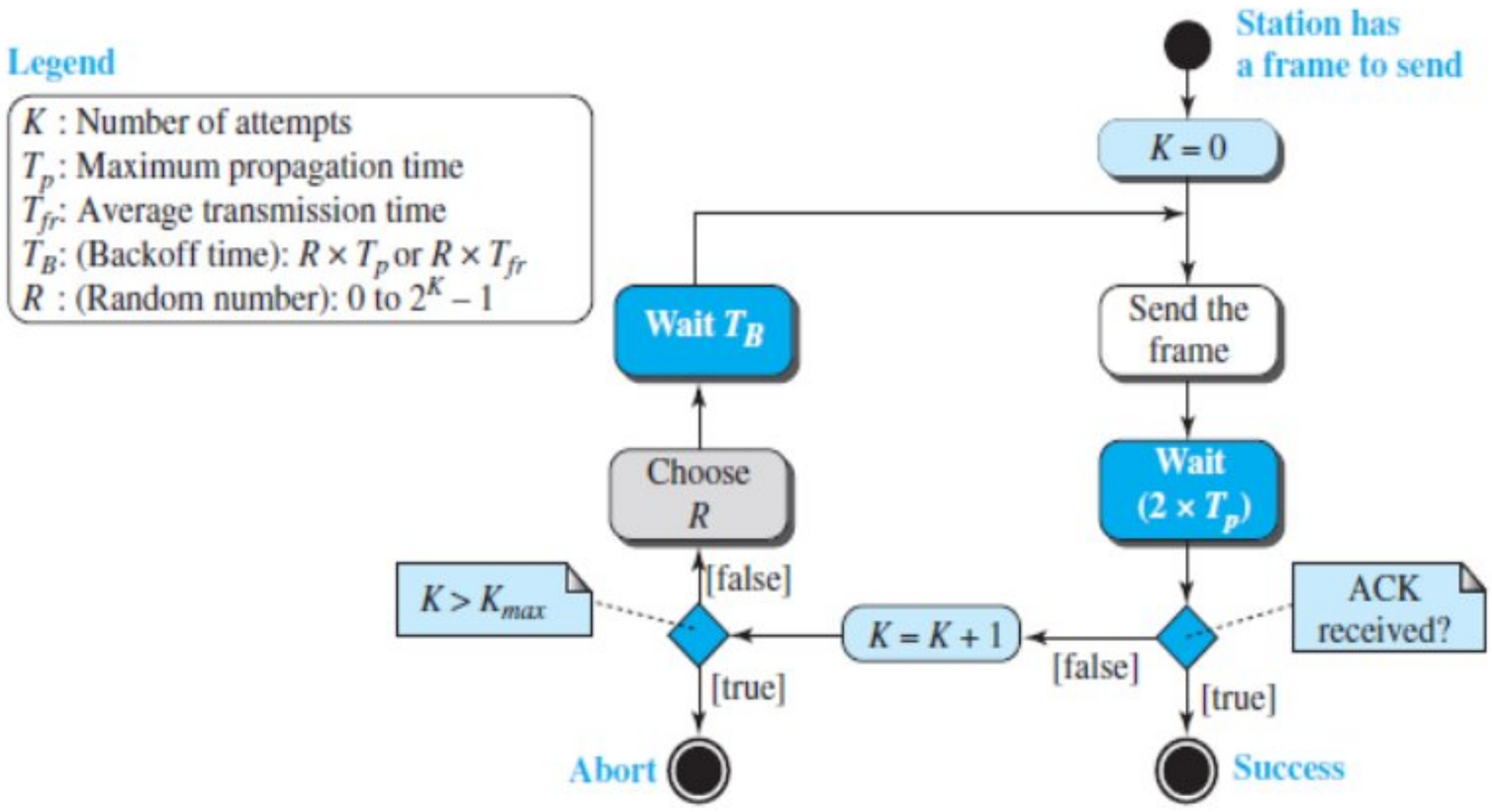


Fig: Procedure for pure ALOHA protocol

☒ *Slotted ALOHA*

A station may send soon after another station has started or just before another station has finished. Slotted ALOHA was invented to improve the efficiency of pure ALOHA.
In **slotted ALOHA** we divide the time into slots of *Tfr* seconds and force the station to send only at the beginning of the time slot. Figure shows an example of frame collisions in slotted ALOHA.

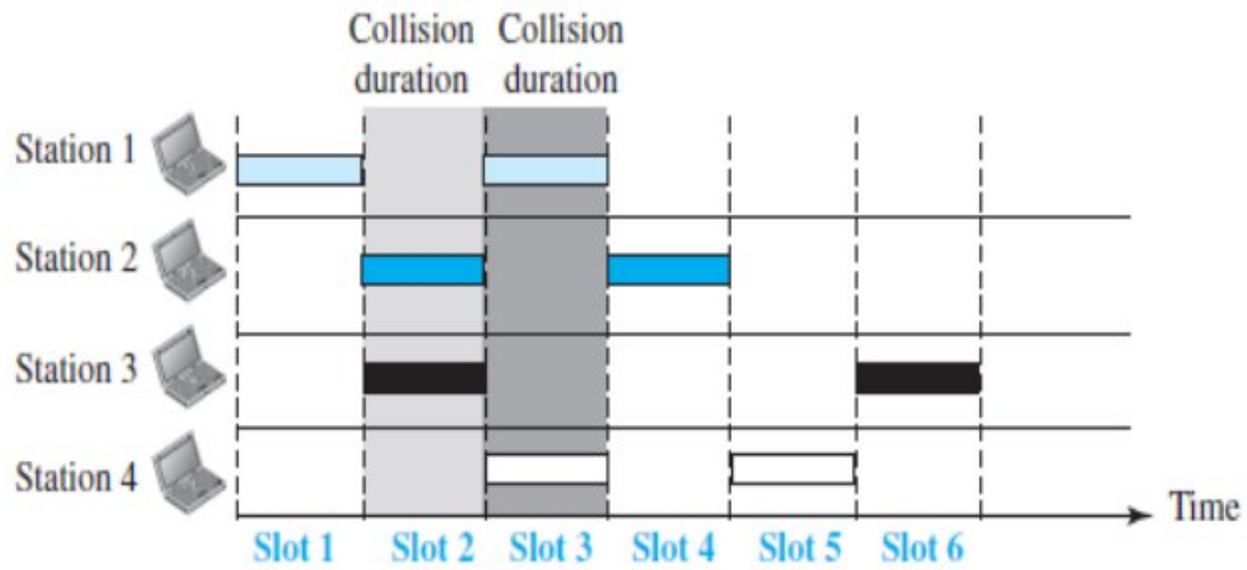


Fig: Frames in a slotted ALOHA network

Because a station is allowed to send only at the beginning of the synchronized time slot, if a station misses this moment, it must wait until the beginning of the next time slot. This means that the station which started at the beginning of this slot has already finished sending its frame.

- **CSMA**

To minimize the chance of collision and, therefore, increase the performance, the CSMA method was developed. The chance of collision can be reduced if a station senses the medium before trying to use it.

Carrier sense multiple access (CSMA) requires that each station first listen to the medium (or check the state of the medium) before sending. In other words, CSMA is based on the principle "sense before transmit".

CSMA can reduce the possibility of collision, but it cannot eliminate it. The reason for this is shown in Figure, a space and time model of a CSMA network. Stations are connected to a shared channel.

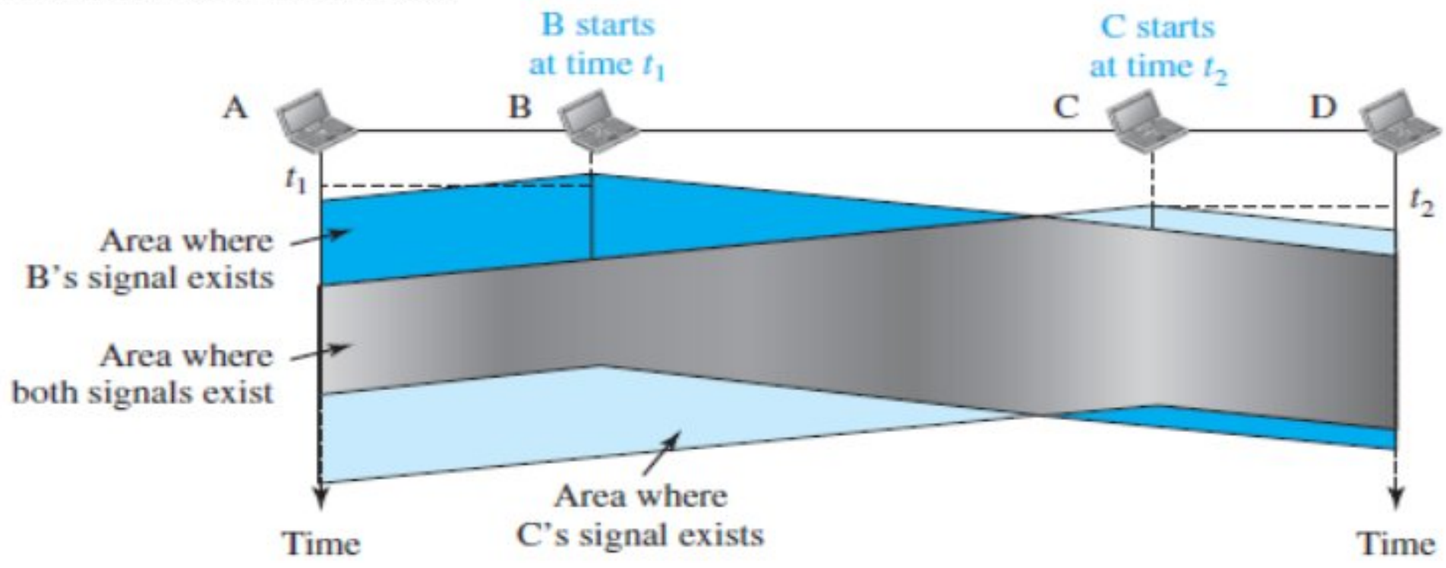


Fig: Space/time model of a collision in CSMA

The possibility of collision still exists because of propagation delay; when a station sends a frame, it still takes time for the first bit to reach every station and for every station to sense it.

