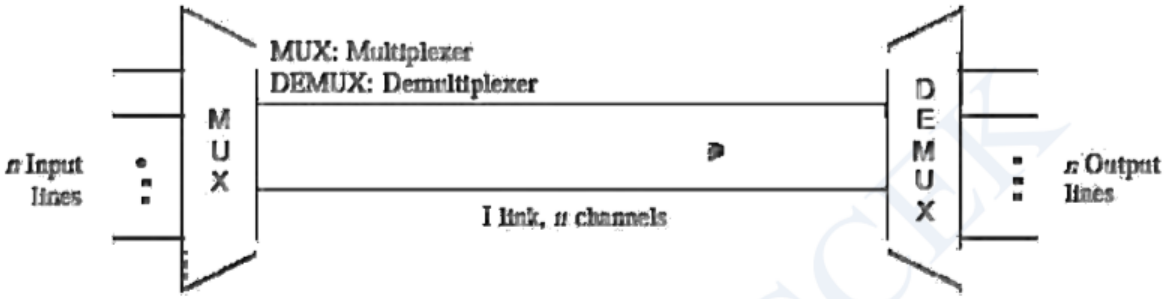
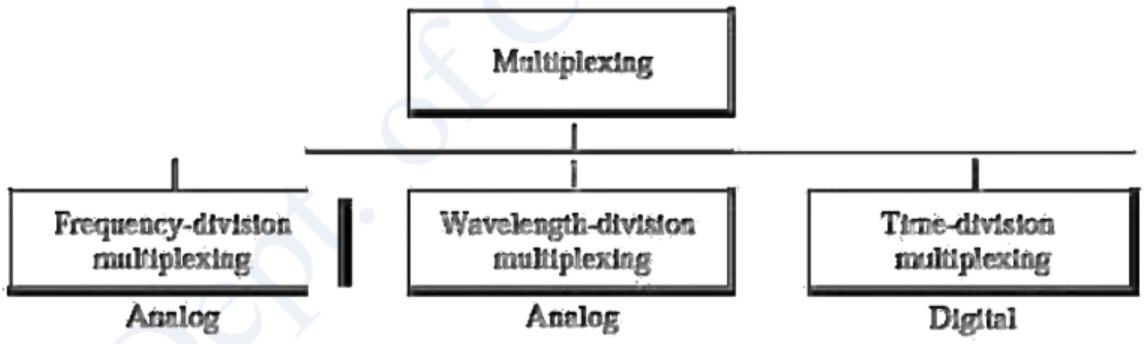
At the receiving end, that stream is fed into a de-multiplexer (DEMUX), which separates the stream back into its component transmissions (one-to-many) and directs them to their corresponding lines. In the figure, the word link refers to the physical path.

The word channel refers to the portion of a link that carries a transmission between a given pair of lines. One link can have many (n) channels.



There are three basic multiplexing techniques: frequency-division multiplexing, wavelength-division multiplexing, and time-division multiplexing. The first two are techniques designed for analog signals, the third, for digital signals.



Frequency-Division Multiplexing

Frequency-division multiplexing (FDM) is an analog technique that can be applied when the bandwidth of a link (in hertz) is greater than the combined bandwidths of the signals to be transmitted.

In FDM, signals generated by each sending device modulate different carrier frequencies. These modulated signals are then combined into a single composite signal that can be transported by the link.

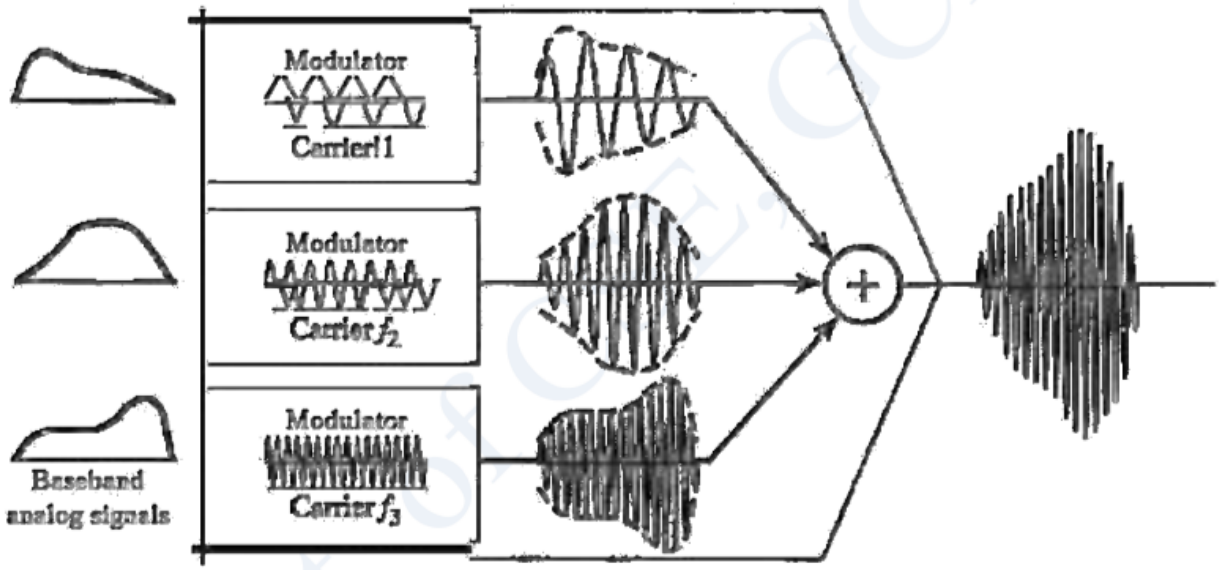
Carrier frequencies are separated by sufficient bandwidth to accommodate the modulated signal.

These bandwidth ranges are the channels through which the various signals travel. Channels can be separated by strips of unused bandwidth-guard bands-to prevent signals from overlapping.

In addition, carrier frequencies must not interfere with the original data frequencies.

