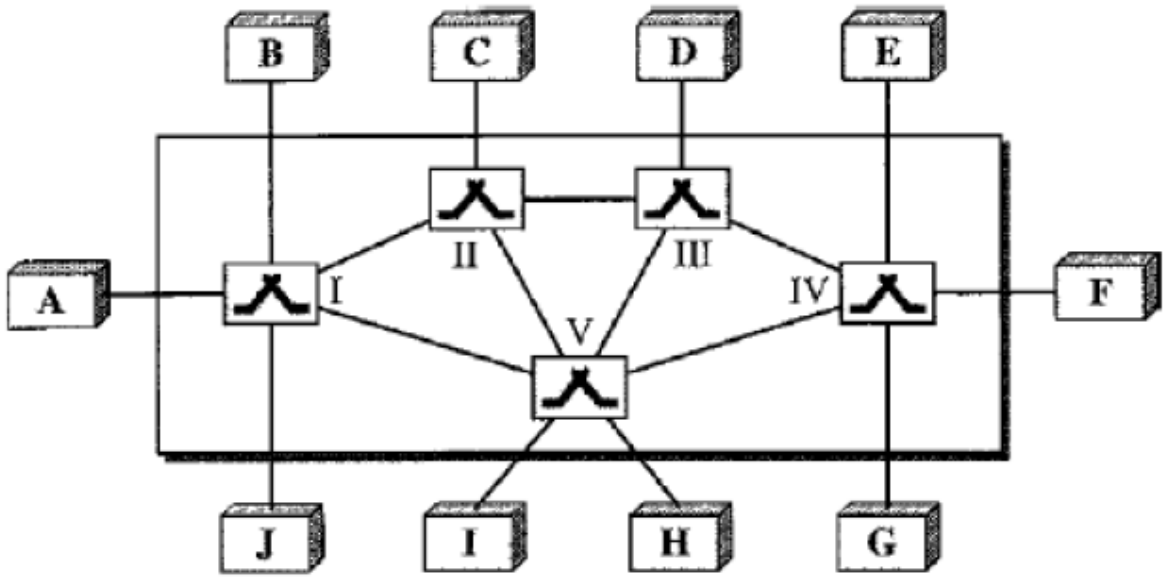
UNIT-II

DATA COMMUNICATIONS AND NETWORKING

Switching

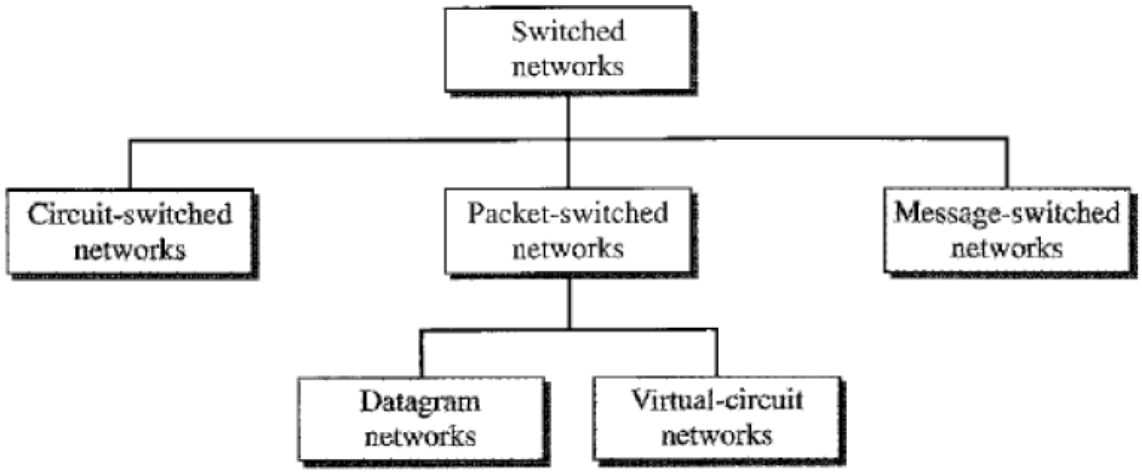
A network is a set of connected devices. Whenever we have multiple devices, we have the problem of how to connect them to make one-to-one communication possible. One solution is to make a point-to-point connection between each pair of devices (a mesh topology) or between a central device and every other device (a star topology). These methods, however, are impractical and wasteful when applied to very large networks. The number and length of the links require too much infrastructure to be cost-efficient, and the majority of those links would be idle most of the time. Other topologies employing multipoint connections, such as a bus, are ruled out because the distances between devices and the total number of devices increase beyond the capacities of the media and equipment.

A better solution is 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 (computers or telephones, for example). Others are used only for routing. Figure shows a switched network.



The end systems (communicating devices) are labeled A, B, C, D, and so on, and the switches are labeled I, II, III, IV, and V. Each switch is connected to multiple links.

Taxonomy of switched networks



2.2.1 CIRCUIT-SWITCHED NETWORKS

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

Figure shows a trivial circuit-switched network with four switches and four links. Each link is divided into *n (n* is 3 in the figure) channels by using FDM or TDM.

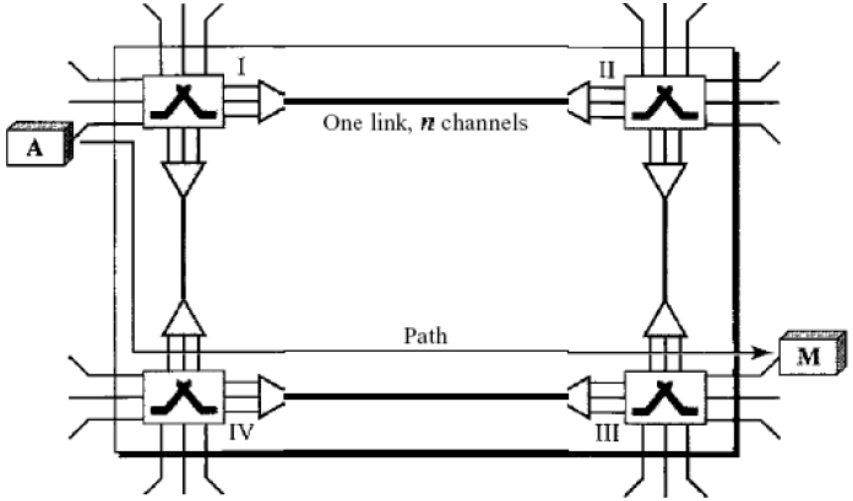


Fig: A trivial circuit-switched network

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. For example, in Figure, when system A needs to connect to system M, it sends a setup request that includes the address of system M, to switch I. Switch I finds a channel between itself and switch IV that can be dedicated for this purpose. Switch I then sends the request to switch IV, which finds a dedicated channel between itself and switch III. Switch III informs system M of system A's intention at this time.

In the next step to making a connection, 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. Note that end-to-end addressing is required for creating a connection between the two end systems. These can be, for example, the addresses of the computers assigned by the administrator in a TDM network, or telephone numbers in an FDM network.

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.

Efficiency:

It can be argued that circuit-switched networks are not as efficient as the other two types of networks because resources are allocated during the entire duration of the connection. These resources are unavailable to other connections. In a telephone network, people normally terminate the communication when they have finished their conversation. However, in computer networks, a computer can be connected to another computer even if there is no activity for a long time. In this case, allowing resources to be dedicated means that other connections are deprived.

Delay :

Although a circuit-switched network normally has low efficiency, the delay in this type of network is minimal. During data transfer the data are not delayed at each switch; the resources are allocated for the duration of the connection. Figure 8.6 shows the idea of delay in a circuit-switched network when only two switches are involved. As Figure shows, there is no waiting time at each switch. The total delay is due to the time needed to create the connection, transfer data, and disconnect the circuit.

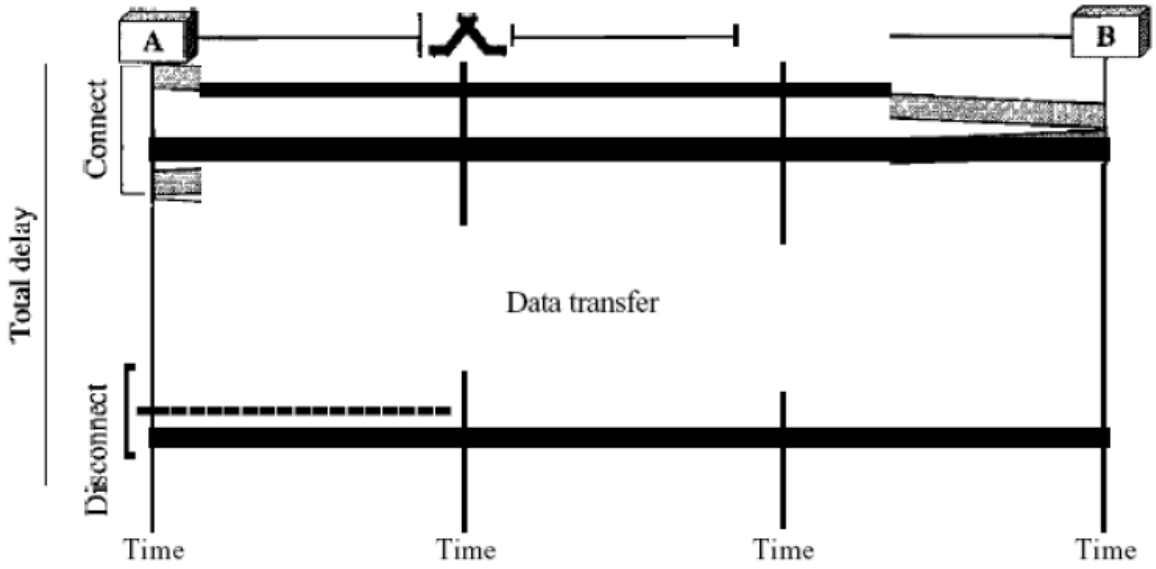


Fig: Delay in a circuit-switched network

The delay caused by the setup is the sum of four parts: the propagation time of the source computer request (slope of the first gray box), the request signal transfer time (height of the first gray box), the propagation time of the acknowledgment from the destination computer (slope of the second gray box), and the signal transfer time of the acknowledgment (height of the second gray box). The delay due to data transfer is the sum of two parts: the propagation time (slope of the colored box) and data transfer time (height of the colored box), which can be very long. The third box shows the time needed to tear down the circuit. We have shown the case in which the receiver requests disconnection, which creates the maximum delay.

2.2.2 DATAGRAM NETWORKS

In a datagram network, each packet is treated independently of all others. Even if a packet is part of a multipacket transmission, the network treats it as though it existed alone. Packets in this approach are referred to as datagrams.

Datagram switching is normally done at the network layer. We briefly discuss datagram networks here as a comparison with circuit-switched and virtual-circuit switched networks

Figure shows how the datagram approach is used to deliver four packets from station A to station X. The switches in a datagram network are traditionally referred to as routers. That is why we use a different symbol for the switches in the figure.

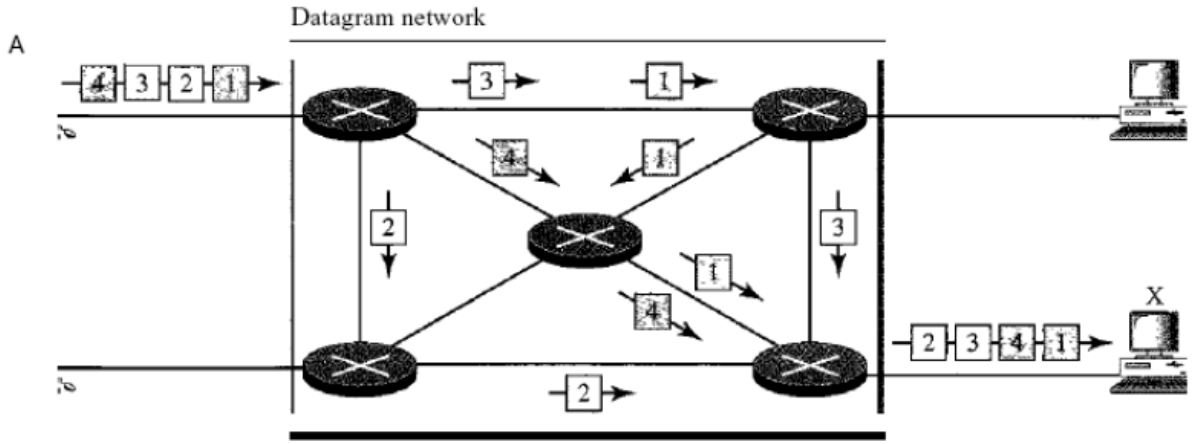


Fig: *A datagram network with four switches (routers)*

In this example, 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. In most protocols, it is the responsibility of an upper-layer protocol to reorder the datagrams or ask for lost datagrams before passing them on to the application.

The datagram networks are sometimes referred to as connectionless networks. The term *connectionless* here means that the switch (packet switch) does not keep information about the connection state. There are no setup or teardown phases. Each packet is treated the same by a switch regardless of its source or destination.

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. Figure shows the routing table for a switch.

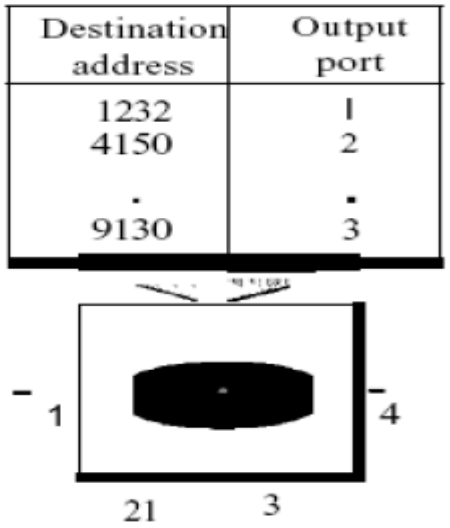


Fig: Routing table in a datagram network

Destination Address

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. In addition, since not all packets in a message necessarily travel through the same switches, the delay is not uniform for the packets of a message.

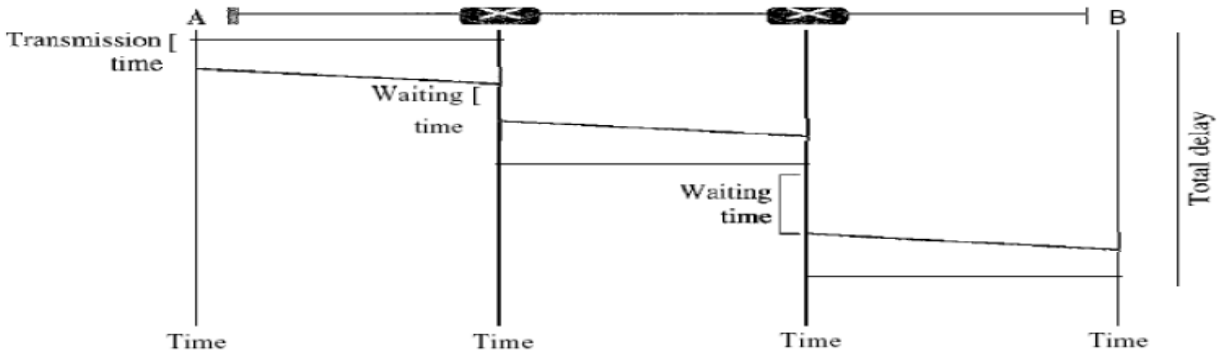


Fig: Delay in a datagram network

The packet travels through two switches. There are three transmission times *(3T)*, three propagation delays (slopes 3't of the lines), and two waiting times (W1 + *w2)'* We ignore the processing time in each switch. The total delay is

Total delay =*3T* + 3t + W1+ W2

2.2.3 VIRTUAL-CIRCUIT NETWORKS:

A virtual-circuit network is a cross between a circuit-switched network and a datagram network. It has some characteristics of both.

1. As in a circuit-switched network, there are setup and teardown phases in addition to the data transfer phase.
2. Resources can be allocated during the setup phase, as in a circuit-switched network, or on demand, as in a datagram network.
3. As in a datagram network, data are packetized and each packet carries an address in the header. However, the address in the header has local jurisdiction (it defines what should be the next switch and the channel on which the packet is being canied), not end-to-end jurisdiction. The reader may ask how the intermediate switches know where to send the packet if there is no final destination address carried by a packet. The answer will be clear when we discuss virtual-circuit identifiers in the next section.
4. As in a circuit-switched network, all packets follow the same path established during the connection.
5. A virtual-circuit network is normally implemented in the data link layer, while a circuit-switched network is implemented in the physical layer and a datagram network in the network layer. But this may change in the future. Figure 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.

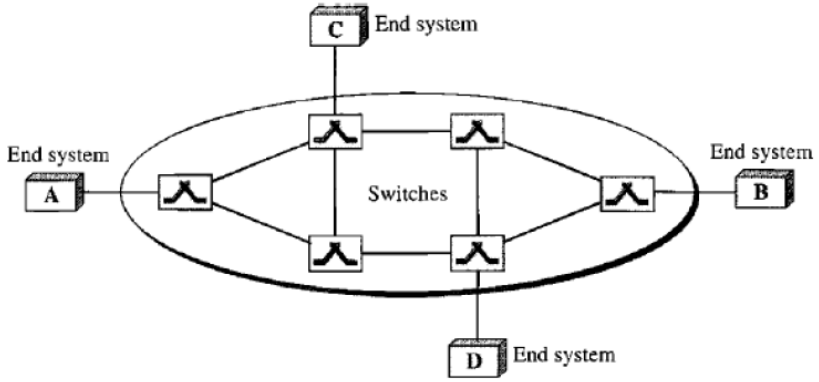


Fig a: *Virtual-circuit network*

Addressing

In a virtual-circuit network, two types of addressing are involved: global and local (virtual-circuit identifier).

Global Addressing: A source or a destination needs to have a global address-an address that can be unique in the scope of the network or internationally if the network is part of an international network. However, we will see that a global address in virtual-circuit networks is used only to create a virtual-circuit identifier, as discussed next.

Virtual-Circuit Identifier: The identifier that is actually used for data transfer is called the virtual-circuit identifier (Vel). A vel, unlike a global address, is a small number that has only switch scope; it is used by a frame between two switches. When a frame arrives at a switch, it has a VCI; when it leaves, it has a different VCl. Figure 8.11 shows how the VCI in a data frame changes from one switch to another. Note that a VCI does not need to be a large number since each switch can use its own unique set of VCls.