At time $t1$, station B senses the medium and finds it idle, so it sends a frame. At time $t2$ ($t2 > t1$), station C senses the medium and finds it idle because, at this time, the first bits from station B have not reached station C. Station C also sends a frame. The two signals collide and both frames are destroyed.

- ***Persistence Methods:*** **Three methods** have been devised to answer these questions: the **1-persistentmethod,** the **nonpersistent method,** and the ***p*-persistent method.** Figure shows the behavior of three persistence methods when a station finds a channel busy.

1-Persistent: The *1-persistent method* is simple and straightforward. In this method, after the station finds the line idle, it sends its frame immediately (with probability 1). This method has the highest chance of collision because two or more stations may find the line idle and send their frames immediately.

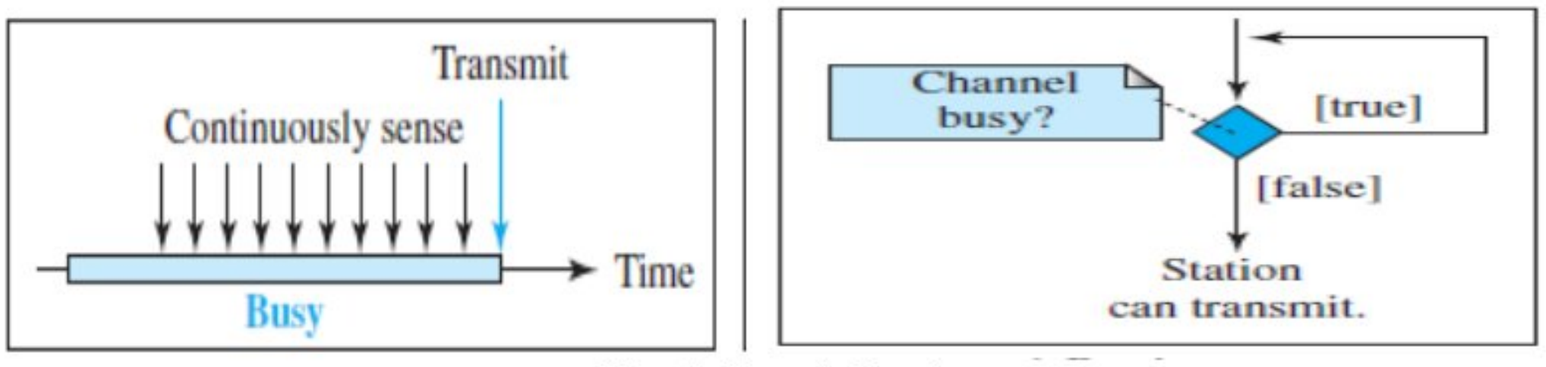


Fig: 1-Persistent

Nonpersistent: In the *nonpersistent method,* a station that has a frame to send senses the line. If the line is idle, it sends immediately. If the line is not idle, it waits a random amount of time and then senses the line again. The nonpersistent approach reduces the chance of collision because it is unlikely that two or more stations will wait the same amount of time and retry to send simultaneously

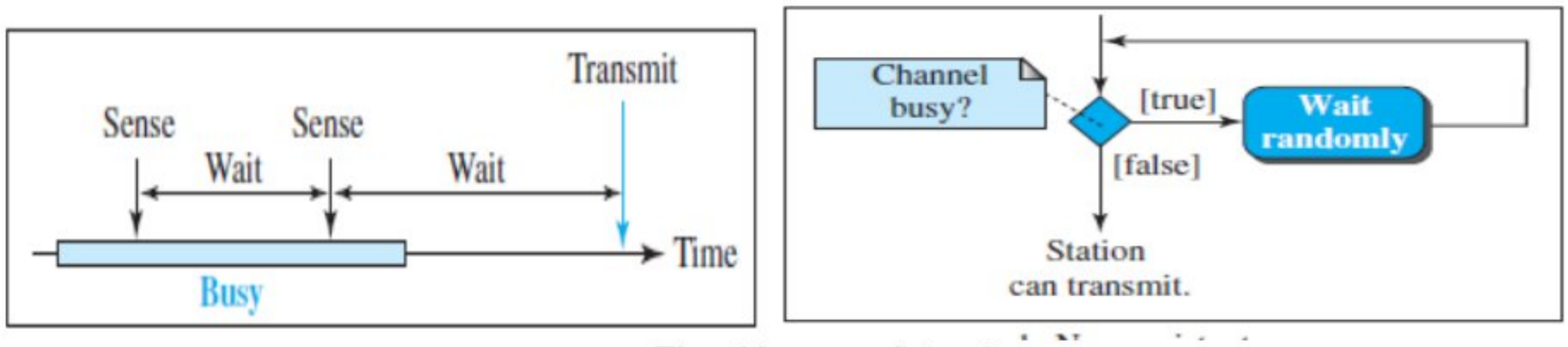


Fig: Nonpersistent

P-Persistent*:* The p-*persistent method* is used if the channel has time slots with a slot duration equal to or greater than the maximum propagation time. It reduces the chance of collision and improves efficiency. In this method, after the station finds the line idle it follows these steps:

1. With probability *p*, the station sends its frame.

2. With probability $q = 1 - p$, the station waits for the beginning of the next time slot and checks the line again.

a. If the line is idle, it goes to step 1.

b. If the line is busy, it acts as though a collision has occurred and uses the backoff procedure.

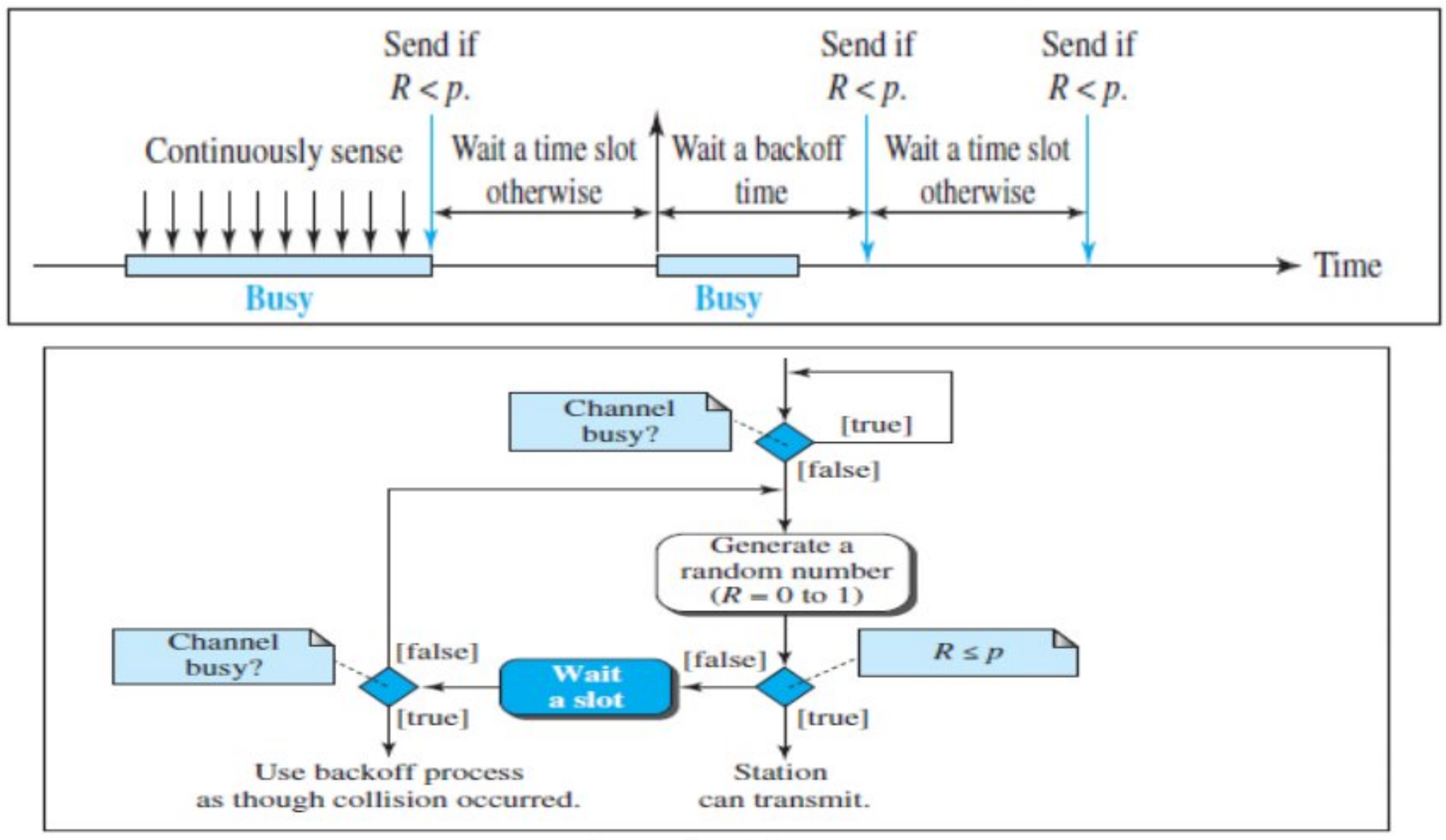


Fig: p-Persistent

⊠ CSMA/CD

The CSMA method does not specify the procedure following a collision. ***Carrier sense multiple access with collision detection (CSMA/CD)*** augments the algorithm to handle the collision.