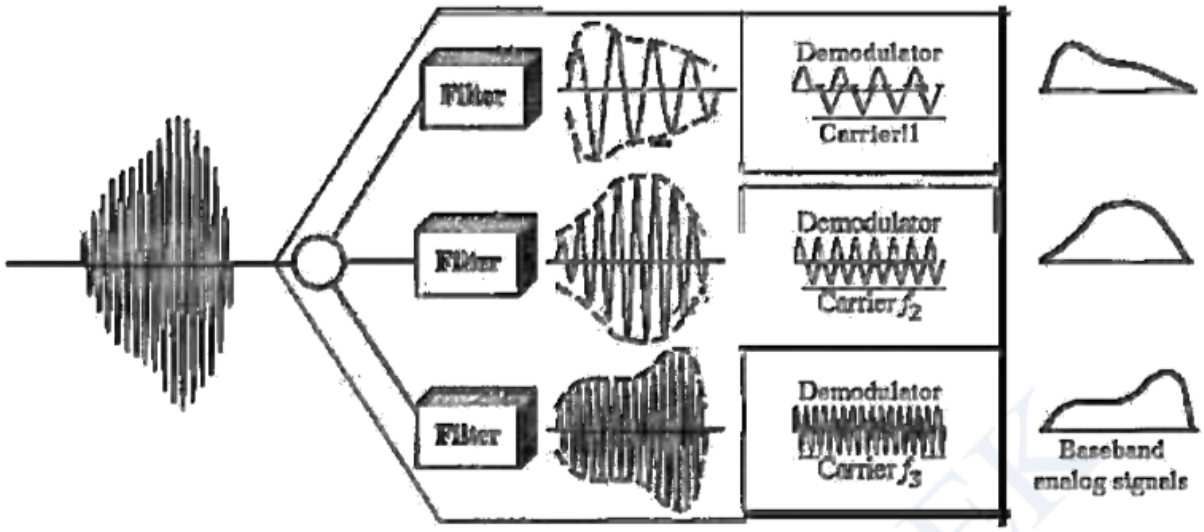
Multiplexing Process

Each source generates a signal of a similar frequency range. Inside the multiplexer, these similar signals modulate different carrier frequencies f1, f2, and f3). The resulting modulated signals are then combined into a single composite signal that is sent out over a media link that has enough bandwidth to accommodate it.



Demultiplexing Process

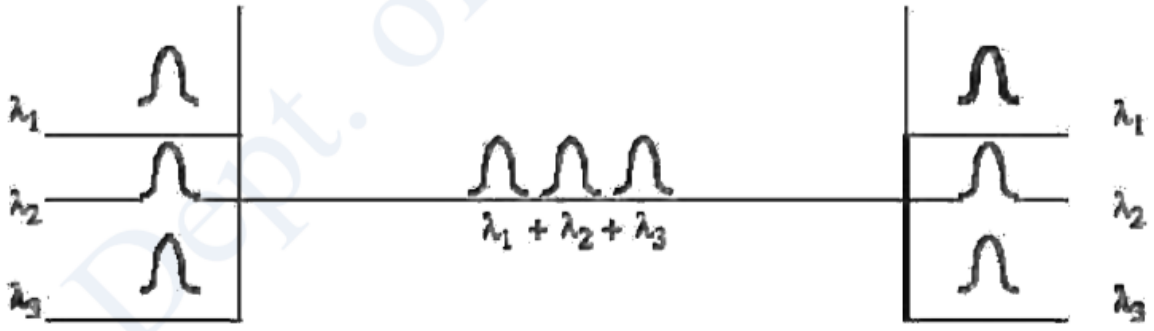
The de-multiplexer uses a series of filters to decompose the multiplexed signal into its constituent component signals. The individual signals are then passed to a demodulator that separates them from their carriers and passes them to the output lines.



Wavelength-Division Multiplexing

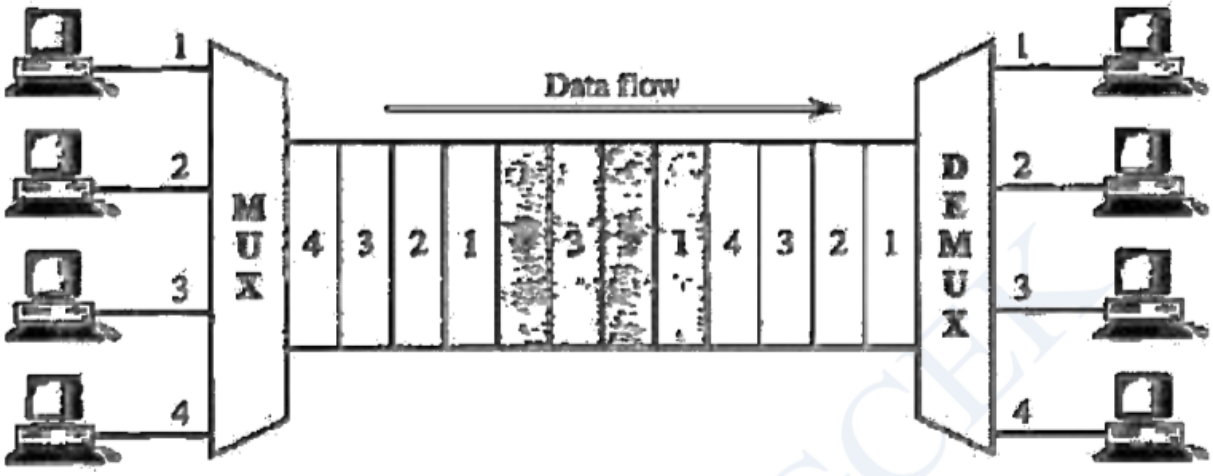
Wavelength-division multiplexing (WDM) is designed to use the high-data-rate capability of fiber-optic cable. The optical fiber data rate is higher than the data rate of metallic transmission cable. Using a fiber-optic cable for one single line wastes the available bandwidth. Multiplexing allows us to combine several lines into one.

WDM is conceptually the same as FDM, except that the multiplexing and de-multiplexing involve optical signals transmitted through fiber-optic channels. The idea is the same: We are combining different signals of different frequencies. The difference is that the frequencies are very high.



