Module-3

- **Bandwidth Utilization**

  Multiplexing and Spread Spectrum

- **Switching**

  Introduction

  Circuit Switched Networks

  Packet switching.

- **Error Detection and Correction**

  Introduction

  Block coding

  Cyclic codes

  Checksum

MULTIPLEXING AND SPREADING

MULTIPLEXING

- When bandwidth of a medium is greater than bandwidth needs of the devices, the link can be shared.
- *Multiplexing* allows simultaneous transmission of multiple signals across a single data-link (Fig 4.21).
- The traffic increases, as data/telecommunications use increases.
- We can accommodate this increase by
  - → adding individual links, each time a new channel is needed or
  - → installing higher-bandwidth links to carry multiple signals.
- Today's technology includes high-bandwidth media such as optical-fiber and satellite microwaves.
- Each has a bandwidth far in excess of that needed for the average transmission-signal.
- If the bandwidth of a link is greater than the bandwidth needs of the devices connected to it, the bandwidth is wasted.
- An efficient system maximizes the utilization of all resources; bandwidth is one of the most precious resources we have in data communications.

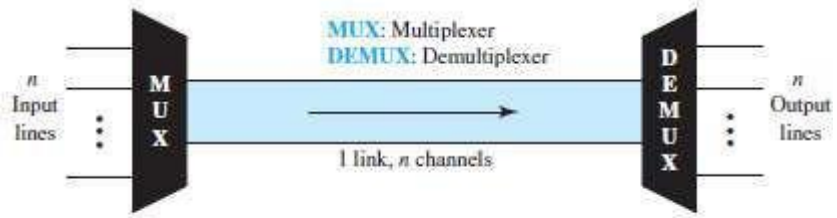


Figure 6.1 *Dividing a link into channels*

- In a multiplexed-system, „n' lines share the bandwidth of one link.
- MUX combines transmission-streams from different input-lines into a single stream (many-to-one).
- At the receiving-end, that stream is fed into a demultiplexer (DEMUX).
- DEMUX
  - → separates the stream back into its component-transmissions (one-to-many) and
  - → directs the transmission-streams to different output-lines.
- Link vs. Channel:
  1) The link refers to the physical path.
  2) The channel refers to the portion of a link that carries a transmission between a given pair of lines. One link can

have many channels.

- Three multiplexing techniques (Figure 6.2):
  1) Frequency-division multiplexing (FDM)
  2) Wavelength-division multiplexing (WDM) and
  3) Time-division multiplexing (TDM).
- The first two techniques are used for analog-signals. The third one technique is used for digital-signals.

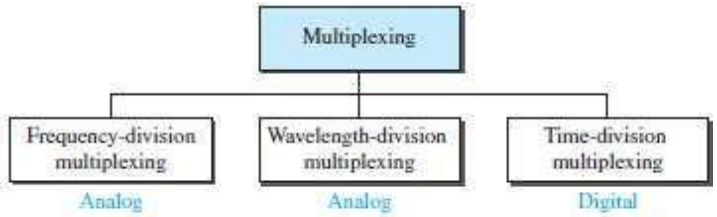


Figure 6.2 *Categories of multiplexing*

Frequency Division Multiplexing (FDM)

- FDM is an analog multiplexing technique that combines analog signals (Figure 6.3).
- FDM can be used when the bandwidth of a link is greater than the combined bandwidths of the signals to be transmitted. (Bandwidth measured in hertz).

Figure 6.3 *Frequency-division multiplexing*

Multiplexing Process

- Here is how it works (Figure 6.4):
  1) Each sending-device generates modulated-signals with different carrier-frequencies (f1, f2, & f3).
  2) Then, these modulated-signals are combined into a single multiplexed-signal.
  3) Finally, the multiplexed-signal is transported by the link.

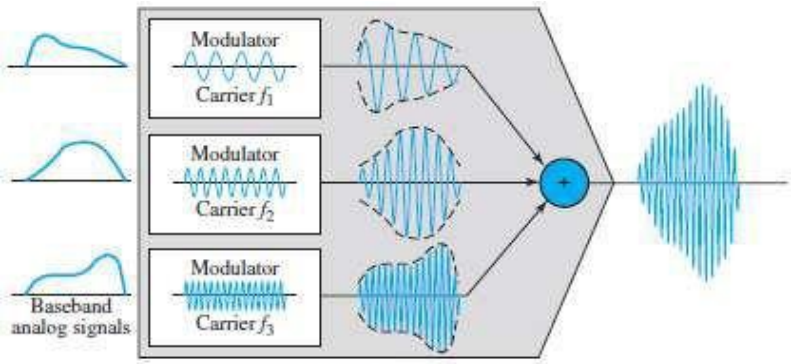


Figure 6.4 *FDM process*

- Carrier-frequencies are separated by sufficient bandwidth to accommodate the modulated-signal.
- Channels can be separated by strips of unused bandwidth called guard bands.
- Guard bands prevent signals from overlapping.
- In addition, carrier-frequencies must not interfere with the original data frequencies.
- Although FDM is considered as analog multiplexing technique, the sources can produce digital-signal.
- The digital-signal can be sampled, changed to analog-signal, and then multiplexed by using FDM.

Demultiplexing Process

- Here is how it works (Figure 6.5):
  4) The demultiplexer uses filters to divide the multiplexed-signal into individual-signals.
  5) Then, the individual signals are passed to a demodulator.
  6) Finally, the demodulator
     → separates the individual signals from the carrier signals and
     → passes the individual signals to the output-lines.

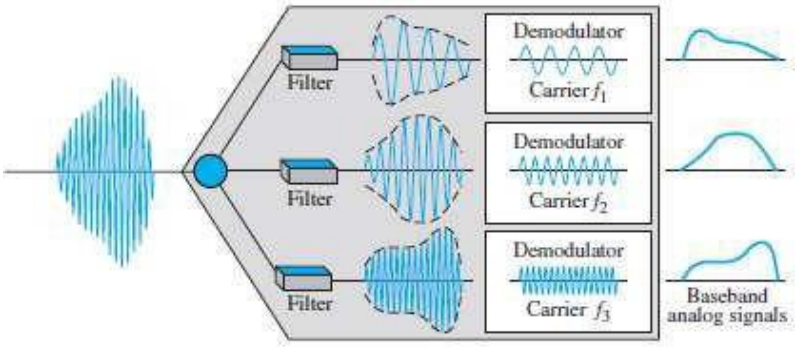


Figure 6.5 *FDM demultiplexing example*

Applications of FDM

- To maximize the efficiency of their infrastructure, Telephone-companies have traditionally multiplexed signals from lower-bandwidth lines onto higher-bandwidth lines.
- A very common application of FDM is AM and FM radio broadcasting.
- The first generation of cellular telephones (still in operation) also uses FDM.

Analog Carrier System

- To maximize the efficiency, telephone-companies have multiplexed-signals from lower-bandwidth lines onto higher-bandwidth lines.
- Many switched or leased lines are combined into bigger channels.
- For analog lines, FDM is used.
- One of these hierarchical systems used by AT&T is made up of (Figure 6.9):

  Groups
  Super groups
  Master groups, and
  Jumbo groups

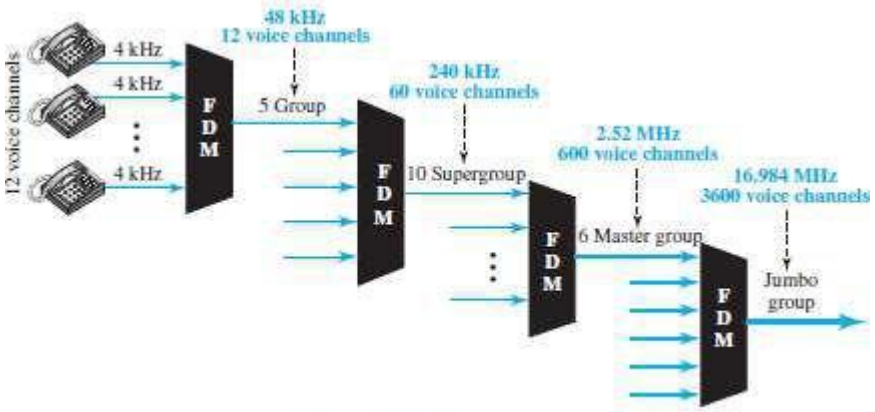


Figure 6.9 *Analog hierarchy*

1) Group: In the analog hierarchy, 12 voice channels are multiplexed onto a higher-bandwidth line to create a group.
- A group has 48 kHz of bandwidth and supports 12 voice channels.

2) Super Group: At the next level, up to five groups can be multiplexed to create a composite- signal called a supergroup.
- A supergroup has a bandwidth of 240 kHz and supports up to 60 voice channels.
- Supergroups can be made up of either five groups or 60 independent voice channels.

3) Master Groups: At the next level, 10 supergroups are multiplexed to create a master group.
- A master group must have 2.40 MHz of bandwidth, but the need for guard bands between the supergroups increases the necessary bandwidth to 2.52 MHz.
- Master groups support up to 600 voice channels.

4) Jumbo Group: Finally, six master groups can be combined into a jumbo group.
- A jumbo group must have 15.12 MHz (6 x 2.52 MHz) of bandwidth, but the need for guard bands b/w the master groups increases the necessary bandwidth to 16.984 MHz

Example 2.15

The Advanced Mobile Phone System (AMPS) uses two bands. The first band of 824 to 849 MHz is used for sending, and 869 to 894 MHz is used for receiving. Each user has a bandwidth of 30 kHz in each direction. The 3-kHz voice is modulated using FM, creating 30 kHz of modulated signal. How many people can use their cellular phones simultaneously?

Solution

Each band is 25 MHz. If we divide 25 MHz by 30 kHz, we get 833.33. In reality, the band is divided into 832 channels. Of these, 42 channels are used for control, which means only 790 channels are available for cellular phone users.

Wavelength Division Multiplexing (WDM)

- WDM is an analog multiplexing technique that combines analog signals (Figure 6.10).
- WDM is designed to use the high-data-rate capability of fiber optical-cable.
- The data-rate of optical-cable is higher than the data-rate of metallic-cable.
- Using an optical-cable for one single line wastes the available bandwidth.
- Multiplexing allows combining several lines into one line.
- WDM is same as FDM with 2 exceptions:
  1) Multiplexing & demultiplexing involve optical-signals transmitted through optical-cable.
  2) The frequencies are very high.

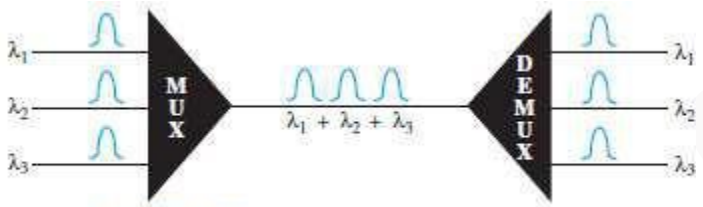


Figure 6.10 *Wavelength-division multiplexing*

- Here is how it works (Figure 6.11):
  - A multiplexer combines several narrow-bands of light into a wider-band of light.
  - A demultiplexer divides a wider-band of light into several narrow-bands of light.
  - A prism is used for combining and splitting of light sources
  - A prism bends a beam of light based on
    → angle of incidence and
    → frequency.

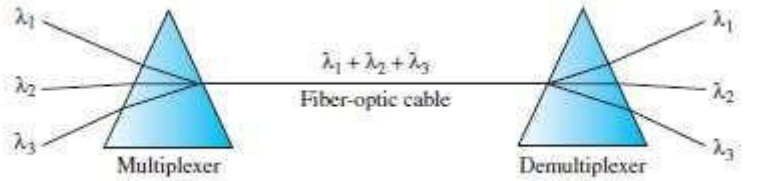


Figure 6.11 *Prisms in wavelength-division multiplexing and demultiplexing*

- Applications of WDM:
  1) SONET network: Multiple optical-fiber lines can be multiplexed and demultiplexed.
  2) Dense WDM (DWDM) can multiplex a very large number of channels by spacing channels very close to one another. DWDM achieves even greater efficiency

Time Division Multiplexing (TDM)

- TDM is a digital multiplexing technique that combines digital signals (Figure 6.12).
- TDM combines several low-rate channels into one high-rate one.
- FDM vs. TDM
    1) In FDM, a portion of the bandwidth is shared.
    2) In TDM, a portion of the time is shared.
- Each connection occupies a portion of time in the link.
- Several connections share the high bandwidth of a line.

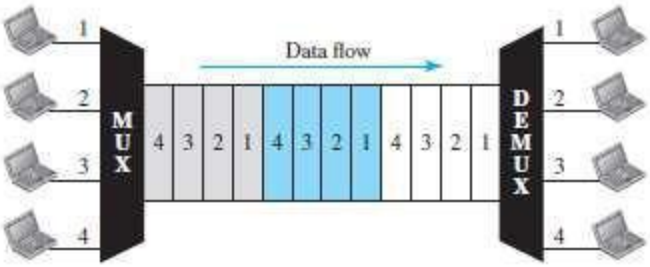


Figure 6.12 *TDM*

- As shown in Figure 6.12, the link is divided by time.
- Portions of signals 1, 2, 3, and 4 occupy the link sequentially.
- Digital-data from different sources are combined into one timeshared link.
- Although TDM is considered as digital multiplexing technique, the sources can produce analog-signal.
- The analog data can be sampled, changed to digital-data, and then multiplexed by using TDM.
- Two types of TDM:
    1) Synchronous and
    2) Statistical.