In this method, a station monitors the medium after it sends a frame to see if the transmission was successful. If so, the station is finished. If, however, there is a collision, the frame is sent again.

To better understand CSMA/CD, let us look at the first bits transmitted by the two stations involved in the collision. Although each station continues to send bits in the frame until it detects the collision, we show what happens as the first bits collide. In Figure, stations A and C are involved in the collision.

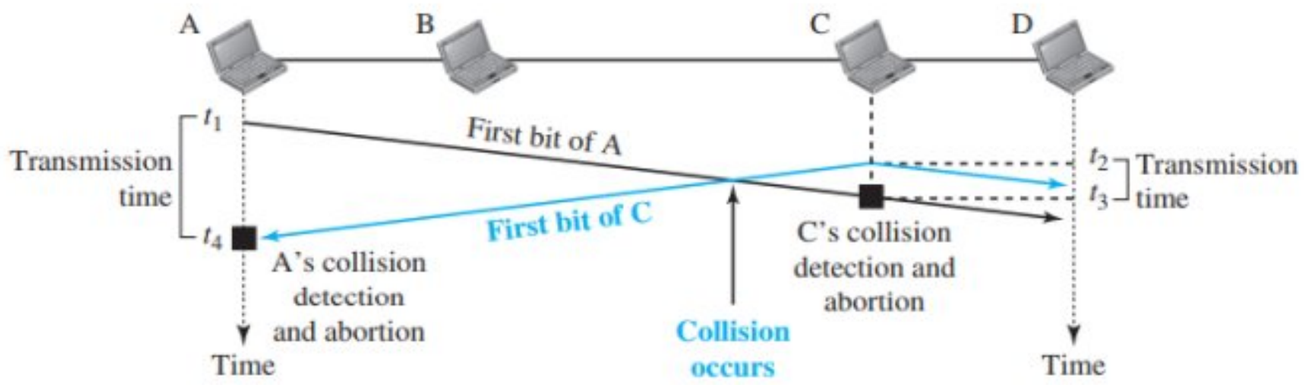


Fig: Collision of the first bits in CSMA/CD

Procedure: Now let us look at the flow diagram for CSMA/CD in Figure.

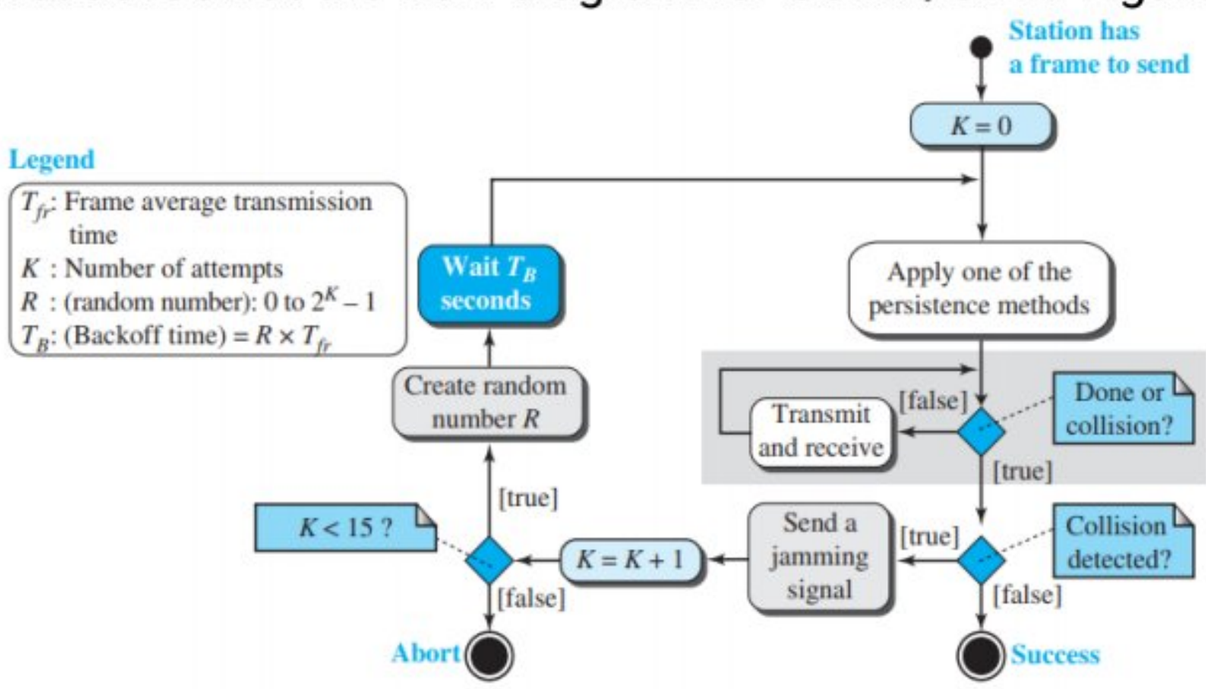


Fig: Flow diagram for the CSMA/CD

The sending of a short jamming signal to make sure that all other stations become aware of the collision.

- **CSMA/CA**

Carrier sense multiple access with collision avoidance (CSMA/CA) was invented for wireless networks. Collisions are avoided through the use of CSMA/CA's three strategies: the inter frame space, the contention window, and acknowledgments, as shown in Figure.

Inter Frame Space (IFS). First, collisions are avoided by deferring transmission even if the channel is found idle. When an idle channel is found, the station does not send immediately. It waits for a period of time called the ***interframe*** space or ***IFS.***

Contention Window. The contention window is an amount of time divided into slots. A station that is ready to send chooses a random number of slots as its wait time. The number of slots in the window changes according to the binary exponential backoff strategy.

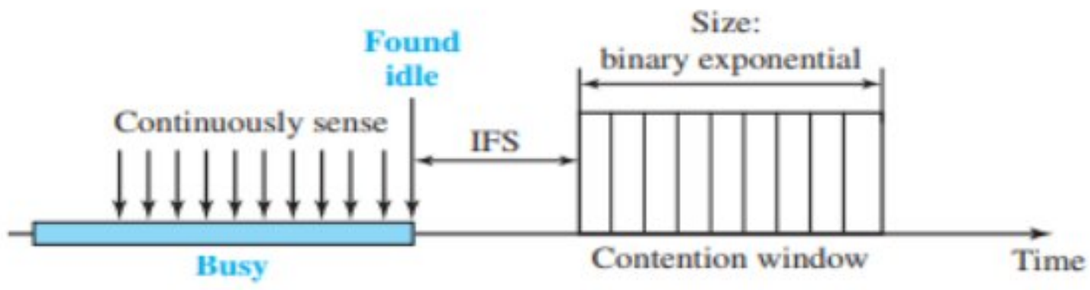


Fig: Contention window

Acknowledgment. With all these precautions, there still may be a collision resulting in destroyed data. In addition, the data may be corrupted during the transmission. The positive acknowledgment and the time-out timer can help guarantee that the receiver has received the frame.

ii) Controlled Access: In controlled access, the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. We discuss ***three*** controlled-access methods.

Reservation:

- In the reservation method, a station needs to make a reservation before sending data. Time is divided into intervals.
- In each interval, a reservation frame precedes the data frames sent in that interval. If there are N stations in the system, there are exactly N reservation minislots in the reservation frame. Each minislot belongs to a station. When a station needs to send a data frame, it makes a reservation in its own minislot.
- The stations that have made reservations can send their data frames after the reservation frame. Figure shows a situation with five stations and a five-minislot reservation frame.
- In the first interval, only stations 1, 3, and 4 have made reservations. In the second interval, only station 1 has made a reservation.

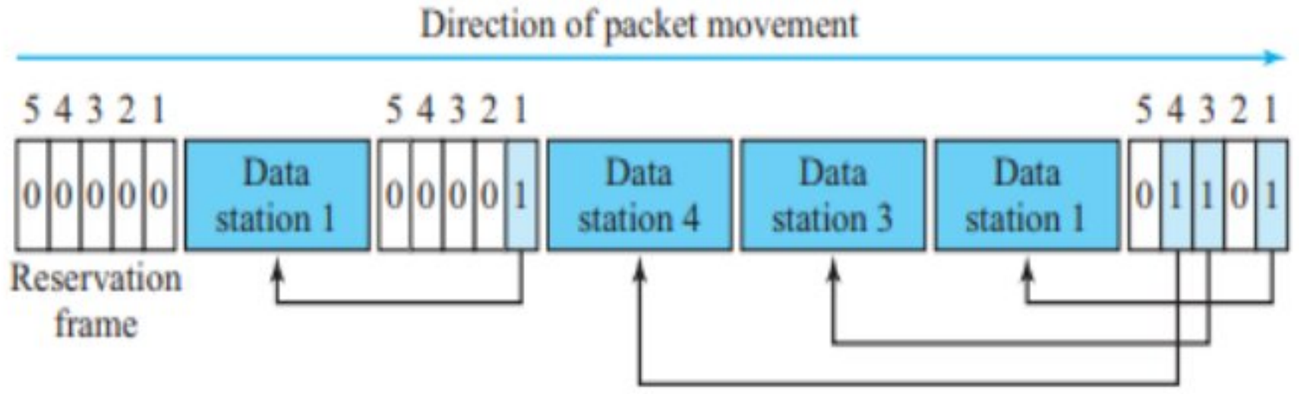


Fig: Reservation access method

Polling

- Polling works with topologies in which one device is designated as a primary station and the other devices are secondary stations.
- All data exchanges must be made through the primary device even when the ultimate destination is a secondary device.
- The primary device controls the link; the secondary devices follow its instructions. It is up to the primary device to determine which device is allowed to use the channel at a given time.
- This method uses ***poll*** and ***select functions*** to prevent collisions.