Time-Division Multiplexing

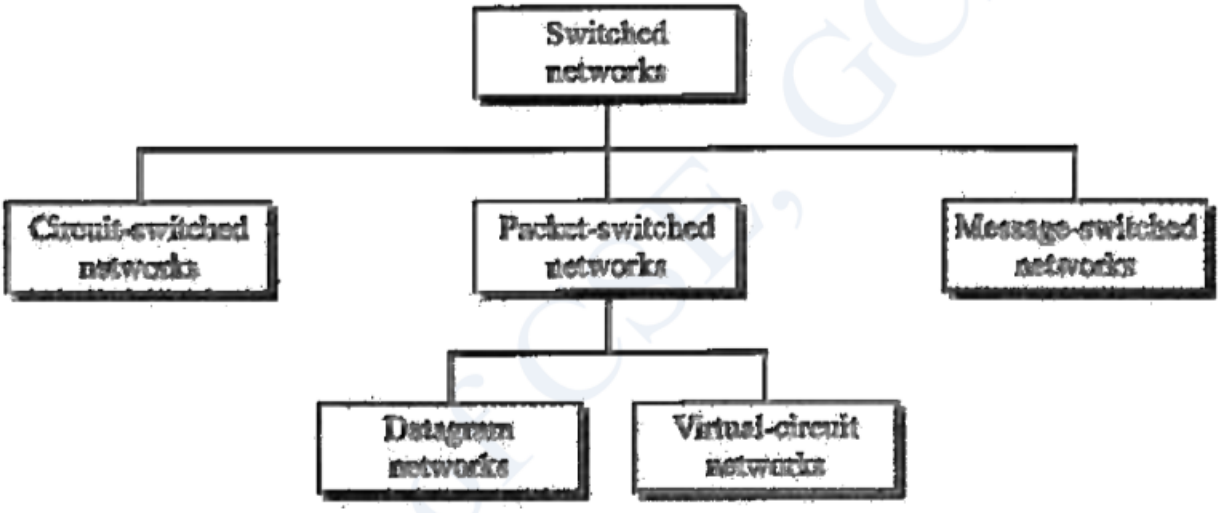
Time-division multiplexing (TDM) is a digital process that allows several connections to share the high bandwidth of a link Instead of sharing a portion of the bandwidth as in FDM, time is shared. Each connection occupies a portion of time in the link.



SWITCHING

A switched network consists of a series of interlinked nodes, called switches. Switches are devices capable of creating temporary connections between two or more devices linked to the switch. In a switched network, some of these nodes are connected to the end systems.

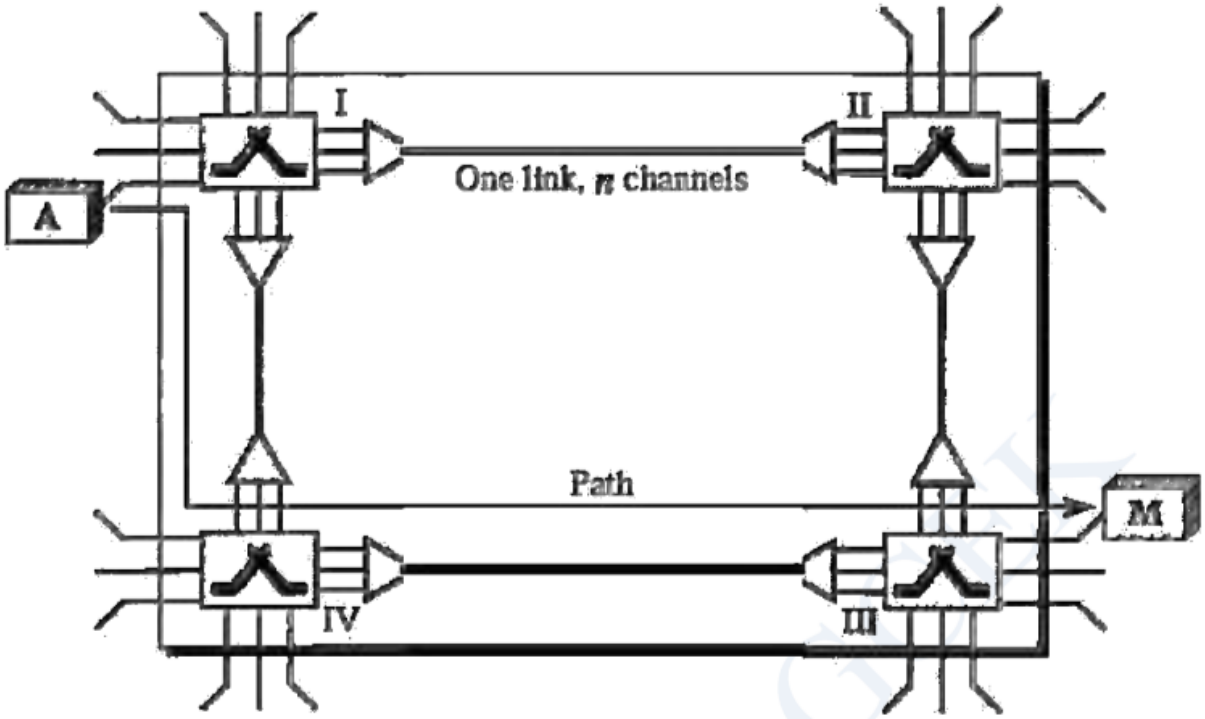
Traditionally, three methods of switching have been important: **circuit switching**, **packet switching**, and **message switching**. The first two are commonly used today. The third has been phased out in general communications but still has networking applications. We can then divide today's networks into three broad categories: circuit-switched networks, packet-switched networks, and message-switched. Packet-switched networks can further be divided into two subcategories-virtual-circuit networks and datagram networks as shown in Figure.



Circuit switching and Telephone Network

A circuit-switched network consists of a set of switches connected by physical links. A connection between two stations is a dedicated path made of one or more links.

However, each connection uses only one dedicated channel on each link. Each link is normally divided into n channels by using FDM or TDM.



Circuit switching takes place at the physical layer.

Before starting communication, the stations must make a reservation for the resources to be used during the communication. These resources, such as channels (bandwidth in FDM and time slots in TDM), switch buffers, switch processing time, and switch input/output ports, must remain dedicated during the entire duration of data transfer until the teardown phase.

Data transferred between the two stations are not packetized (physical layer transfer of the signal). The data are a continuous flow sent by the source station and received by the destination station, although there may be periods of silence.

There is no addressing involved during data transfer. The switches route the data based on their occupied band (FDM) or time slot (TDM). Of course, there is end-to-end addressing used during the setup phase.

Three Phases

The actual communication in a circuit-switched network requires three phases: connection setup, data transfer, and connection teardown.