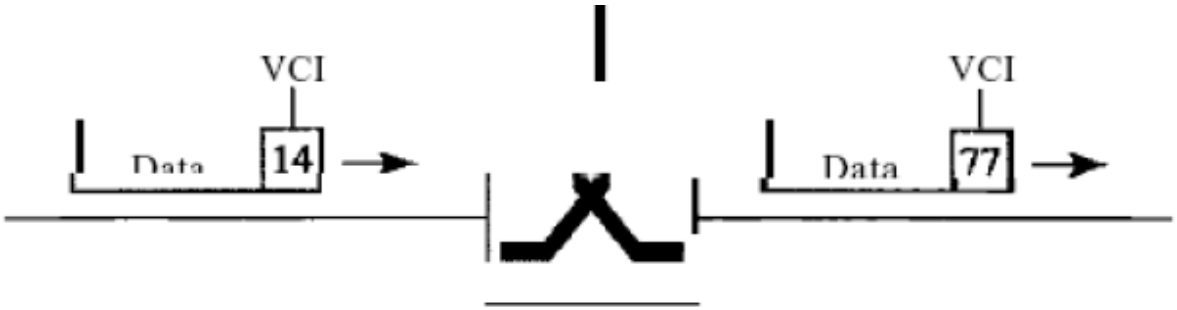


Figure 1 *Virtual-circuit identifier*

Three Phases

As in a circuit-switched network, a source and destination need to go through three phases in a virtual-circuit network: setup, data transfer, and teardown. In the setup phase, the source and destination use their global addresses to help switches make table entries for the connection. In the teardown phase, the source and destination inform the switches to delete the corresponding entry. Data transfer occurs between these two phases. We first discuss the data transfer phase, which is more straightforward; we then talk about the setup and teardown 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, in its simplest form, has four columns. This means that the switch holds four pieces of information for each virtual circuit that is already set up. We show later how the switches make their table entries, but for the moment we assume that each switch has a table with entries for all active virtual circuits. Figure 2 shows such a switch and its corresponding table. And also shows a frame arriving 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 and send out the frame from port 3. Figure 3 shows how a frame from source A reaches destination B and how its VCI changes during the trip. 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

In the 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: the setup request and the acknowledgment.

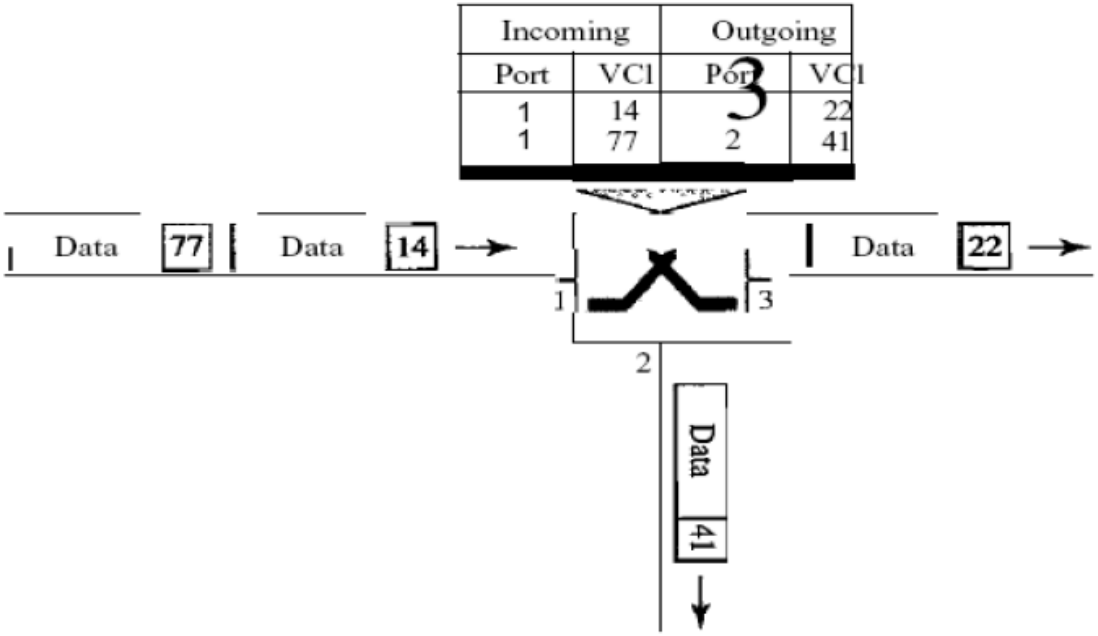


Figure 2 *Switch and tables in a virtual-circuit network*

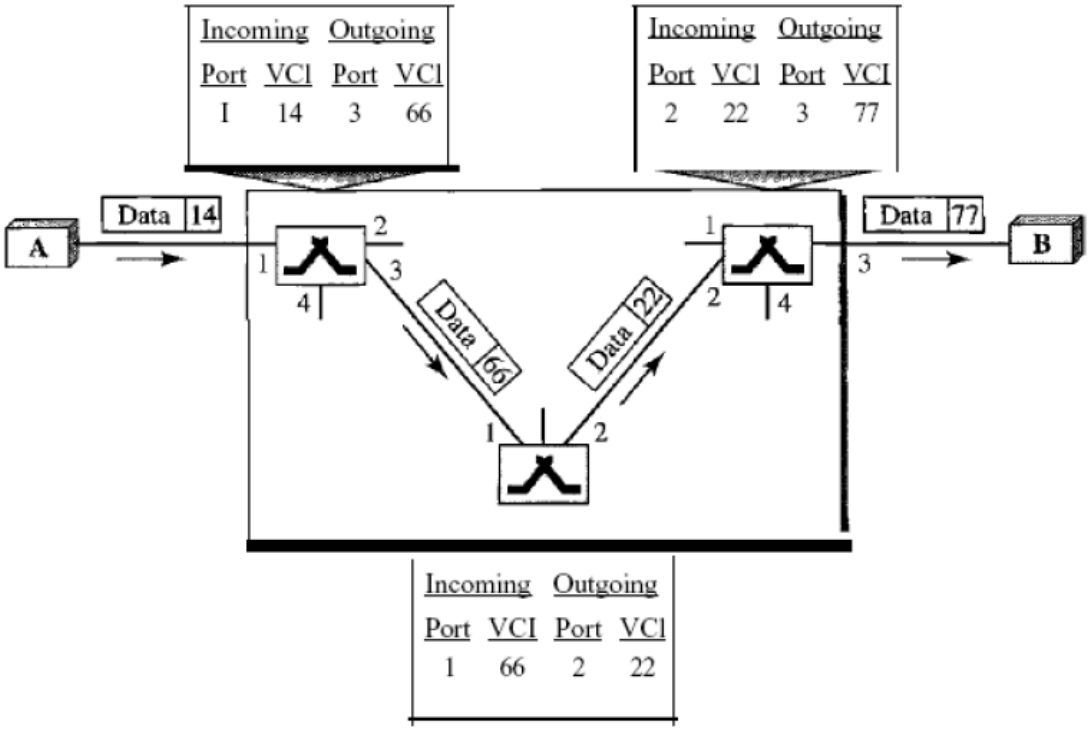


Figure 3 *Source-to-destination data transfer in a virtual-circuit network*

Setup Request

A setup request frame is sent from the source to the destination.

Figure 4 shows the process.

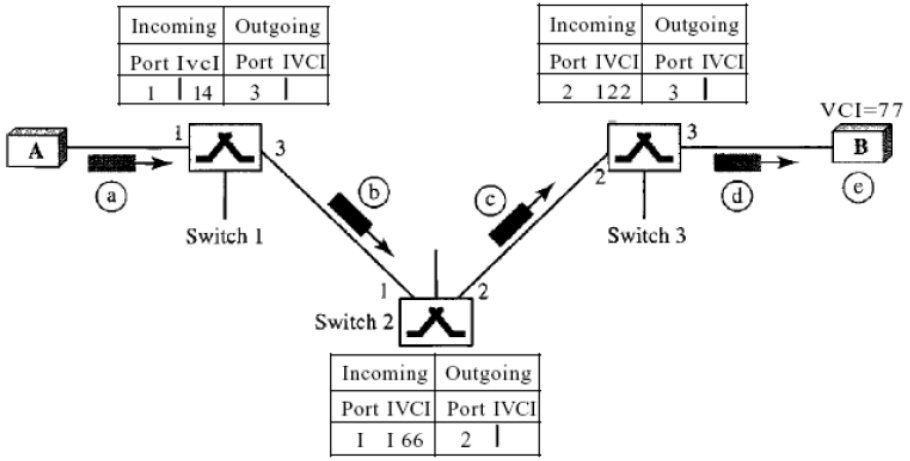


Figure 4 *Setup request in a virtual-circuit network*

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

b. Switch 1 receives the setup request frame. It knows that a frame going from A to B goes out through port 3. How the switch has obtained this information is a point covered in future chapters. The switch, in the setup phase, acts as a packet switch; it has a routing table which is different from the switching table. For the moment, assume that it knows the output port. The switch creates an entry in its table for this virtual circuit, but it is only able to fill three of the four columns. The switch assigns the incoming port (1) and chooses an available incoming VCI (14) and the outgoing port (3). It 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 (l), 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, and if it is ready to receive frames from A, it 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 not other sources.

Acknowledgment A special frame, called the acknowledgment frame, completes the entries in the switching tables. Figure 8.15 shows the process.

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. Note that 77 is the incoming VCI for destination B, but the outgoing VCI for switch 3.

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.

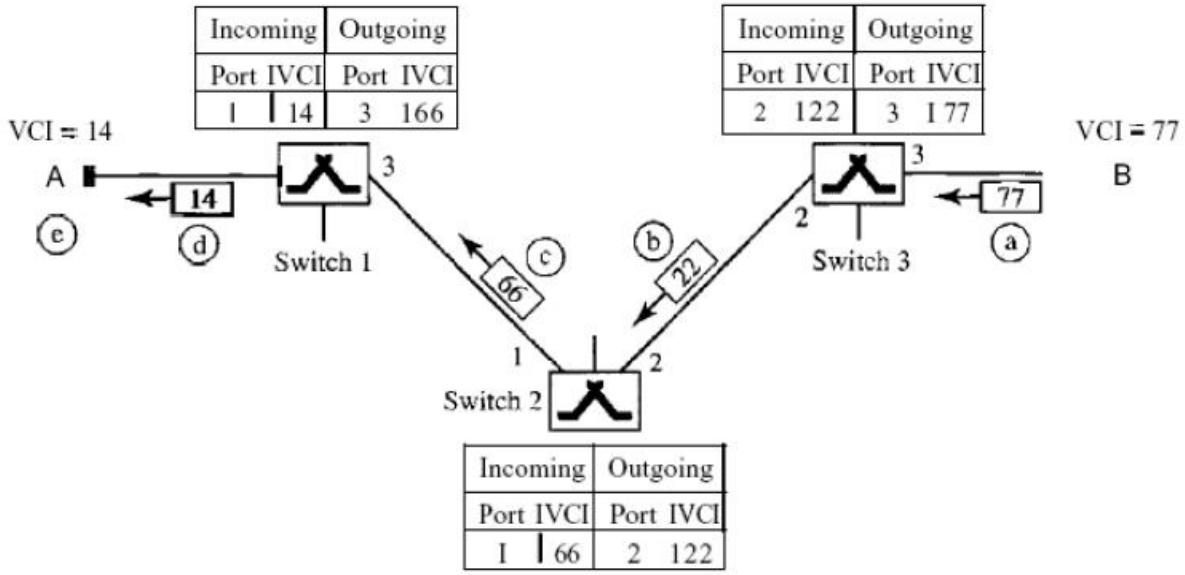


Figure 5 *Setup acknowledgments in a virtual-circuit network*

Teardown Phase

In this 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

As we said before, resource reservation in a virtual-circuit network can be made during the setup or can be on demand during the data transfer phase. In the first case, the delay for each packet is the same; in the second case, each packet may encounter different delays. There is one big advantage in a virtual-circuit network even if resource allocation is on demand. The source can check the availability of the resources, without actually reserving it. Consider a family that wants to dine at a restaurant. Although the restaurant may not accept reservations (allocation of the tables is on demand), the family can call and find out the waiting time. This can save the family time and effort.

Delay in Virtual-Circuit Networks

In a virtual-circuit network, there is a one-time delay for setup and a one-time delay for teardown. If resources are allocated during the setup phase, there is no wait time for individual packets. Below Figure shows the delay for a packet traveling through two switches in a virtual-circuit network

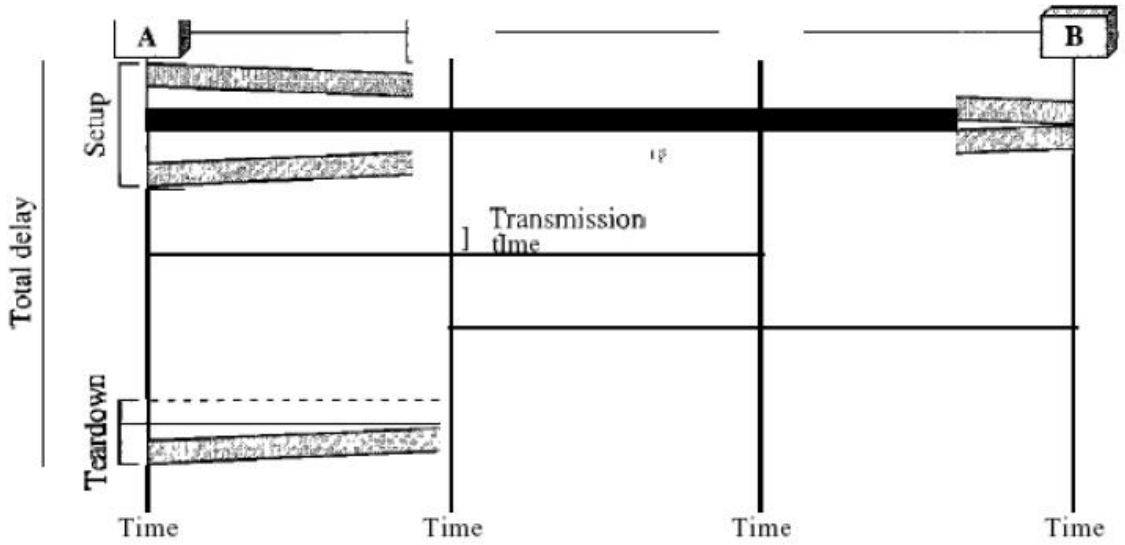


Fig: Delay in a virtual-circuit network

The packet is traveling through two switches (routers). There are three transmission times *(3T)*, three propagation times (3't), data transfer depicted by the sloping lines, a setup delay (which includes transmission and propagation in two directions), and a teardown delay (which includes transmission and propagation in one direction). We ignore the processing time in each switch. The total delay time is

Total delay = *3T*+ 3't + setup delay + teardown delay

HDLC Protocol - High-level Data Link Control

HDLC - Short for High-level Data Link Control, a transmission protocol used at the data link layer (layer 2) of the OSI seven layer model for data communications. The HDLC protocol embeds information in a data frame that allows devices to control data flow and correct errors. HDLC is an ISO standard developed from the Synchronous Data Link Control (SDLC) standard proposed by IBM in the 1970's. HDLC NRM (also known as SDLC)

HDLC is a bit oriented protocol that supports both half-duplex and full-duplex communication over point to point & multipoint link.

For any **HDLC communications session**, one station is designated primary and the other secondary. A session can use one of the following connection modes, which determine how the primary and secondary stations interact.

• **Normal unbalanced**: The secondary station responds only to the primary station.

• **Asynchronous**: The secondary station can initiate a message

• **Asynchronous balanced**: Both stations send and receive over its part of a duplex line.

This mode is used for X.25 packet-switching networks.

The Link Access Procedure-Balanced (LAP-B) and Link Access Procedure D-channel (LAP-D) protocols are subsets of HDLC.

LAPB is a bit-oriented synchronous protocol that provides complete data transparency in a full-duplex point-to-point operation. It supports a peer-to-peer link in that neither end of the link plays the role of the permanent master station. HDLC NRM, on the other hand, has a permanent primary station with one or more secondary stations.

HDLC LAPB is a very efficient protocol, which requires a minimum of overhead to ensure flow control, error detection and recovery. If data is flowing in both directions (full duplex), the data frames themselves carry all the information required to ensure data integrity.

The concept of a frame window is used to send multiple frames before receiving confirmation that the first frame has been correctly been received. This means that data can continue to flow in situations where there may be long "turn-around" time lags without stopping to wait for an acknowledgement. This kind of situation occurs, for instance in satellite communication.

Types of Frames in HDLC

HDLC defines three types of frames:

1. Information frames (I-frame)

2. Supervisory frame (S-frame)

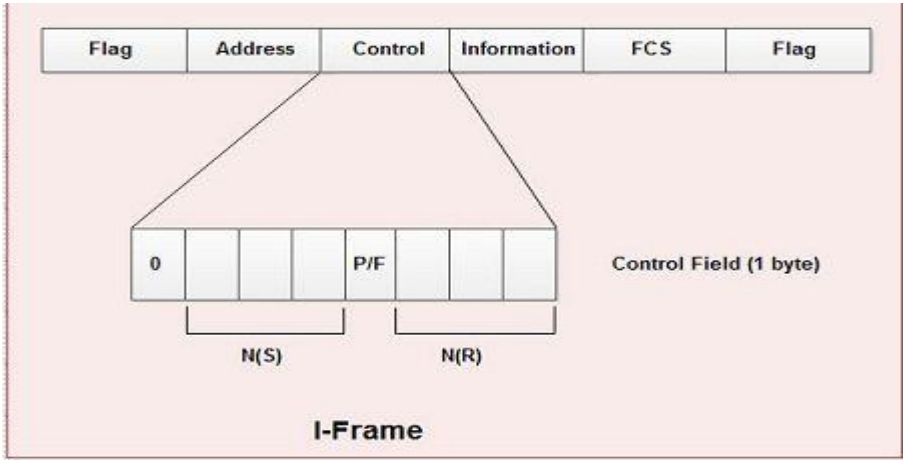
3. Unnumbered frame (U-frame)

1. Information frames

• I-frames carry user's data and control information about user's data.

• I-frame carries user data in the information field.

• The I-frame format is shown in diagram.



• The first bit of control field is always zero, *i.e.* the presence of zero at this place indicates that it is I-frame.

• Bit number 2, 3 & 4 in control field is called N(S) that specifies the sequence number of the frame. Thus it specifies the number of the frame that is currently being sent. Since it is a 3.bit field, only eight sequence numbers are possible 0, 1,2,3,4,5,6, 7 (000 to 111).

• Bit number 5 in control field is P/F i.e. Poll/Final and is used for these two purposes. It has, meaning only when it is set i.e. when P/F=1. It can represent the following two cases.

(i) It means poll when frame is sent by a primary station to secondary (when address field contains the address of receiver).

(ii) It means final when frame is sent by secondary to a primary (when the address field contains the address of the sender).

• Bit number 6, 7, and 8 in control field specifies N(R) i.e. the sequence number of the frame expected in return in two-way communication.

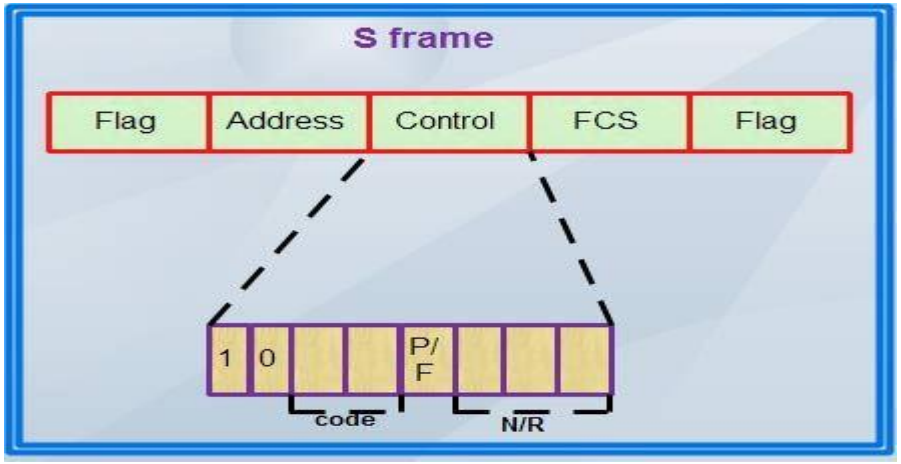
If last frame received was error-free then N(R) number will be that of the next frame is sequence. If the last frame was not received correctly, the N(R) number will be the number of the damaged frame, asking for its retransmission.