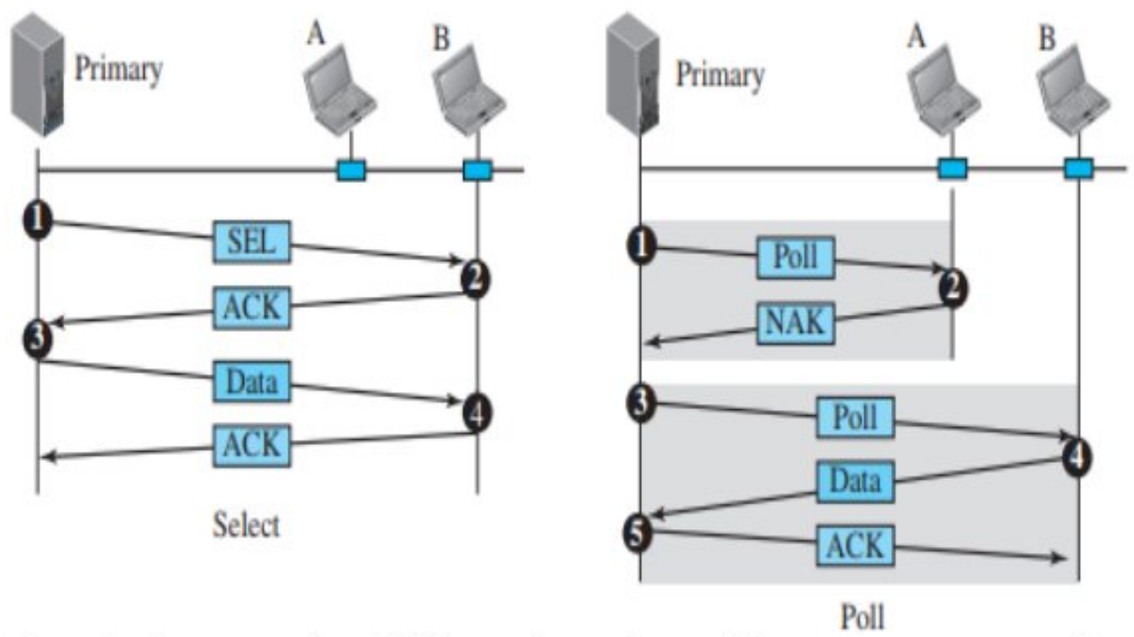


Fig: Select and poll functions in polling-access method

Select: The select function is used whenever the primary device has something to send. Remember that the primary controls the link. If the primary is neither sending nor receiving data, it knows the link is available.

- The primary must alert the secondary to the upcoming transmission and wait for an acknowledgment of the secondary's ready status.
- Before sending data, the primary creates and transmits a select (SEL) frame, one

field of which includes the address of the intended secondary.

Poll: The poll function is used by the primary device to request transmissions from the secondary devices.

- When the primary is ready to receive data, it must ask (poll) each device in turn if it has anything to send.
- When the first secondary is approached, it responds either with a NAK frame if it has nothing to send or with data (in the form of a data frame) if it does.
- If the response is negative (a NAK frame), then the primary polls the next secondary in the same manner until it finds one with data to send.
- When the response is positive (a data frame), the primary reads the frame and returns an acknowledgment (ACK frame), verifying its receipt.

Token Passing

- In the token-passing method, the stations in a network are organized in a logical ring. For each station, there is a predecessor and a successor.
- The predecessor is the station which is logically before the station in the ring; the successor is the station which is after the station in the ring.
- The current station is the one that is accessing the channel now. The right to this access has been passed from the predecessor to the current station.
- The right will be passed to the successor when the current station has no more data to send.

But how is the right to access the channel passed from one station to another? In this method, a special packet called a ***token*** circulates through the ring.

Token management is needed for this access method. Stations must be limited in the time they can have possession of the token. The token must be monitored to ensure it has not been lost or destroyed.
Another function of token management is to assign priorities to the stations and to the types of data being transmitted. And finally, token management is needed to make low-priority stations release the token to high-priority stations.

Logical Ring: In a token-passing network, stations do not have to be physically connected in a ring; the ring can be a logical one. Figure shows ***four*** different physical topologies that can create a logical ring.

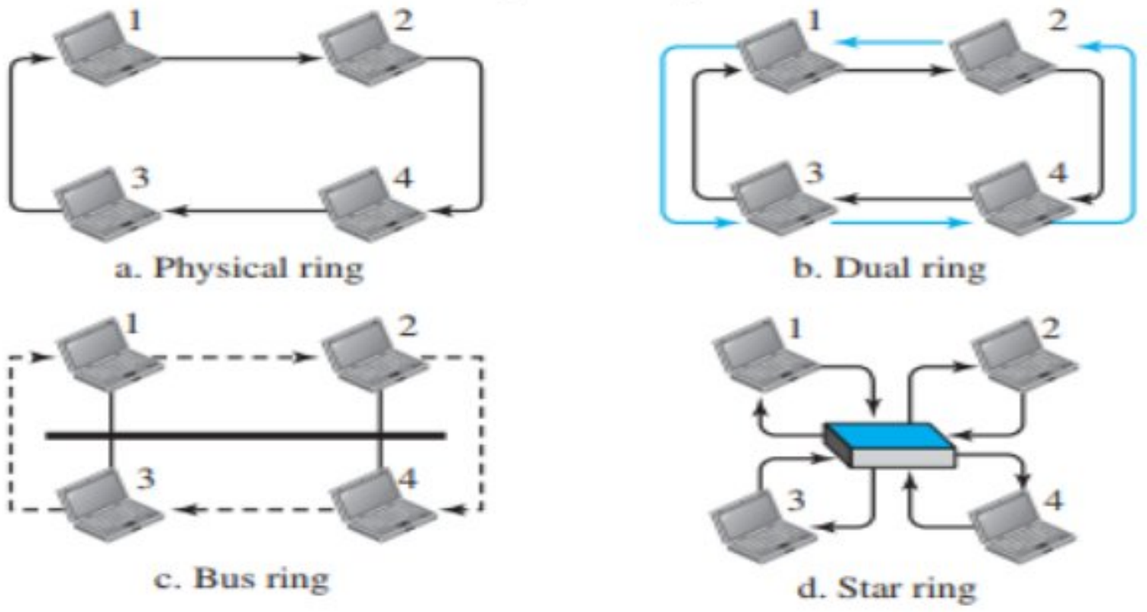


Fig: Logical ring and physical topology in token-passing access method

iii) Channelization: Channelization (or channel partition, as it is sometimes called) is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, among different stations. We discuss three channelization protocols: FDMA, TDMA, and CDMA.

FDMA

- In frequency-division multiple access (FDMA), the available bandwidth is divided into frequency bands.

- Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time.
- Each station also uses a bandpass filter to confine the transmitter frequencies.
- To prevent station interferences, the allocated bands are separated from one another by small guard bands. Figure shows the idea of FDMA.

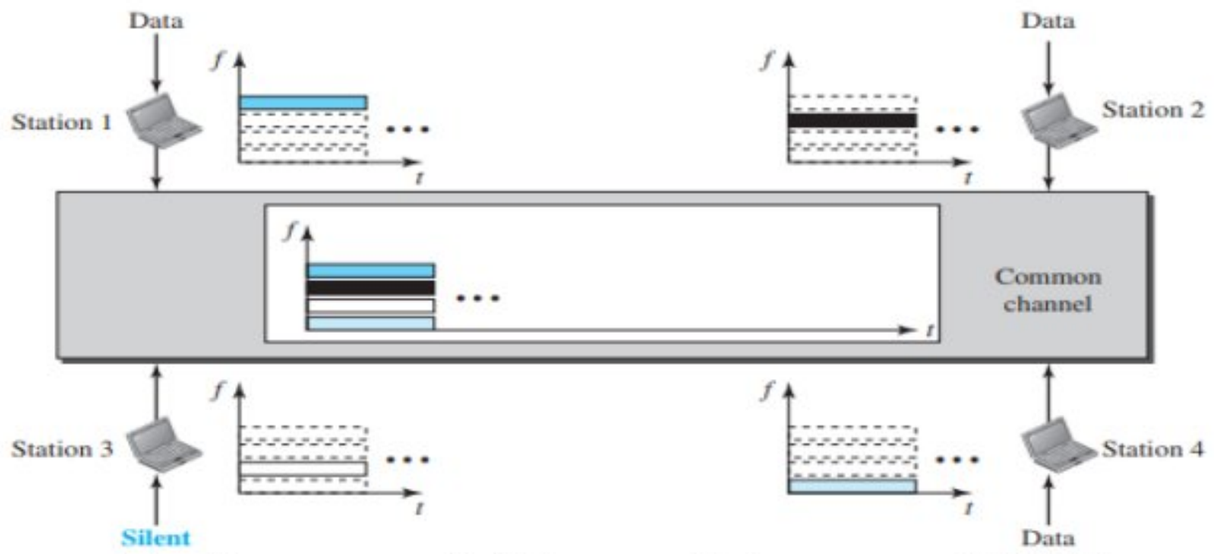


Fig: Frequency-division multiple access (FDMA)

TDMA

In time-division multiple access (TDMA), the stations share the bandwidth of the channel in time. Each station is allocated a time slot during which it can send data. Each station transmits its data in its assigned time slot. Figure shows the idea behind TDMA.