Synchronous TDM

Time Slots & Frames

- Each input-connection has an allotment in the output-connection even if it is not sending data.
- The data-flow of input-connection is divided into units (Figure 6.13).
- A unit can be 1 bit, 1 character, or 1 block of data.
- Each input-unit occupies one input-time-slot.
- Each input-unit
  → becomes one output-unit and
  → occupies one output-time-slot.
- However, duration of output-time-slot is „n' times shorter than duration of input-time-slot.
- If an input-time-slot is T s, the output-time-slot is T/n s
  where n = No. of connections.
- In the output-connection, a unit has a shorter duration & therefore travels faster.

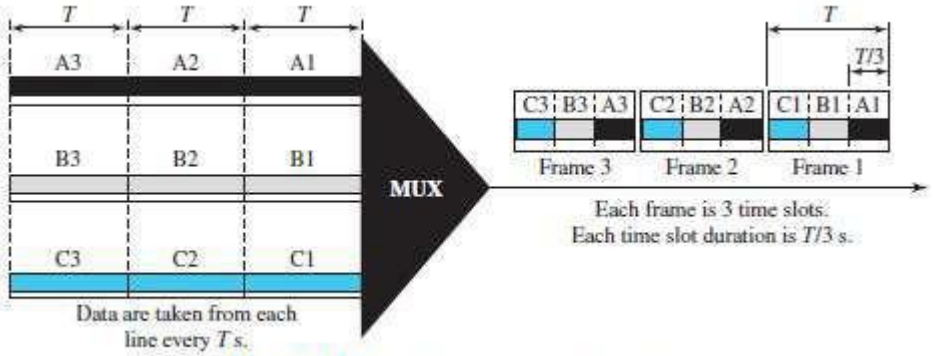


Figure 6.13 *Synchronous time-division multiplexing*

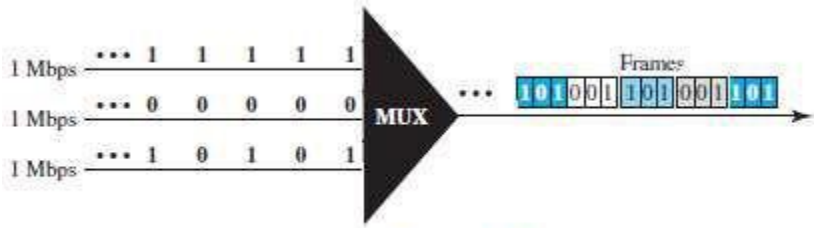


Figure 6.14

- In Figure 6.14, n = 3.
- A set of data-units from each input-connection is grouped into a frame.
- For example:
  If there are 3 connections, a frame is divided into 3 time-slots.
  One slot is allocated for each data-unit.
  One data-unit is used for each input-line.

In Figure 6.13, the data rate for each input connection is 1 kbps. If 1 bit at a time is multiplexed (a unit is 1 bit), what is the duration of

1. each input slot,
2. each output slot, and
3. each frame?

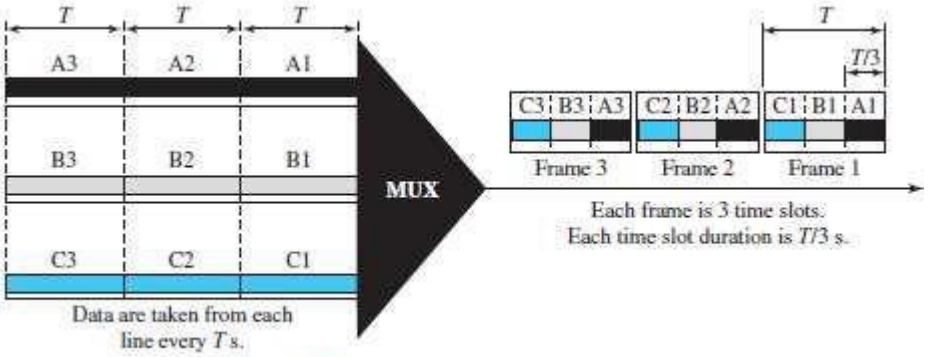


Figure 6.13 *Synchronous time-division multiplexing*

Solution

We can answer the questions as follows:

1. The data rate of each input connection is 1 kbps. This means that the bit duration is 1/1000 s or 1 ms. The duration of the input time slot is 1 ms (same as bit duration).
2. The duration of each output time slot is one-third of the input time slot. This means that the duration of the output time slot is 1/3 ms.
3. Each frame carries three output time slots. So the duration of a frame is 3 × 1/3 ms, or 1 ms. The duration of a frame is the same as the duration of an input unit.

Example 2.17

Figure 6.14 shows synchronous TDM with a data stream for each input and one data stream for the output. The unit of data is 1 bit. Find (1) the input bit duration, (2) the output bit duration, (3) the output bit rate, and (4) the output frame rate.

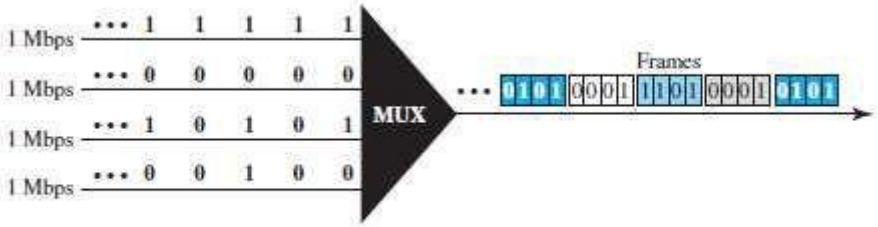


Figure 6.14

Solution

We can answer the questions as follows:

1. The input bit duration is the inverse of the bit rate: 1/1 Mbps = 1 μs.
2. The output bit duration is one-fourth of the input bit duration, or 1/4 μs.
3. The output bit rate is the inverse of the output bit duration, or 1/4 μs, or 4 Mbps. This can also be deduced from the fact that the output rate is 4 times as fast as any input rate; so the output rate = 4 × 1 Mbps = 4 Mbps.
4. The frame rate is always the same as any input rate. So the frame rate is 1,000,000 frames per second. Because we are sending 4 bits in each frame, we can verify the result of the previous question by multiplying the frame rate by the number of bits per frame.

Example 2.18

Four 1-kbps connections are multiplexed together. A unit is 1 bit. Find (1) the duration of 1 bit before multiplexing, (2) the transmission rate of the link, (3) the duration of a time slot, and (4) the duration of a frame.

Solution

We can answer the questions as follows:

1. The duration of 1 bit before multiplexing is 1/1 kbps, or 0.001 s (1 ms).
2. The rate of the link is 4 times the rate of a connection, or 4 kbps.
3. The duration of each time slot is one-fourth of the duration of each bit before multiplexing, or 1/4 ms or 250 μs. Note that we can also calculate this from the data rate of the link, 4 kbps. The bit duration is the inverse of the data rate, or 1/4 kbps or 250 μs.
4. The duration of a frame is always the same as the duration of a unit before multiplexing, or 1 ms. We can also calculate this in another way. Each frame in this case has four time slots. So the duration of a frame is 4 times 250 μs, or 1 ms.

INTERLEAVING

- TDM can be seen as 2 fast-rotating switches (Figure 6.15):
  1) First switch on the multiplexing-side and
  2) Second switch on the demultiplexing-side.
- The switches are synchronized and rotate at the same speed, but in opposite directions.
  1) On the multiplexing-side (Figure 6.16)
  - As the switch opens in front of a connection, that connection has the opportunity to send a unit onto the path. This process is called *interleaving*.
  2) On the demultiplexing-side
  - As the switch opens in front of a connection, that connection has the opportunity to receive a unit from the path.

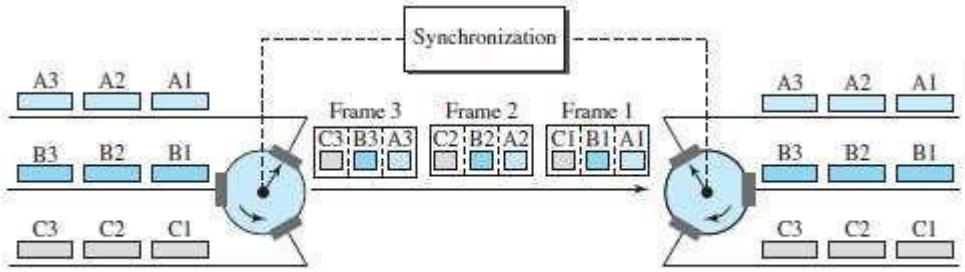


Figure 6.15 *Interleaving*

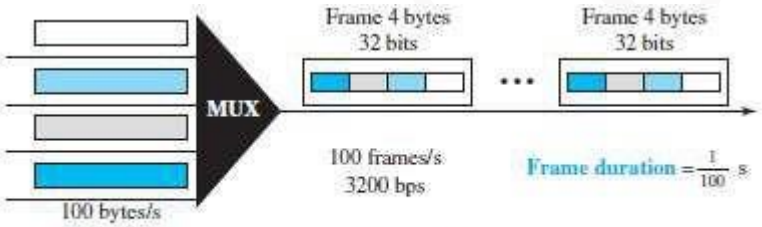


Figure 6.16

Example 2.19

Four channels are multiplexed using TDM. If each channel sends 100 bytes/s and we multiplex 1 byte per channel, show the frame traveling on the link, the size of the frame, the duration of a frame, the frame rate, and the bit rate for the link.

Solution

The multiplexer is shown in Figure 6.16. Each frame carries 1 byte from each channel; the size of each frame, therefore, is 4 bytes, or 32 bits. Because each channel is sending 100 bytes/s and a frame carries 1 byte from each channel, the frame rate must be 100 frames per second. The duration of a frame is therefore 1/100 s. The link is carrying 100 frames per second, and since each frame contains 32 bits, the bit rate is 100×32, or 3200 bps. This is actually 4 times the bit rate of each channel, which is $100 \times 8 = 800$ bps.

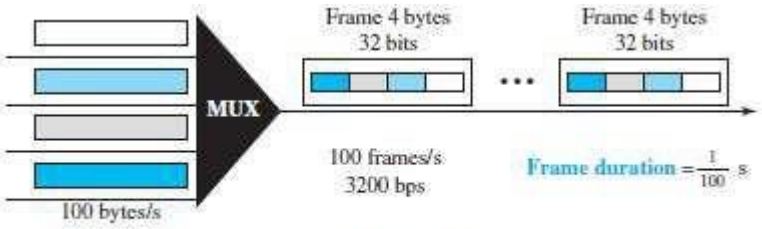


Figure 6.16

Example 2.20

A multiplexer combines four 100-kbps channels using a time slot of 2 bits. Show the output with four arbitrary inputs. What is the frame rate? What is the frame duration? What is the bit rate? What is the bit duration?

Solution

Figure 6.17 shows the output for four arbitrary inputs. The link carries 50,000 frames per second since each frame contains 2 bits per channel. The frame duration is therefore 1/50,000 s or 20 µs. The frame rate is 50,000 frames per second, and each frame carries 8 bits; the bit rate is 50,000 × 8 = 400,000 bits or 400 kbps. The bit duration is 1/400,000 s, or 2.5 µs. Note that the frame duration is 8 times the bit duration because each frame is carrying 8 bits.

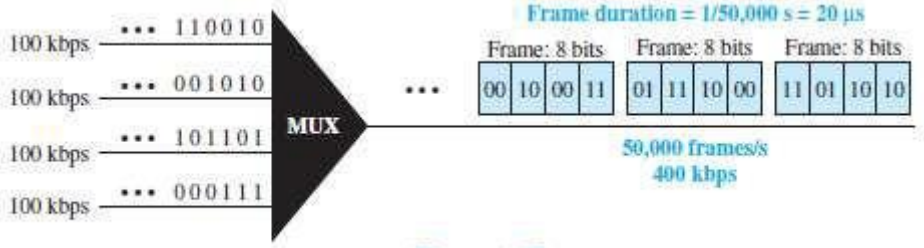


Figure 6.17

Empty Slots

- Problem: Synchronous TDM is not efficient.

  For example: If a source does not have data to send, the corresponding slot in the output-frame is empty.

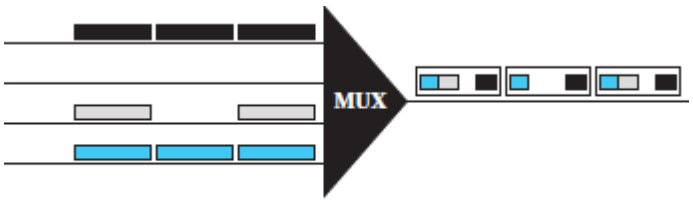


Figure 6.18 *Empty slots*

- As shown in Figure 6.18,

  Second input-line has no data to send

  Third input-line has discontinuous data.
- The first output-frame has 3 slots filled. The second frame has 2 slots filled.

  The third frame has 3 slots filled.

  No frame is full.
- Solution: Statistical TDM can improve the efficiency by removing the empty slots from the frame.

Data Rate Management

- Problem in TDM: How to handle differences in the input data-rates?
- If data-rates are not the same, three strategies can be used.
- Three different strategies: 1) Multilevel multiplexing 2) Multiple-slot allocation and 3) Pulse stuffing

1) Multilevel Multiplexing

➢ This technique is used when the data-rate of an input-line is a multiple of others.