2. Supervisory frame

• S-frame carries control information, primarily data link layer flow and error controls.

• It does not contain information field.

• The format of S-frame is shown in diagram.



• The first two bits in the control field of S-frame are always 10.

• Then there is a bit code field that specifies four types of S-frame with combination 00,01, 10, 11 as shown in table :-

Table: Types of S-frame

Code	Command
00	RR Receive Ready
01	REJ Reject
10	RNR Receive Not Ready
11	SREJ Selective Reject

1. RR, Receive Ready-used to acknowledge frames when no I-frames are availab1e to piggyback the acknowledgement.
2. REJ Reject-used by the receiver to send a NAK when error has occurred.
3. RNR Receive Not Ready-used for flow control.
4. SREJ Selective Reject-indicates to the transmitter that it should retransmit the frame indicated in the N(R) subfield.

• There is no N(S) field in control field of S-frame as S-frames do not transmit data.

P/F bit is the fifth bit and serves the same purpose as discussed earlier.

• Last three bits in control field indicates N(R) *i.e.* they correspond to the ACK or NAK value.

3. Unnumbered frame

• U-frames are reserved for system management and information carried by them is used for managing the link

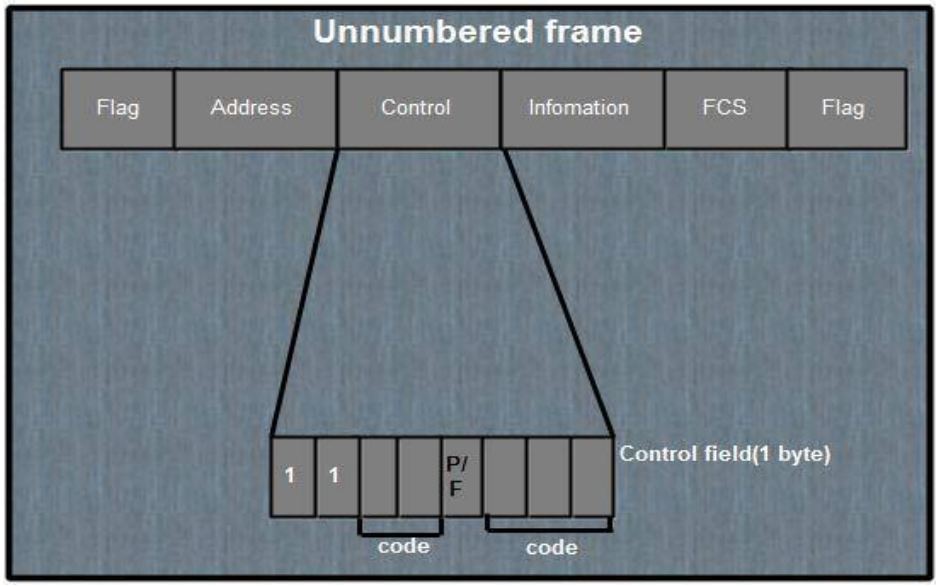
• U-frames are used to exchange session management and control information between the two connected devices.

• Information field in U-frame does not carry user information rather, it carries system management information.

• The frame format of U-frame is shown in diagram.

• U-frame is identified by the presence of 11 in the first and second bit position in control field.

• These frames do not contain N(S) or N(R) in control field.



• U-frame contains two code fields, one two hit and other three bit.

• These five bits can create upto 32 different U-frames.

• .*P/F* bit in control field has same purpose in V-frame as discussed earlier.

Protocol Structure - HDLC: High Level Data Link Control

Flag - The value of the flag is always (0x7E).

Address field - Defines the address of the secondary station which is sending the frame or the destination of the frame sent by the primary station. It contains Service Access Point (6bits), a Command/Response bit to indicate whether the frame relates to information frames (I-frames) being sent from the node or received by the node, and an address extension bit which is usually set to true to indicate that the address is of length one byte. When set to false it indicates an additional byte follows.

Extended address - HDLC provides another type of extension to the basic format. The address field may be extended to more than one byte by agreement between the involved parties.

Control field - Serves to identify the type of the frame. In addition, it includes sequence numbers, control features and error tracking according to the frame type.

FCS - The Frame Check Sequence (FCS) enables a high level of physical error control by allowing the integrity of the transmitted frame data to be checked.

Point-to-Point Protocol (PPP)

PPP was devised by IETF (Internet Engineering Task Force) to create a data link protocol for point to point lines that can solve all the problems present in SLIP.

• PPP is most commonly used data link protocol. It is used to connect the Home PC to the server of ISP via a modem.

• This protocol offers several facilities that were not present in SLIP. Some of these facilities are:

1. PPP defines the format of the frame to be exchanged between the devices.

2. It defines link control protocol (LCP) for:-

(a) Establishing the link between two devices.

(b) Maintaining this established link.

(c) Configuring this link.

(d) Terminating this link after the transfer.

3. It defines how network layer data are encapsulated in data link frame.

4. PPP provides error detection.

5. Unlike SLIP that supports only IP, PPP supports multiple protocols.

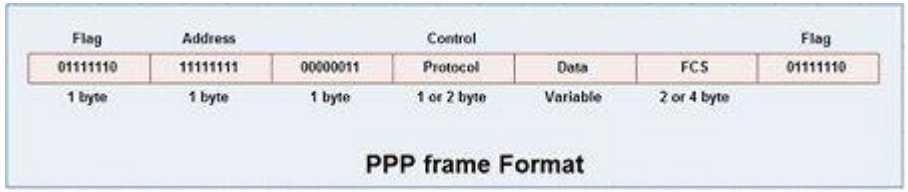
6. PPP allows the IP address to be assigned at the connection time i.e. dynamically. Thus a temporary IP address can be assigned to each host.

7. PPP provides multiple network layer services supporting a variety of network layer protocol. For this PPP uses a protocol called NCP (Network Control Protocol).

8. It also defines how two devices can authenticate each other.

PPP Frame Format

The frame format of PPP resembles HDLC frame. Its various fields are:



1. **Flag field**: Flag field marks the beginning and end of the PPP frame. Flag byte is 01111110. (1 byte).

2. **Address field**: This field is of 1 byte and is always 11111111. This address is the broadcast address *i.e.* all the stations accept this frame.

3. **Control field**: This field is also of 1 byte. This field uses the format of the U-frame (unnumbered) in HDLC. The value is always 00000011 to show that the frame does not contain any sequence numbers and there is no flow control or error control.

4. **Protocol field**: This field specifies the kind of packet in the data field *i.e.* what is being carried in data field.

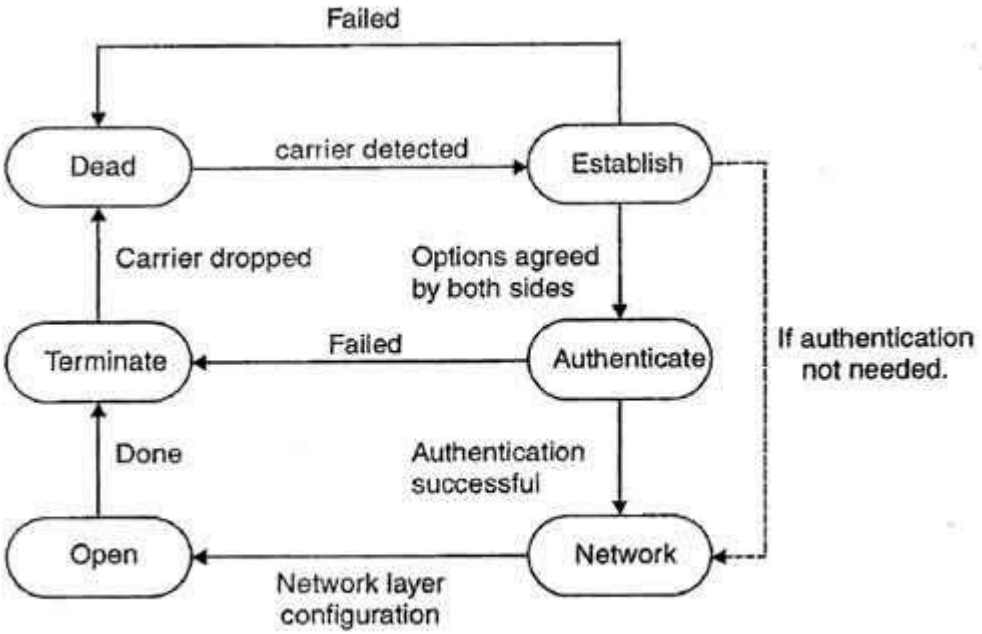
5. **Data field**: Its length is variable. If the length is not negotiated using LCP during line set up, a default length of 1500 bytes is used. It carries user data or other information.

6. **FCS field**: The frame checks sequence. It is either of 2 bytes or 4 bytes. It contains the checksum.

Transition Phases in PPP

The PPP connection goes through different states as shown in fig.

1. **Dead**: In dead phase the link is not used. There is no active carrier and the line is quiet.



Transition phases

2. **Establish**: Connection goes into this phase when one of the nodes start communication. In this phase, two parties negotiate the options. If negotiation is successful, the system goes into authentication phase or directly to networking phase. LCP packets are used for this purpose.

3. **Authenticate**: This phase is optional. The two nodes may decide during the establishment phase, not to skip this phase. However if they decide to proceed with authentication, they send several authentication packets. If the result is successful, the connection goes to the networking phase; otherwise, it goes to the termination phase.

4. **Network**: In network phase, negotiation for the network layer protocols takes place. PPP specifies that two nodes establish a network layer agreement before data at the network layer can be exchanged. This is because PPP supports several protocols at network layer. If a node is running multiple protocols simultaneously at the network layer, the receiving node needs to know which protocol will receive the data.

5. **Open**: In this phase, data transfer takes place. The connection remains in this phase until one of the endpoints wants to end the connection.

6. **Terminate**: In this phase connection is terminated.

Point-to-point protocol Stack

PPP uses several other protocols to establish link, authenticate users and to carry the network layer data.

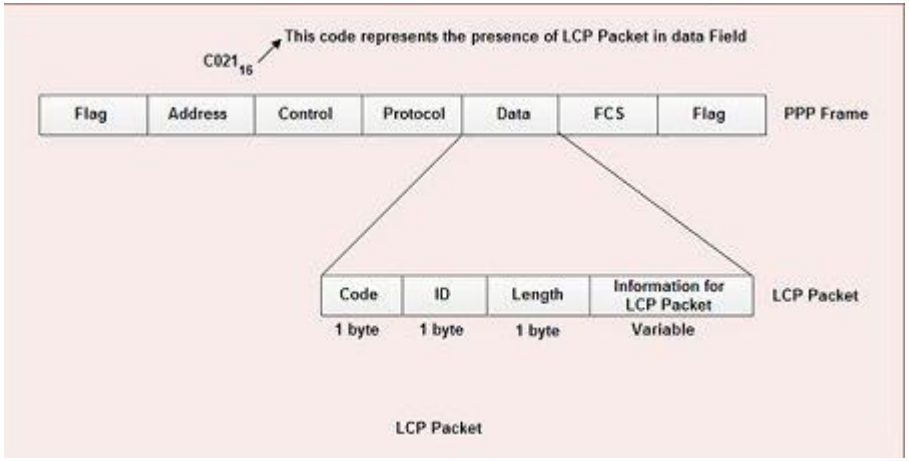
The various protocols used are:

1. Link Control Protocol
2. Authentication Protocol
3. Network Control Protocol

1. Link Control Protocol

• It is responsible for establishing, maintaining, configuring and terminating the link.

• It provides negotiation mechanism to set options between two endpoints.



• All LCP packets are carried in the data field of the PPP frame.

• The presence of a value $C021_{16}$ in the protocol field of PPP frame indicates that LCP packet is present in the data field.

• The various fields present in LCP packet are:

1. **Code**: 1 byte-specifies the type of LCP packet.
2. I**D**: 1 byte-holds a value used to match a request with the reply.
3. **Length**: 2 byte-specifies the length of entire LCP packet.

4. **Information**: Contains extra information required for some LCP packet.

• There are eleven different type of LCP packets. These are categorized in three groups:

1. **Configuration packet**: These are used to negotiate options between the two ends. For example: configure-request, configure-ack, configure-nak, configure-reject are some configuration packets.

2. **Link termination packets**: These are used to disconnect the link between two end points. For example: terminate-request, terminate-ack, are some link termination packets.

3. **Link monitoring and debugging packets**: These are used to monitor and debug the links. For example: code-reject, protocol-reject, echo-request, echo-reply and discard-request are some link monitoring and debugging packets.

2. Authentication Protocol

Authentication protocols help to validate the identity of a user who needs to access the resources.

There are two authentication protocols:

1. Password Authentication Protocols (PAP)

2. Challenge Handshake Authentication Protocol (CHAP)

1. PAP (Password Authentication Protocol)

This protocol provides two step authentication procedures:

Step 1: User name and password is provided by the user who wants to access a system.

Step 2: The system checks the validity of user name and password and either accepts or denies the connection.

• PAP packets are also carried in the data field of PPP frames.

• The presence of PAP packet is identified by the value $C023_{16}$ in the protocol field of PPP frame.

• There are three PAP packets.

1. **Authenticate-request**: used to send user name & password.

2. **Authenticate-ack**: used by system to allow the access.

3. **Authenticate-nak**: used by system to deny the access.

2. CHAP (Challenge Handshake Authentication Protocol)

• It provides more security than PAP.

• In this method, password is kept secret, it is never sent on-line.

• It is a three-way handshaking authentication protocol:

1. System sends. a challenge packet to the user. This packet contains a value, usually a few bytes.

2. Using a predefined function, a user combines this challenge value with the user password and sends the resultant packet back to the system.

3. System then applies the same function to the password of the user and challenge value and creates a result. If result is same as the result sent in the response packet, access is granted, otherwise, it is denied.

• There are 4 types of CHAP packets:

1. Challenge-used by system to send challenge value.

2. Response-used by the user to return the result of the calculation.

3. Success-used by system to allow access to the system.

4. Failure-used by the system to deny access to the system.

3. Network Control Protocol (NCP)

• After establishing the link and authenticating the user, PPP connects to the network layer. This connection is established by NCP.

• Therefore NCP is a set of control protocols that allow the encapsulation of the data coming from network layer.

• After the network layer configuration is done by one of the NCP protocols, the users can exchange data from the network layer.

• PPP can carry a network layer data packet from protocols defined by the Internet, DECNET, Apple Talk, Novell, OSI, Xerox and so on.

• None of the NCP packets carry networks layer data. They just configure the link at the network layer for the incoming data.

ERROR DETECTION AND CORRECTION

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

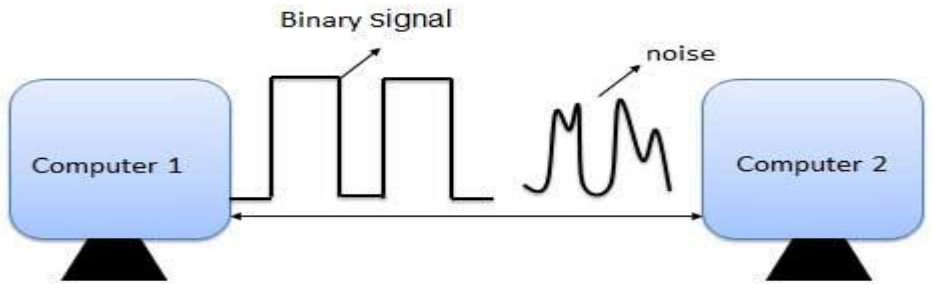
Topics discussed in this section:

1.Types of Errors
2.Redundancy
3.Detection Versus Correction
4.Forward Error Correction Versus Retransmission
5.Coding

6.Modular Arithmetic

ERROR

Error is a condition when the output information does not match with the input information. During transmission, digital signals suffer from noise that can introduce errors in the binary bits travelling from one system to other. That means a 0 bit may change to 1 or a 1 bit may change to 0.



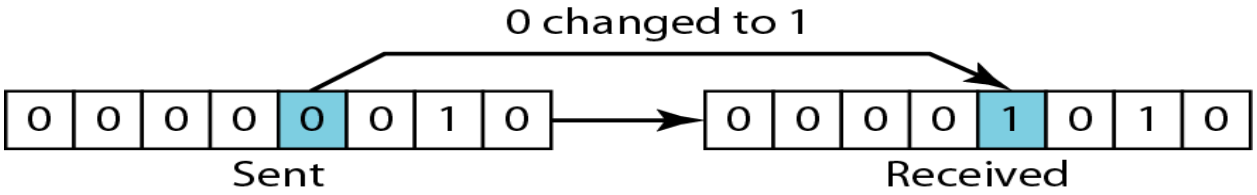
TYPES OF ERRORS

1.Single bit error

2.Burst error

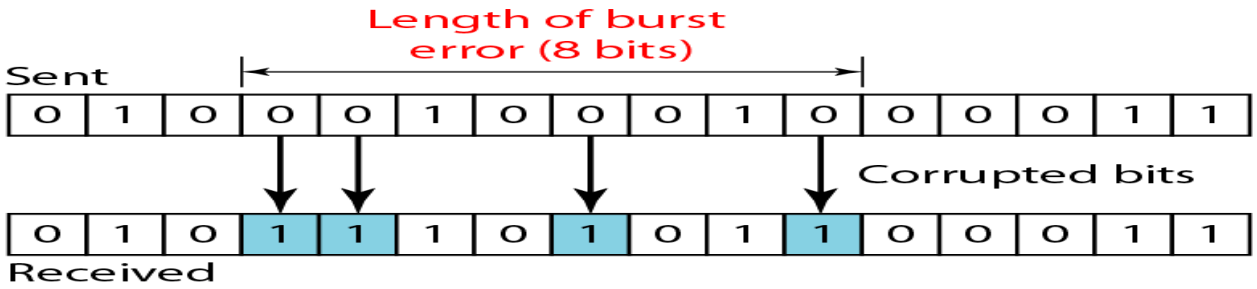
Single bit error

In a single-bit error, only 1 bit in the data unit has changed



Burst error

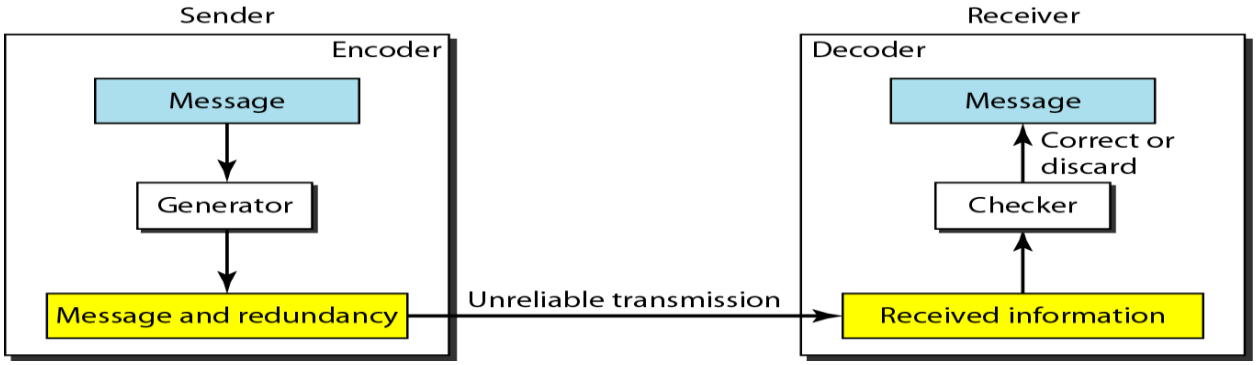
A burst error means that 2 or more bits in the data unit have changed.