Setup Phase

Before the two parties (or multiple parties in a conference call) can communicate, a dedicated circuit (combination of channels in links) needs to be established. The end systems are normally connected through dedicated lines to the switches, so connection setup means creating dedicated channels between the switches.

Data Transfer Phase

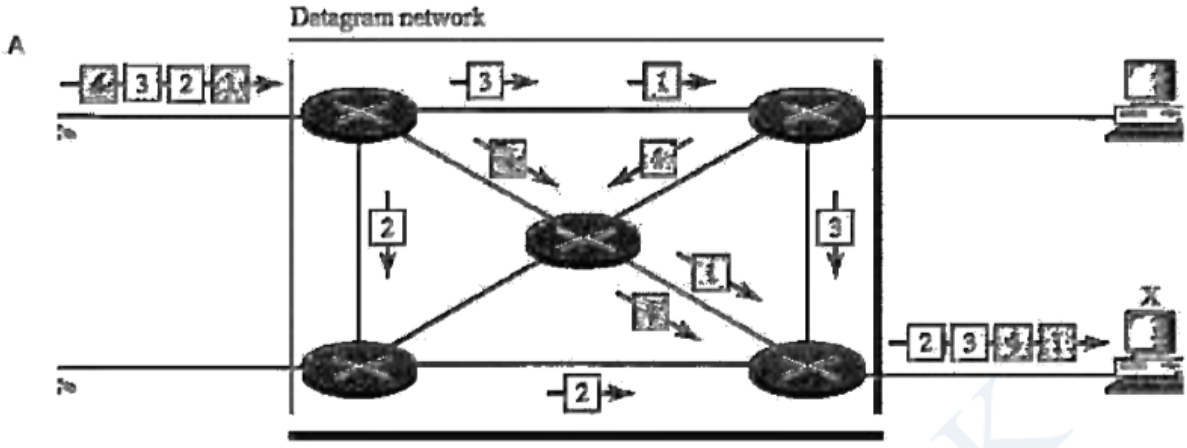
After the establishment of the dedicated circuit (channels), the two parties can transfer data.

Teardown Phase

When one of the parties needs to disconnect, a signal is sent to each switch to release the resources.

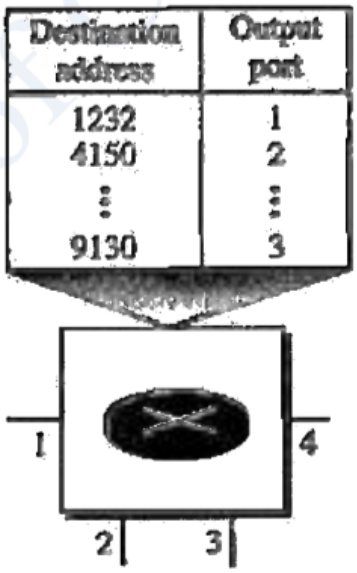
Circuit-Switched Technology in Telephone Networks

The telephone companies have previously chosen the circuit switched approach to switching in the physical layer; today the tendency is moving toward other switching techniques. For example, the telephone number is used as the global address, and a signaling system (called SS7) is used for the setup and teardown phases.



Routing Table

If there are no setup or teardown phases, how are the packets routed to their destinations in a datagram network? In this type of network, each switch (or packet switch) has a routing table which is based on the destination address. The routing tables are dynamic and are updated periodically. The destination addresses and the corresponding forwarding output ports are recorded in the tables. This is different from the table of a circuit switched network in which each entry is created when the setup phase is completed and deleted when the teardown phase is over.



[Routing Table]

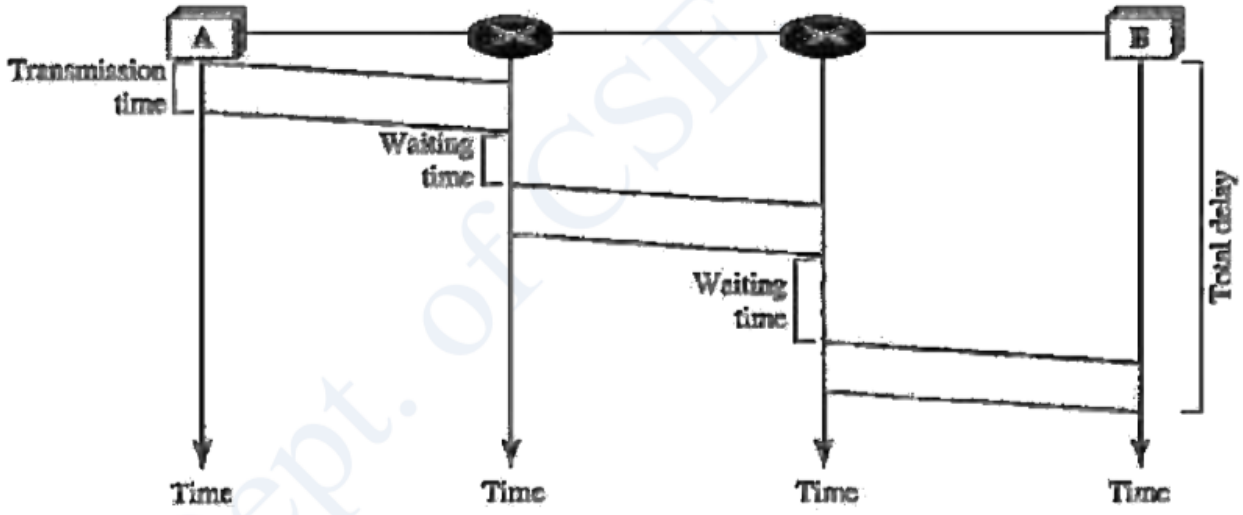
Every packet in a datagram network carries a header that contains, among other information, the destination address of the packet. When the switch receives the packet, this destination address is examined; the routing table is consulted to find the corresponding port through which the packet should be forwarded. This address, unlike the address in a virtual-circuit-switched network, remains the same during the entire journey of the packet.

Efficiency

The efficiency of a datagram network is better than that of a circuit-switched network; resources are allocated only when there are packets to be transferred. If a source sends a packet and there is a delay of a few minutes before another packet can be sent, the resources can be reallocated during these minutes for other packets from other sources.

Delay

There may be greater delay in a datagram network than in a virtual-circuit network. Although there are no setup and teardown phases, each packet may experience a wait at a switch before it is forwarded.