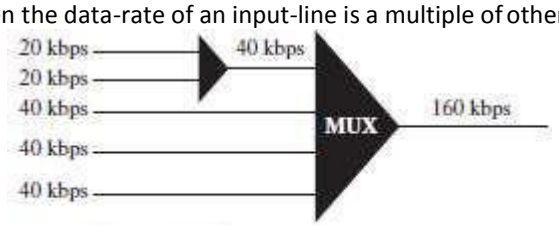


Figure 6.19 *Multilevel multiplexing*

➢ For example:
  - ➢ As shown in Figure 6.19, we have 2 inputs of 20 kbps and 3 inputs of 40 kbps.
  - ➢ The first 2 input-lines can be multiplexed to provide a data-rate of 40 kbps.

2) Multiple Slot Allocation

➢ Sometimes it is more efficient to allot more than 1 slot in a frame to a single input-line.

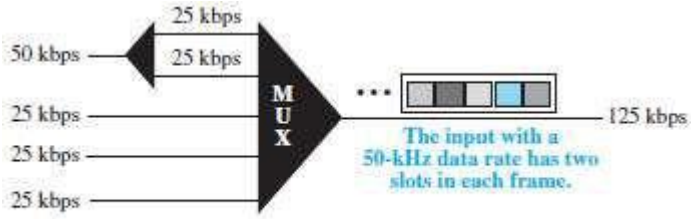


Figure 6.20 *Multiple-slot multiplexing*

➢ For example:

Data-rate of multiple input-lines can be data-rate of one input-line.

➢ As shown in Figure 6.20, the input-line with a 50-kbps data-rate can be given 2 slots in the output-line.

➢ In first input line, serial-to-parallel converter is used. The converter creates two 25 kbps input lines out of one 50 kbps input line.

3) Pulse Stuffing

➢ Sometimes the bit-rates of sources are not multiple integers of each other. ∴ above 2 techniques cannot be used.

➢ Solution:

→ Make the highest input data-rate the dominant data-rate.
→ Then, add dummy bits to the input-lines with lower rates.
→ This will increase data rates of input-line.
→ This technique is called pulse stuffing, bit padding, or bit stuffing.

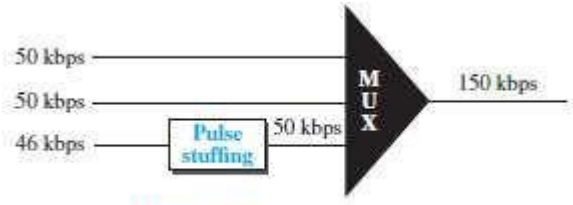


Figure 6.21 *Pulse stuffing*

➢ As shown in Figure 6.21, the input-line with a lower data-rate = 46kbps is pulse-stuffed to increase the data-rate to 50 kbps.

➢ Now, multiplexing can take place.

Frame Synchronizing

- Problem: Synchronization between the multiplexer and demultiplexer is a major issue.

If the multiplexer and the demultiplexer are not synchronized, a bit belonging to one channel may be received by the wrong channel.

Solution: Usually, one or more synchronization-bits are added to the beginning of each frame. These bits are called *framing-bits*.

The framing-bits follow a pattern (frame-to-frame) that allows multiplexer and demultiplexer to synchronize.

As shown in Figure 6.22, the synchronization-information

→ consists of 1 bit per frame and

→ alternates between 0 & 1.

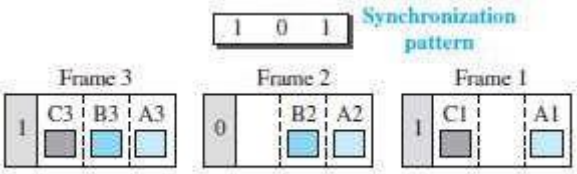


Figure 6.22 *Framing bits*

Example 2.21

We have four sources, each creating 250 characters per second. If the interleaved unit is a character and 1 synchronizing bit is added to each frame, find (1) the data rate of each source, (2) the duration of each character in each source, (3) the frame rate, (4) the duration of each frame, (5) the number of bits in each frame, and (6) the data rate of the link.

Solution

We can answer the questions as follows:

1. The data rate of each source is $250 \times 8 = 2000$ bps $= 2$ kbps.
2. Each source sends 250 characters per second; therefore, the duration of a character is 1/250 s, or 4 ms.
3. Each frame has one character from each source, which means the link needs to send 250 frames per second to keep the transmission rate of each source.
4. The duration of each frame is 1/250 s, or 4 ms. Note that the duration of each frame is the same as the duration of each character coming from each source.
5. Each frame carries 4 characters and 1 extra synchronizing bit. This means that each frame is $4 \times 8 + 1 = 33$ bits.
6. The link sends 250 frames per second, and each frame contains 33 bits. This means that the data rate of the link is 250×33, or 8250 bps. Note that the bit rate of the link is greater than the combined bit rates of the four channels. If we add the bit rates of four channels, we get 8000 bps. Because 250 frames are traveling per second and each contains 1 extra bit for synchronizing, we need to add 250 to the sum to get 8250 bps.

Example 2.22

Two channels, one with a bit rate of 100 kbps and another with a bit rate of 200 kbps, are to be multiplexed. How this can be achieved? What is the frame rate? What is the frame duration? What is the bit rate of the link?

Solution

We can allocate one slot to the first channel and two slots to the second channel. Each frame carries 3 bits. The frame rate is 100,000 frames per second because it carries 1 bit from the first channel. The frame duration is 1/100,000 s, or 10 ms. The bit rate is 100,000 frames/s $\times$ 3 bits per frame, or 300 kbps. Note that because each frame carries 1 bit from the first channel, the bit rate for the first channel is preserved. The bit rate for the second channel is also preserved because each frame carries 2 bits from the second channel.

Statistical TDM

- Problem: Synchronous TDM is not efficient.

  For ex: If a source does not have data to send, the corresponding slot in the output-frame is empty.

  Solution: Use statistical TDM.

  Slots are dynamically allocated to improve bandwidth-efficiency.

  Only when an input-line has data to send, the input-line is given a slot in the output-frame.
- The number of slots in each frame is less than the number of input-lines.
- The multiplexer checks each input-line in round robin fashion. If the line has data to send;

  Then, multiplexer allocates a slot for an input-line;

  Otherwise, multiplexer skips the line and checks the next line.

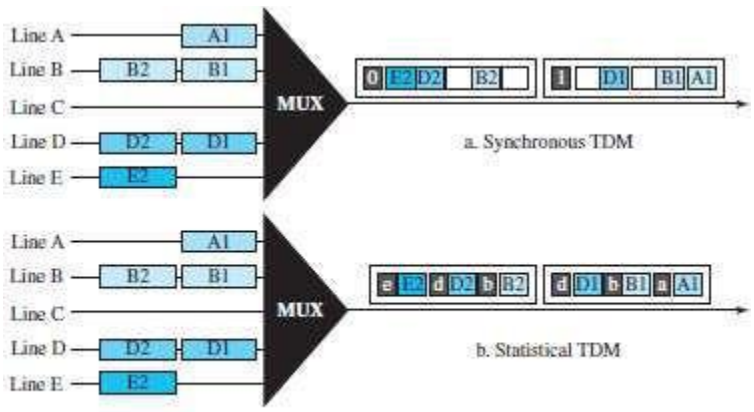


Figure 6.26 *TDM slot comparison*

- In synchronous TDM (Figure 6.26a), some slots are empty because the corresponding line does not have data to send.
- In statistical TDM (Figure 6.26b), no slot is left empty.

1) Addressing

Synchronous TDM	Statistical TDM
An output-slot needs to carry only data of the destination (Figure 6.26a).	An output-slot needs to carry both data & address of the destination (Figure 6.26b).
There is no need for addressing. Synchronization and pre-assigned relationships between the inputs and outputs serve as an address.	There is no fixed relationship between the inputs and outputs because there are no pre-assigned or reserved slots. We need to include the address of the receiver inside each slot to show where it is to be delivered.

2) Slot Size

- Usually, a block of data is many bytes while the address is just a few bytes.
- A slot carries both data and address.
- Therefore, address-size must be very small when compared to data-size. This results in efficient transmission.
- For example:

  It will be inefficient to send 1 bit per slot as data, when the address is 3 bits.

  This means an overhead of 300%.

3) No Synchronization Bit

- In statistical TDM, the frames need not be synchronized, so synchronization-bits are not needed.

4) Bandwidth

- Normally, the capacity of the link is less than the sum of the capacities of each channel.
- The designers define the capacity of the link based on the statistics of the load for each channel.

SPREAD SPECTRUM

- Spread-spectrum is used in wireless applications (Figure 6.27).
- In wireless applications, all stations use air (or a vacuum) as the medium for communication.
- Goal: Stations must be able to share the air medium without interception by an attacker.

Solution: Spread-spectrum techniques add redundancy i.e. they spread the original spectrum needed for each station.

- If the required bandwidth for each station is B, spread-spectrum expands it to B_{ss} such that $B_{ss} >> B$.
- The expanded-bandwidth allows the source to place its message in a protective envelope for a more secure transmission.

(An analogy is the sending of a delicate, expensive gift. We can insert the gift in a special box to prevent it from being damaged during transportation).

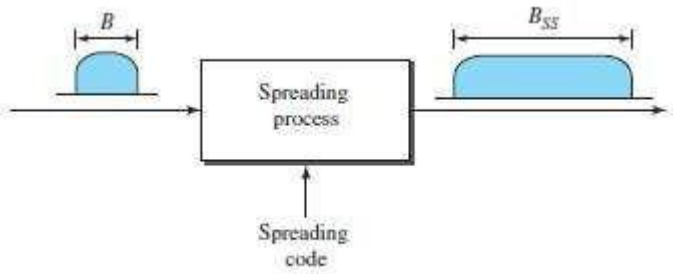


Figure 6.27 *Spread spectrum*

- Spread-spectrum achieves its goal through 2 principles:
  - **1)** The bandwidth allocated to each station needs to be, by far, larger than what is needed. This allows redundancy.
  - **2)** The spreading process must occur after the signal is created by the source.
- After the signal is created by the source, the spreading process
  - → uses a spreading-code and
  - → spreads the bandwidth.
- The spreading-code is a series of numbers that look random, but are actually a pattern.
- Two types of spread-spectrum:
  1) Frequency hopping spread-spectrum (FHSS) and
  2) Direct sequence spread-spectrum (DSSS).

Frequency Hopping Spread Spectrum (FHSS)

- This technique uses „M' different carrier-frequencies that are modulated by the source-signal.
- At one moment, the signal modulates one carrier-frequency.
  At the next moment, the signal modulates another carrier-frequency.
- Although the modulation is done using one carrier-frequency at a time, 'M' frequencies are used in the long run.
- The bandwidth occupied by a source is given by

$$B_{FHSS} >> B$$

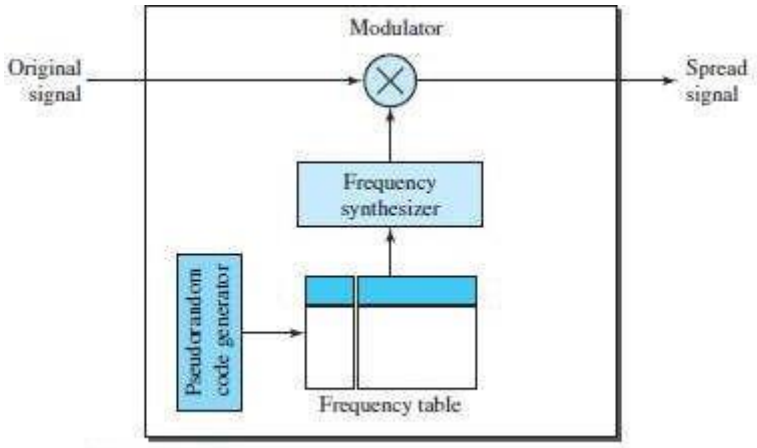


Figure 6.28 *Frequency hopping spread spectrum (FHSS)*

- As shown in Figure 6.28.
  - A pseudorandom code generator (PN) creates a k-bit pattern for every hopping period T_h.
  - The frequency-table
    → uses the pattern to find the frequency to be used for this hopping period and
    → passes the frequency to the frequency-synthesizer.
  - The frequency-synthesizer creates a carrier-signal of that frequency.
  - The source-signal modulates the carrier-signal.

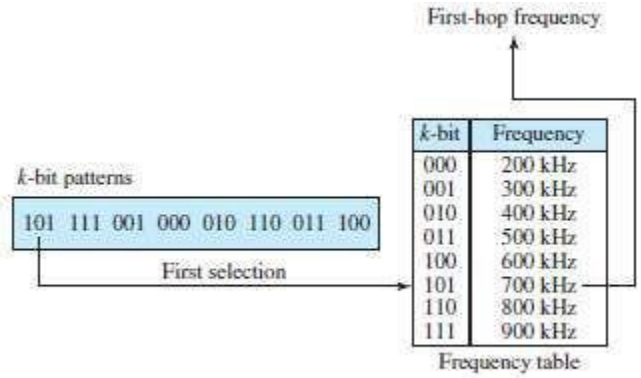


Figure 6.29 *Frequency selection in FHSS*

- As shown in Figure 6.29, assume we have 8 hopping frequencies.
  - Here, M = 8 and k = 3.
  - The pseudorandom code generator will create 8 different 3-bit patterns.
  - These are mapped to 8 different frequencies in the frequency table (see Figure 6.29).
  - The pattern for this station is 101, 111, 001, 000, 010, 111 & 100.
    1) At hopping-period 1, the pattern is 101.
       The frequency selected is 700 kHz; the source-signal modulates this carrier-frequency.
    2) At hopping-period 2, the pattern is 111.
       The frequency selected is 900 kHz; the source-signal modulates this carrier-frequency.