Redundancy:

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

The structure of encoder and decoder



Error-Detecting codes

Whenever a message is transmitted, it may get scrambled by noise or data may get corrupted. To avoid this, we use error-detecting codes which are additional data added to a given digital message to help us detect if an error occurred during transmission of the message. A simple example of error-detecting code is **parity check**.

Error-Correcting codes

Along with error-detecting code, we can also pass some data to figure out the original message from the corrupt message that we received. This type of code is called an error-correcting code. Error-correcting codes also deploy the same strategy as error-detecting codes but additionally, such codes also detect the exact location of the corrupt bit.

In error-correcting codes, parity check has a simple way to detect errors along with a sophisticated mechanism to determine the corrupt bit location. Once the corrupt bit is located, its value is reverted (from 0 to 1 or 1 to 0) to get the original message.

How to Detect and Correct Errors?

To detect and correct the errors, additional bits are added to the data bits at the time of transmission.

- The additional bits are called **parity bits**. They allow detection or correction of the errors.
- The data bits along with the parity bits form a **code word**.

Forward error correction versus retransmission:

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

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.

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 actual data bits .The receiver checks the relationship between the two sets of bits to detect or correct errors.The ratio of redundant bits to the data bits and robustness of the process are important factors in any coding scheme.

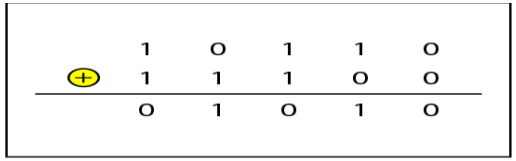
Modular arithmetic

In modulo-N arithmetic, we use only the integers in the range 0 to N −1, inclusive

XORing of two single bits or two words

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

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

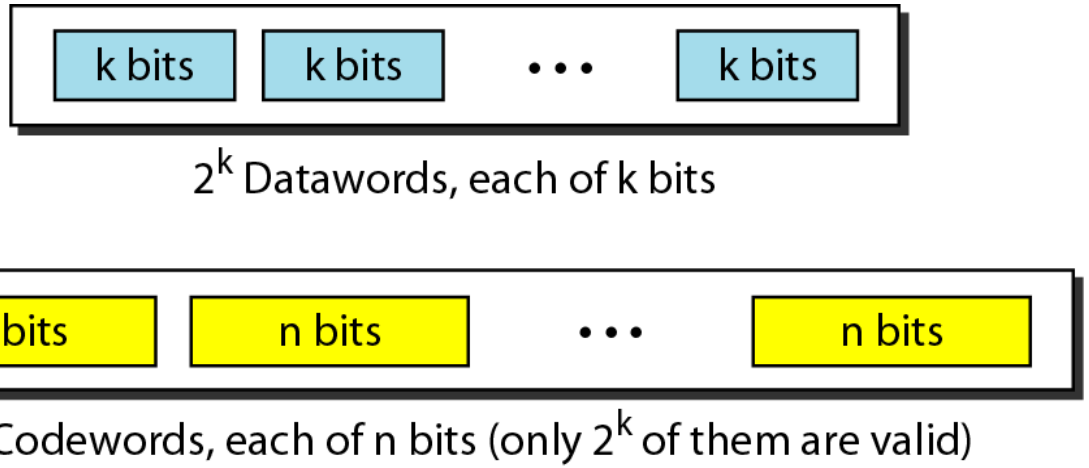


c. Result of XORing two patterns

BLOCK CODING

In block coding, we divide our message into blocks, each of k bits, called datawords. We add r redundant bits to each block to make the length n = k + r. The resulting n-bit blocks are called codewords.

Datawords and codewords in block coding



Process of error detection in block coding

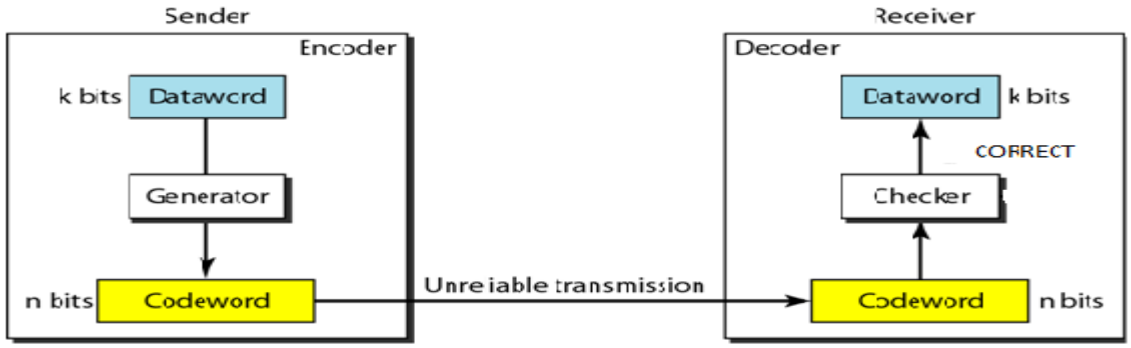
Errors can be detected by using block coding, if the following two conditions are met, the receiver can detect a change in the original codeword.

1.The receiver has(or can find) a list of valid code words.

2.The original codeword has changed to an invalid one.

Process of error correction in block coding

In error correction the receiver to find (guess)the original codeword sent.



EXAMPLE :

Let us assume that k = 2 and n = 3. Table 10.1 shows the list of datawords and codewords. Later, we will see how to derive a codeword from a dataword. Assume the sender encodes the dataword 01 as 011 andsends 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, and111 is received. This is not a valid codeword and is discarded.

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

A code for error detection example

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

HAMMING DISTANCE

The Hamming distance between two strings of equal length is the number of positions at which the corresponding character are different. The Hamming distance between two words x and y as d(x,y).

The hamming distance can be easily found if we apply the XOR operation on the two words and count the number of ones in the result.

EXAMPLE

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

1. The Hamming distance d(000, 011) is 2 because

$$000 \oplus 011 \text{ is } 011 \text{ (two 1s)}$$

2. The Hamming distance d(10101, 11110) is 3 because

$$10101 \oplus 11110 \text{ is } 01011 \text{ (three 1s)}$$

Minimum Hamming distance

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

EXAMPLE: Find the minimum Hamming distance of the coding scheme in Table 10.1.

Solution

We first find all Hamming distances.

$$d(000, 011) = 2 \quad d(000, 101) = 2 \quad d(000, 110) = 2 \quad d(011, 101) = 2$$
$$d(011, 110) = 2 \quad d(101, 110) = 2$$

The d_{min} in this case is 2.

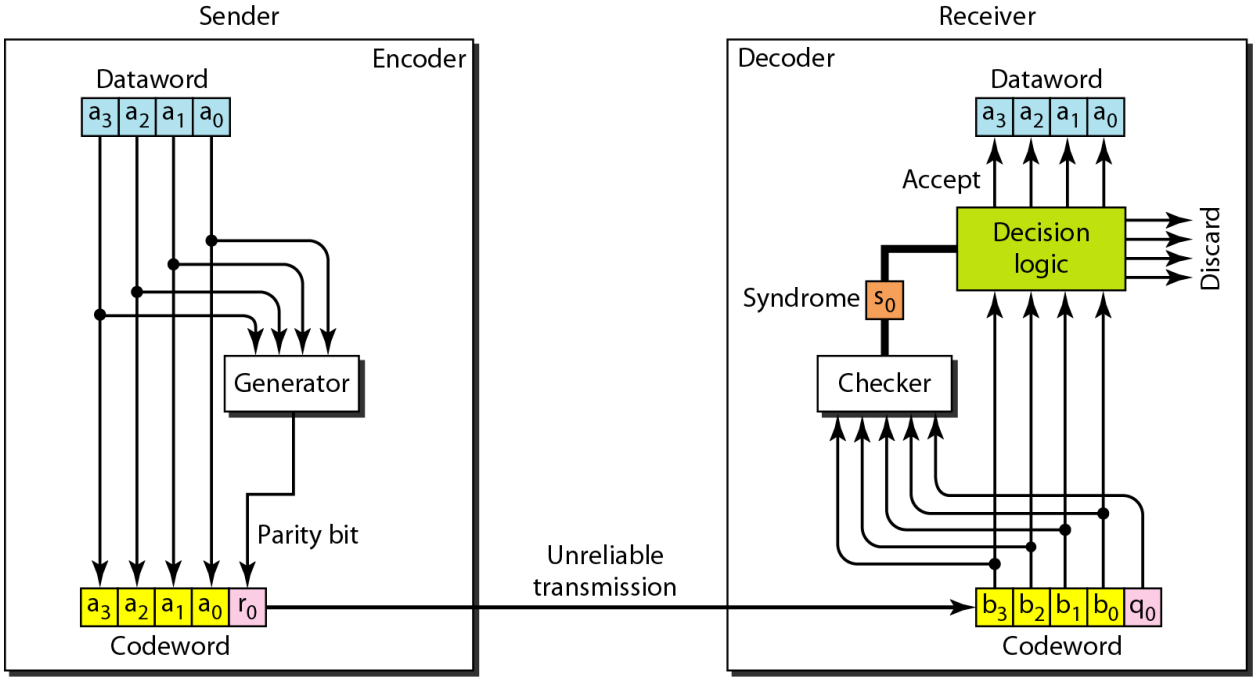
simple parity-check code

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

EXAMPLE: Simple parity-check code C(5, 4)

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

Encoder and decoder for simple parity-check code



EXAMPLE:

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 receive codeword is 10011. The syndrome is 1. No dataword is created.

3. One single-bit error changes r_0 . The received codewo is 10110. The syndrome is 1. No dataword is created.

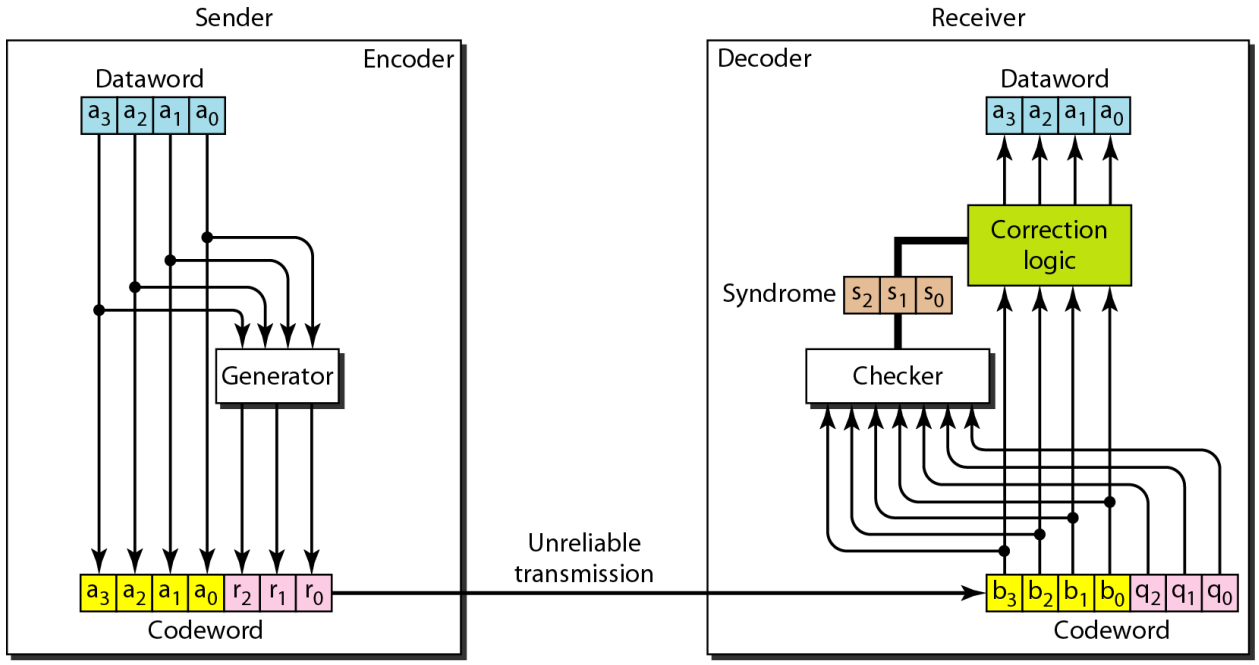
4.. An error changes r_0 and a second error changes a_3 .The received codeword is 00110. The syndrome is 0. The dataword 0011 is created at the receiver. Note that here the dataword is wrongly created due to the syndrome value.

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.

Hamming code C(7, 4)

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	0000000	1000	1000110
0001	0001101	1001	1001011
0010	0010111	1010	1010001
0011	0011010	1011	1011100
0100	0100011	1100	1100101
0101	0101110	1101	1101000
0110	0110100	1110	1110010
0111	0111001	1111	1111111

The structure of the encoder and decoder for a Hamming code



Logical decision made by the correction logic analyzer

Syndrome	000	001	010	011	100	101	110	111
Error	None	q_0	q_1	b_2	q_2	b_0	b_3	b_1

EXAMPLE:

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

1. The dataword 0100 becomes the codeword 0100011. The codeword 0100011 is received. The syndrome is 000, the final dataword is 0100.

2. The dataword 0111 becomes the codeword 0111001. The syndrome is 011. After flipping b_2 (changing the 1 to 0), the final dataword is 0111.

3. The dataword 1101 becomes the codeword 1101000.The syndrome is 101. After flipping b_0, we get 0000, the wrong dataword. This shows that our code cannot correct two errors.

EXAMPLE

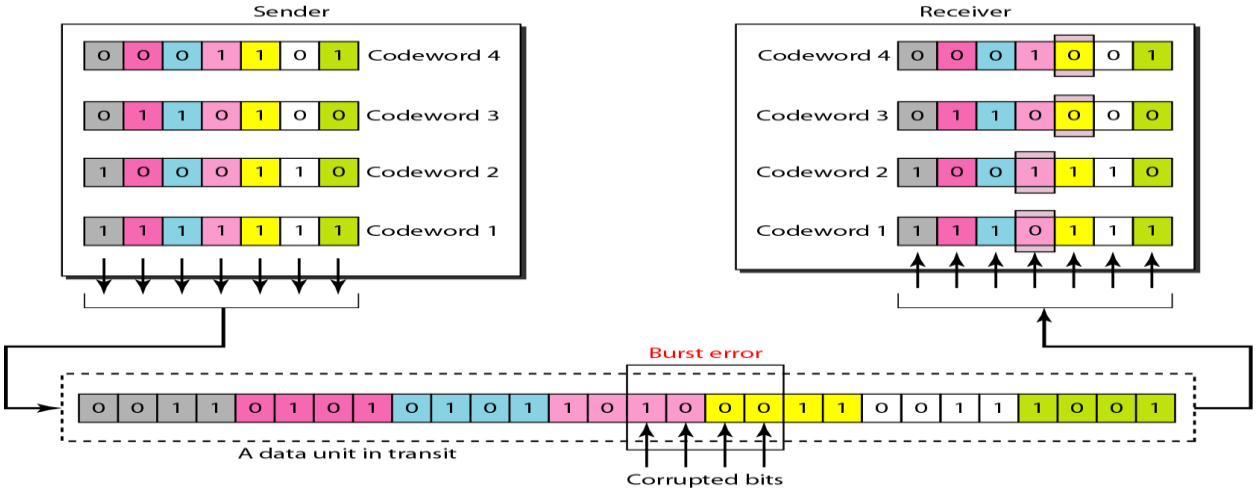
We need a dataword of at least 7 bits. Calculate values of k and n that satisfy this requirement.

We need to make $k = n - m$ greater than or equal to 7, or $2m - 1 - m \geq 7$.

1. If we set $m = 3$, the result is $n = 23 - 1$ and $k = 7 - 3$, or 4, which is not acceptable.

2. If we set $m = 4$, then $n = 24 - 1 = 15$ and $k = 15 - 4 = 11$, which satisfies the condition. So the code is C(15, 11)

Burst error correction using Hamming code



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.

Topics discussed in this section:

Cyclic Redundancy Check
Hardware Implémentation
Polynomial
Cyclic Code Analysis

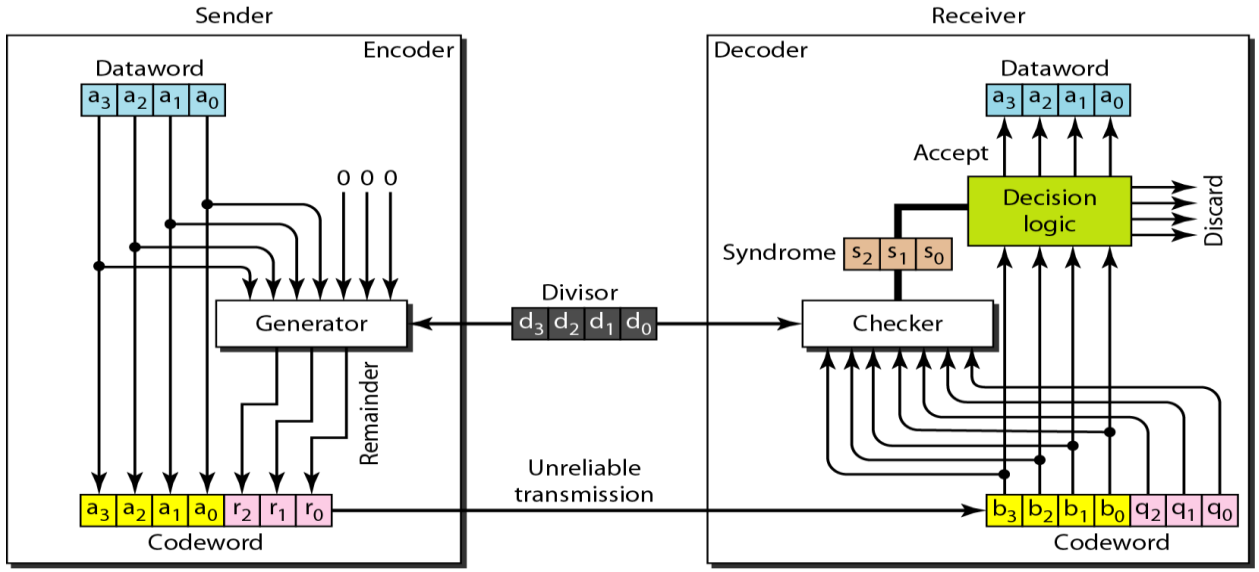
Advantages of Cyclic Codes
Other Cyclic Code