Packet Switching

Packet switching is a digital networking communications method that groups all transmitted data – regardless of content, type, or structure – into suitably sized blocks, called packets. Packet switching features delivery of variable-bit-rate data streams (sequences of packets) over a shared network. When

traversing network adapters, switches, routers and other network nodes, packets are buffered and queued, resulting in variable delay and throughput depending on the traffic load in the network.

Packet switching contrasts with another principal networking paradigm, circuit switching, a method which sets up a limited number of dedicated connections of constant bit rate and constant delay between nodes for exclusive use during the communication session. In case of traffic fees (as opposed to flat rate), for example in cellular communication services, circuit switching is characterized by a fee per time unit of connection time, even when no data is transferred, while packet switching is characterized by a fee per unit of information.

Two major packet switching modes exist:

(1) Connectionless packet switching, also known as datagram switching, and

(2) Connection-oriented packet switching, also known as virtual circuit switching.

In the first case each packet includes complete addressing or routing information. The packets are routed individually, sometimes resulting in different paths and out-of-order delivery.

In the second case a connection is defined and preallocated in each involved node during a connection phase before any packet is transferred.

ERROR DETECTION AND CORRECTION

Data can be corrupted during transmission. Some applications require that errors be detected and corrected.

Types of Errors

Whenever bits flow from one point to another, they are subject to unpredictable changes because of interference. This interference can change the shape of the signal. Errors are of two types:

Single-Bit Error

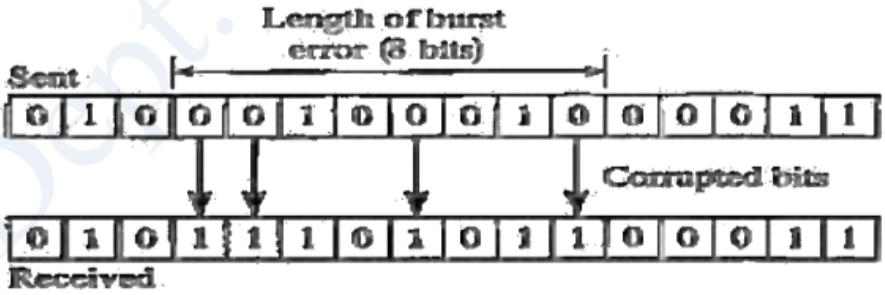
The term single-bit error means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1.

For a single-bit error to occur, the noise must have a duration of only 1) ls, which is very rare; noise normally lasts much longer than this.

Burst Error

The term burst error means that 2 or more bits in the data unit have changed from 1 to 0 or from 0 to 1.

A burst error is more likely to occur than a single-bit error. The duration of noise is normally longer than the duration of 1 bit, which means that when noise affects data, it affects a set of bits. The number of bits affected depends on the data rate and duration of noise.



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 looking only to see if any error has occurred. The answer is a simple yes or no.

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.

Forward Error Correction versus Retransmission

There are two main methods of error correction.

Forward error correction is the process in which the receiver tries to guess the message by using redundant bits. This is possible, as we see later, if the number of errors is small.

Correction by **retransmission** is a 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 (usually, not all errors can be detected).

Coding

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.

In modular arithmetic, we use only a limited range of integers. We define an upper limit, called a modulus N. We then use only the integers 0 to N -1, inclusive. This is modulo-N arithmetic.

For example, if the modulus is 12, we use only the integers 0 to 11, inclusive.

Of particular interest is modulo-2 arithmetic. In this arithmetic, the modulus N is 2. We can use only 0 and 1.

Addition	**Subtraction**
0+0=0	0-0=0
1+0=1	1-0=1
1+1=0	1-1=0

0+1=1 0-1=1

Particularly that addition and subtraction give the same results. In this arithmetic we use the XOR (exclusive OR) operation for both addition and subtraction. The result of an XOR operation is 0 if two bits are the same; the result is 1 if two bits are different.

If the modulus is not 2, addition and subtraction are distinct.

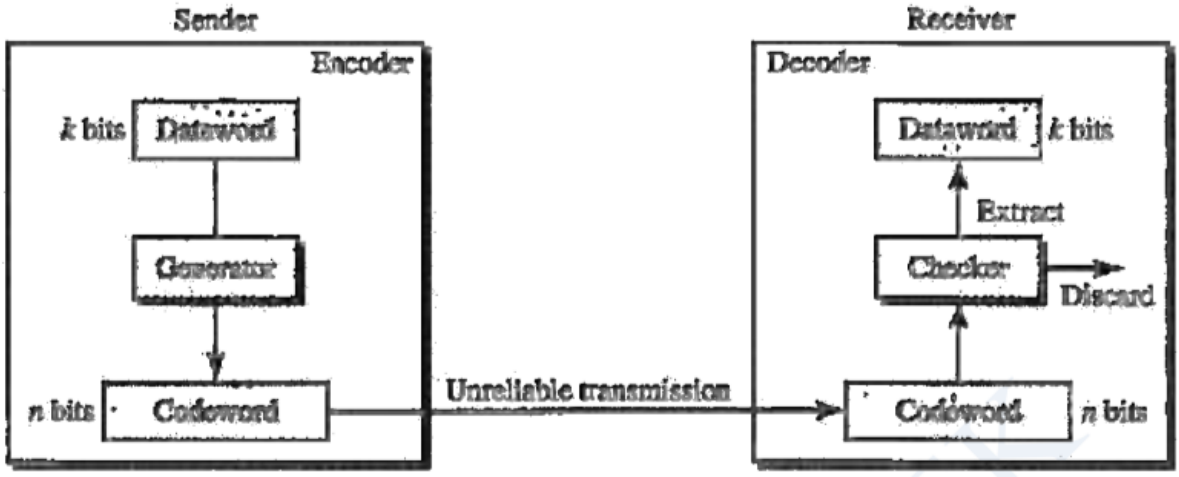
BLOCK CODING

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.

With k bits, we can create a combination of 2k data words; with n bits, we can create a combination of 2n code words. Since n > k, the number of possible code words is larger than the number of possible data words. The block coding process is one-to-one; the same data word is always encoded as the same codeword. This means that we have 2n - 2k code words that are not used.