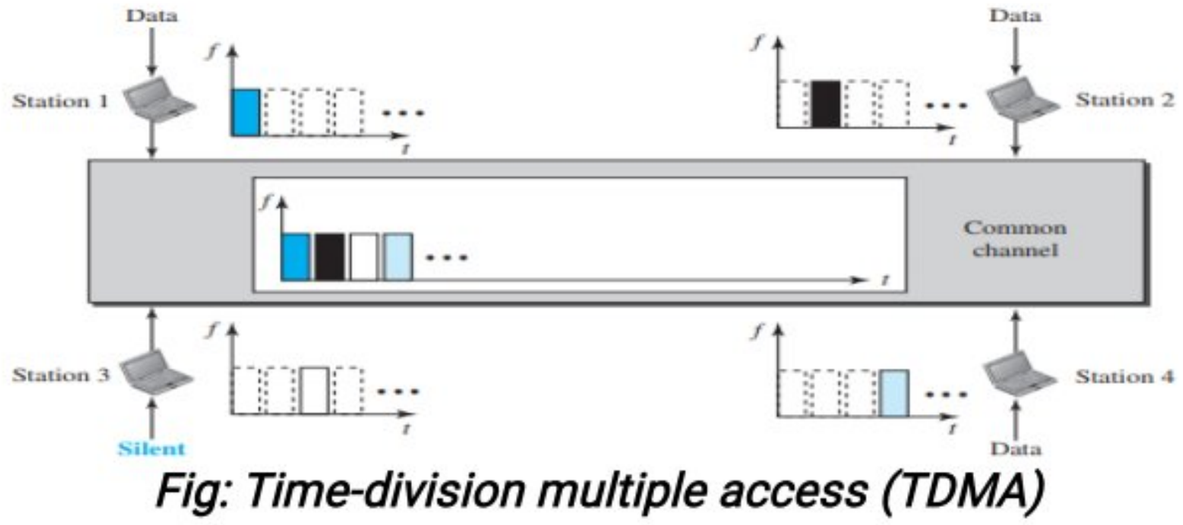
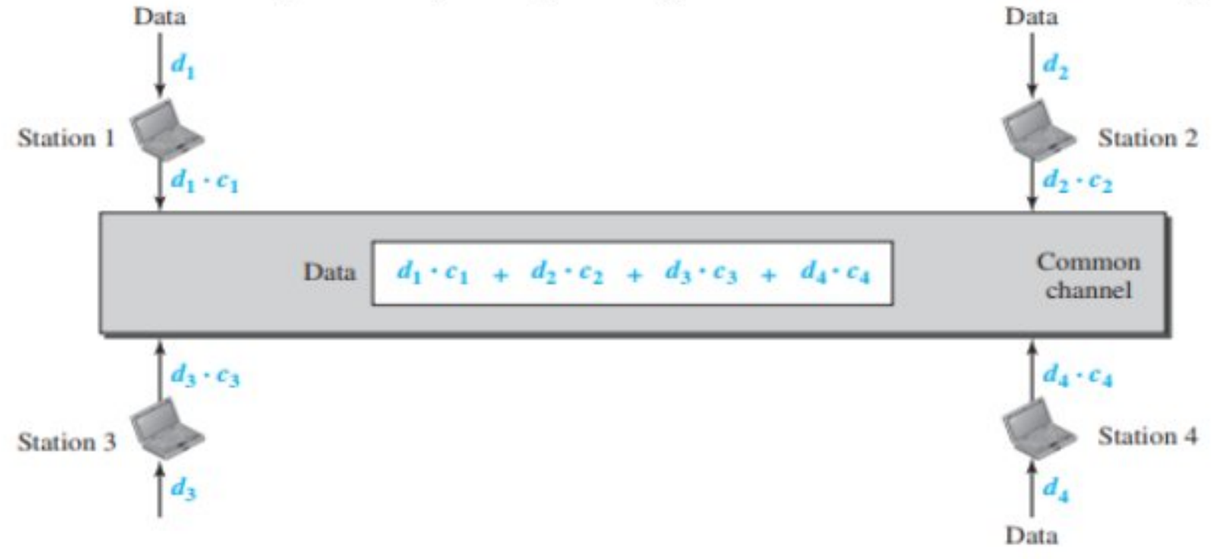


Fig: Time-division multiple access (TDMA)

CDMA

Code-division multiple access (CDMA) was conceived several decades ago. In CDMA, one channel carries all transmissions simultaneously.
In CDMA, the stations use different codes to achieve multiple accesses. CDMA is based on coding theory and uses sequence of numbers called chips.

Let us assume we have four stations, 1, 2, 3, and 4, connected to the same channel. The data from station 1 are d1, from station 2 are d2, and so on. The code assigned to the first station is c1, to the second is c2, and so on.
We assume that the assigned codes have two properties.
1. If we multiply each code by another, we get 0.
2. If we multiply each code by itself, we get 4 (the number of stations).



$$\begin{aligned} \text{data} &= (d_1 \cdot c_1 + d_2 \cdot c_2 + d_3 \cdot c_3 + d_4 \cdot c_4) \cdot c_1 \\ &= d_1 \cdot c_1 \cdot c_1 + d_2 \cdot c_2 \cdot c_1 + d_3 \cdot c_3 \cdot c_1 + d_4 \cdot c_4 \cdot c_1 = 4 \times d_1 \end{aligned}$$

Fig: Simple idea of communication with code

5. Switched Local Area Networks

Figure shows a switched local network connecting three departments, two servers and a router with four switches. Because these switches operate at the link layer, they switch link-layer frames.

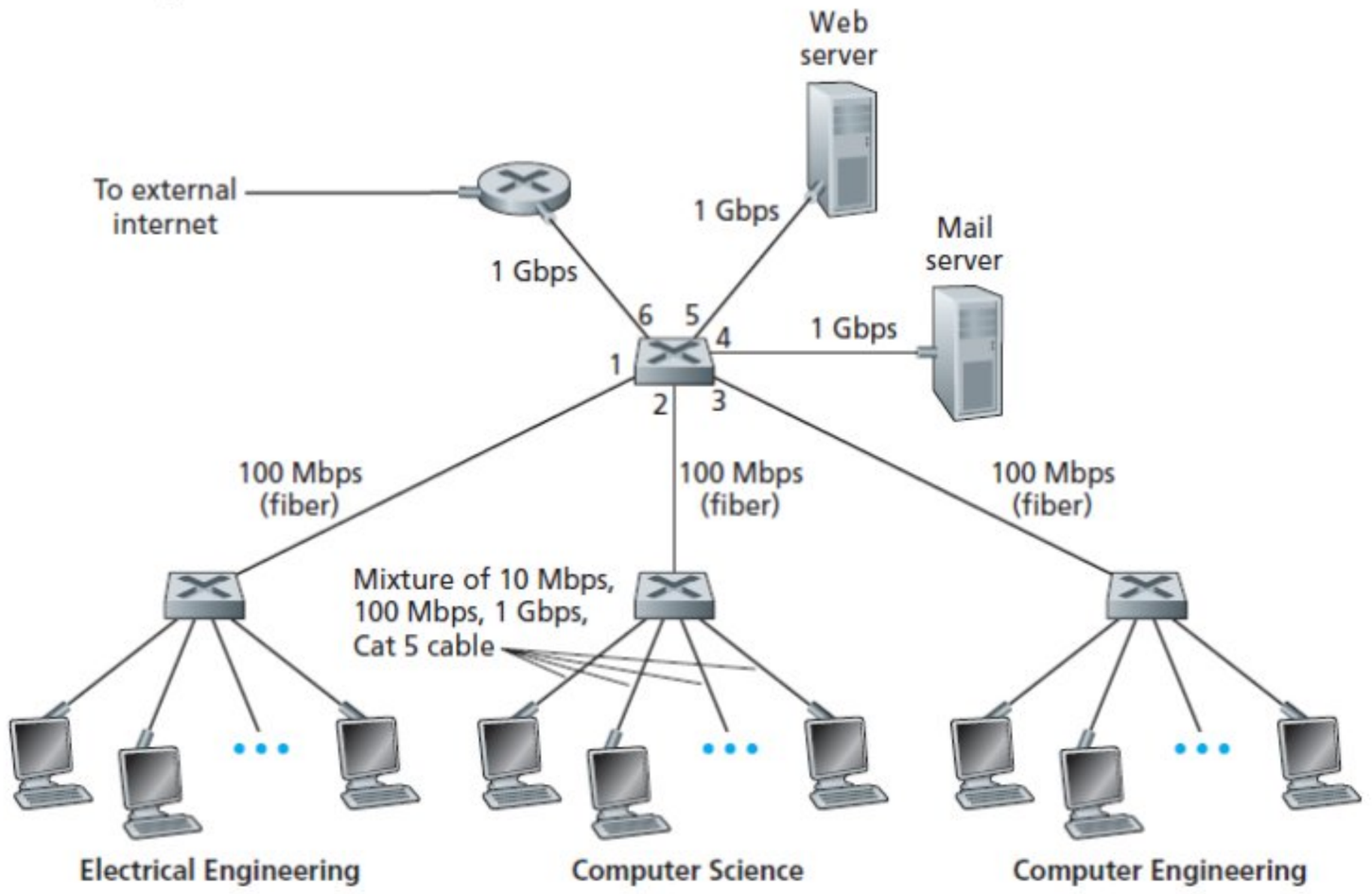


Fig: An institutional network connected together by four switches

Instead of using IP addresses, we will soon see that they use link-layer addresses to forward link-layer frames through the network of switches.

