Computer Networking with TCP/IP

Module 1

Introduction

A network is a group of connected, communicating devices such as computers and printers. An internet is two or more networks that can communicate with each other. The most notable internet is called the internet, composed of hundreds of thousands of interconnected networks. Private individuals as well as various organizations such as government agencies, schools, research facilities, corporations, and libraries in more than 100 countries use the Internet.

PROTOCOLS AND STANDARDS

First, we define protocol, which is synonymous with "rule." Then we discuss standards, which are agreed-upon rules.

Networking Components

1. **Message.** The message is the information (data) to be communicated. Popular forms of information include text, numbers, pictures, audio, and video.
2. **Sender**. The sender is the device that sends the data message. It can be a computer, workstation, telephone handset, video camera, and so on.
3. **Receiver**. The receiver is the device that receives the message. It can be a computer, workstation, telephone handset, television, and so on.
4. **Transmission medium**. The transmission medium is the physical path by which a message travels from sender to receiver. Some examples of transmission media include twisted-pair wire, coaxial cable, fiber-optic cable, and radio waves.
5. **Protocol.** A protocol is a set of rules that govern data communications. It represents an agreement between the communicating devices. Without a protocol, two devices may be connected but not communicating, just as a person speaking French cannot be understood by a person who speaks only Japanese.

Protocols

Communication between two people or two devices needs to follow some protocol. A protocol is a set of rules that governs communication.

- **Syntax.** The term syntax refers to the structure or format of the data, meaning the order in which they are presented.
- **Semantics**. The word semantics refers to the meaning of each section of bits. How is a particular pattern to be interpreted, and what action is to be taken based on that interpretation?
- **Timing**. The term timing refers to two characteristics: when data should be sent and how fast they can be sent.

Standards

Standards are essential in creating and maintaining an open and competitive market for equipment manufacturers and in guaranteeing national and international interoperability of data and telecommunications technology and processes.

Data communication standards fall into two categories: de facto (meaning "by fact" or "by convention") and de jure (meaning "by law" or "by regulation").

- De facto. Standards that have not been approved by an organized body but have been adopted as standards through widespread use are de facto standards.
- De jure. Those standards that have been legislated by an officially recognized body are de jure standards.

Internet Standards

An Internet standard is a thoroughly tested specification that is useful to and adhered to by those who work with the Internet. **An Internet draft** is a working document may be published as a **Request for Comment (RFC)**. Each RFC is edited, assigned a number, and made available to all interested parties. RFCs go through maturity levels and are categorized according to their requirement level.

Networking Models

A network is a combination of hardware and software that sends data from one location to another. The hardware consists of the physical equipment that carries signals from one point of the network to another. The software consists of instruction sets that make possible the services that we expect from a network.

A networking model, sometimes also called either a *networking architecture or networking blueprint*, refers to a comprehensive set of documents. Individually, each document describes one small function required for a network; collectively, these documents define everything that should happen for a computer network to work. Some documents define a *protocol*, which is a set of logical rules that devices must follow to communicate. Other documents define some physical requirements for networking. For example, a document could define the voltage and current levels used on a particular cable when transmitting data.

PROTOCOL LAYERING the Layered Approach.

Networking model addresses all the process required for effective communication and divides them into logical groupings called Layers. When communication system is designed in this manner, it is known as ***layered architecture.***

When communication is simple, we may need only one simple protocol; when the communication is complex, we may need to divide the task between different layers, in which case we need a protocol at each layer, or protocol layering.

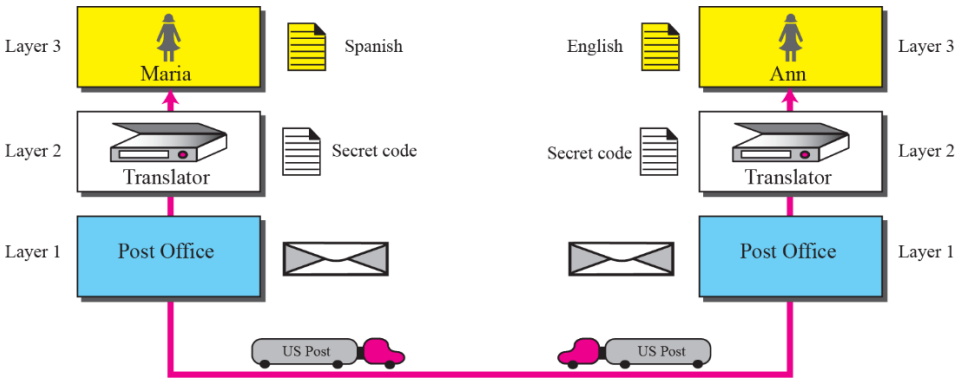
Principles of Protocol Layering

First Principle The first principle dictates that if we want bidirectional communication, we need to make each layer so that it can perform two opposite tasks, one in each direction.