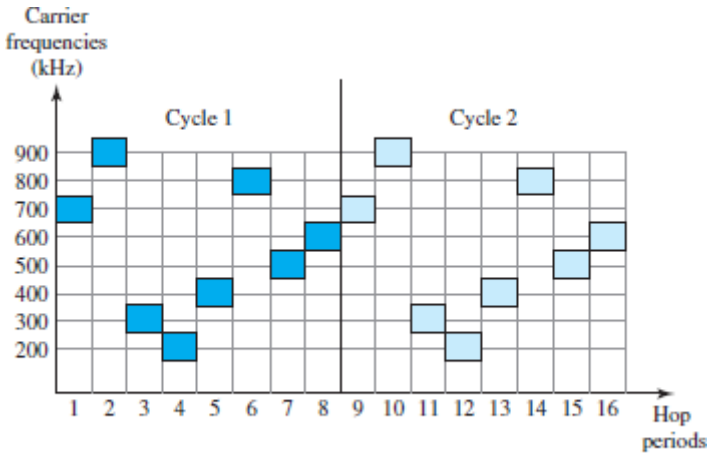


Figure 6.30 *FHSS cycles*

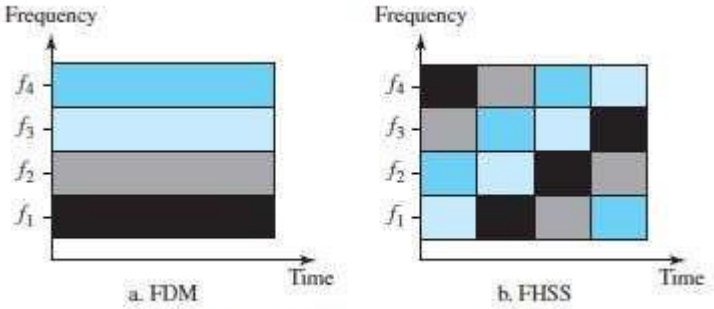


Figure 6.31 *Bandwidth sharing*

- If there are many k-bit patterns & the hopping period is short, a sender & receiver can have privacy. If an attacker tries to intercept the transmitted signal, he can only access a small piece of data because he does not know the spreading sequence to quickly adapt himself to the next hop.
- The scheme has also an anti-jamming effect.

  A malicious sender may be able to send noise to jam the signal for one hopping period (randomly), but not for the whole period.

Bandwidth Sharing

- If the number of hopping frequencies is M, we can multiplex M channels into one by using the same B_{ss} bandwidth.
- This is possible because
  1) A station uses just one frequency in each hopping period.
  2) Other M-1 stations uses other M-1 frequencies.
- In other words, M different stations can use the same B_{ss} if a multiple FSK (MFSK) is used.

Direct Sequence Spread Spectrum (DSSS)

- This technique expands the bandwidth of the original signal.
- Each data-bit is replaced with „n' bits using a spreading-code.
- Each bit is assigned a code of „n' bits called chips.
- The chip-rate is „n' times that of the data-bit (Figure 6.32).

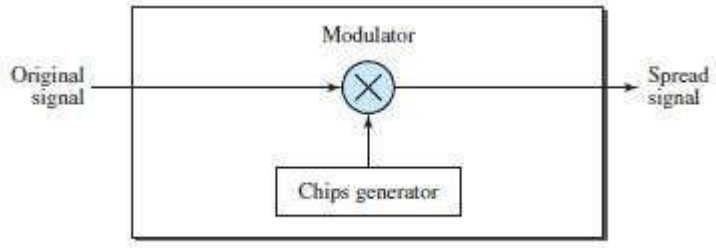


Figure 6.32 *DSSS*

- For example (Figure 6.33):
  - Consider the Barker sequence used in a wireless LAN. Here n =11.
  - Assume that the original signal and the chips in the chip-generator use polar NRZ encoding.
  - The spreading-code is 11 chips having the pattern 10110111000.
  - If the original signal-rate is N, the rate of the spread signal is 1/N.
  - This means that the required bandwidth for the spread signal is 11 times larger thanthe bandwidth of the original signal.

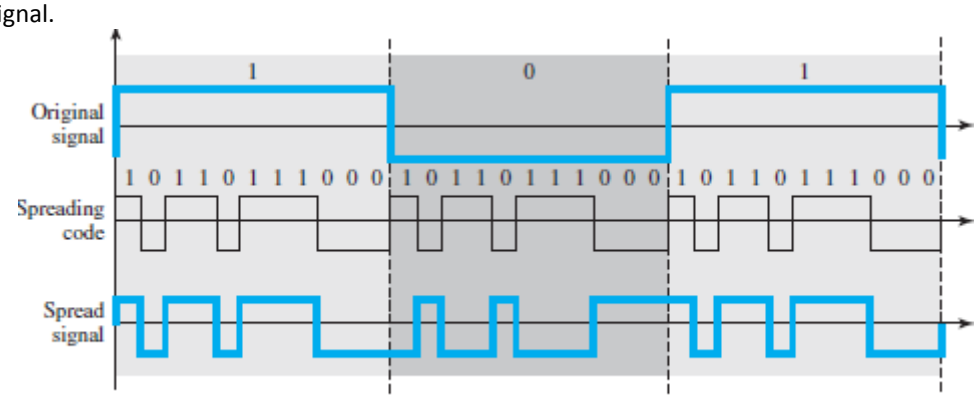


Figure 6.33 *DSSS example*

- The spread signal can provide privacy if the attacker does not know the code.
- It can also provide immunity against interference if each station uses a different code.

Bandwidth Sharing

- Can we share a bandwidth in DSSS?
- The answer is no and yes.

3) If we use a spreading-code that spreads signals that cannot be combined and separated, we cannot share a bandwidth.

- For example:

Some wireless LANs use DSSS and the spread bandwidth cannot be shared.

4) If we use a special spreading-code that spreads signals that can be combined and separated, we can share a bandwidth.

- For example:

Cellular telephony uses DSSS and the spread bandwidth is shared b/w several user

SWITCHING

SWITCHING

- A network is a set of connected-devices.
- Problem: Whenever we have multiple-devices, we have the problem of how to connect them to make one-to-one communication possible.
- Solution: Use 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.
- In a switched-network,
    1) Some nodes are connected to the end-systems (For example: PC or TP).
    2) Some nodes are used only for routing.

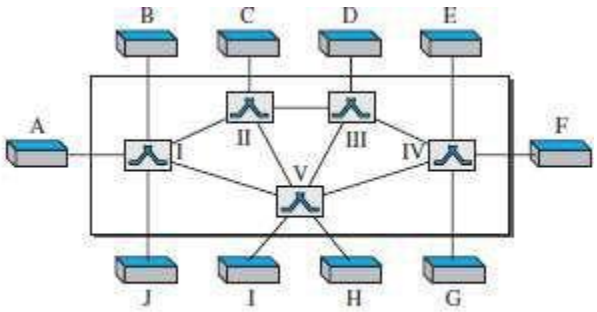


Figure 8.1 *Switched network*

- As shown in Figure 8.1,
    1) The end-systems are labeled A, B, C, D, and so on.
    2) The switches are labeled I, II, III, IV, and V. Each switch is connected to multiple links.

Three Methods of Switching

- Three methods of Switching are (Figure 8.2):
    1) Circuit Switching
    2) Packet Switching and
    3) Message Switching.
- The first two are commonly used today.
- The third has been phased out in general communications but still has networking applications.
- Packet switching can further be divided into two subcategories—virtual circuit approach and datagram approach

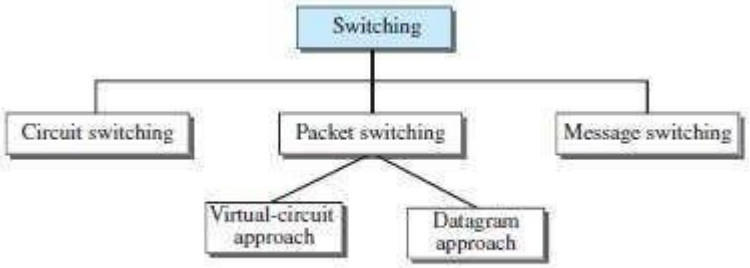


Figure 8.2 *Taxonomy of switched networks*

Switching and TCP/IP Layers

- Switching can happen at several layers of the TCP/IP protocol suite.

4) Switching at Physical Layer

- At the physical layer, we can have only circuit switching.
- There are no packets exchanged at the physical layer.
- The switches at the physical layer allow signals to travel in one path or another.

5) Switching at Data-Link Layer

- At the data-link layer, we can have packet switching.
- However, the term packet in this case means frames or cells.
- Packet switching at the data-link layer is normally done using a virtual-circuit approach.

6) Switching at Network Layer

- At the network layer, we can have packet switching.
- In this case, either a virtual-circuit approach or a datagram approach can be used.
- Currently the Internet uses a datagram approach, but the tendency is to move to a virtual- circuit approach.

7) Switching at Application Layer

- At the application layer, we can have only message switching.
- The communication at the application layer occurs by exchanging messages.
- Conceptually, we can say that communication using e-mail is a kind of message-switched communication, but we do not see any network that actually can be called a message-switched network.

CIRCUIT SWITCHED NETWORK

- This is similar to telephone system.
- Fixed path (connection) is established between a source and a destination prior to the transfer of packets.
- A circuit-switched-network consists of a set of switches connected by physical-links (Figure 8.3).
- A connection between 2 stations is a dedicated-path made of one or more links.
- However, each connection uses only one dedicated-channel on each link.
- Normally, each link is divided into „n' channels by using FDM or TDM.
- The resources need to be reserved during the setup phase.

The resources remain dedicated for the entire duration of data transfer until the teardown phase.

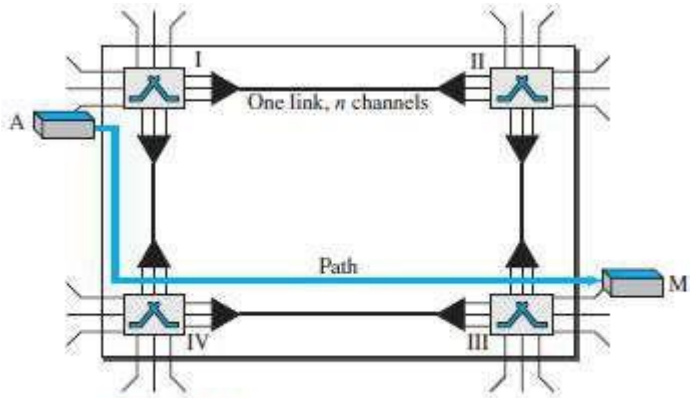


Figure 8.3 *A trivial circuit-switched network*

- The virtual-circuit setup procedure
  → first determines a path through the network &
  → sets parameters in the switches by exchanging connect-request & connect-confirm messages
- If a switch does not have enough resources to set up a virtual circuit, the switch responds with a connect-reject message and the setup procedure fails (Figure 7.15).
- A connection-release procedure may also be required to terminate the connection.

Three Phases

- The communication requires 3 phases:
  1) Connection-setup
  2) Data-transfer
  3) Connection teardown.

1) Setup Phase

- Before the 2 parties can communicate, a dedicated-circuit needs to be established.
- Normally, the end-systems are connected through dedicated-lines to the switches. So, connection-setup means creating dedicated-channels between the switches.
- For ex: Assume system-A needs to connect to system-M. For this, following events occur:
  i) System-A sends a setup-request to switch-I.
  ii) Switch-I finds a channel between itself and switch-IV that can be dedicated for this purpose.
  iii) Switch-I then sends the request to switch-IV, which finds a dedicated-channel between itself and switch-III.
  iv) Switch-III informs system-M of system-A's intention at this time.
  v) Finally, an acknowledgment from system-M needs to be sent in the opposite direction to system-A.
- Only after system A receives this acknowledgment is the connection established.

2) Data Transfer Phase

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

3) Teardown Phase

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

Efficiency

- Circuit-switched-networks are inefficient when compared to other two types of networks because
    8) Resources are allocated during the entire duration of the connection.
    9) These resources are unavailable to other connections.

Delay

- Circuit-switched-networks have minimum delay when compared to other two types of networks
- During data-transfer,
    10) The data are not delayed at each switch.
    11) The resources are allocated for the duration of the connection.

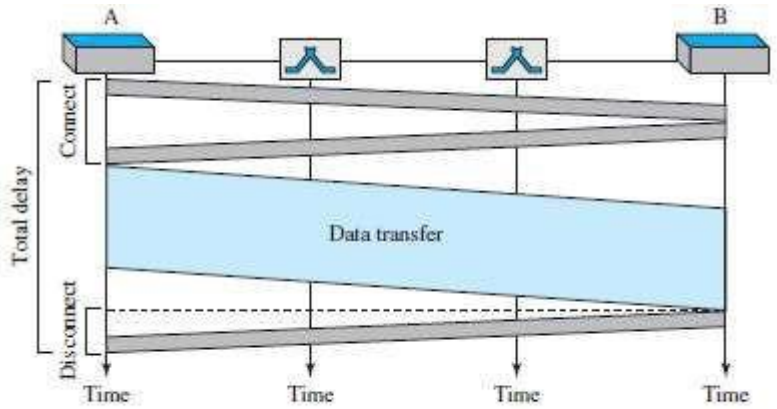


Figure 8.6 *Delay in a circuit-switched network*

- As in the above figure, there is no waiting time at each switch.
- The total delay is the time needed to
    1) Create the connection
    2) Transfer-data and
    3) Disconnect the circuit.
- The delay caused by the setup is the sum of 4 parts:
    1) The propagation time of the source-computer request.
    2) The request signal transfer time.
    3) The propagation time of the acknowledgment from the destination computer.
    4) The signal transfer time of the acknowledgment.
- The delay due to data-transfer is the sum of 2 parts:
    1) The propagation time.
    2) Data-transfer time which can be very long.