Link-Layer Addressing

In a connectionless internetwork such as the Internet we cannot make a datagram reach its destination using only IP addresses. The reason is that each datagram in the Internet, from the same source host to the same destination host, may take a different path.

The source and destination IP addresses define the two ends but cannot define which links the datagram should pass through.

So, we need another addressing mechanism in a connectionless internetwork: the **link-layer addresses** of the two nodes.

A link-layer address is sometimes called a **link address,** sometimes **a physical address,** and sometimes **a MAC address**.

When a datagram passes from the network layer to the data-link layer, the datagram will be encapsulated in a frame and two data-link addresses are added to the frame header. These two addresses are changed every time the frame moves from one link to another.

Address Resolution Protocol (ARP): the IP address of the next node is not helpful in moving a frame through a link; we need the link-layer address of the next node. This is the time when the Address Resolution Protocol (ARP) becomes helpful.

The ARP protocol is one of the auxiliary protocols defined in the network layer. It belongs to the network layer; it maps an IP address to a logical-link address.

ARP accepts an IP address from the IP protocol, maps the address to the corresponding link-layer address, and passes it to the data-link layer.

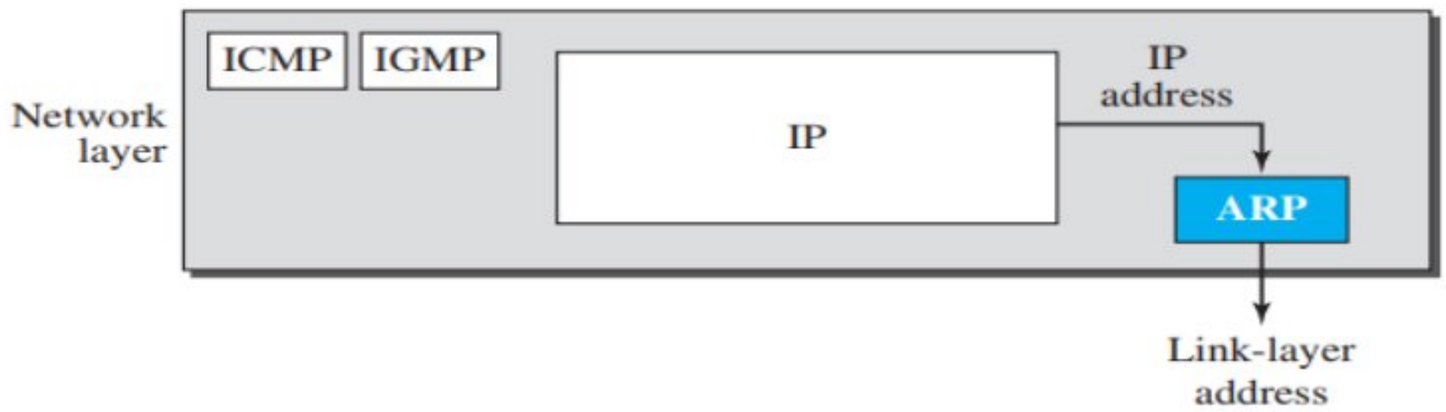
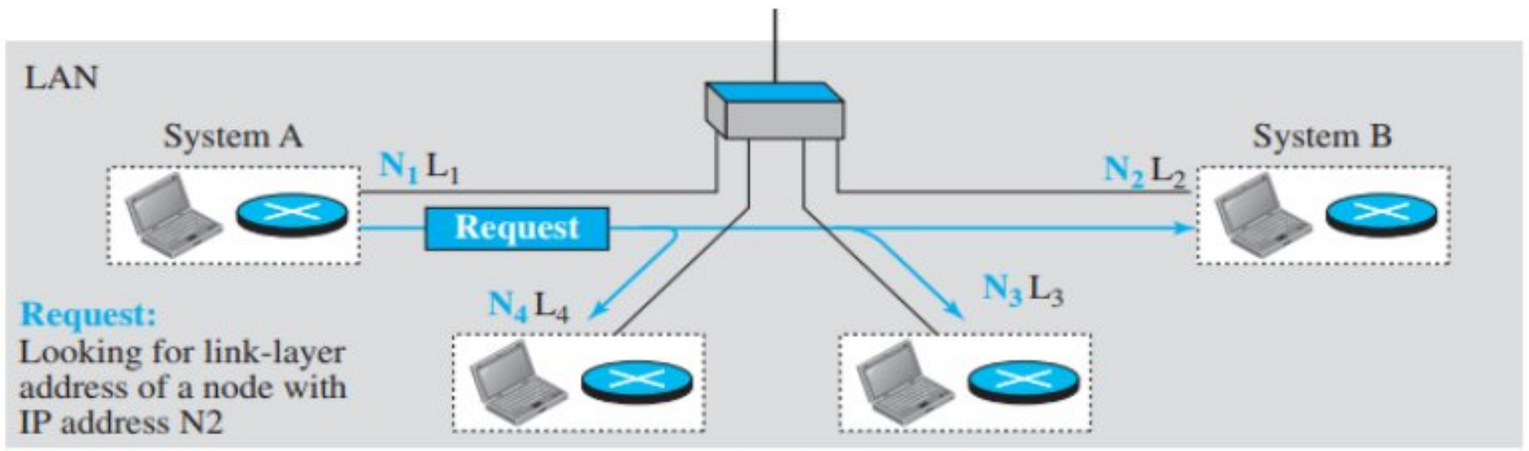


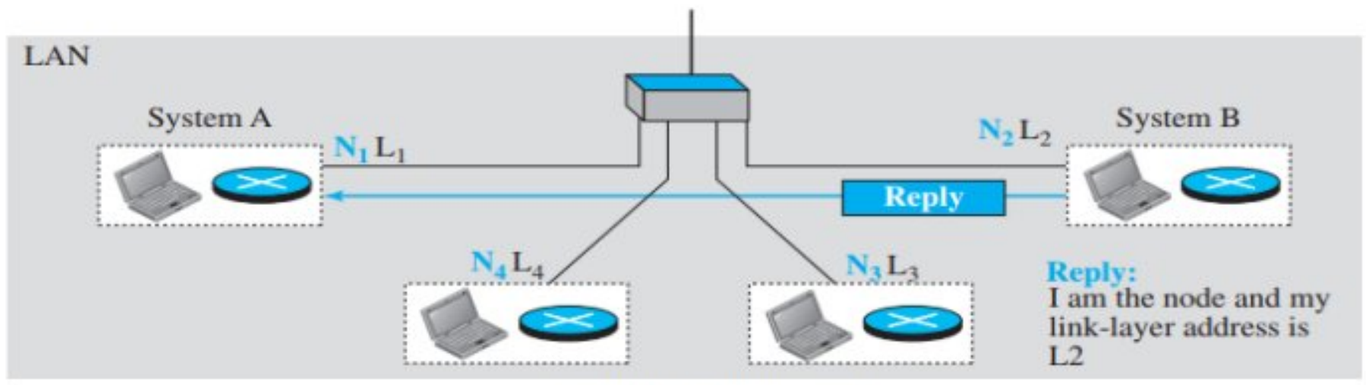
Fig: Position of ARP in TCP/IP protocol suite

ARP operation: ARP request, ARP response.
Anytime a host or a router needs to find the link-layer address of another host or router in its network, it sends an ***ARP request packet***. The packet includes the link-layer and IP addresses of the sender and the IP address of the receiver. Because the sender does not know the link-layer address of the receiver, the query is broadcast over the link using the link-layer broadcast address.



a. ARP request is broadcast

Every host or router on the network receives and processes the ARP request packet, but only the intended recipient recognizes its IP address and sends back an ***ARP response packet***. The response packet contains the recipient's IP and link-layer addresses. The packet is unicast directly to the node that sent the request packet.



b. ARP reply is unicast

ARP Packet Format: Figure shows the format of an ARP packet. The hardware type field defines the type of the link-layer protocol; the protocol type field defines the network-layer protocol.
The source hardware and source protocol addresses are variable-length fields defining the link-layer and network-layer addresses of the sender.
The destination hardware address and destination protocol address fields define the receiver link-layer and network-layer addresses. An ARP packet is encapsulated directly into a data-link frame.