Cyclic Redundancy Check

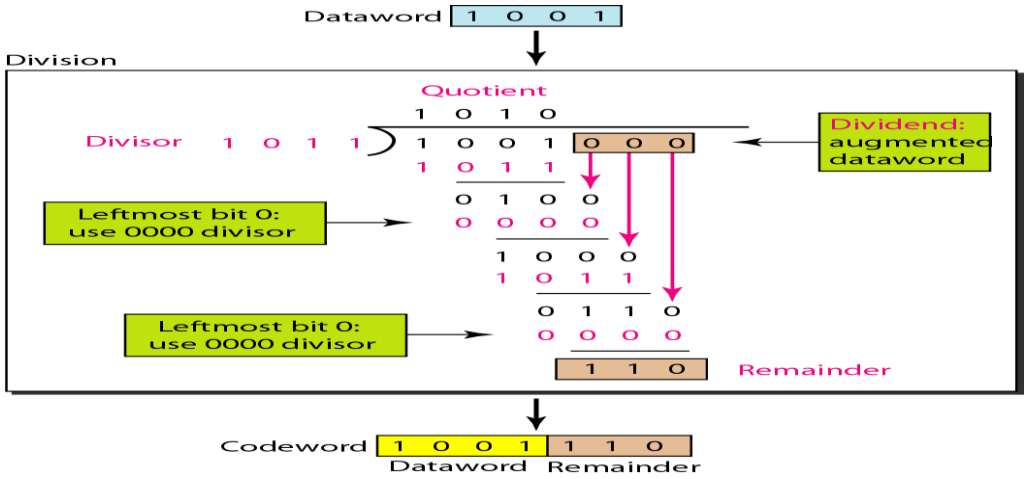
CRC code with C(7, 4)

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

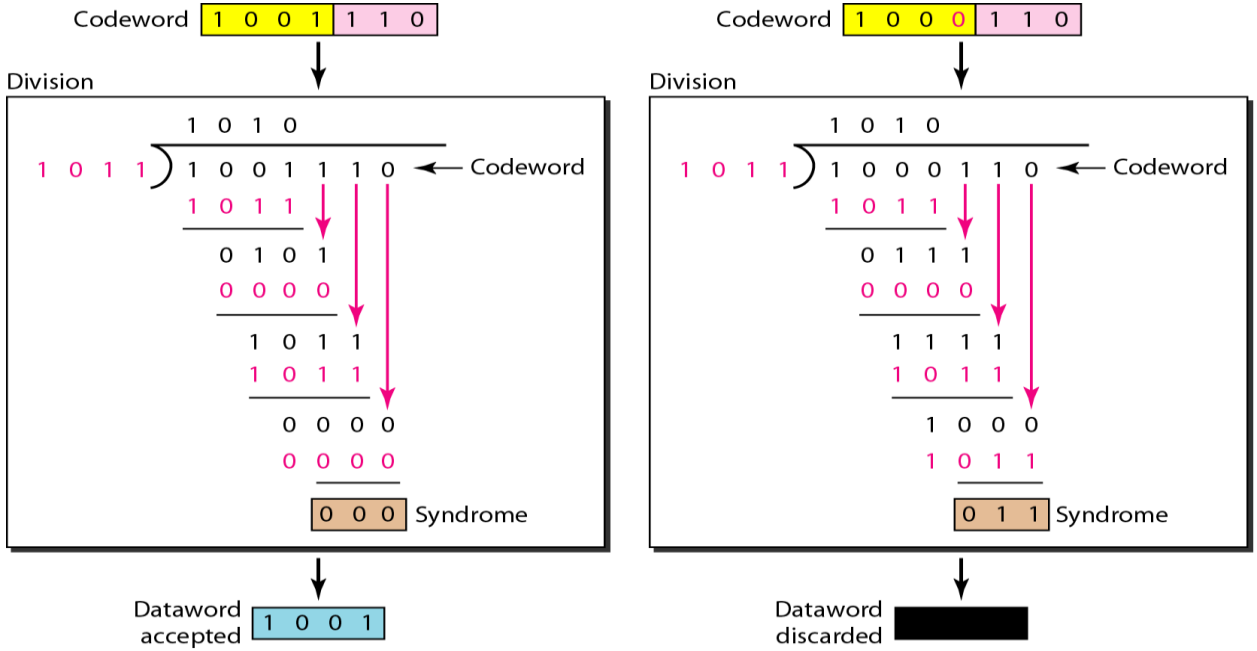
CRC encoder and decoder



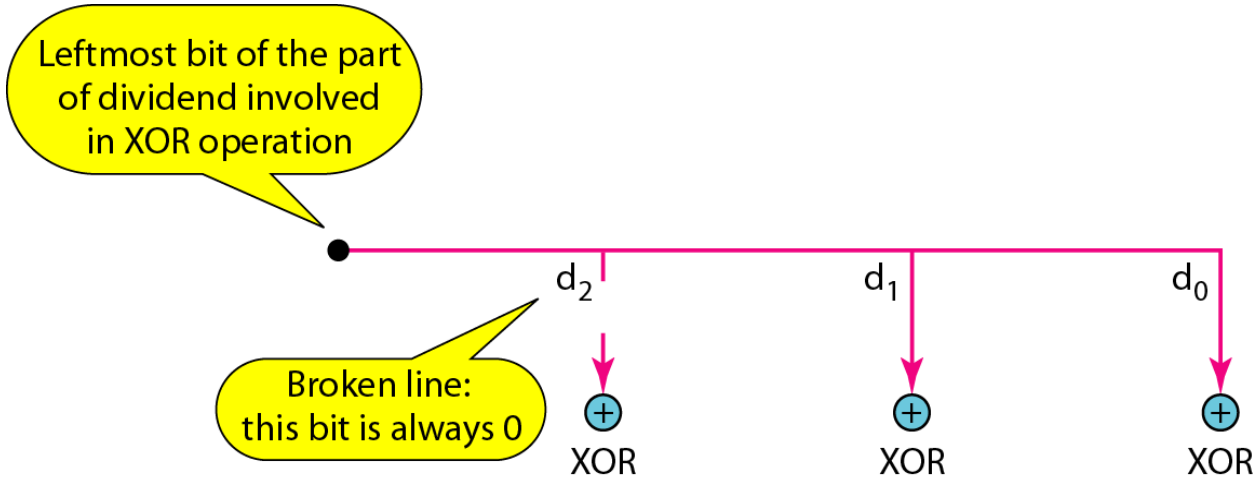
Division in CRC encoder



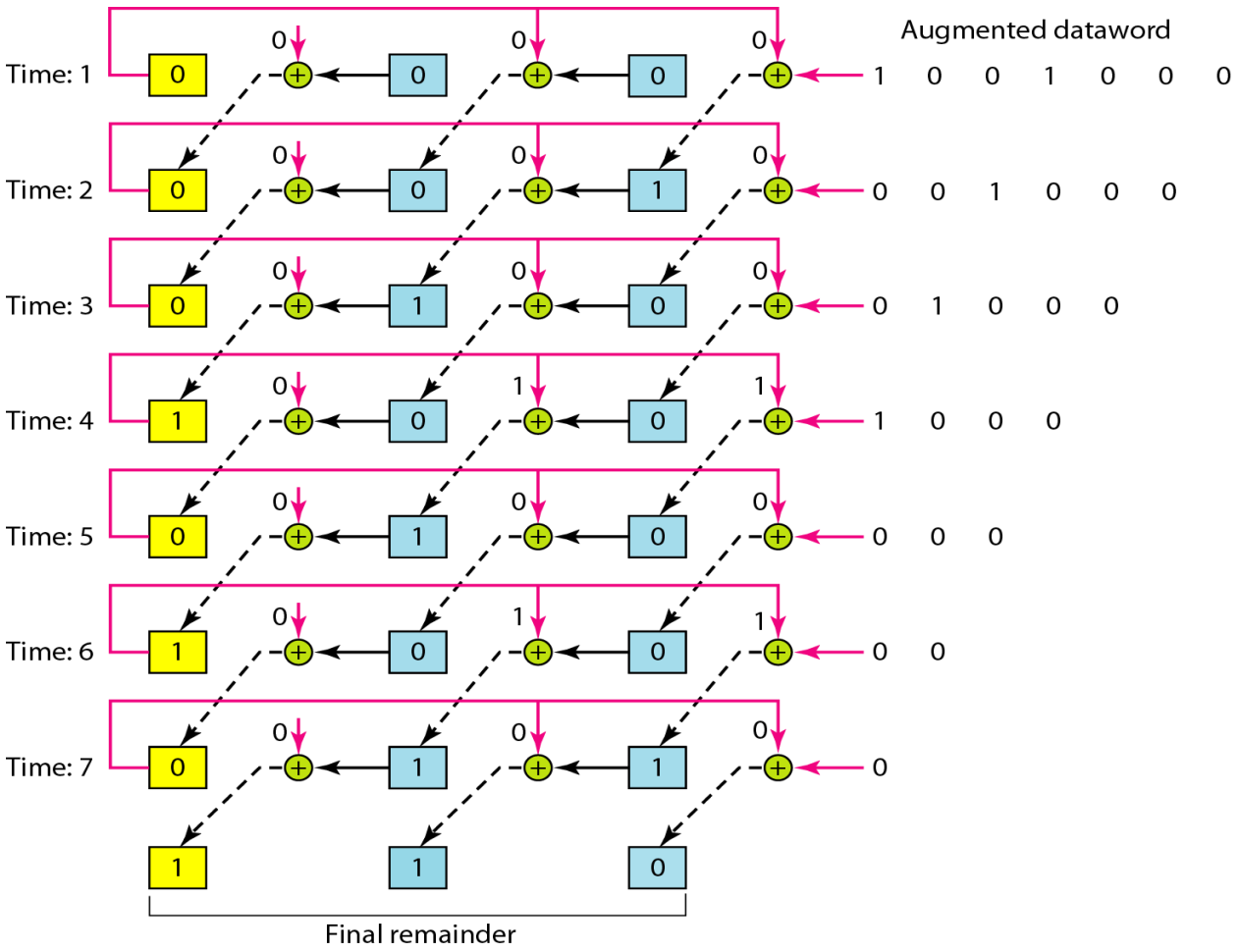
Division in the CRC decoder for two cases



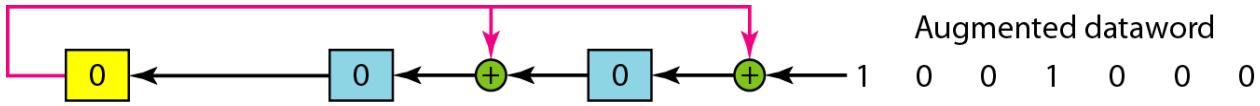
Hardwired design of the divisor in CRC



Simulation of division in CRC encoder



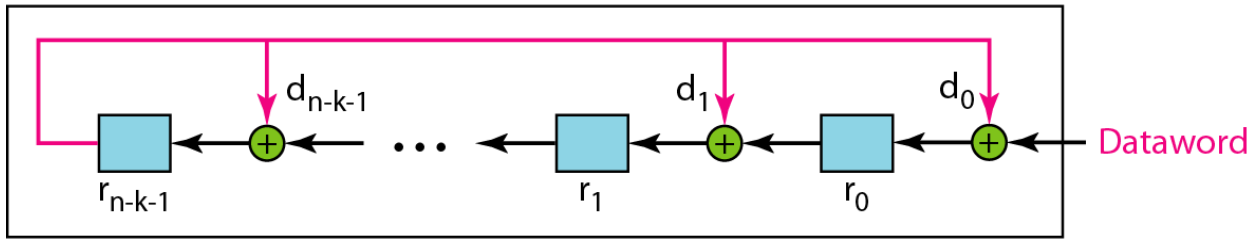
The CRC encoder design using shift registers



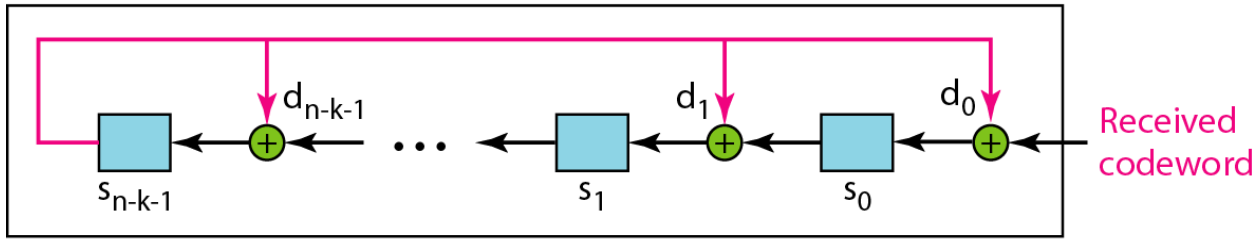
General design of encoder and decoder of a CRC code

Note:

The divisor line and XOR are missing if the corresponding bit in the divisor is 0.

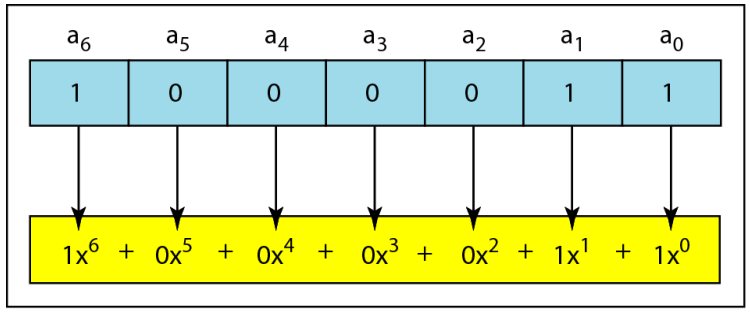


a. Encoder

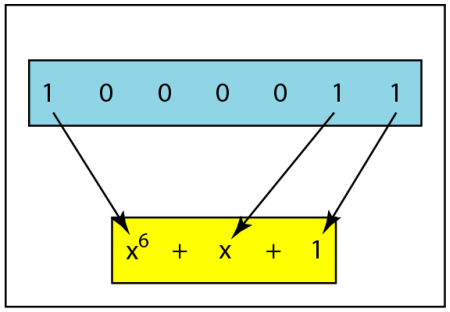


b. Decoder

A polynomial to represent a binary word

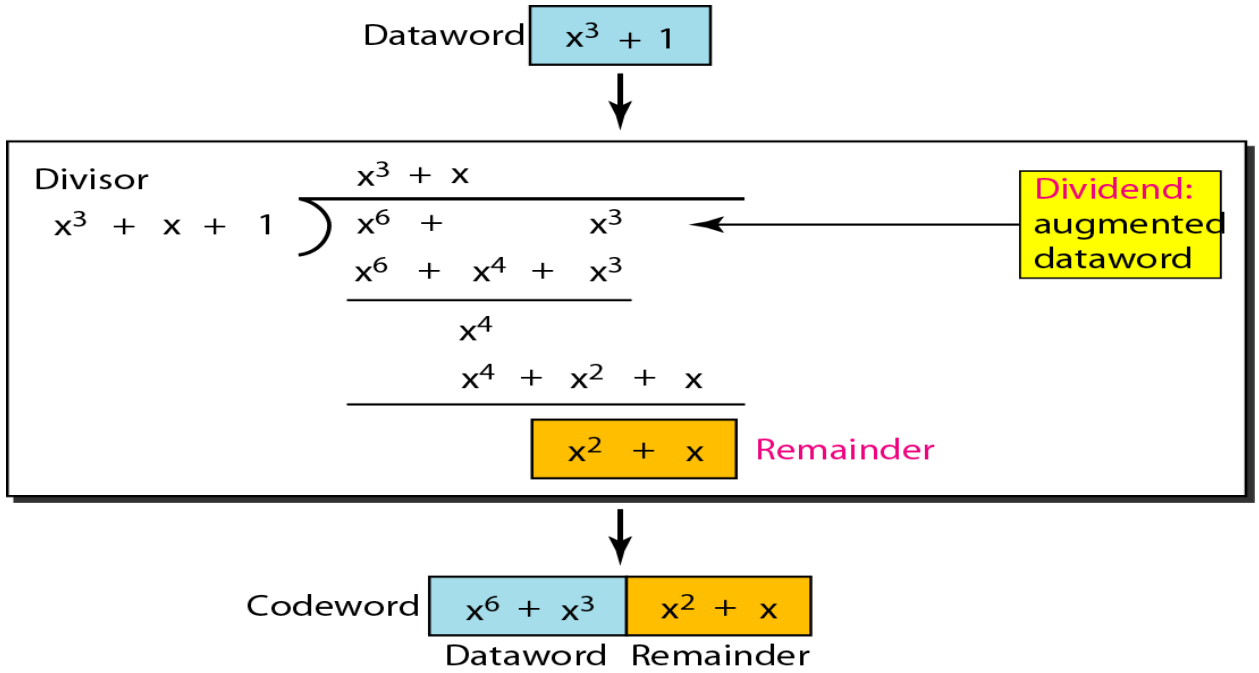


a. Binary pattern and polynomial



b. Short form

CRC division using polynomials



The divisor in a cyclic code is normally called the generator polynomial or simply the generator.

In a cyclic code,

If $s(x) \neq 0$, one or more bits is corrupted.

If $s(x) = 0$, either

a. No bit is corrupted. or

b. Some bits are corrupted, but the decoder failed to detect them.

In a cyclic code, those $e(x)$ errors that are divisible by $g(x)$ are not caught.

EXAMPLE

Which of the following g(x) values guarantees that a single-bit error is caught? For each case, what is the error that cannot be caught?
a. x + 1 b. x^3 c. 1

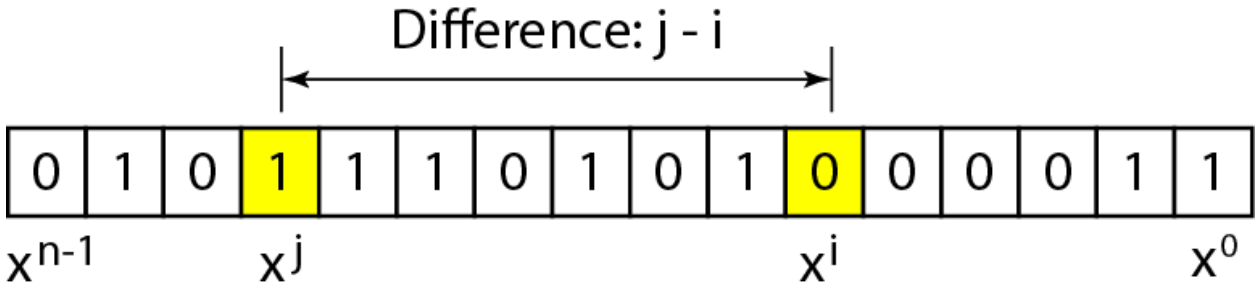
Solution

a. No x^i can be divisible by x + 1. Any single-bit error can be caught.

b. If i is equal to or greater than 3, x^i is divisible by g(x). All single-bit errors in positions 1 to 3 are caught.

c. All values of i make x^i divisible by g(x). No single-bit error can be caught. This g(x) is useless.

Representation of two isolated single-bit errors using polynomials



If a generator cannot divide $x^t + 1$ (t between 0 and n – 1), then all isolated double errors can be detected.

Find the status of the following generators related to two isolated, single-bit errors.

a. x + 1 b. $x^4 + 1$ c. $x^7 + x^6 + 1$ d. $x^{15} + x^{14} + 1$

Solution

a. This is a very poor choice for a generator. Any two errors next to each other cannot be detected.

b. This generator cannot detect two errors that are four positions apart.

c. This is a good choice for this purpose.

d. This polynomial cannot divide $x^t + 1$ if t is less than 32,768. A codeword with two isolated errors

NOTE:

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

2.All burst errors with $L \leq r$ will be detected.

3. All burst errors with $L = r + 1$ will be detected with probability $1 - (1/2)^{r-1}$.

4. All burst errors with $L > r + 1$ will be detected with probability $1 - (1/2)^r$.

CHECKSUM

The last error detection method we discuss here is called the checksum. The checksum is used in the Internet by several protocols although not at the data link layer. However, we briefly discuss it here to complete our discussion on error checking

Topics discussed in this section:

Idea
One's Complement
Internet Checksum

EXAMPLE 1

Suppose 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, if the set of numbers is (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 two 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 2

We can make the job of the receiver easier 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.