-PACKET SWITCHED NETWORK

- The message is divided into packets of fixed or variable size.
- The packet-size is determined by
  → network and
  → governing protocol.
- There is no resource reservation; resources are allocated on-demand.

2.8.1 Datagram Networks

- This is analogous to postal system.
- Each packet is routed independently through the network.
- Each packet has a header that contains source and destination addresses.
- Each switch examines the header to determine the next hop in the path to the destination.
- If the transmission line is busy
  then the packet is placed in the queue until the line becomes free.
- Packets are referred to as datagrams.
- Datagram switching is normally done at the network layer.
- In Internet, switching is done by using the datagram switching.
- Advantage:
  1) High utilization of transmission-line can be achieved by sharing among multiple packets.
- Disadvantages:
  1) Packets may arrive out-of-order, and re-sequencing may be required at the destination
  2) Loss of packets may occur when a switch has insufficient buffer

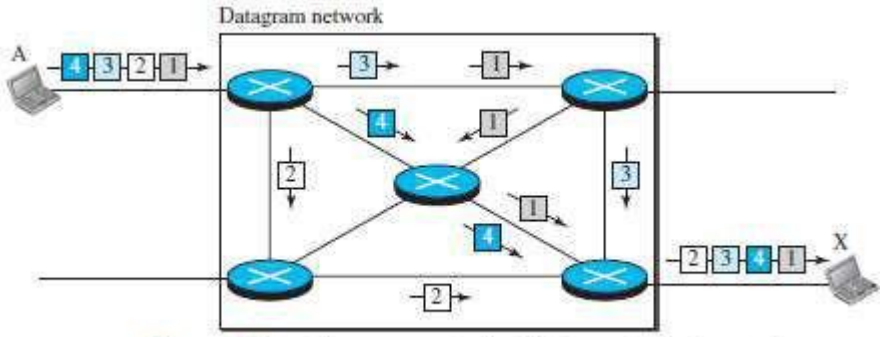


Figure 8.7 *A datagram network with four switches (routers)*

- The Figure 8.7 shows how the 4 packets are transferred from station-A to station-X.
- The switches are referred to as routers.
- All four packets (or datagrams) belong to the same message, but may travel different paths to reach their destination.
- This is so because the links may be involved in carrying packets from other sources and do not have the necessary bandwidth available to carry all the packets from A to X.
- This approach can cause the datagrams of a transmission to arrive at their destination out-of- order with different delays between the packets.
- Packets may also be lost or dropped because of a lack-of-resources.
- It is the responsibility of an upper-layer protocol to
  → reorder the datagrams or
  → ask for lost datagrams.
- The datagram-networks are referred to as connectionless networks. This is because
  1) The switch does not keep information about the connection state.
  2) There are no setup or teardown phases.
  3) Each packet is treated the same by a switch regardless of its source or destination.

Routing Table

- Each switch has a routing-table which is based on the destination-address.
- The routing-tables are dynamic & updated periodically.
- The destination-addresses and the corresponding forwarding output-ports are recorded in the tables.

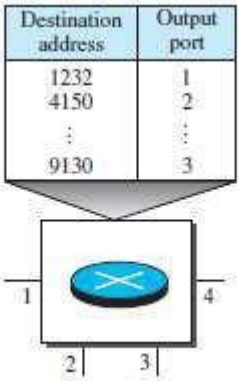


Figure 8.8 *Routing table in a datagram network*

2.8.1.1.1 Destination Address

- Every packet carries a header that contains 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.
- The destination address in the header of a packet in remains the same during the entire journey of the packet.

2.8.1.1.2 Efficiency

- Datagram-networks are more efficient when compared to circuit-switched-network. This is because
  1) Resources are allocated only when there are packets to be transferred.
  2) 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 re-allocated during these minutes for other packets from other sources.

2.8.1.1.3 Delay

- Datagram-networks may have greater delay when compared to circuit-switched-network. This is because
  1) Each packet may experience a wait at a switch before it is forwarded.
  2) Since not all packets in a message necessarily travel through the same switches, the delay is not uniform for the packets of a message.

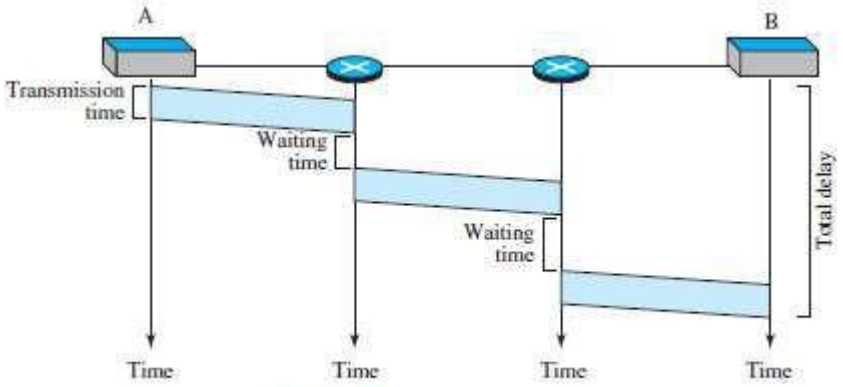


Figure 8.9 *Delay in a datagram network*

- The Figure 8.9 gives an example of delay for one single packet.
- The packet travels through two switches.
- There are three transmission times (3T), three propagation delays (slopes 3t of the lines), and two waiting times (W1+ W2).

$$\text{Total delay} = 3T + 3\tau + w_1 + w_2$$

Virtual Circuit Network (VCN)

- This is similar to telephone system.
- A virtual-circuit network is a combination of circuit-switched-network and datagram-network.
- Five characteristics of VCN:
    1) As in a circuit-switched-network, there are setup & teardown phases in addition to the data transfer phase.
    2) As in a circuit-switched-network, resources can be allocated during the setup phase. As in a datagram-network, resources can also be allocated on-demand.
    3) As in a datagram-network, data is divided into packets. Each packet carries an address in the header.
        However, the address in the header has local jurisdiction, not end-to-end jurisdiction.
    4) As in a circuit-switched-network, all packets follow the same path established during the connection.
    5) A virtual-circuit network is implemented in the data link layer.
        A circuit-switched-network is implemented in the physical layer.
        A datagram-network is implemented in the network layer.

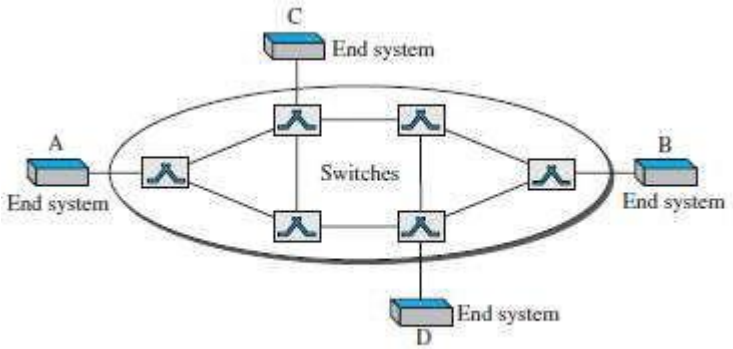


Figure 8.10 *Virtual-circuit network*

- The Figure 8.10 is an example of a virtual-circuit network.
- The network has switches that allow traffic from sources to destinations.
- A source or destination can be a computer, packet switch, bridge, or any other device that connects other networks.

Addressing

- Two types of addressing: 1) Global and 2) Local (virtual-circuit identifier).

1) Global Addressing

- A source or a destination needs to have a global address.
- Global address is an address that can be unique in the scope of the network or internationally if the network is part of an international network.

2) Virtual Circuit Identifier

- The identifier used for data-transfer is called the virtual-circuit identifier (VCI).
- A VCI, unlike a global address, is a small number that has only switch scope.
- VCI is used by a frame between two switches.
- When a frame arrives at a switch, it has a VCI. When the frame leaves, it has a different VCI.

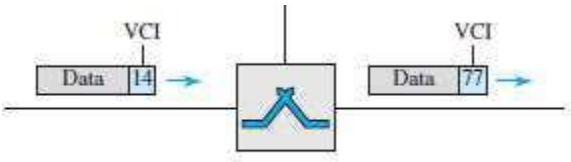


Figure 8.11 *Virtual-circuit identifier*

- Figure 8.11 show how the VCI in a data-frame changes from one switch to another.

Three Phases

- A source and destination need to go through 3 phases: setup, data-transfer, and teardown.
    3) In setup phase, the source and destination use their global addresses to help switches make table entries for the connection.
    4) In the teardown phase, the source and destination inform the switches to delete the corresponding entry.
    5) Data-transfer occurs between these 2 phases.

Data Transfer Phase

- To transfer a frame from a source to its destination, all switches need to have a table-entry for this virtual-circuit.
- The table has four columns.
- The switch holds 4 pieces of information for each virtual-circuit that is already set up.

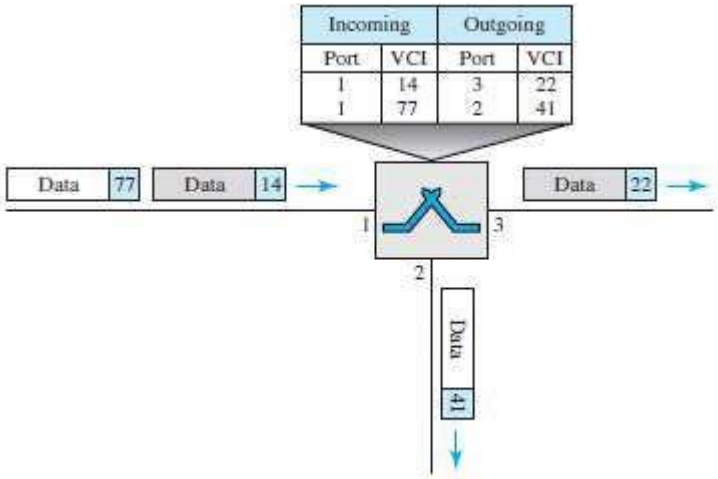


Figure 8.12 *Switch and tables in a virtual-circuit network*

- As shown in Figure 8.12, a frame arrives at port 1 with a VCI of 14.
- When the frame arrives, the switch looks in its table to find port 1 and a VCI of 14.
- When it is found, the switch knows to change the VCI to 22 & send out the frame from port 3.

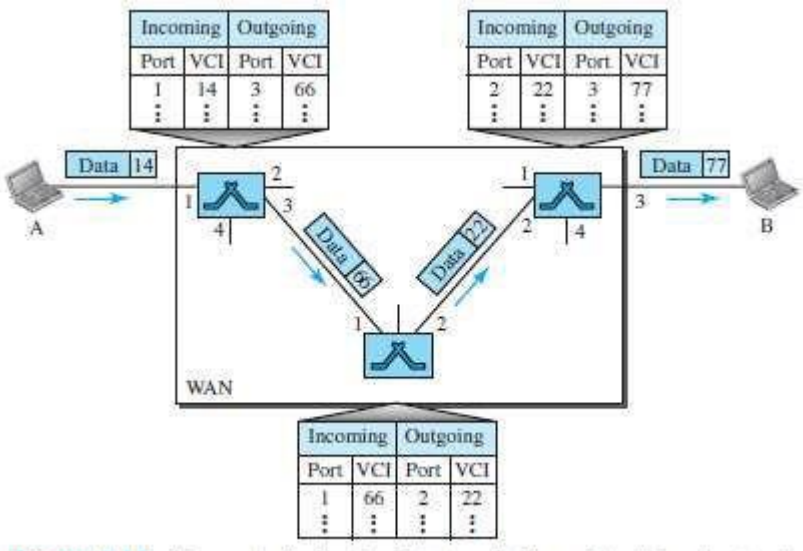


Figure 8.13 *Source-to-destination data transfer in a virtual-circuit network*

- As shown in Figure 8.13, each switch changes the VCI and routes the frame.
- The data-transfer phase is active until the source sends all its frames to the destination.
- The procedure at the switch is the same for each frame of a message.
- The process creates a virtual circuit, not a real circuit, between the source and destination.

Setup Phase

- A switch creates an entry for a virtual-circuit.
- For example, suppose source A needs to create a virtual-circuit to B.
- Two steps are required: 1) Setup-request and
  2) Acknowledgment.

1) Setup Request

➢ A setup-request frame is sent from the source to the destination (Figure 8.14).

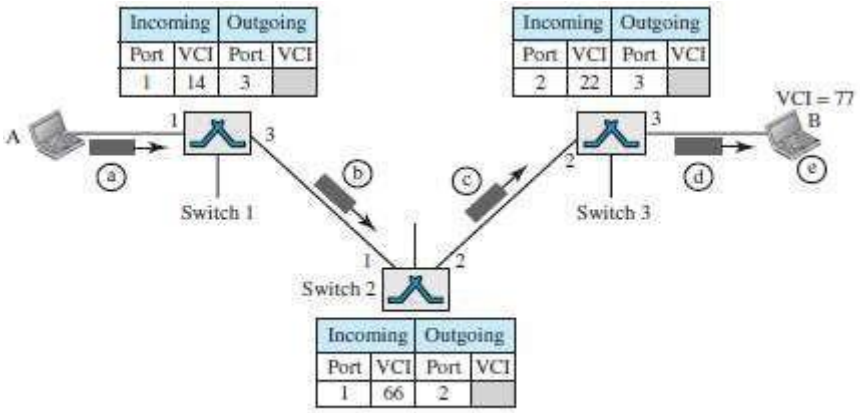


Figure 8.14 *Setup request in a virtual-circuit network*

➢ Following events occurs:

a) Source-A sends a setup-frame to switch-1.

b) Switch-1 receives the setup-frame.