Second Principle The second principle that we need to follow in protocol layering is that the two objects under each layer at both sites should be identical.

Logical Connections

We have a logical connection between each layer This means that we have layer-to-layer communication. There is a logical (imaginary) connection at each layer through which they can send the object created from that layer.



Advantages of Reference Models

The OSI model is hierarchical, and the same benefits and advantages can apply to any layered model. The primary purpose of all such models, especially the OSI model, is to allow different vendors' networks to interoperate.

Advantages of using the OSI layered model include, but are not limited to, the following:

- It divides the network communication process into smaller and simpler components, thus aiding component development, design, and troubleshooting.
- It allows multiple-vendor development through standardization of network components.
- It encourages industry standardization by defining what functions occur at each layer of the model.
- It allows various types of network hardware and software to communicate.
- It prevents changes in one layer from affecting other layers, so it does not hamper development.

The OSI Reference Model

In the late 1970s, the Open Systems Interconnection (OSI) reference model was created by the International Organization for Standardization (ISO) to break this barrier.

The OSI model was meant to help vendors create interoperable network devices and software in the form of protocols so that different vendor networks could work with each other. The OSI model is the primary architectural model for networks. It describes how data and network information are communicated from an application on one computer, through the network media, to an application on another computer. The OSI reference model breaks this approach into layers.

The OSI model is a **layered framework** for the design of network systems that allows communication between all types of computer systems. It consists of seven separate but related layers, each of which defines a part of the process of moving information across a network.

The OSI has seven different layers, divided into two groups. The top three layers define how the applications within the end stations will communicate with each other as well as with users. The bottom four layers define how data is transmitted end to end.

The Upper Layers

Application	• Provides a user interface
Presentation	• Presents data • Handles processing such as encryption
Session	• Keeps different applications' data separate

The Lower Layers

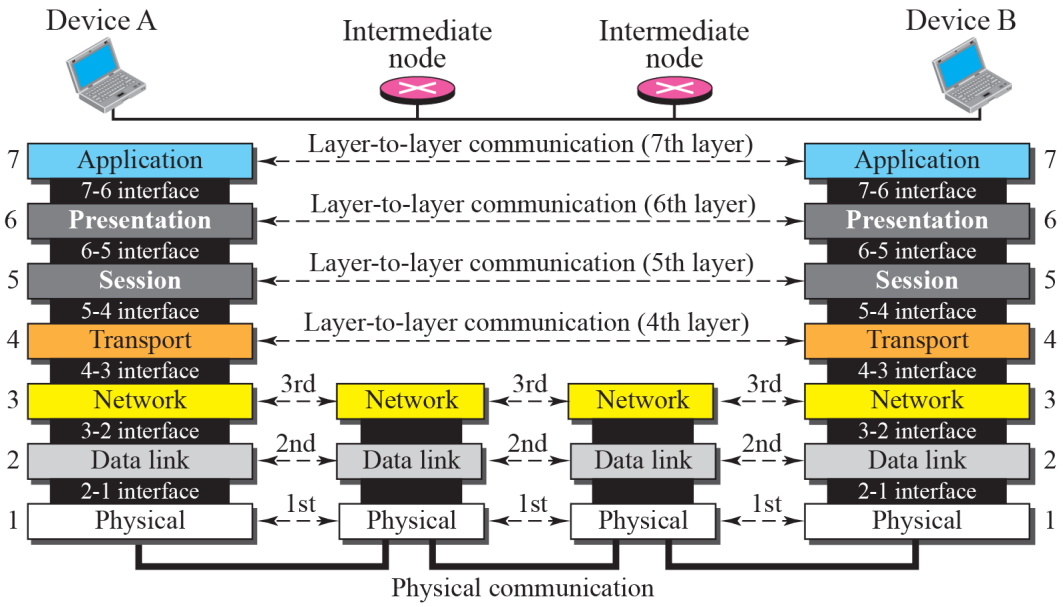
Transport	• Provides reliable or unreliable delivery • Performs error correction before retransmit
Network	• Provides logical addressing, which routers use for path determination
Data Link	• Combines packets into bytes and bytes into frames • Provides access to media using MAC address • Performs error detection not correction
Physical	• Moves bits between devices • Specifies voltage, wire speed, and pinout of cables