Error Detection

The sender creates code words out of data words by using a generator that applies the rules and procedures of encoding. Each codeword sent to the receiver may change during transmission. If the received codeword is the same as one of the valid code words, the word is accepted; the corresponding data word is extracted for use. If the received codeword is not valid, it is discarded. However, if the codeword is corrupted during transmission but the received word still matches a valid codeword, the error remains undetected. This type of coding can detect only single errors. Two or more errors may remain undetected.



[Diagram: Structure of encoder and decoder for Error detection]

Example

Let us assume that k =2 and n =3. Table 1 shows the list of data words and code words. Later, we will see how to derive a codeword from a data word.

TABLE 1

Data words	Code words
00	000
01	011
10	101
11	110

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

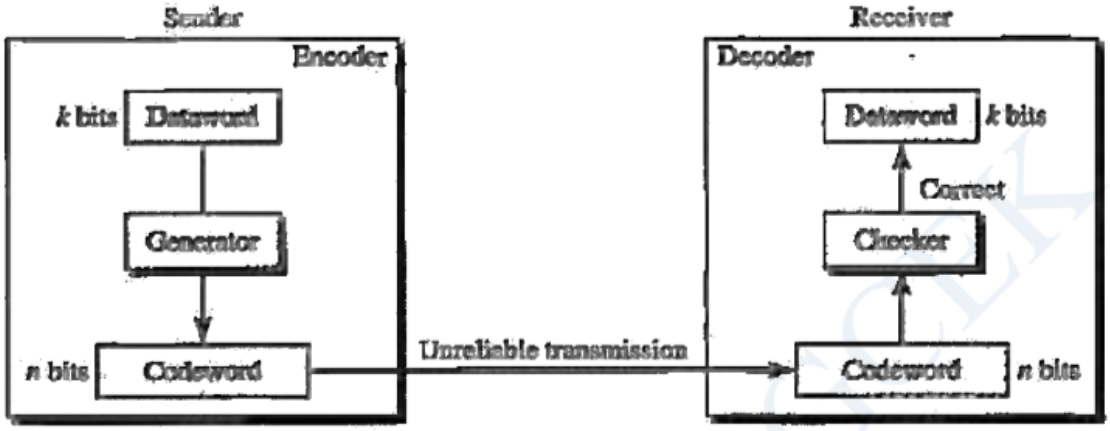
1. The receiver receives 011. It is a valid codeword. The receiver extracts the data word 01 from it.

2. The codeword is corrupted during transmission, and 111 is received (the leftmost bit is corrupted). This is not a valid codeword and is discarded.

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

Error Correction

As we said before, error correction is much more difficult than error detection. In error detection, the receiver needs to know only that the received codeword is invalid; in error correction the receiver needs to find (or guess) the original codeword sent.



[Diagram: Structure of encoder and decoder for Error correction]

Example

Let us assume that k =2 and n =3. Below table shows the list of data words and code words.

Dataword(k)	Codeword(n=k+r)
00	00000
01	01011
10	10101
11	11110

Assume the dataword is 01. The sender consults the table (or uses an algorithm) to create the codeword 01011. The codeword is corrupted during transmission, and 01001 is received (error in the second bit from the right). First, the receiver finds that the received codeword is not in the table. This means an error has occurred. (Detection must come before correction.) The receiver, assuming that there is only 1 bit corrupted, uses the following strategy to guess the correct dataword.

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.

Hamming Distance

The Hamming distance between two words (of the same size) is the number of differences between the corresponding bits.

The Hamming distance can easily be found if we apply the *XOR operation on the two words and count the number of 1s in the result*. Note that the Hamming distance is a value greater than zero.

The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.

Example:

Let us find the Hamming distance between two pairs of words.

1. The Hamming distance d (000, 011) is 2 because 000 **XOR** 011 is 011 (two 1s).
2. The Hamming distance d (10101, 11110) is 3 because 10101 **XOR** 11110 is 01011 (three 1s).

Linear Block Codes

In a linear block code, the exclusive OR (XOR) of any two valid code words creates another valid codeword.

The scheme in Table 1 is a linear block code because the result of XORing any codeword with any other codeword is a valid codeword. For example, the XORing of the second and third code words creates the fourth one.

Let us now show some linear block codes. These codes are trivial because we can easily find the encoding and decoding algorithms and check their performances.

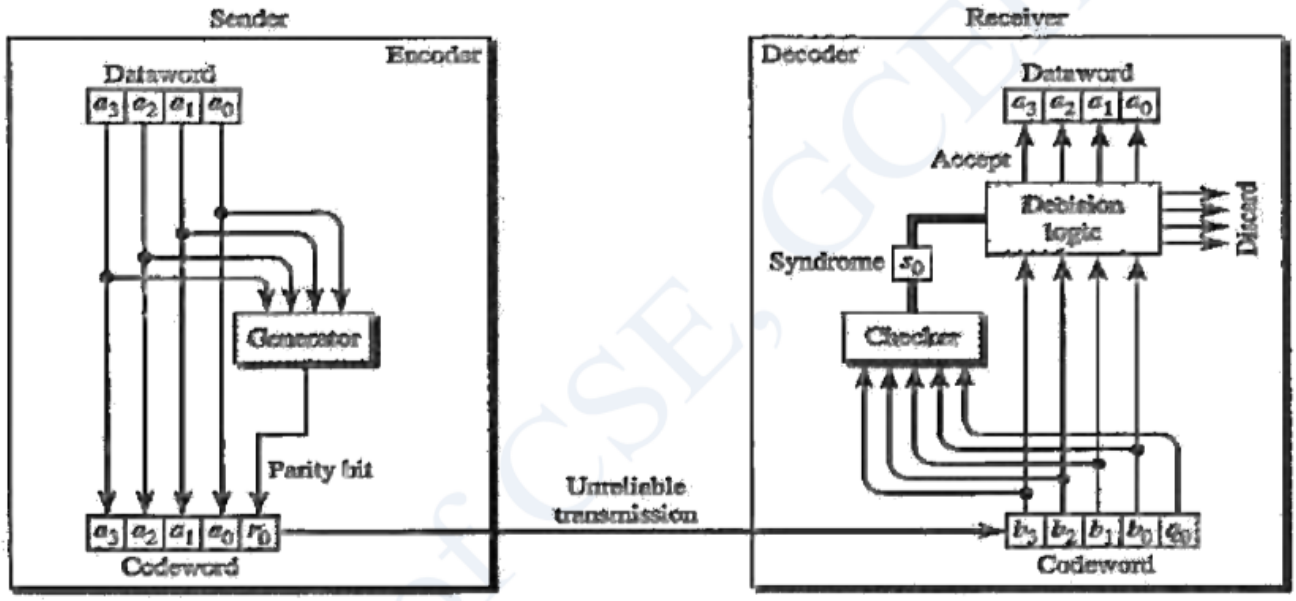
Simple Parity-Check Code

A simple parity-check code is a single-bit error-detecting code in which n =k + 1 with d_{min} =2.