- ¤ Switch-1 knows that a frame going from A to B goes out through port 3.
- ¤ The switch-1 has a routing table.
- ¤ The switch
  - → creates an entry in its table for this virtual-circuit
  - → is only able to fill 3 of the 4 columns.
- ¤ The switch
  - → assigns the incoming port (1) and
  - → chooses an available incoming-VCI (14) and the outgoing-port (3).
  - → does not yet know the outgoing VCI, which will be found during the acknowledgment step.
- ¤ The switch then forwards the frame through port-3 to switch-2.

c) Switch-2 receives the setup-request frame.

- ¤ The same events happen here as at switch-1.
- ¤ Three columns of the table are completed: In this case, incoming port (1), incoming-VCI (66), and outgoing port (2).

d) Switch-3 receives the setup-request frame.

- ¤ Again, three columns are completed: incoming port (2), incoming-VCI (22), and outgoing-port (3).

e) Destination-B

- → receives the setup-frame
- → assigns a VCI to the incoming frames that come from A, in this case 77.
- ¤ This VCI lets the destination know that the frames come from A, and no other sources.

2) Acknowledgment

- A special frame, called the acknowledgment-frame, completes the entries in the switching- tables (Figure 8.15).

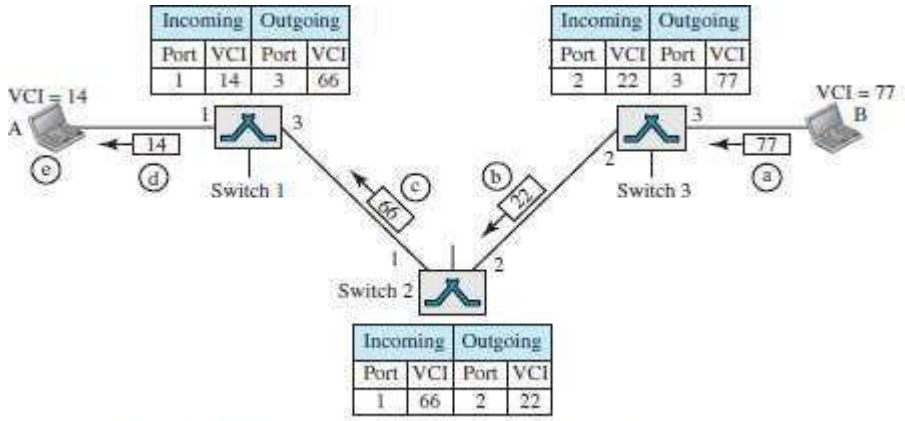


Figure 8.15 *Setup acknowledgment in a virtual-circuit network*

a) The destination sends an acknowledgment to switch-3.

- The acknowledgment carries the global source and destination-addresses so the switch knows which entry in the table is to be completed.
- The frame also carries VCI 77, chosen by the destination as the incoming-VCI for frames from A.
- Switch 3 uses this VCI to complete the outgoing VCI column for this entry.

b) Switch 3 sends an acknowledgment to switch-2 that contains its incoming-VCI in the table, chosen in the previous step.

- Switch-2 uses this as the outgoing VCI in the table.

c) Switch-2 sends an acknowledgment to switch-1 that contains its incoming-VCI in the table, chosen in the previous step.

- Switch-1 uses this as the outgoing VCI in the table.

d) Finally switch-1 sends an acknowledgment to source-A that contains its incoming-VCI in the table, chosen in the previous step.

e) The source uses this as the outgoing VCI for the data-frames to be sent to destination-B.

Teardown Phase

- Source-A, after sending all frames to B, sends a special frame called a teardown request.
- Destination-B responds with a teardown confirmation frame.
- All switches delete the corresponding entry from their tables.

Efficiency

- Resource reservation can be made in 2 cases:
  1) During the setup: Here, the delay for each packet is the same.
  2) On demand: Here, each packet may encounter different delays.
- Advantage of on demand resource allocation:
  The source can check the availability of the resources, without actually reserving it.

Delay in Virtual Circuit Networks

- There is a one-time delay for setup and a one-time delay for teardown (Figure 8.16).
- If resources are allocated during the setup phase, there is no wait time for individual packets.
- The packet is traveling through two switches (routers).
- There are three transmission times (3T), three propagation times (3τ), data transfer delay, a setup delay and a teardown delay.
- The total delay time is

$$\text{Total delay} + 3T + 3\tau + \text{setup delay} + \text{teardown delay}$$

Circuit Switching	Datagram Packet Switching	Virtual circuit Packet switching
Dedicate transmission path	No dedicate path	No dedicate path
Continuous transmission of data	Transmission of packets	Transmission of packets
Fast enough for interactive	Fast enough for interactive	Fast enough for interactive
Message are not stored	Packets may be stored until delivered	Packets stored until delivered
The path is established for entire conversation	Route established for each packet	Route established for entire conversation
Call setup delay; negligible transmission delay	Packet transmission delay	Call setup delay; Packet transmission delay
Busy signal if called party busy	Sender may be notified if packet not delivered	Sender notified of connection denial
Overload may block call setup; no delay for established calls	Overload increases packet delay	Overload may block call setup; increases packet delay
Electromechanical or computerized switching nodes	Small switching nodes	Small switching nodes
User responsible for message loss protection	Network may be responsible for individual packets	Network may be responsible for packet sequences
Usually no speed or code conversion	Speed and code conversion	Speed and code conversion
Fixed bandwidth	Dynamic use of bandwidth	Dynamic use of bandwidth
No overhead bits after call setup	Overhead bits in each packet	Overhead bits in each packet

ERROR-DETECTION AND CORRECTION

INTRODUCTION

Types of Errors

- When bits flow from 1 point to another, they are subject to unpredictable-changes '.' of interference.
- The interference can change the shape of the signal.
- Two types of errors: 1) Single-bit error 2) Burst-error.

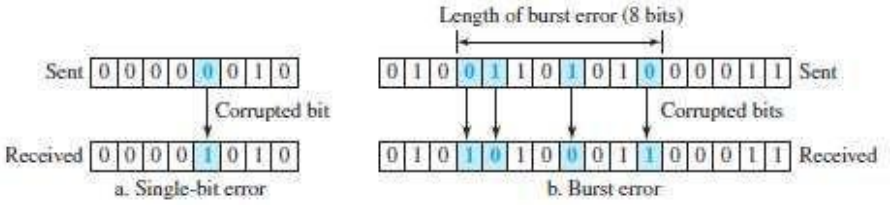


Figure 10.1 *Single-bit and burst error*

Single-Bit Error

- Only 1 bit of a given data is changed
  - → from 1 to 0 or
  - → from 0 to 1 (Figure 10.1a).

Burst Error

- Two or more bits in the data have changed
  - → from 1 to 0 or
  - → from 0 to 1 (Figure 10.1b).
- A burst-error occurs more than a single-bit error. This is because:
  - Normally, the duration of noise is longer than the duration of 1-bit.
- When noise affects data, the noise also affects the bits.
- The no. of corrupted-bits depends on
  - → data-rate and
  - → duration of noise.

Redundancy

- The central concept in detecting/correcting errors is *redundancy*.
- Some extra-bits along with the data have to be sent to detect/correct errors. These extra bits are called redundant-bits.
- The redundant-bits are
  - → added by the sender and
  - → removed by the receiver.
- The presence of redundant-bits allows the receiver to detect/correct errors.

Error Detection vs. Error Correction

- Error-correction is more difficult than error-detection.

Error Detection

- Here, we are checking whether any error has occurred or not.
- The answer is a simple YES or NO.
- We are not interested in the number of corrupted-bits.

Error Correction

- Here, we need to know
  - → exact number of corrupted-bits and
  - → location of bits in the message.
- Two important factors to be considered:
  1) Number of errors and
  2) Message-size.

Coding

- Redundancy is achieved through various coding-schemes.
  2) Sender adds redundant-bits to the data-bits. This process creates a relationship between
     → redundant-bits and
     → data-bits.
  3) Receiver checks the relationship between redundant-bits & data-bits to detect/correct errors.
- Two important factors to be considered:
  1) Ratio of redundant-bits to the data-bits and
  2) Robustness of the process.
- Two broad categories of coding schemes: 1) Block-coding and 2) Convolution coding.

Block Coding

- The message is divided into k-bit blocks. These blocks are called data-words.
- Here, r-redundant-bits are added to each block to make the length n=k+r.
- The resulting n-bit blocks are called code-words.
- Since n>k, the number of possible code-words is larger than the number of possible data-words.
- Block-coding process is 1-to-1; the same data-word is always encoded as the same code-word.
- Thus, we have 2^n-2^k code-words that are not used. These code-words are invalid or illegal.

Error Detection

- If the following 2 conditions are met, the receiver can detect a change in the original code-word:
  1) The receiver has a list of valid code-words.
  2) The original code-word has changed to an invalid code-words.

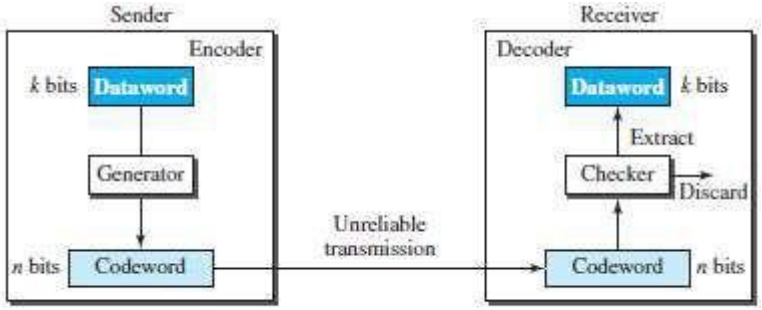


Figure 10.2 *Process of error detection in block coding*

- Here is how it works (Figure 10.2):

At Sender

i) The sender creates code-words out of data-words by using a generator.
The generator applies the rules and procedures of encoding.

ii) During transmission, each code-word sent to the receiver may change.

At Receiver

i) a) If the received code-word is the same as one of the valid code-words,
the code-word is accepted; the corresponding data-word is extracted for use.

b) If the received code-word is invalid, the code-word is discarded.

ii) However, if the code-word is corrupted but the received code-word still matches a valid code-word, the error remains undetected.

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

Example 3.1

Let us assume that $k = 2$ and $n = 3$. Table 10.1 shows the list of datawords and codewords.

Table 10.1 *A code for error detection*

Dataword	*Codeword*	*Dataword*	*Codeword*
00	000	10	101
01	011	11	110

Assume the sender encodes the dataword 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 dataword 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 dataword 00. Two corrupted bits have made the error undetectable.

Hamming Distance

- The main concept for error-control: Hamming distance.
- The Hamming distance b/w 2 words is the number of differences between the corresponding bits.
- Let d(x,y) = Hamming distance b/w 2 words x and y.
- Hamming distance can be found by
  → applying the XOR operation on the 2 words and
  → counting the number of 1s in the result.
- For example:
  1) The Hamming distance d(000, 011) is 2 because 000⊕011= 011 (two 1s).
  2) The Hamming distance d(10101, 11110) is 3 because 10101⊕11110 = 01011 (three 1s).

Hamming Distance and Error

➢ Hamming distance between the received word and the sent code-word is the number of bits that are corrupted during transmission.

➢ For example: Let Sent code-word = 00000
Received word = 01101
Hamming distance = d(00000, 01101) =3. Thus, 3 bits are in error.

Minimum Hamming Distance for Error Detection

- Minimum Hamming distance is the smallest Hamming distance b/w all possible pairs of code-words.
- Let d_{min} = minimum Hamming distance.
- To find d_{min} value, we find the Hamming distances between all words and select the smallest one.

Minimum-distance for Error-detection

➢ If 's' errors occur during transmission, the Hamming distance b/w the sent code-word and received code-word is 's' (Figure 10.3).

➢ If code has to detect upto 's' errors, the minimum-distance b/w the valid codes must be 's+1'
i.e. $d_{min}=s+1$.

➢ We use a geometric approach to define $d_{min}=s+1$.

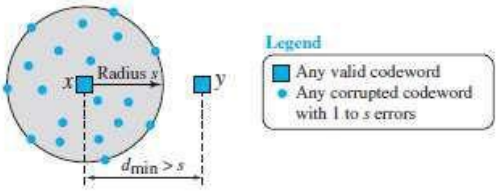


Figure 10.3 *Geometric concept explaining d_{min} in error detection*

¤ Let us assume that the sent code-word x is at the center of a circle with radius s.
¤ All received code-words that are created by 0 to s errors are points inside the circle or on the perimeter of the circle.
¤ All other valid code-words must be outside the circle

- For example: A code scheme has a Hamming distance $d_{min} = 4$.
  This code guarantees the detection of upto 3 errors (d = s + 1 or s = 3).

Linear Block Codes

- Almost all block codes belong to a subset of block codes called linear block codes.
- A linear block code is a code in which the XOR of 2 valid code-words creates another valid code-word. (XOR → Addition modulo-2)

Table 10.1 *A code for error detection*

Dataword	Codeword	Dataword	Codeword
00	000	10	101
01	011	11	110

- The code in Table 10.1 is a linear block code because the result of XORing any code-word with any other code-word is a valid code-word.