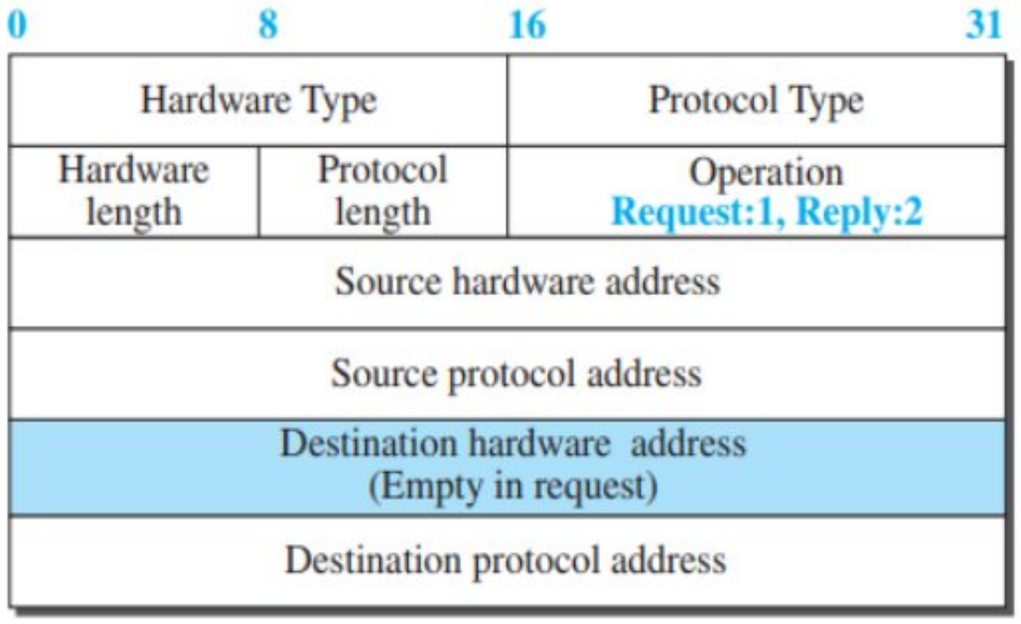


Fig: ARP packet

Ethernet (IEEE Standard 802.3)

Ethernet is the traditional technology for connecting devices in a wired local area network (LAN) or wide area network (WAN). Ethernet was the first widely deployed high-speed LAN.

- **Frame Format**: The Ethernet frame contains seven fields, as shown in Figure.

Preamble. This field contains 7 bytes (56 bits) of alternating 0s and 1s that alert the receiving system to the coming frame and enable it to synchronize its clock if it's out synchronization. The pattern provides only an alert and a timing pulse.

Destination address (DA). This field is six bytes (48 bits) and contains the link-layer address of the destination station or stations to receive the packet.

Source address (SA). This field is also six bytes and contains the link-layer address of the sender of the packet.

Type. This field defines the upper-layer protocol whose packet is encapsulated in the frame. This protocol can be IP, ARP, OSPF, and so on.

Data. This field carries data encapsulated from the upper-layer protocols. It is a minimum of 46 and a maximum of 1500 bytes.

Cyclic redundancy check (CRC) *(4 bytes)*. the purpose of the CRC field is to allow the receiving adapter, adapter B, to detect bit errors in the frame.

Ethernet Technologies:

1. Standard Ethernet (l0 Mbps),
2. Fast Ethernet (100 Mbps),
3. Gigabit Ethernet (l Gbps)
4. Ten-Gigabit Ethernet (l0 Gbps)

Standard Ethernet: A standard Ethernet network can transmit data at a rate up to 10 Megabits per second (10 Mbps). The Institute for Electrical and Electronic Engineers developed an Ethernet standard known as IEEE Standard 802.3. This standard defines rules for configuring an Ethernet network and specifies how the elements in an Ethernet network interact with one another.

Fast Ethernet (100 Mbps): 802.3u

Ethernet made a big jump by increasing the transmission rate to 100 Mbps, and the new generation was called the *Fast Ethernet*.
The goals of Fast Ethernet can be summarized as follows:
1. Upgrade the data rate to 100 Mbps.
2. Make it compatible with Standard Ethernet.
3. Keep the same 48-bit address.
4. Keep the same frame format.
5. Keep the same minimum and maximum frame lengths.

Gigabit Ethernet (1 Gbps): 802.3z
The need for an even higher data rate resulted in the design of the Gigabit Ethernet Protocol (1000 Mbps). The IEEE committee calls it the Standard 802.3z. The goals of the Gigabit Ethernet were to upgrade the data rate to 1 Gbps.
The goals of the Gigabit Ethernet design can be summarized as follows:
1. Upgrade the data rate to 1 Gbps.
2. Make it compatible with Standard or Fast Ethernet.
3. Use the same 48-bit address.
4. Use the same frame format.
5. Keep the same minimum and maximum frame lengths.

10 Gigabit Ethernet (10 Gbps): 802.3ae
1. Upgrade the data rate to 10 Gbps.
2. Make it compatible with Standard or Fast Ethernet.
3. Use the same 48-bit address.
4. Use the same frame format.
5. Keep the same minimum and maximum frame lengths.

6. Link Virtualization: A Network as a Link Layer

- A switch that supports **virtual local area networks** (**VLANs**). As the name suggests, a switch that supports VLANs allows multiple *virtual* local area networks to be defined over a single physical local area network infrastructure.
- Hosts within a VLAN communicate with each other as if they (and no other hosts) were connected to the switch. In a port-based VLAN, the switch's ports (interfaces) are divided into groups by the network manager.
- Each group constitutes a VLAN, with the ports in each VLAN forming a broadcast domain (i.e., broadcast traffic from one port can only reach other ports in the group).
- Figure shows a single switch with 16 ports. Ports2 to 8 belong to the EE VLAN, while ports 9 to 15 belong to the CS VLAN (ports1 and 16 are unassigned).

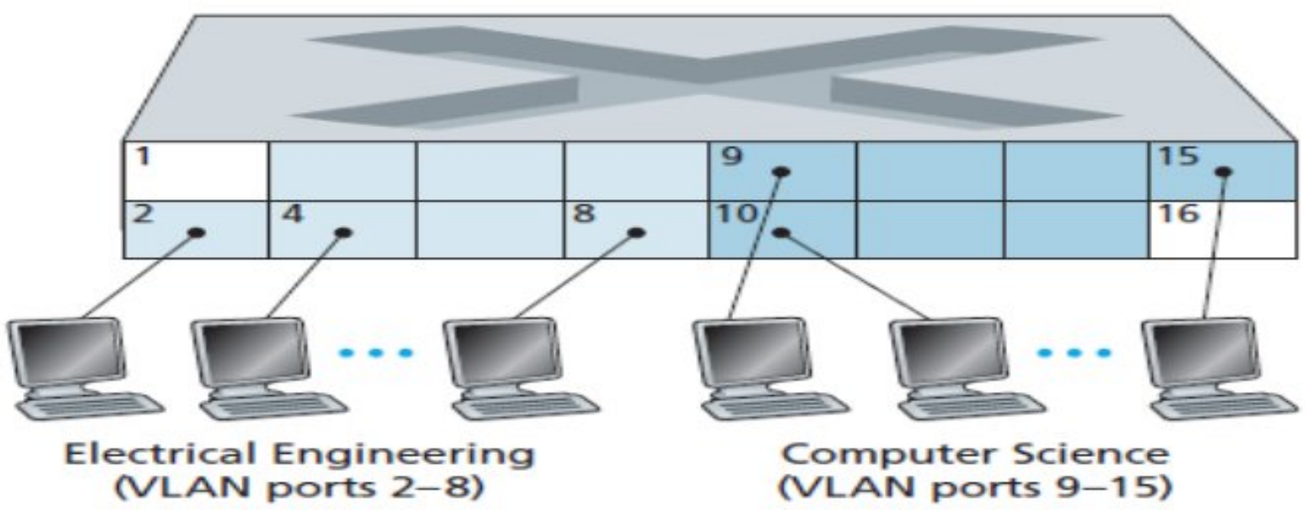


Fig: A single switch with two configured VLANs

A more scalable approach to interconnecting VLAN switches is known as **VLAN trunking**. In the VLAN trunking approach shown in Figure, a special port on each switch (port 16 on the left switch and port 1 on the right switch) is configured as a trunk port to interconnect the two VLAN switches. The trunk port belongs to all VLANs, and frames sent to any VLAN are forwarded over the trunk link to the other switch.

The IEEE has defined an extended Ethernet frame format, 802.1Q, for frames crossing a VLAN trunk. As shown in Figure, the 802.1Q frame consists of the standard Ethernet frame with a four-byte **VLAN tag** added into the header that carries the identity of the VLAN to which the frame belongs.