EXAMPLE 3

How can we represent the number 21 in one's complement arithmetic using only four bits?

Solution

The number 21 in binary is 10101 (it needs five bits). We can wrap the leftmost bit and add it to the four rightmost bits. We have (0101 + 1) = 0110 or 6.

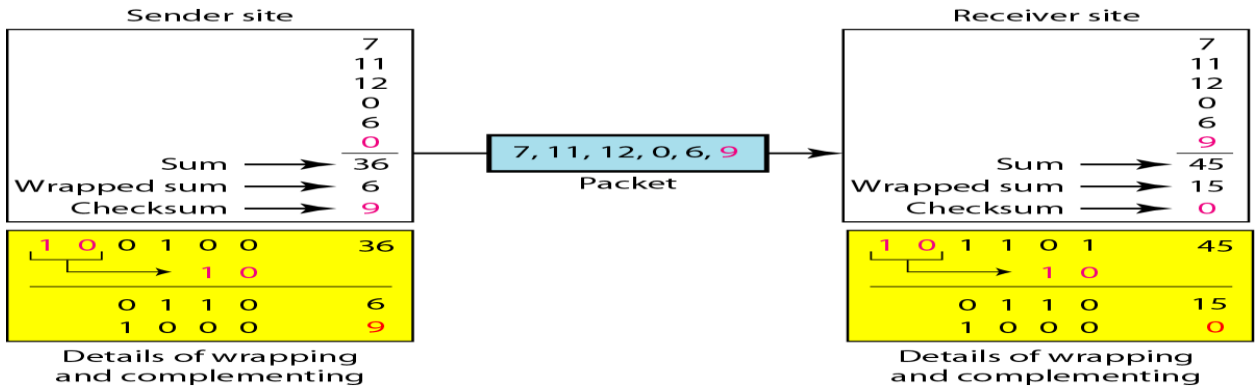
Example 4

How can we represent the number −6 in one's complement arithmetic using only four bits?

Solution

In one's complement arithmetic, the negative or complement of a number is found by inverting all bits. Positive 6 is 0110; negative 6 is 1001. If we consider only unsigned numbers, this is 9. In other words, the complement of 6 is 9. Another way to find the complement of a number in one's complement arithmetic is to subtract the number from $2^n - 1$ (16 − 1 in this case).

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.



Sender site:

1. The message is divided into 16-bit words.

2. The value of the checksum word is set to 0.

3. All words including the checksum are added using one's complement addition.

4. The sum is complemented and becomes the checksum.

5. The checksum is sent with the data.

Receiver site:

1. The message (including checksum) is divided into 16-bit words.

2. All words are added using one's complement addition.

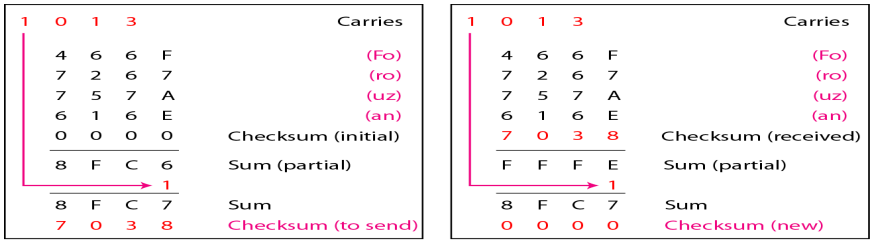
3. The sum is complemented and becomes the new checksum.

4. If the value of checksum is 0, the message is accepted; otherwise, it is rejected.

EXAMPLE 5

Let us calculate the checksum for a text of 8 characters ("Forouzan"). The text needs to be divided into 2-byte (16-bit) words. We use ASCII (see Appendix A) to change each byte to a 2-digit hexadecimal number. For example, F is represented as 0x46 and o is represented as 0x6F. Figure 10.25 shows how the checksum is calculated at the sender and receiver sites. In part a of the figure, the value of partial sum for the first column is 0x36. We keep the rightmost digit (6) and insert the leftmost digit (3) as the carry in the second column. The process is repeated for each column. Note that if there is any corruption, the checksum recalculated by the receiver is not all 0s. We leave this an exercise.

Figure 10.25 *Example 10.23*



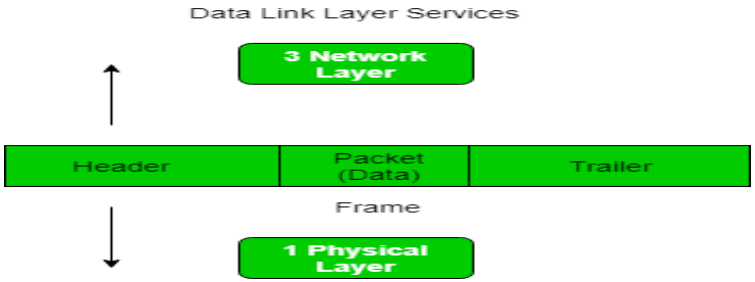
a. Checksum at the sender site

a. Checksum at the receiver site

DATA LINK CONTROL

FRAMING

Framing is a point-to-point connection between two computers or devices consists of a wire in which data is transmitted as a stream of bits. However, these bits must be framed into discernible blocks of information. Framing is a function of the data link layer. It provides a way for a sender to transmit a set of bits that are meaningful to the receiver. Ethernet, token ring, frame relay, and other data link layer technologies have their own frame structures. Frames have headers that contain information such as error-checking codes.



At data link layer, it extracts message from sender and provide it to receiver by providing sender's and receiver's address. The advantage of using frames is that data is broken up into recoverable chunks that can easily be checked for corruption.

Problems in Framing –

- **Detecting start of the frame:** When a frame is transmitted, every station must be able to detect it. Station detect frames by looking out for special sequence of bits that marks the beginning of the frame i.e. SFD (Starting Frame Delimeter).
- **How do station detect a frame:** Every station listen to link for SFD pattern through a sequential circuit. If SFD is detected, sequential circuit alerts station. Station checks destination address to accept or reject frame.
- **Detecting end of frame:** When to stop reading the frame.

Types of framing – There are two types of framing:

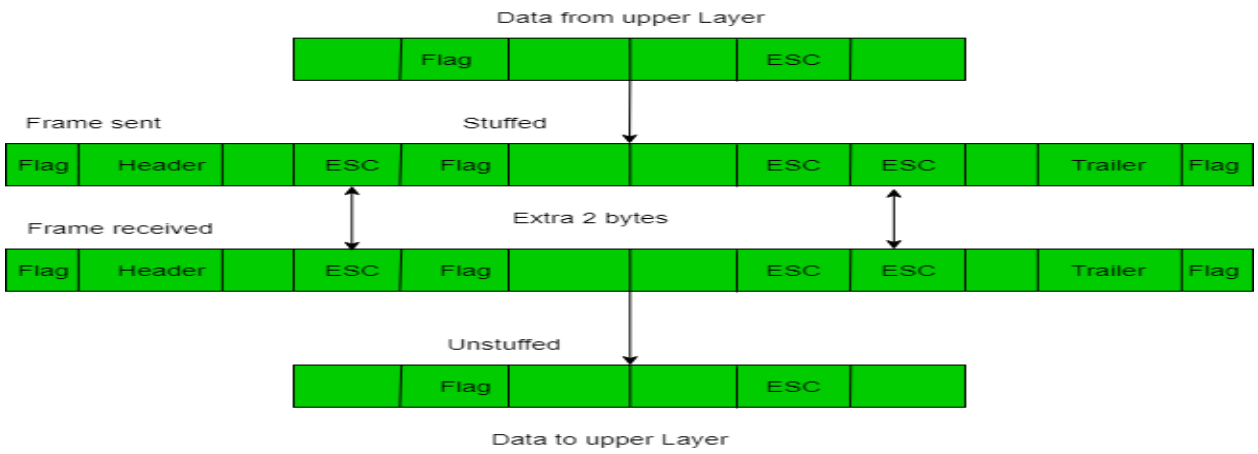
1. Fixed size – The frame is of fixed size and there is no need to provide boundaries to the frame, length of the frame itself acts as delimiter.

- **Drawback:** It suffers from internal fragmentation if data size is less than frame size
- **Solution:** Padding

2. Variable size – In this there is need to define end of frame as well as beginning of next frame to distinguish. This can be done in two ways:

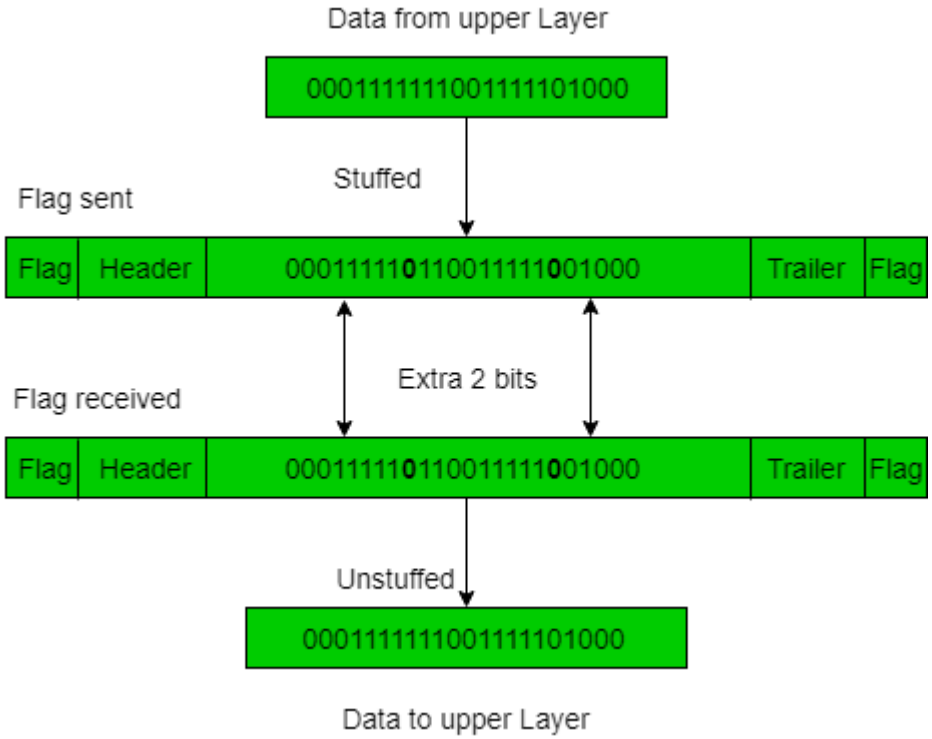
1. **Length field –** We can introduce a length field in the frame to indicate the length of the frame. Used in **Ethernet(802.3)**. The problem with this is that sometimes the length field might get corrupted.
2. **End Delimeter (ED) –** We can introduce an ED(pattern) to indicate the end of the frame. Used in **Token Ring**. The problem with this is that ED can occur in the data. This can be solved by:

   1. Character/Byte Stuffing: Used when frames consist of character. If data contains ED then, byte is stuffed into data to diffentiate it from ED.
   Let ED = "$" –> if data contains '$' anywhere, it can be escaped using '\O' character.
   –> if data contains '\O$' then, use '\O\O\O$'($ is escaped using \O and \O is escaped using \O).



Disadvantage – It is very costly and obsolete method.

2. Bit Stuffing: Let ED = 01111 and if data = 01111
–> Sender stuffs a bit to break the pattern i.e. here appends a 0 in data = 0111**0**1.
–> Receiver receives the frame.
–> If data contains 011101, receiver removes the 0 and reads the data.



Examples –

- If Data –> 011100011110 and ED –> 01111 then, find data after bit stuffing ?
  –> 0111**0**000111**0**10
- If Data –> 110001001 and ED –> 1000 then, find data after bit stuffing ?
  –> 1100**1**010**1**1

Flow Control

- Flow control coordinates the amount of data that can be sent before receiving acknowledgement
- It is one of the most important functions of data link layer.
- Flow control is a set of procedures that tells the sender how much data it can transmit before it must wait for an acknowledgement from the receiver.
- Receiver has a limited speed at which it can process incoming data and a limited amount of memory in which to store incoming data.
- Receiver must inform the sender before the limits are reached and request that the transmitter to send fewer frames or stop temporarily.
- Since the rate of processing is often slower than the rate of transmission, receiver has a block of memory (buffer) for storing incoming data until they are processed.

Error Control

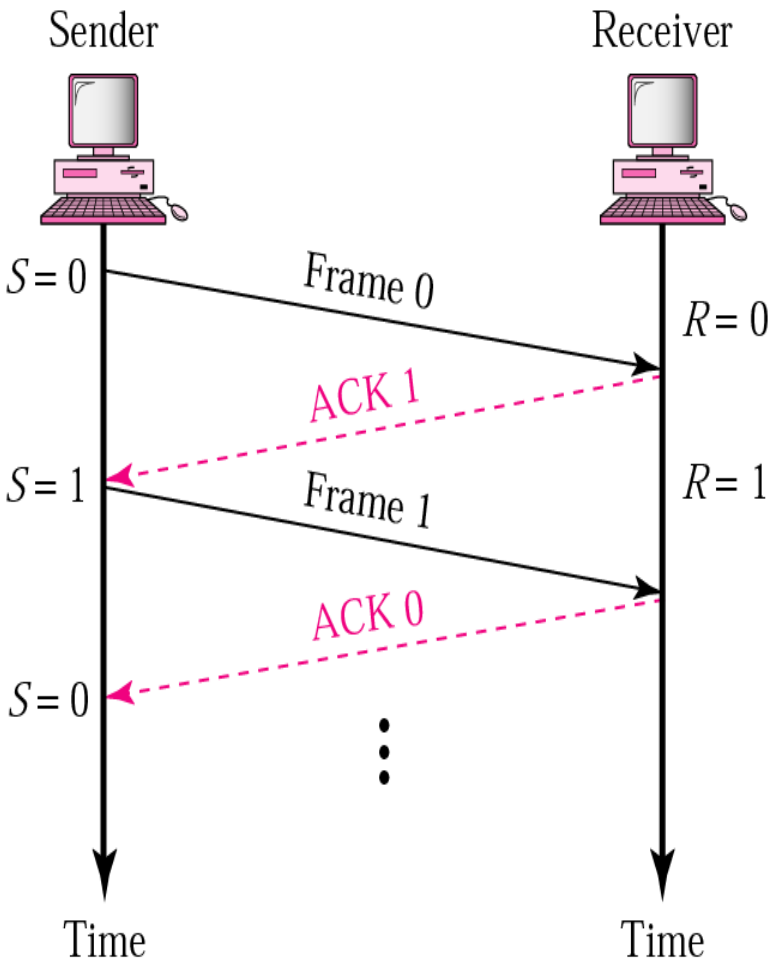
Error control includes both error detection and error correction.

- It allows the receiver to inform the sender if a frame is lost or damaged during transmission and coordinates the retransmission of those frames by the sender.
- Error control in the data link layer is based on automatic repeat request (ARQ). Whenever an error is detected, specified frames are retransmitted.

Error and Flow Control Mechanisms

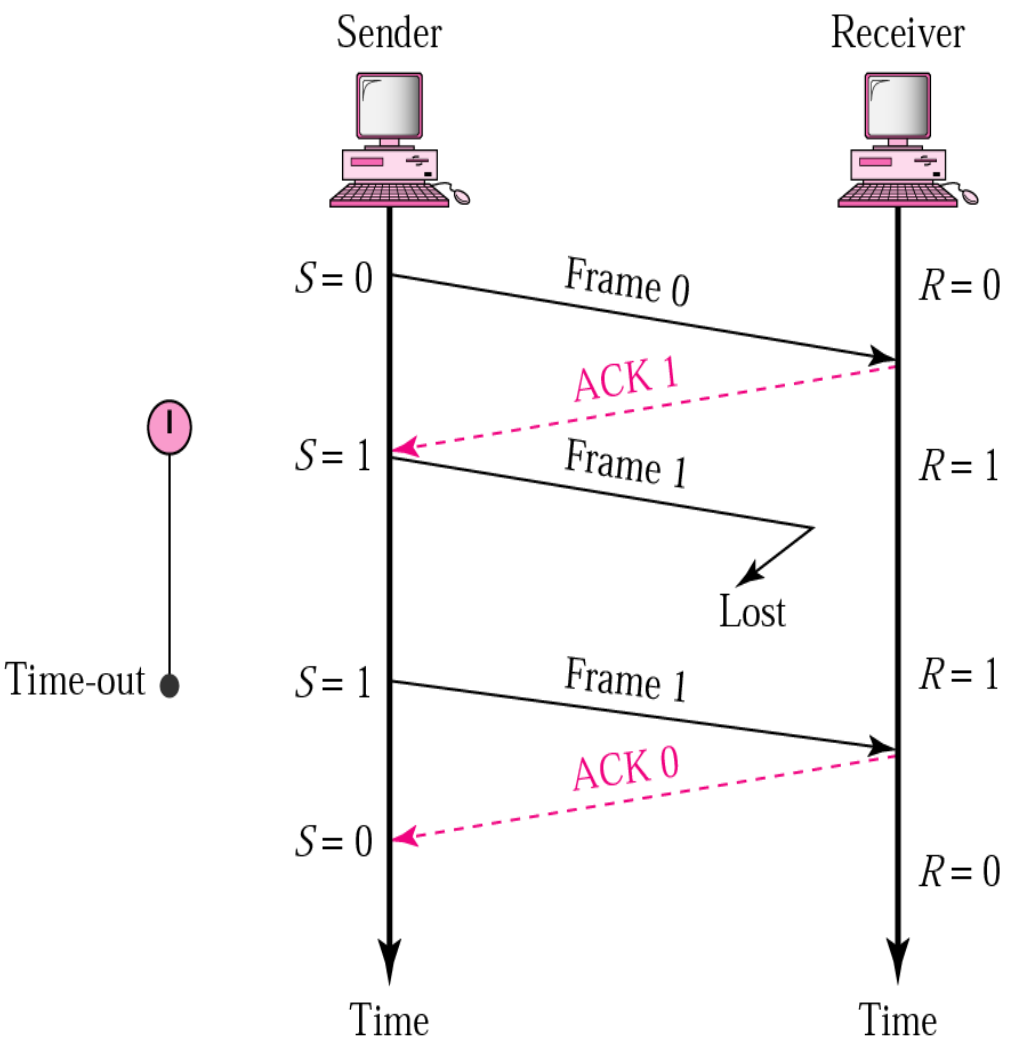
- Stop-and-Wait
- Go-Back-N ARQ
- Selective-Repeat ARQ