In this code, a k-bit data word is changed to an n-bit codeword where n = k + 1. The extra bit, called the parity bit, is selected to make the total number of 1s in the codeword even.

The minimum Hamming distance for this category is d_{min} =2, which means that the code is a single-bit error-detecting code; it cannot correct any error.

Table 1 is a parity-check code with k = 2 and n =3.



[Encoder and Decoder for Simple parity check]

The encoder uses a generator that takes a copy of a 4-bit dataword (a0, a1, a2 and a3) and generates a parity bit ro. The dataword bits and the parity bit create the 5-bit codeword. *The parity bit that is added makes the number of 1s in the codeword even.*

This is normally done by adding the 4 bits of the dataword (modulo-2); the result is the parity bit. In other words,

$$r_0=a_3+a_2+a_1+a_0 \text{ (modulo 2)}$$

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.

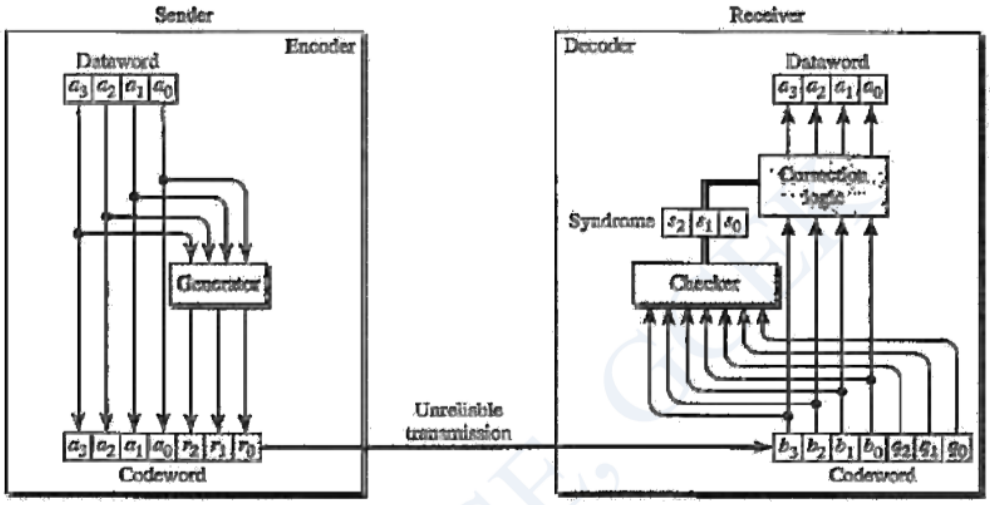
The sender sends the codeword which may be corrupted during transmission. The receiver receives a 5-bit word. The checker at the receiver does the same thing as the generator in the

These codes were originally designed with $d_{min} = 3$, which means that they can detect up to two errors or correct one single error. Although there are some Hamming codes that can correct more than one error, our discussion focuses on the single-bit error-correcting code.

Let us find the relationship between n and k in a Hamming code. We need to choose an integer m >= 3.

The values of n and k are then calculated from m as $n = 2^m - 1$and k=n - m. The number of check bits r =m.

A Hamming code can only correct a single error or detect a double error.



[Encoder and Decoder for Hamming Code]

A copy of a 4-bit dataword is fed into the generator that creates three parity checks r_0, r_1 and r_2 as shown below:

$r_0=a_2+a_1+a_o$ modulo-2

$r_1 =a_3 + a_2 + a_1$ modulo-2

$r_2=a_1+a_0+a_3$ modulo-2

In other words, each of the parity-check bits handles 3 out of the 4 bits of the dataword. The total number of 1s in each 4-bit combination (3 dataword bits and 1 parity bit) must be even.

The checker in the **decoder** creates a 3-bit syndrome (s2s1s0) in which each bit is the parity check for 4 out of the 7 bits in the received codeword:

$s_0=b_2+b_1+b_o+q_0$ modulo-2

$s_1 =b_3 + b_2 + b_1 +q_1$ modulo-2

$s_2=b_1+b_0+b_3+q_2$ modulo-2

The equations used by the checker are the same as those used by the generator with the parity-check bits added to the right-hand side of the equation. The 3-bit syndrome creates eight different bit patterns

(000 to 111) that can represent eight different conditions. These conditions define a lack of error or an error in 1 of the 7 bits of the received codeword, as shown in Table:

Syndrome	000	001	010	011	100	101	110	111
Error	None	q_0	q_1	b2	q2	b_0	b_3	b_1

Let us trace the path of three datawords from the sender to the destination:

1. The dataword 0100 becomes the codeword 0100011. The codeword 01 00011 is received. The syndrome is 000 (no error), the final dataword is 0100.

2. The dataword 0111 becomes the codeword 0111001. The codeword 0011001 is received. The syndrome is 011. According to Table, b2 is in error. After flipping b2 (changing the 1 to 0), the final dataword is 0111.

3. The dataword 1101 becomes the codeword 1101000. The codeword 0001000 is received (two errors). The syndrome is 101, which means that b_0 is in error. After flipping b_0, we get 0000, the wrong dataword. This shows that our code cannot correct two errors.