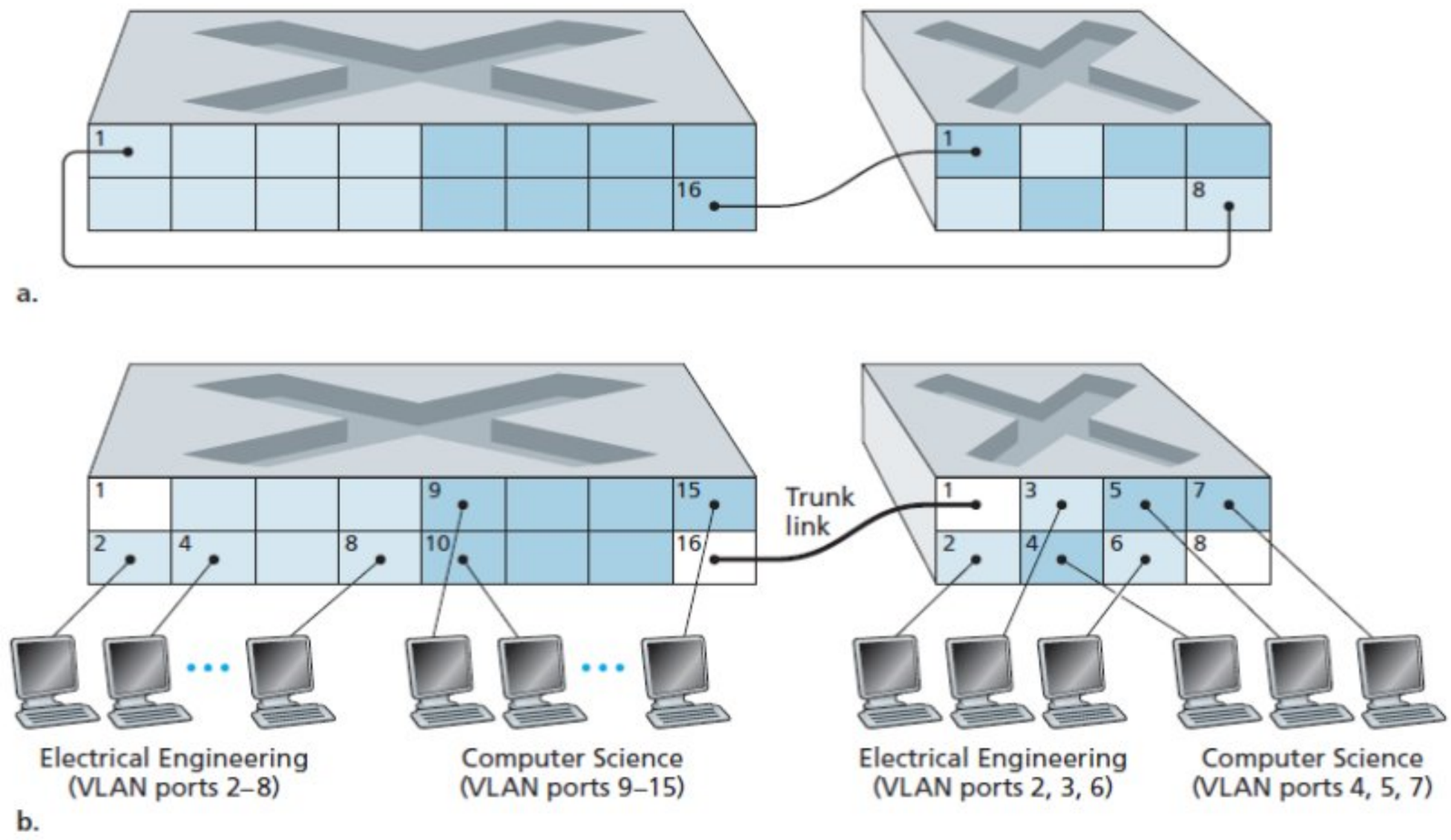


Fig: Connecting two VLAN switches with two VLANs: (a) two cables (b) trunked

7. Data Center Networking

In recent years, Internet companies such as Google, Microsoft, Facebook, and Amazon have built massive data centers, each housing tens to hundreds of thousands of hosts, and concurrently supporting many distinct cloud applications (e.g., search, email, social networking, and e-commerce).

- A data center is a facility that centralizes an organization's shared IT operations and equipment for the purposes of storing, processing, and disseminating data and applications.
- Each data center has its own data center network that interconnects its hosts with each other and interconnects the data center with the Internet.
- The hosts in data centers, called **blades** and like pizza boxes, are generally commodity hosts that include CPU, memory, and disk storage. The hosts are stacked in racks, with each rack typically having 20 to 40 blades.
- At the top of each rack there is a switch, aptly named the **Top of Rack (TOR)** switch, that interconnects the hosts in the rack with each other and with other switches in the data center.
- Specifically, each host in the rack has a network interface card that connects to its TOR switch, and each TOR switch has additional ports that can be connected to other switches. Each host is also assigned its own data-center-internal IP address.
- The data center network supports **two types of traffic**: traffic flowing between external clients and internal hosts and traffic flowing between internal hosts. To handle flows between external clients and internal hosts, the data center network includes one or more **border routers**, connecting the data center network to the public Internet. The data center network therefore interconnects the racks with each other and connects the racks to the border routers.
- Figure shows an example of a data center network. **Data center network design**, the art of designing the interconnection network and protocols that connect the racks with each other and with the border routers.

Fig: A data center network with a hierarchical topology

Load Balancing

- A cloud data center, such as a Google or Microsoft data center, provides many applications concurrently, such as search, email, and video applications.
- To support requests from external clients, each application is associated with a publicly visible IP address to which clients send their requests and from which they receive responses.
- Inside the data center, the external requests are first directed to a load balancer whose job it is to distribute requests to the hosts, balancing the load across the hosts as a function of their current load.
- A large data center will often have several load balancers, each one devoted to a set of specific cloud applications.

8. Retrospective: A Day in the Life of a Web Page Request

Getting Started: DHCP, UDP, IP, and Ethernet

Let's suppose that Bob boots up his laptop and then connects it to an Ethernet cable connected Ethernet switch.

When Bob first connects his laptop to the network, he can't do anything (e.g., download a Web page) without an IP address. Thus, the first network-related action taken by Bob's laptop is to run the DHCP protocol to obtain an IP address, as well as other information, from the local DHCP server:

1. The operating system on Bob's laptop creates a DHCP request message and puts this message within a UDP segment with destination port 67 (DHCP server) and source port 68 (DHCP client). The UDP segment is then placed within an IP datagram with a broadcast IP destination address (255.255.255.255) and a source IP address of 0.0.0.0, since Bob's laptop doesn't yet have an IP address.

2. The IP datagram containing the DHCP request message is then placed within an Ethernet frame. The Ethernet frame has a destination MAC addresses of FF:FF:FF:FF:FF:FF so that the frame will be broadcast to all devices connected to the switch (hopefully including a DHCP server); the frame's source MAC address is that of Bob's laptop, 00:16:D3:23:68:8A.

3. The broadcast Ethernet frame containing the DHCP request is the first frame sent by Bob's laptop to the Ethernet switch. The switch broadcasts the incoming frame on all outgoing ports, including the port connected to the router.

4. The router receives the broadcast Ethernet frame containing the DHCP request on its interface with MAC address 00:22:6B:45:1F:1B and the IP datagram is extracted from the Ethernet frame. The DHCP request message is extracted from the UDP segment. The DHCP server now has the DHCP request message.

5. Let's suppose the DHCP server allocates address 68.85.2.101 to Bob's laptop. The DHCP server creates a DHCP ACK message containing this IP address. The DHCP message is put inside a UDP segment, which is put inside an IP datagram, which is put inside an Ethernet frame.

6. The Ethernet frame containing the DHCP ACK is sent (unicast) by the router to the switch. Because the switch is self-learning and previously received an Ethernet frame (containing the DHCP request) from Bob's laptop.

7. Bob's laptop receives the Ethernet frame containing the DHCP ACK, extracts the IP datagram from the Ethernet frame. At this point, Bob's laptop has initialized its networking components and is ready to begin processing the Web page fetch.

Still Getting Started: DNS and ARP

When Bob types the URL for www.google.com into his Web browser, he begins the long chain of events that will eventually result in Google's home page being displayed by his Web browser.

8. The operating system on Bob's laptop thus creates a DNS query message, putting the string "www.google.com" in the question section of the DNS message. This DNS message is then placed within a UDP segment with a destination port of 53 (DNS server).

9. Bob's laptop then places the datagram containing the DNS query message in an Ethernet frame. However, even though Bob's laptop knows the IP address of the gateway router (68.85.2.1) via the DHCP ACK message in step 5 above, it doesn't know the gateway router's MAC address. To obtain the MAC address of the gateway router, Bob's laptop will need to use the ARP protocol.

10. Bob's laptop creates an ARP query message with a target IP address of 68.85.2.1 (the default gateway), places the ARP message within an Ethernet frame with a broadcast destination address and sends the Ethernet frame to the switch, which delivers the frame to all connected devices, including the gateway router.

11. The gateway router receives the frame containing the ARP query message on the interface and finds that the target IP address of 68.85.2.1 in the ARP message matches the IP address of its interface. The gateway router thus prepares an ARP reply, indicating that its MAC address of 00:22:6B:45:1F:1B corresponds to IP address 68.85.2.1.

12. Bob's laptop receives the frame containing the ARP reply message and extracts the MAC address of the gateway router (00:22:6B:45:1F:1B) from the ARP reply message.

13. Bob's laptop extracts the IP address of the server www.google.com from

the DNS message. Finally, after a lot of work, Bob's laptop is now ready to contact the www.google.com server!

Web Client-Server Interaction: TCP and HTTP

14. Now that Bob's laptop has the IP address of www.google.com,it can create the TCP socket that will be used to send the HTTP GET message to www.google.com. When Bob creates the TCP socket, the TCP in Bob's laptop must first perform a three-way handshake with the TCP in www.google.com. Bob's laptop thus first creates a TCP SYN segment with destination port 80 (for HTTP), places the TCP segment inside an IP datagram with a destination IP address of 64.233.169.105 (www.google.com), places the datagram inside a frame with a destination MAC address of 00:22:6B:45:1F:1B (the gateway router) and sends the frame to the switch.

15. Eventually, the datagram containing the TCP SYN arrives at www.google.com. A connection socket is created for the TCP connection between the Google HTTP server and Bob's laptop. A TCP SYNACK segment is generated, placed inside a datagram addressed to Bob's laptop.

16. The datagram containing the TCP SYNACK segment is forwarded through the Google, Bob's network, eventually arriving at the Ethernet card in Bob's laptop. The datagram is demultiplexed within the operating system to the TCP socket enters the connected state.

17. With the socket on Bob's laptop now (finally!) ready to send bytes to www.google .com, Bob's browser creates the HTTP GET message containing the URL to be fetched. The TCP segment is placed in a datagram and sent and delivered to www.google.com.

18. The HTTP server at www.google.com reads the HTTP GET message from the TCP socket, creates an HTTP response message, places the requested Web page content in the body of the HTTP response message, and sends the message into the TCP socket.

19. The datagram containing the HTTP reply message is forwarded through the Google, bob's network, and arrives at Bob's laptop. Bob's Web browser program reads the HTTP response from the socket, extracts the html for the Web page from the body of the HTTP response, and finally displays the Web page!