For example, the XORing of the 2nd and 3rd code-words creates the 4th one.

3.2.1.2.1 Minimum Distance for Linear Block Codes

- Minimum Hamming distance is no. of 1s in the nonzero valid code-word with the smallest no. of 1s.
- In Table 10.1,

The numbers of 1s in the nonzero code-words are 2, 2, and 2. So the minimum Hamming distance is $d_{min} = 2$.

Parity Check Code

- This code is a linear block code. This code can detect an odd number of errors.
- A k-bit data-word is changed to an n-bit code-word where n=k+1.
- One extra bit is called the parity-bit.
- The parity-bit is selected to make the total number of 1s in the code-word even.
- Minimum hamming distance $d_{min} = 2$. This means the code is a single-bit error-detecting code.

Table 10.2 *Simple parity-check code C(5, 4)*

Dataword	Codeword	Dataword	Codeword
0000	**00000**	1000	**10001**
0001	**00011**	1001	**10010**
0010	**00101**	1010	**10100**
0011	**00110**	1011	**10111**
0100	**01001**	1100	**11000**
0101	**01010**	1101	**11011**
0110	**01100**	1110	**11101**
0111	**01111**	1111	**11110**

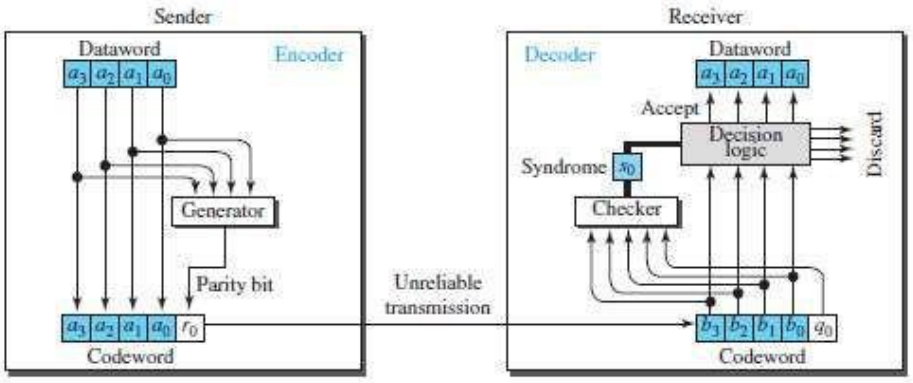


Figure 10.4 *Encoder and decoder for simple parity-check code*

- Here is how it works (Figure 10.4):

At Sender

- The encoder uses a generator that takes a copy of a 4-bit data-word (a_0, a_1, a_2, and a_3) and generates a parity-bit r_0.
- The encoder
  - → accepts a copy of a 4-bit data-word (a_0, a_1, a_2, and a_3) and
  - → generates a parity-bit r_0 using a generator
  - → generates a 5-bit code-word
- The parity-bit & 4-bit data-word are added to make the number of 1s in the code-word even.
- The addition is done by using the following:

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

- The result of addition is the parity-bit.
  1) If the no. of 1s in data-word = even, result = 0. (r_0=0)
  2) If the no. of 1s in data-word = odd, result = 1. (r_0=1)
  3) In both cases, the total number of 1s in the code-word is even.
- The sender sends the code-word, which may be corrupted during transmission.

At Receiver

- The receiver receives a 5-bit word.
- The checker performs the same operation as the generator with one exception: The addition is done over all 5 bits.

$$s_0 = b_3 + b_2 + b_1 + b_0 + q_0 \quad \text{(modulo-2)}$$

- The result is called the syndrome bit (s_0).
- Syndrome bit = 0 when the no. of 1s in the received code-word is even; otherwise, it is 1.
- The syndrome is passed to the decision logic analyzer.
  4) If s_0=0, there is no error in the received code-word. The data portion of the received code-word is accepted as the data-word.
  5) If s_0=1, there is error in the received code-word. The data portion of the received code-word is discarded. The data-word is not created.

Example 3.2

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

1. No error occurs; the received codeword is 10111. The syndrome is 0. The dataword 1011 is created.
2. One single-bit error changes a_1. The received codeword is 10011. The syndrome is 1. No dataword is created.
3. One single-bit error changes r_0. The received codeword is 10110. The syndrome is 1. No dataword is created. Note that although none of the dataword bits are corrupted, no dataword is created because the code is not sophisticated enough to show the position of the corrupted bit.
4. An error changes r_0 and a second error changes a_3. The received codeword is 00110. The syndrome is 0. The dataword 0011 is created at the receiver. Note that here the dataword is wrongly created due to the syndrome value. The simple parity-check decoder cannot detect an even number of errors. The errors cancel each other out and give the syndrome a value of 0.
5. Three bits—a_3, a_2, and a_1—are changed by errors. The received codeword is 01011. The syndrome is 1. The dataword is not created. This shows that the simple parity check, guaranteed to detect one single error, can also find any odd number of errors.

Cyclic Codes

- Cyclic codes are special linear block codes with one extra property:

  If a code-word is cyclically shifted (rotated), the result is another code-word.

  For ex: if code-word = 1011000 and we cyclically left-shift, then another code-word = 0110001.
- Let First-word = a_0 to a_6 and 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 (CRC)

- CRC is a cyclic code that is used in networks such as LANs and WANs.

Table 10.3 *A CRC code with C(7, 4)*

Dataword	Codeword	Dataword	Codeword
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

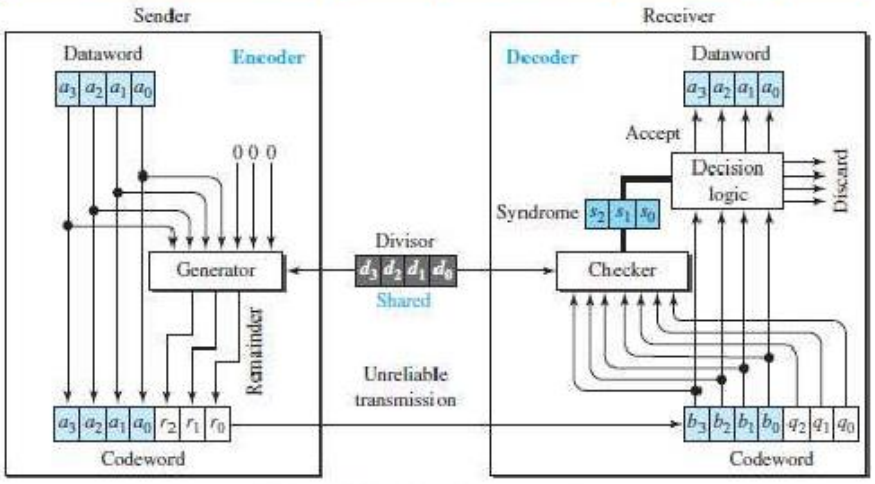


Figure 10.5 *CRC encoder and decoder*

- Let Size of data-word = k bits (here k=4). Size of code-word = n bits (here n=7).

Size of divisor = n-k+1 bits (here n-k+1=4). (Augmented → increased)

- Here is how it works (Figure 10.5):

At Sender

- n-k 0s is appended to the data-word to create augmented data-word. (here n-k=3).
- The augmented data-word is fed into the generator (Figure 10.6).
- The generator divides the augmented data-word by the divisor.
- The remainder is called check-bits ($r_2r_1r_0$).
- The check-bits ($r_2r_1r_0$) are appended to the data-word to create the code-word.

At Receiver

- The possibly corrupted code-word is fed into the checker.
- The checker is a replica of the generator.
- The checker divides the code-word by the divisor.
- The remainder is called syndrome bits ($r_2r_1r_0$).
- The syndrome bits are fed to the decision-logic-analyzer.
- The decision-logic-analyzer performs following functions:

i) For No Error

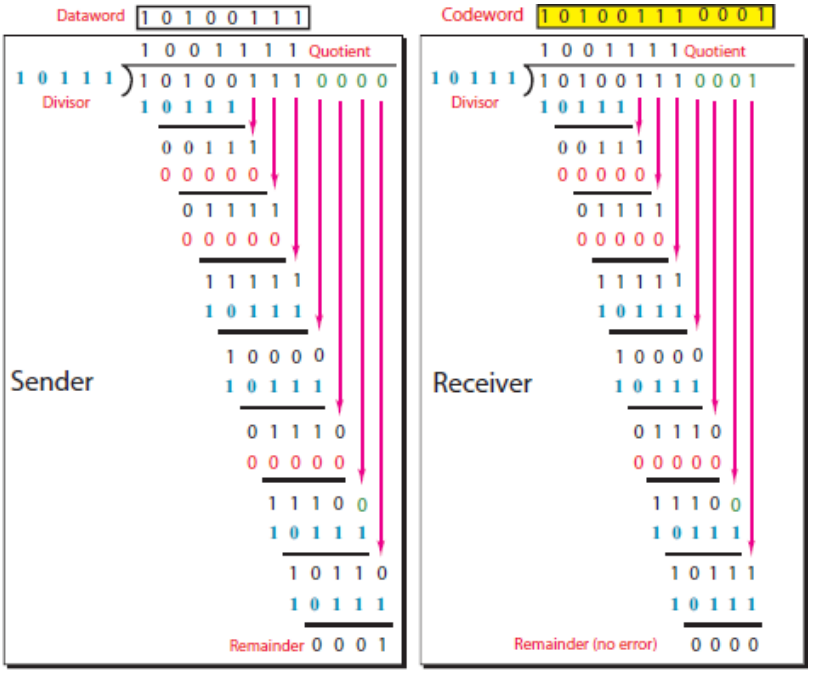
- If all syndrome-bits are 0s, the received code-word is accepted.
- Data-word is extracted from received code-word (Figure 10.7a).

ii) For Error

- If all syndrome-bits are not 0s, the received code-word is discarded (Figure 10.7b).

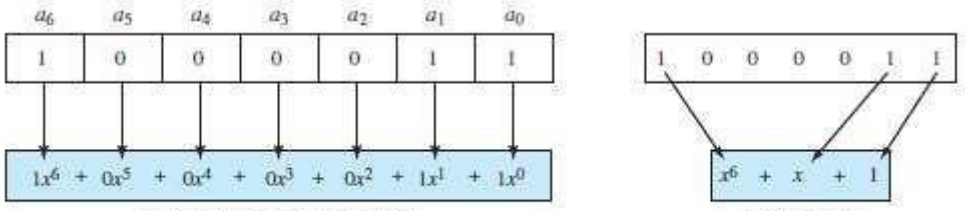
Example 3.3

Given the dataword 10100111 and the divisor 10111, show the generation of the CRC codeword at the sender site (using binary division).



Polynomials

- A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0 and 1 (Figure 10.8).
- The power of each term shows the position of the bit; the coefficient shows the value of the bit.



a. Binary pattern and polynomial b. Short form

Figure 10.8 *A polynomial to represent a binary word*

Cyclic Code Encoder Using Polynomials

- Let Data-word = 1001 = x^3+1. Divisor = 1011 = x^3+x+1.
- In polynomial representation, the divisor is referred to as generator polynomial t(x) (Figure 10.9).

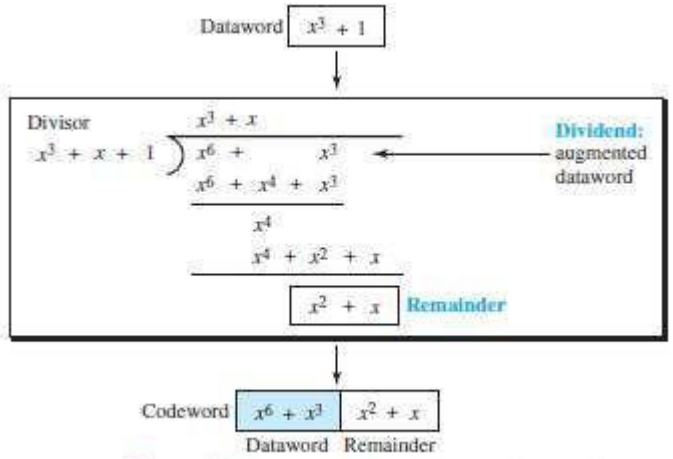


Figure 10.9 *CRC division using polynomials*

3.3.2 Cyclic Code Analysis

- We define the following, where *f*(*x*) is a polynomial with binary coefficients:

Dataword: $d(x)$ Codeword: $c(x)$ Generator: $g(x)$ Syndrome: $s(x)$ Error: $e(x)$

In a cyclic code,

1. If $s(x) \neq 0$, one or more bits is corrupted.
2. If $s(x) = 0$, either
   a. No bit is corrupted, or
   b. Some bits are corrupted, but the decoder failed to detect them.

Single Bit Error

- If the generator has more than one term and the coefficient of x^0 is 1, all single-bit errors can be caught.

Two Isolated Single-Bit Errors

- If a generator cannot divide x^i+1 (t between 0 & n-1), then all isolated double errors can be detected (Figure 10.10).

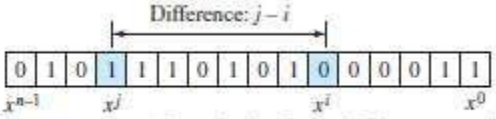


Figure 10.10 *Representation of two isolated single-bit errors using polynomials*

Odd Numbers of Errors

- A generator that contains a factor of x+1 can detect all odd-numbered errors.