Cyclic Codes

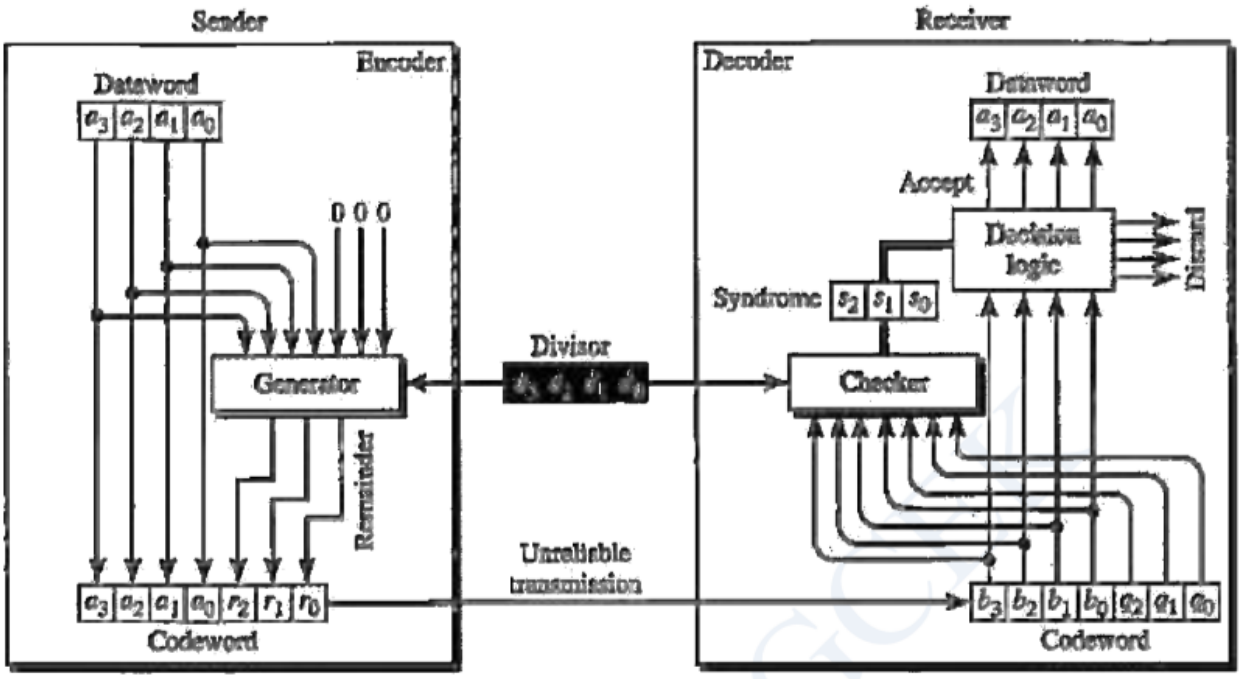
Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword. For example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword.

If we call the bits in the first word a_0 to a_6 and the bits in the second word b_0 to b_6, we can shift the bits by using the following:

$$b_1=a_0 \quad b_2=a_1 \quad b_3=a_2 \quad b_4=a_3 \quad b_5=a_4 \quad b_6=a_5 \quad b_0=a_6$$

Cyclic Redundancy Check

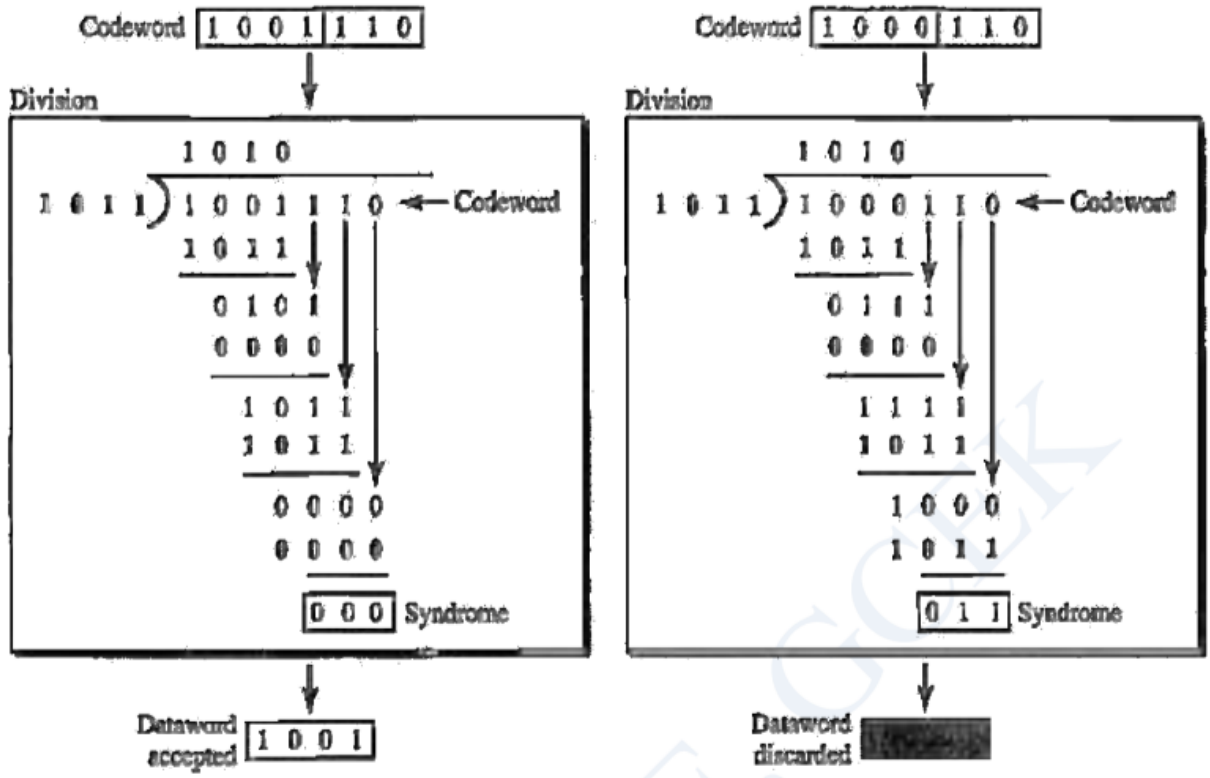
We can create cyclic codes to correct errors. However, the theoretical background required is beyond the scope of this book. In this section, we simply discuss a category of cyclic codes called the cyclic redundancy check (CRC) that is used in networks such as LANs and WANs.



[Encoder and Decoder for CRC]

In the encoder, the dataword has k bits (4 here); the codeword has n bits (7 here). The size of the dataword is augmented by adding n - k (3 here) 0s to the right-hand side of the word. The n-bit result is fed into the generator. The generator uses a divisor of size n - k + 1 (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder ($r_2r_1r_0$) is appended to the dataword to create the codeword.

The decoder receives the possibly corrupted codeword. A copy of all n bits is fed to the checker which is a replica 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 as, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).



Polynomial

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. Below figure shows a binary pattern and its polynomial representation.