Stop-and-Wait



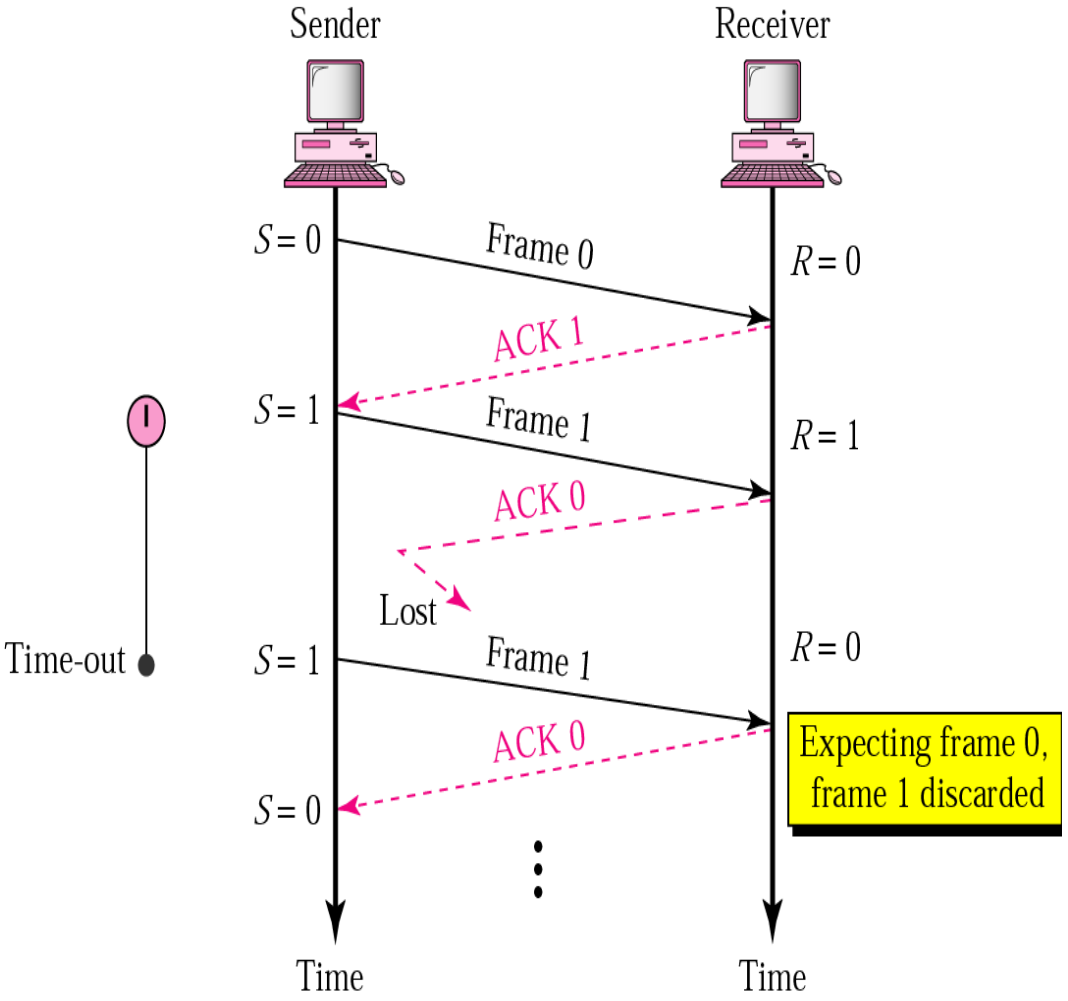
- Sender keeps a copy of the last frame until it receives an acknowledgement.
- For identification, both data frames and acknowledgements (ACK) frames are numbered alternatively 0 and 1.
- Sender has a control variable (S) that holds the number of the recently sent frame. (0 or 1)
- Receiver has a control variable ® that holds the number of the next frame expected (0 or 1).
- Sender starts a timer when it sends a frame. If an ACK is not received within a allocated time period, the sender assumes that the frame was lost or damaged and resends it
- Receiver send only positive ACK if the frame is intact.
- ACK number always defines the number of the next expected frame

Stop-and-Wait ARQ, lost ACK frame



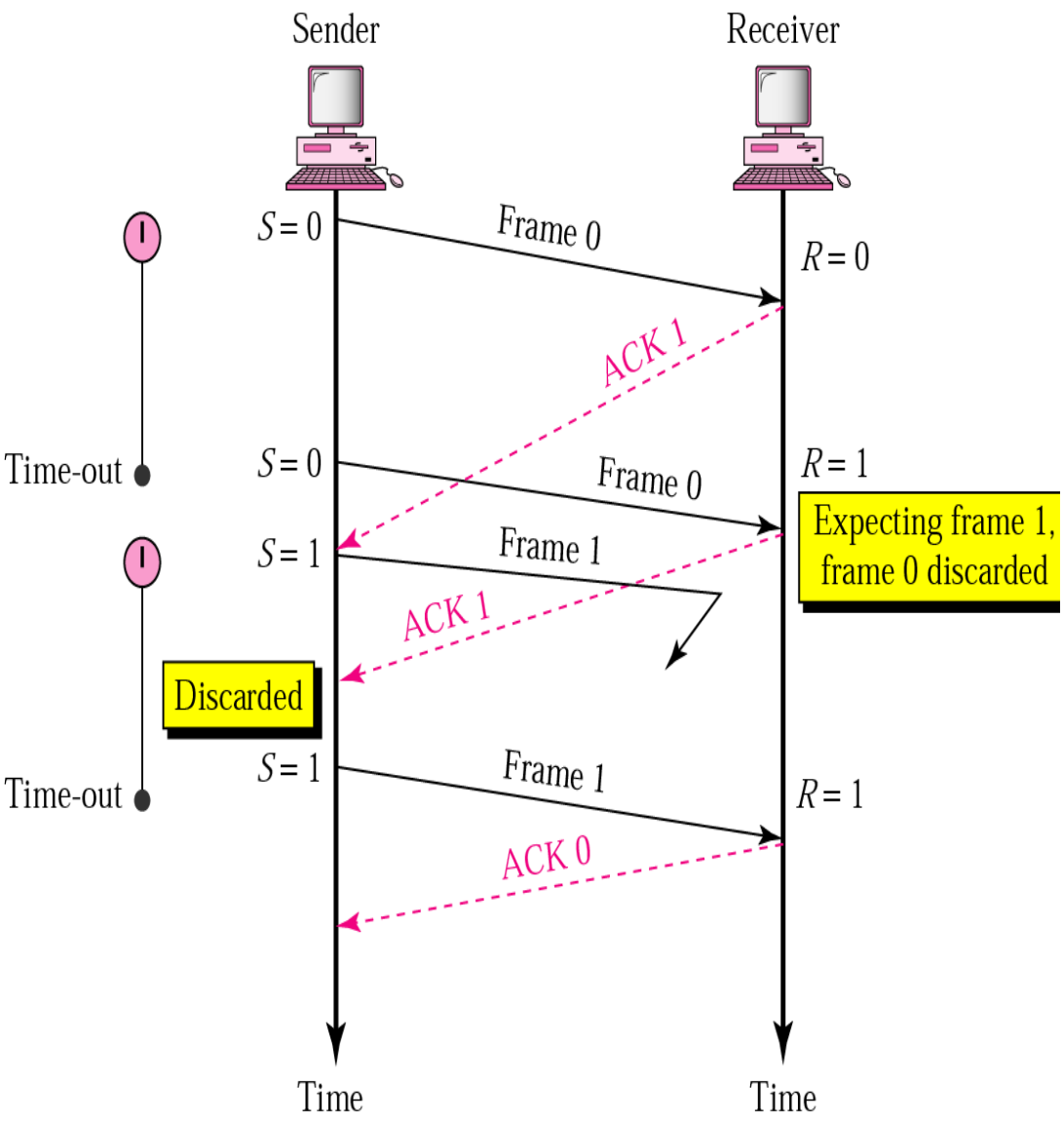
- When a receiver receives a damaged frame, it discards it and keeps its value of R.
- After the timer at the sender expires, another copy of frame1 is sent.

Stop-and-Wait, lost ACK frame



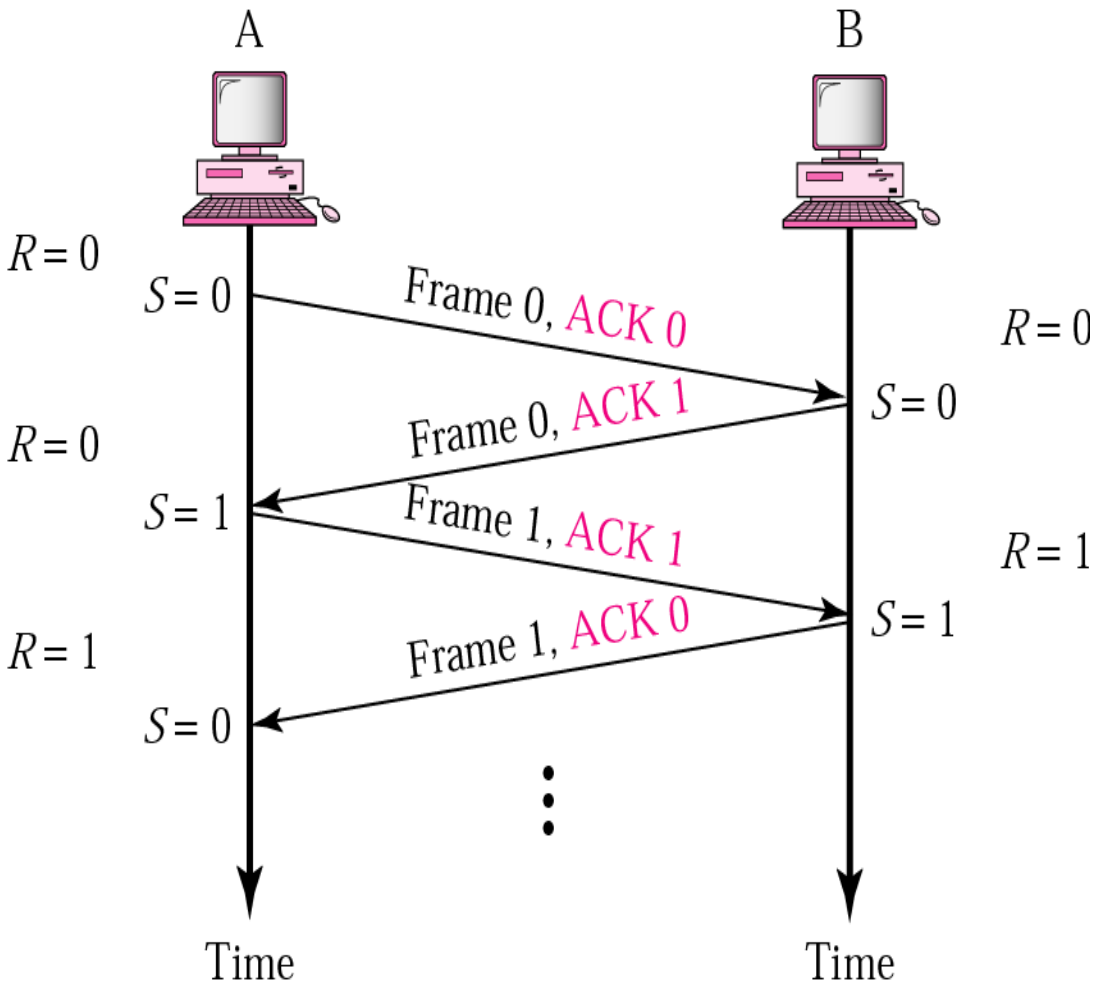
- If the sender receives a damaged ACK, it discards it.
- When the timer of the sender expires, the sender retransmits frame 1.
- Receiver has already received frame 1 and expecting to receive frame 0 (R=0). Therefore it discards the second copy of frame 1.

Stop-and-Wait, delayed ACK frame



- The ACK can be delayed at the receiver or due to some problem
- It is received after the timer for frame 0 has expired.
- Sender retransmitted a copy of frame 0. However, R =1 means receiver expects to see frame 1. Receiver discards the duplicate frame 0.
- Sender receives 2 ACKs, it discards the second ACK.

Piggybacking



- A method to combine a data frame with ACK.
- Station A and B both have data to send.
- Instead of sending separately, station A sends a data frame that includes an ACK.
- Station B does the same thing.
- Piggybacking saves bandwidth.

Disadvantage of Stop-and-Wait

- In stop-and-wait, at any point in time, there is only one frame that is sent and waiting to be acknowledged.
- This is not a good use of transmission medium.

To improve efficiency, multiple frames should be in transition while waiting for ACK.

- Two protocol use the above concept,
  - **Go-Back-N ARQ**
  - **Selective Repeat ARQ**

Go-Back-N ARQ

1.We can send up to W frames before worrying about ACKs.

2.We keep a copy of these frames until the ACKs arrive.

3.This procedure requires additional features to be added to Stop-and-Wait ARQ.

Sequence Numbers

- Frames from a sender are numbered sequentially.

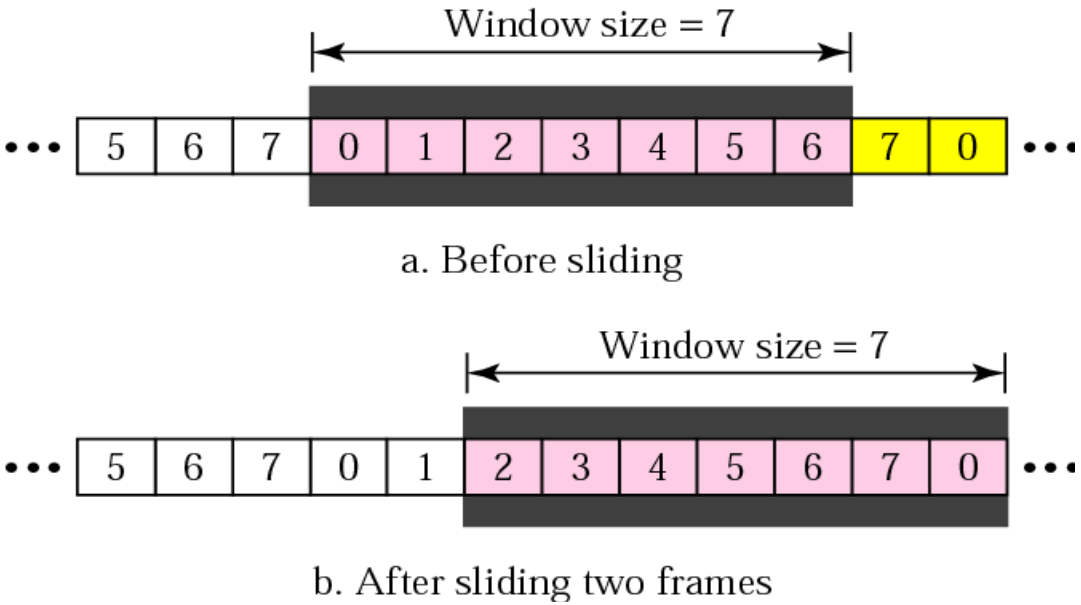
- We need to set a limit since we need to include the sequence number of each frame in the header.

- If the header of the frame allows m bits for sequence number, the sequence numbers range from 0 to $2^m - 1$. for m = 3, sequence numbers are: 1, 2, 3, 4, 5, 6, 7.

- We can repeat the sequence number.
- Sequence numbers are:
  0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0, 1, …

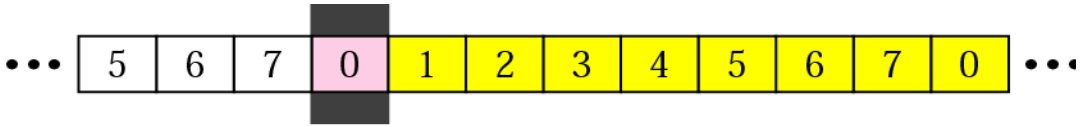
Sender Sliding Window

- At the sending site, to hold the outstanding frames until they are acknowledged, we use the concept of a window.
- The size of the window is at most $2^m - 1$ where m is the number of bits for the sequence number.
- Size of the window can be variable, e.g. TCP.
- The window slides to include new unsent frames when the correct ACKs are received

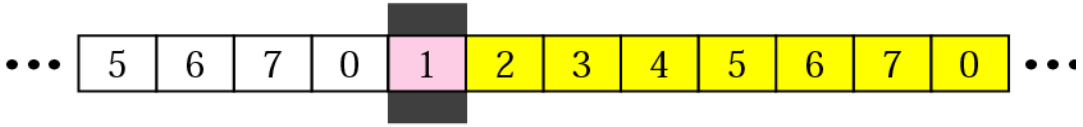


Receiver Sliding Window

- Size of the window at the receiving site is always 1 in this protocol.
- Receiver is always looking for a specific frame to arrive in a specific order.
- Any frame arriving out of order is discarded and needs to be resent.
- Receiver window slides as shown in fig. Receiver is waiting for frame 0 in part a.



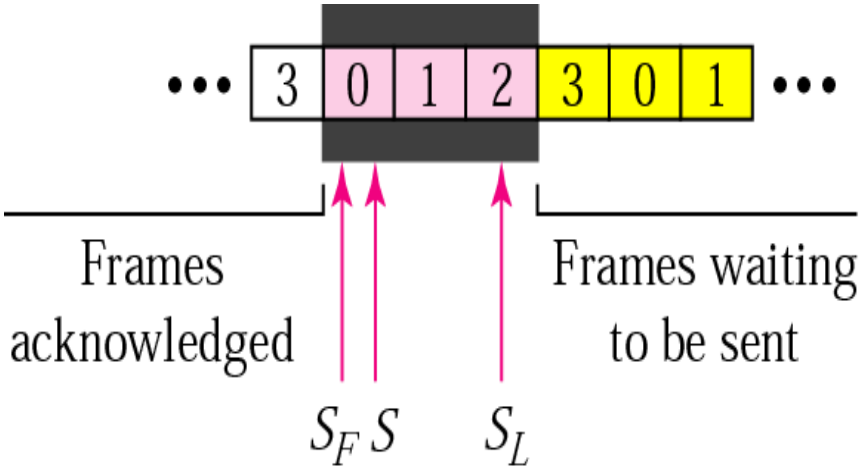
a. Before sliding



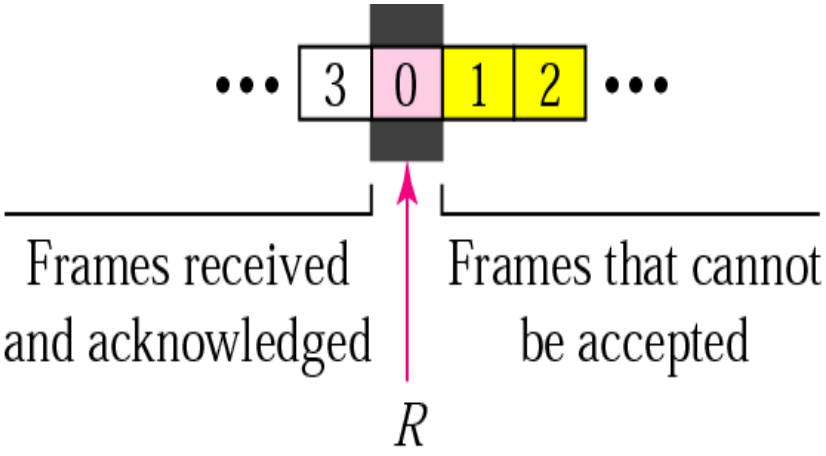
b. After sliding

Control Variables

- Sender has 3 variables: S, S_F, and S_L
- S holds the sequence number of recently sent frame
- S_F holds the sequence number of the first frame
- S_L holds the sequence number of the last frame
- Receiver only has the one variable, R, that holds the sequence number of the frame it expects to receive. If the seq. no. is the same as the value of R, the frame is accepted, otherwise rejected.