A good polynomial generator needs to have the following characteristics:

1. **It should have at least two terms.**
2. **The coefficient of the term x^0 should be 1.**
3. **It should not divide $x^t + 1$, for t between 2 and $n - 1$.**
4. **It should have the factor $x + 1$.**

Standard Polynomials

Table 10.4 *Standard polynomials*

Name	*Polynomial*	*Used in*
CRC-8	$x^8 + x^2 + x + 1$ **100000111**	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$ **11000110101**	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$ **10001000000100001**	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ **10000010011000001000111011011 0111**	LANs

3.3.3 Advantages of Cyclic Codes

- The cyclic codes have a very good performance in detecting
  - → single-bit errors
  - → double errors
  - → odd number of errors and
  - → burst-errors.
- They can easily be implemented in hardware and software. They are fast when implemented in hardware.

Checksum

- Checksum is an error-detecting technique.
- In the Internet,
  - → The checksum is mostly used at the network and transport layer.
  - → The checksum is not used in the data link layer.
- Like linear and cyclic codes, the checksum is based on the concept of redundancy.
- Here is how it works (Figure 10.15):

1) At Source

- Firstly the message is divided into m-bit units.
- Then, the generator creates an extra m-bit unit called the checksum.
- The checksum is sent with the message.

2) At Destination

- The checker creates a new checksum from the combination of the message and sent- checksum.
  - i) If the new checksum is all 0s, the message is accepted.
  - ii) If the new checksum is not all 0s, the message is discarded.

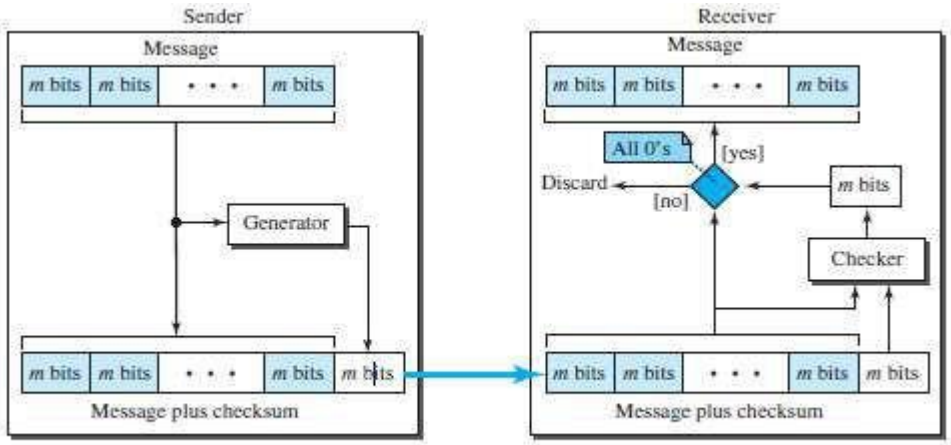


Figure 10.15 *Checksum*

Checksum

- Checksum is an error-detecting technique.
- In the Internet,
  - → The checksum is mostly used at the network and transport layer.
  - → The checksum is not used in the data link layer.
- Like linear and cyclic codes, the checksum is based on the concept of redundancy.
- Here is how it works (Figure 10.15):

At Source

- Firstly the message is divided into m-bit units.
- Then, the generator creates an extra m-bit unit called the checksum.
- The checksum is sent with the message.

At Destination

- The checker creates a new checksum from the combination of the message and sent- checksum.
  - iii) If the new checksum is all 0s, the message is accepted.
  - iv) If the new checksum is not all 0s, the message is discarded.

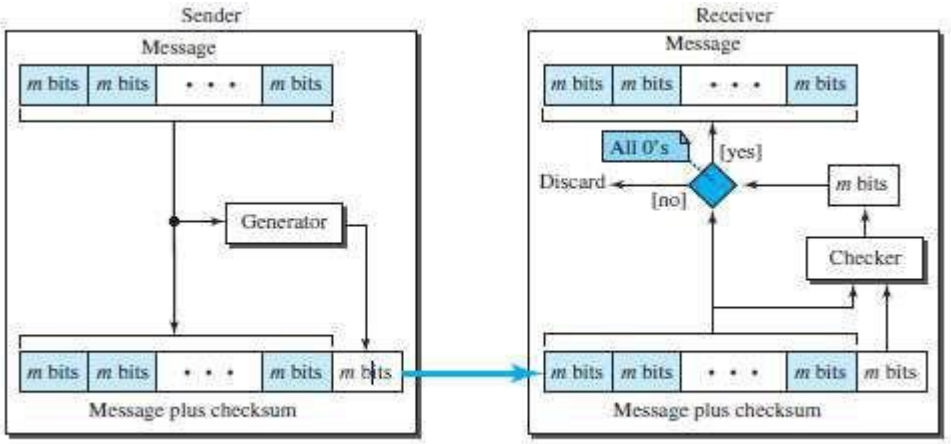


Figure 10.15 *Checksum*

Concept of Checksum

Consider the following example:

Example 3.4

- Our data is a list of five 4-bit numbers that we want to send to a destination.
- In addition to sending these numbers, we send the sum of the numbers.
- For example:

  Let set of numbers = (7, 11, 12, 0, 6).

  We send (7, 11, 12, 0, 6, 36), where 36 is the sum of the original numbers.
- The receiver adds the five numbers and compares the result with the sum.
- If the result & the sum are the same,

  The receiver assumes no error, accepts the five numbers, and discards the sum.

  Otherwise, there is an error somewhere and the data are not accepted.

Example 3.5

- To make the job of the receiver easy if we send the negative (complement) of the sum, called the checksum.
- In this case, we send (7, 11, 12, 0, 6, -36).
- The receiver can add all the numbers received (including the checksum).
- If the result is 0, it assumes no error; otherwise, there is an error.

One's Complement

- The previous example has one major drawback.

  All of our data can be written as a 4-bit word (they are less than 15) except for the checksum.
- Solution: Use one's complement arithmetic.
  - We can represent unsigned numbers between 0 and 2^n-1 using only n bits.
  - If the number has more than n bits, the extra leftmost bits need to be added to the n rightmost bits (wrapping).
  - A negative number can be represented by inverting all bits (changing 0 to 1 and 1 to 0).
  - This is the same as subtracting the number from 2^n-1.

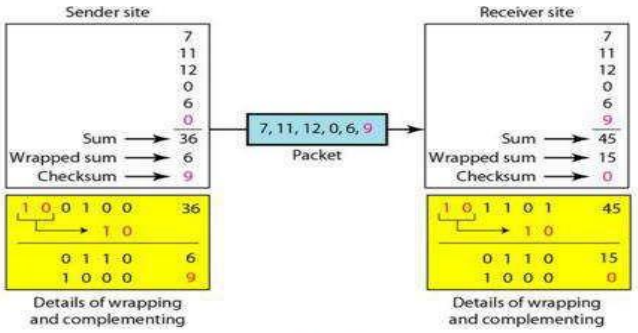


Figure 10.16

- Here is how it works (Figure 10.16):

At Sender

- The sender initializes the checksum to 0 and adds all data items and the checksum.
- The result is 36.
- However, 36 cannot be expressed in 4 bits.
- The extra two bits are wrapped and added with the sum to create the wrapped sum value 6.
- The sum is then complemented, resulting in the checksum value 9 (15 - 6 = 9).
- The sender now sends six data items to the receiver including the checksum 9.

At Receiver

- The receiver follows the same procedure as the sender.
- It adds all data items (including the checksum); the result is 45.
- The sum is wrapped and becomes 15. The wrapped sum is complemented and becomes 0.
- Since the value of the checksum is 0, this means that the data is not corrupted. The receiver drops the checksum and keeps the other data items.
- If the checksum is not zero, the entire packet is dropped.

Internet Checksum

- Traditionally, the Internet has been using a 16-bit checksum.
- The sender or the receiver uses five steps.

Table 10.5 *Procedure to calculate the traditional checksum*

Sender	*Receiver*
1. The message is divided into 16-bit words.	1. The message and the checksum are received.
2. The value of the checksum word is initially set to zero.	2. The message is divided into 16-bit words.
3. All words including the checksum are added using one's complement addition.	3. All words are added using one's complement addition.
4. The sum is complemented and becomes the checksum.	4. The sum is complemented and becomes the new checksum.
5. The checksum is sent with the data.	5. If the value of the checksum is 0, the message is accepted; otherwise, it is rejected.

Algorithm

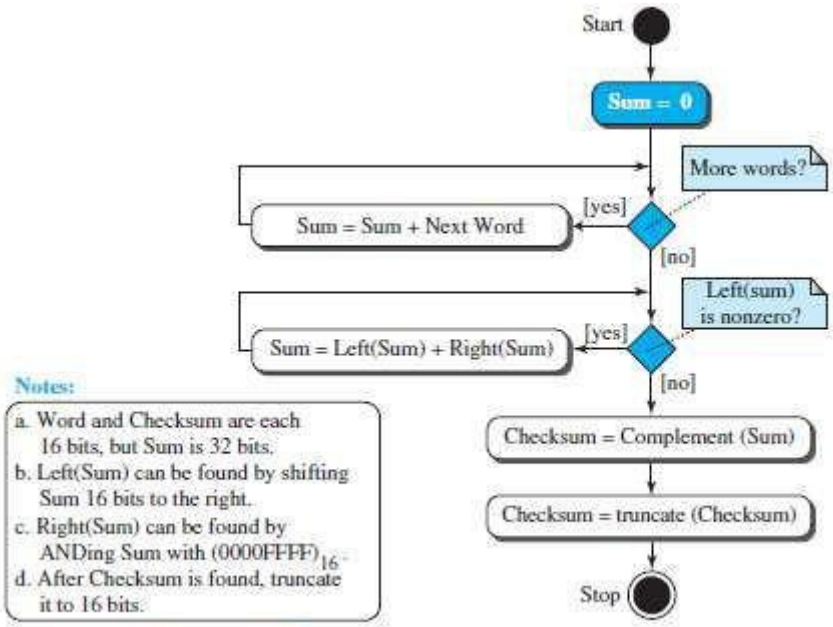


Figure 10.17 *Algorithm to calculate a traditional checksum*

Other Approaches to the Checksum

- If two 16-bit items are transposed in transmission, the checksum cannot catch this error.
- The reason is that the traditional checksum is not weighted: it treats each data item equally.
- In other words, the order of data items is immaterial to the calculation.
- Two approaches have been used to prevent this problem: 1)Fletcher and 2)Adler

Fletcher Checksum

- The Fletcher checksum was devised to weight each data item according to its position.
- Fletcher has proposed two algorithms: 8-bit and 16-bit (Figure 10.18).
- The first, 8-bit Fletcher, calculates on 8-bit data items and creates a 16-bit checksum.

The second, 16-bit Fletcher, calculates on 16-bit data items and creates a 32-bit checksum.

- The 8-bit Fletcher is calculated over data octets (bytes) and creates a 16-bit checksum.
- The calculation is done modulo 256 (2^8), which means the intermediate results are divided by 256 and the remainder is kept.
- The algorithm uses two accumulators, L and R.
- The first simply adds data items together;

The second adds a weight to the calculation.

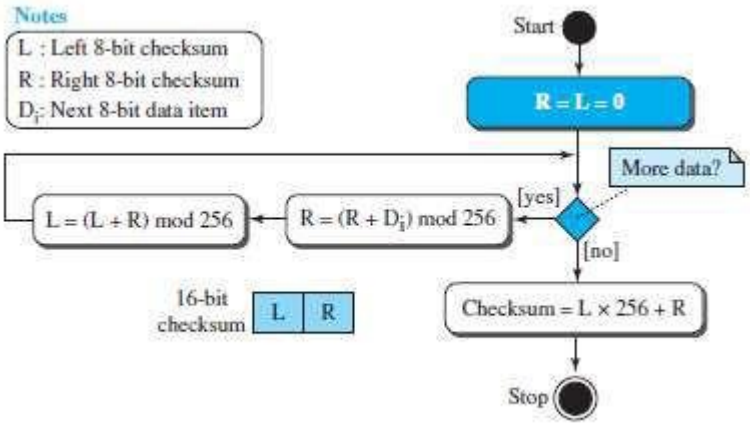


Figure 10.18 *Algorithm to calculate an 8-bit Fletcher checksum*

Adler Checksum

- The Adler checksum is a 32-bit checksum.
- It is similar to the 16-bit Fletcher with three differences (Figure 10.19).
  1) Calculation is done on single bytes instead of 2 bytes at a time.
  2) The modulus is a prime number (65,521) instead of 65,536.
  3) L is initialized to 1 instead of 0.

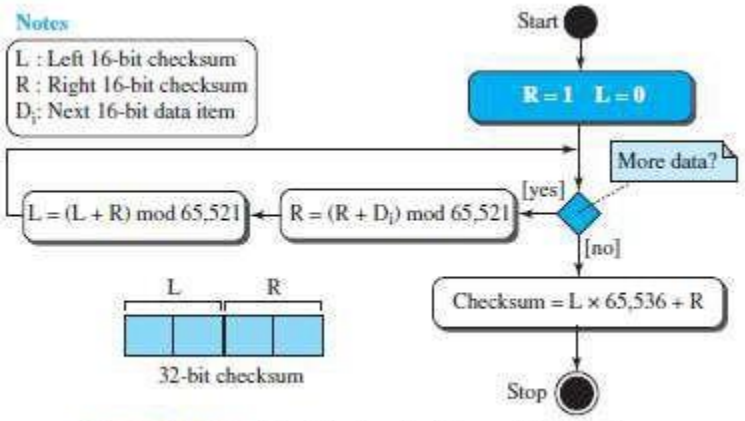


Figure 10.19 *Algorithm to calculate an Adler checksum*

- A prime modulo has a better detecting capability in some combinations of data.