a. Sender window



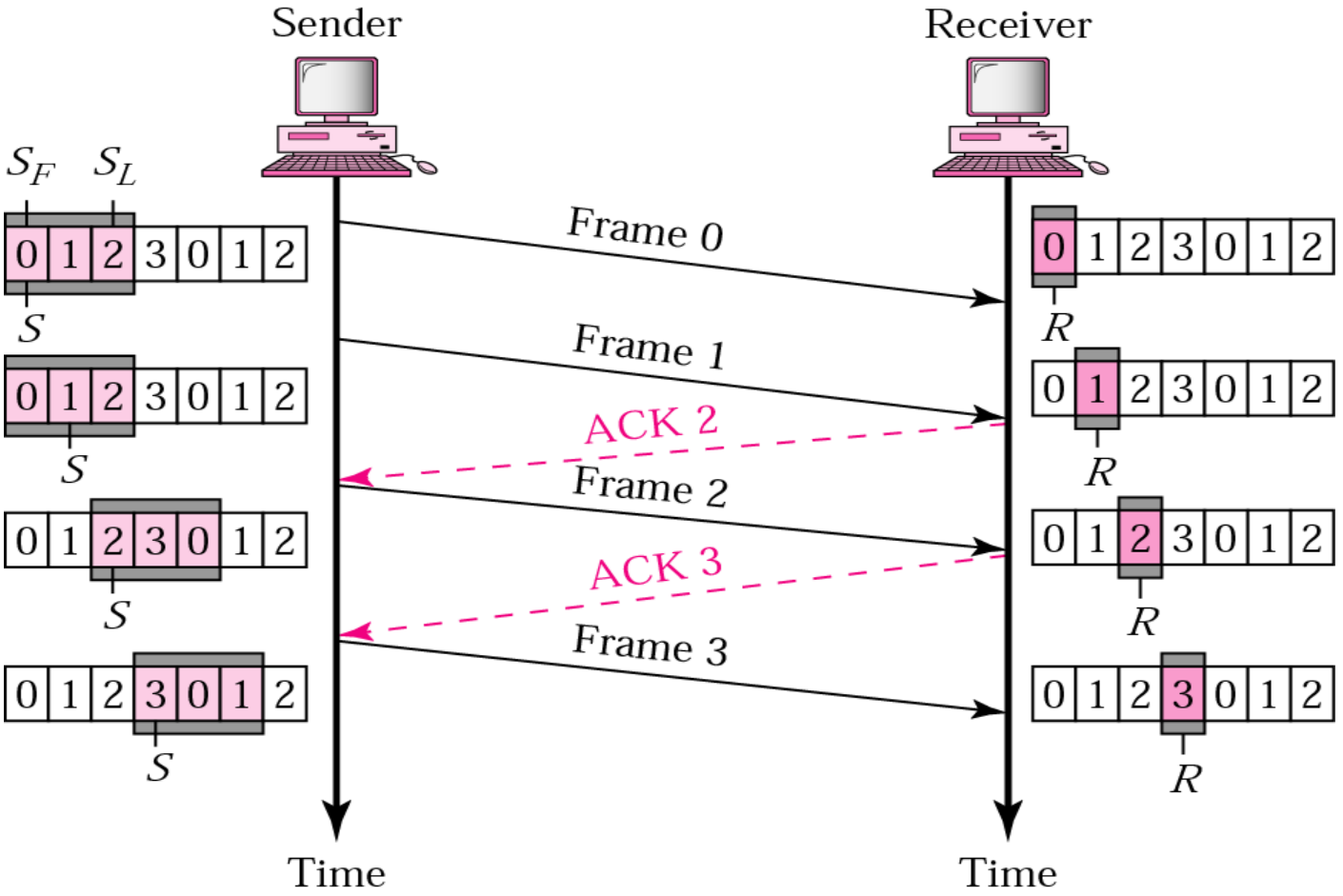
b. Receiver window

Acknowledgement

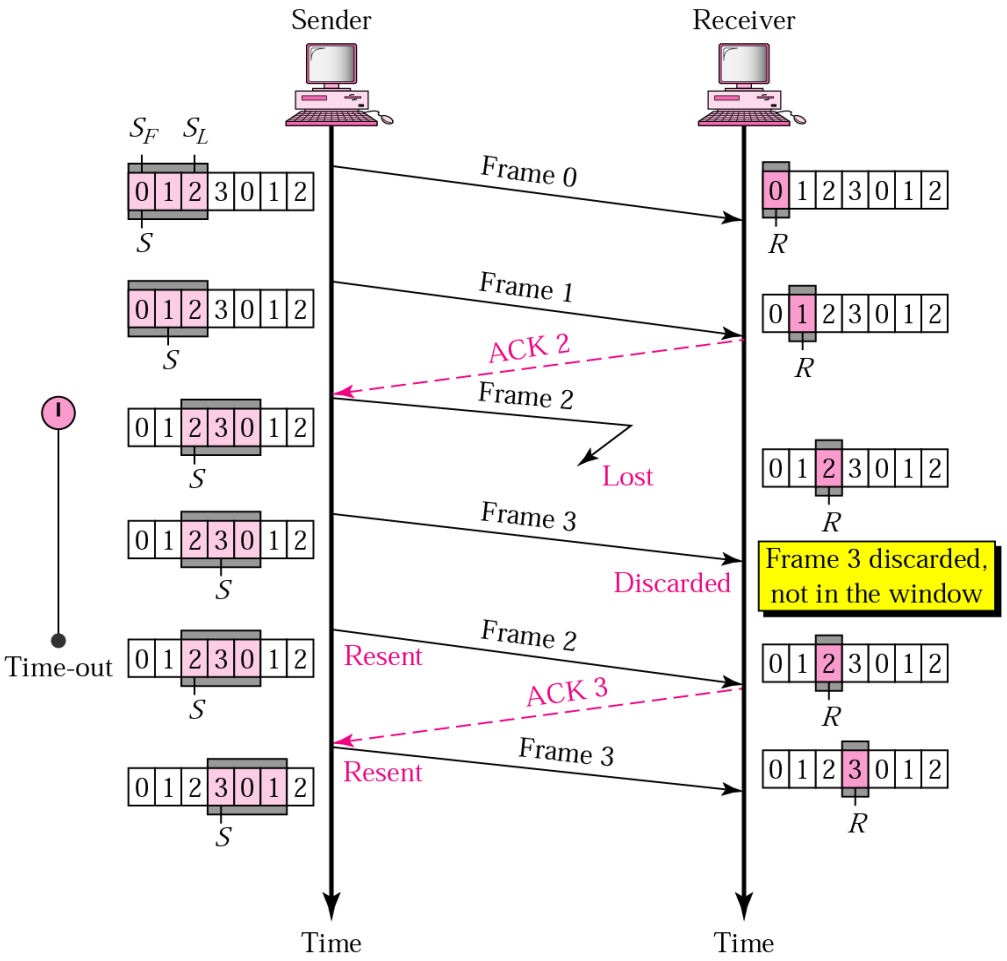
- Receiver sends positive ACK if a frame arrived safe and in order.
- If the frames are damaged/out of order, receiver is silent and discard all subsequent frames until it receives the one it is expecting.
- The silence of the receiver causes the timer of the unacknowledged frame to expire.
- Then the sender resends all frames, beginning with the one with the expired timer.
- For example, suppose the sender has sent frame 6, but the timer for frame 3 expires (i.e. frame 3 has not been acknowledged), then the sender goes back and sends frames 3, 4, 5, 6 again. Thus it is called Go-Back-N-ARQ
- The receiver does not have to acknowledge each frame received, it can send one cumulative ACK for several frames.

Go-Back-N ARQ, normal operation

- The sender keeps track of the outstanding frames and updates the variables and windows as the ACKs arrive.



Go-Back-N ARQ, lost frame



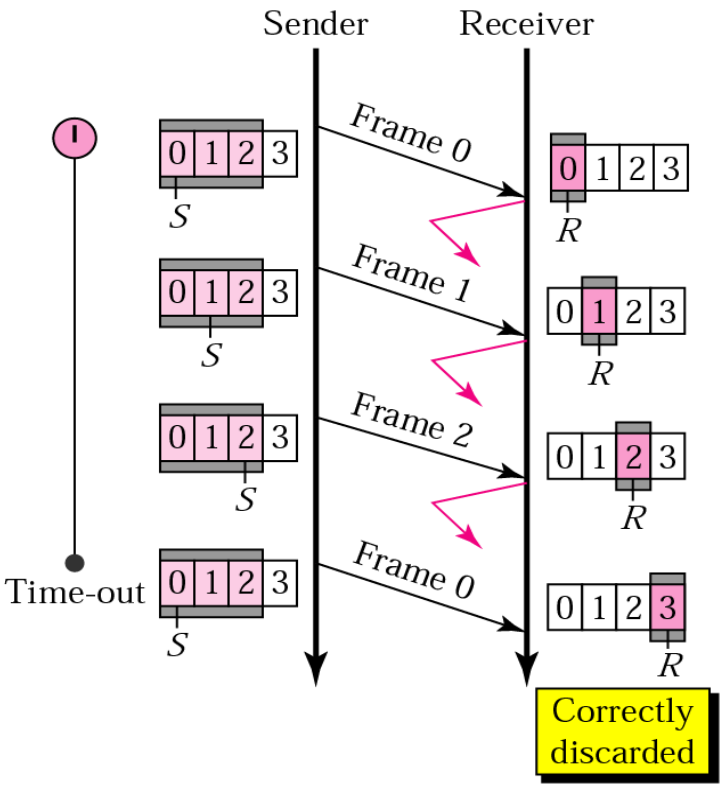
- Frame 2 is lost
  - When the receiver receives frame 3, it discards frame 3 as it is expecting frame 2 (according to window).
  - After the timer for frame 2 expires at the sender site, the sender sends frame 2 and 3.(go back to 2)

Go-Back-N ARQ, damaged/lost/delayed ACK

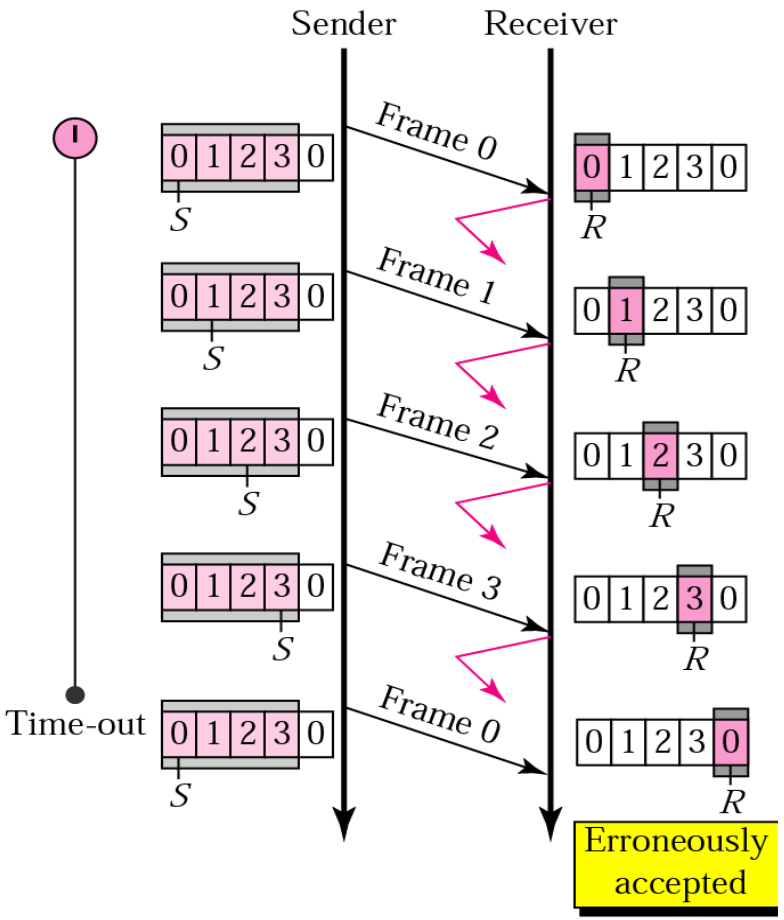
- If an ACK is damaged/lost, we can have two situations:
- If the next ACK arrives before the expiration of any timer, there is no need for retransmission of frames because ACKs are cumulative in this protocol.
- If ACK1, ACK2, and ACk3 are lost, ACK4 covers them if it arrives before the timer expires.
- If ACK4 arrives after time-out, the last frame and all the frames after that are resent.
- Receiver never resends an ACK.
- A delayed ACK also triggers the resending of frames

Go-Back-N ARQ, sender window size

- Size of the sender window must be less than 2^m. Size of the receiver is always 1. If m = 2, window size = $2^m - 1 = 3$.
- Fig compares a window size of 3 and 4.



a. Window size < 2^m

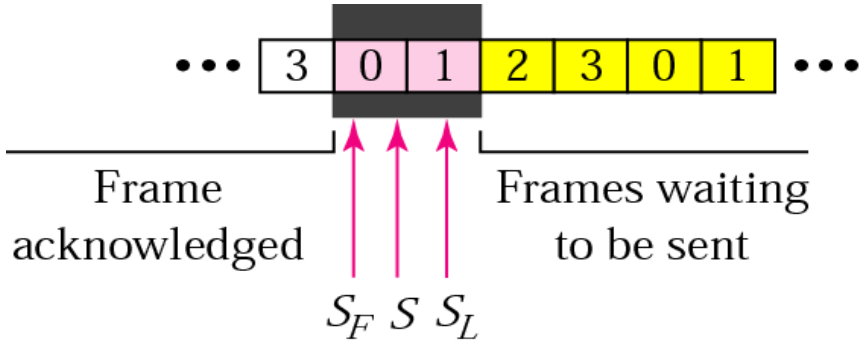


b. Window size = 2^m

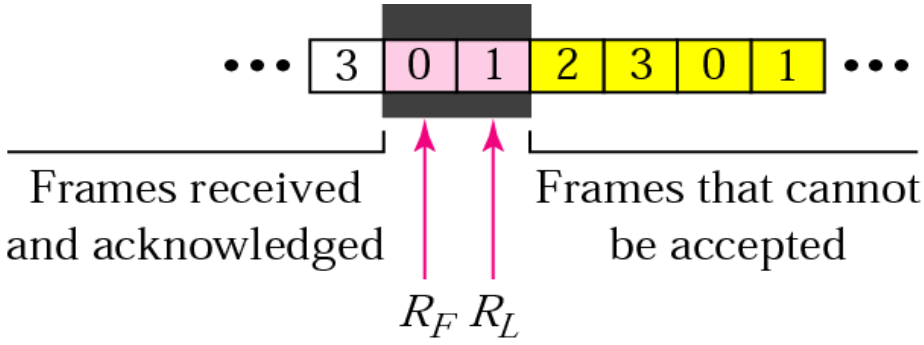
Accepts as the 1st frame in the next cycle-an **error**

Selective Repeat ARQ, sender and receiver windows

- Go-Back-N ARQ simplifies the process at the receiver site. Receiver only keeps track of only one variable, and there is no need to buffer out-of-order frames, they are simply discarded.
- However, Go-Back-N ARQ protocol is inefficient for noisy link. It bandwidth inefficient and slows down the transmission.
- In Selective Repeat ARQ, only the damaged frame is resent. More bandwidth efficient but more complex processing at receiver.
- It defines a negative ACK (NAK) to report the sequence number of a damaged frame before the timer expires.

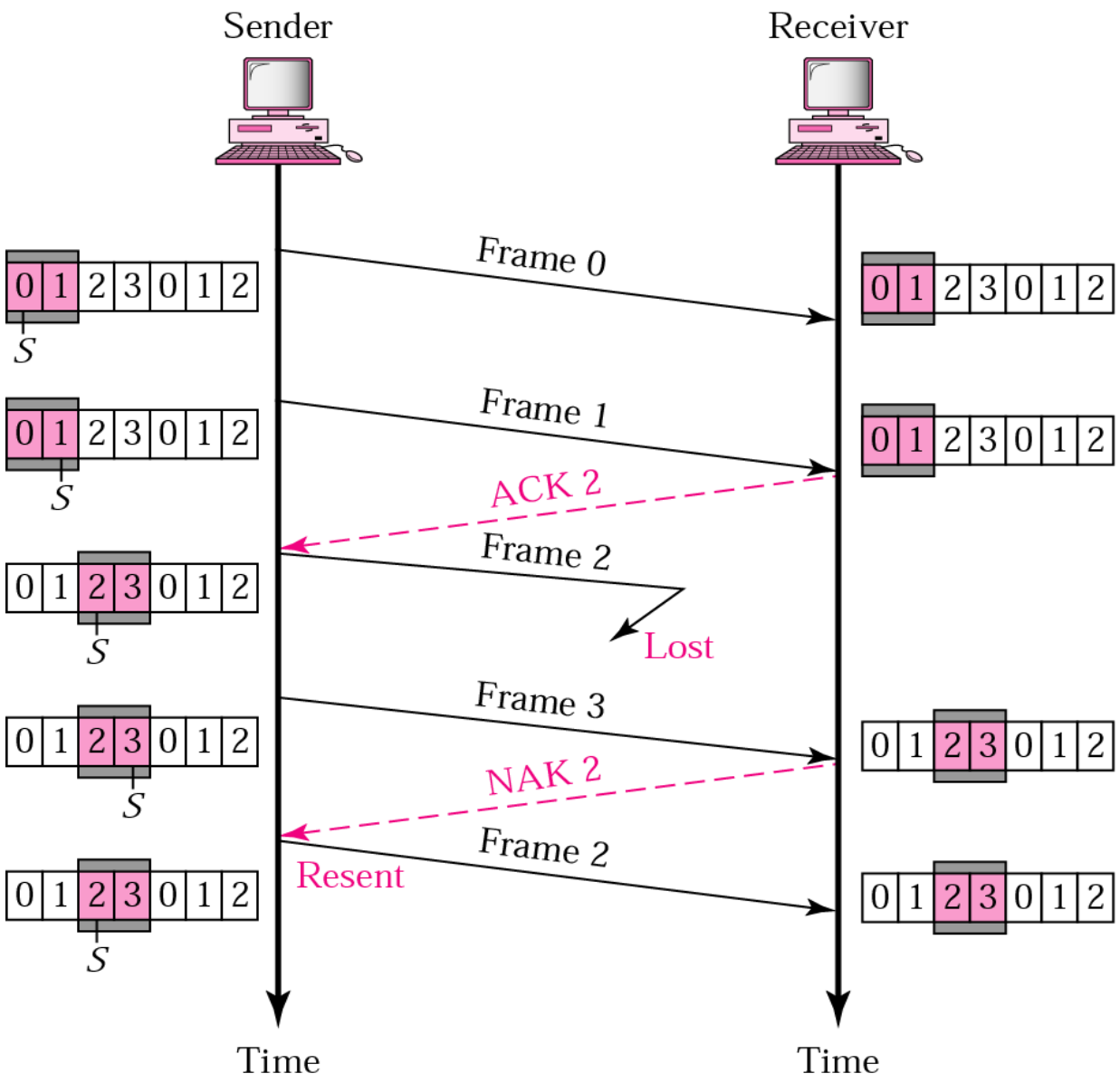


a. Sender window



b. Receiver window

Selective Repeat ARQ, lost frame



- Frames 0 and 1 are accepted when received because they are in the range specified by the receiver window. Same for frame 3.
- Receiver sends a NAK2 to show that frame 2 has not been received and then sender resends only frame 2 and it is accepted as it is in the range of the window.

Selective Repeat ARQ, sender window size

Size of the sender and receiver windows must be at most one-half of 2^m. If m =2, window size should be $2^m/2 = 2$. Fig compares a window size of 2 with a window size of 3. Window size is 3 and all ACKs are lost, sender sends duplicate of frame 0, window of the receiver expect to receive frame 0 (part of thewindow), so accepts frame 0, as the 1st frame of the next cycle – an **error**.