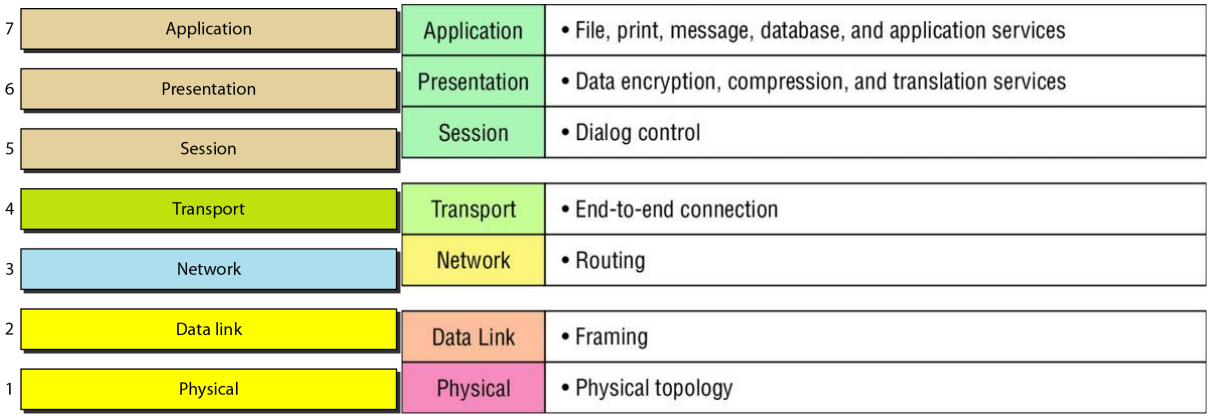
The interaction between layers in the OSI model

Peer-to-Peer Processes

At the physical layer, communication is direct: Device A sends a stream of bits to device B (through intermediate nodes). At the higher layers, however, communication must move down through the layers on device A, over to device B, and then back up through the layers. Each layer in the sending device adds its own information to the message it receives from the layer just above it and passes the whole package to the layer just below it. At layer I the entire package is converted to a form that can be transmitted to the receiving device.

At the receiving machine, the message is unwrapped layer by layer, with each process receiving and removing the data meant for it. For example, layer 2 removes the data meant for it, then passes the rest to layer 3. Layer 3 then removes the data meant for it and passes the rest to layer 4, and so on.





Connection-oriented and connectionless services

- The service each layer provides to the layer above can be classified into two different types,
- **Connection oriented service** requires a pair of processes in the two computers to establish a connection before sending data.
- **In connectionless service**, data packets are independent of each other, and they may not arrive at the destination in the proper order.
- In **connection-oriented service**, a logical connection is established by having both sides initialize variables and counters needed to keep track of which frames have been received and which ones have not
- In **connectionless service**, this connection does not need to be established, therefore data frames can get lost, corrupted, duplicated, and out of order.
- In **connection-oriented service**, a path from the source to the destination must be established before any data packets can be sent. This connection is called a VC (virtual circuit).
- In **connectionless service**, no advance setup is needed. In this context, the packets are called datagrams. The datagrams carry the full destination address, and each is routed through the system independent of all the others.

The Application Layer (7th Layer of OSI)

The Application layer of the OSI model marks the spot where users communicate to the computer and comes into play only when access to the network will be needed soon.

Here are a few good examples of these kinds of events:

- File transfers
- Email
- Enabling remote access
- Network management activities
- Client/server processes
- Information location

The Presentation Layer (6th Layer of OSI)

The Presentation layer gets its name from its purpose: It presents data to the Application layer and is responsible for data translation and code formatting.

Performs the necessary conversion if two application processes use different syntaxes. Performs encryption using a secured key if requested by the Application Layer.

Translation. The processes (running programs) in two systems are usually exchanging information in the form of character strings, numbers, and so on.

Encryption. To carry sensitive information, a system must be able to ensure privacy. Encryption means that the sender transforms the original information to another form and sends the resulting message out over the network. Decryption reverses the original process to transform the message back to its original form.

Compression. Data compression reduces the number of bits contained in the information.

The Session Layer (5th Layer of OSI)

The Session layer is responsible for setting up, managing, and dismantling sessions between Presentation layer entities and keeping user data separate. Dialog control between devices also occurs at this layer. The PDU is – Message.

The session layer is the network dialog controller. It establishes, maintains, and synchronizes the interaction among communicating systems.

Dialog control. The session layer allows two systems to enter a dialog. It allows the communication between two processes to take place in either half-duplex (one way at a time) or full-duplex (two ways at a time) mode.

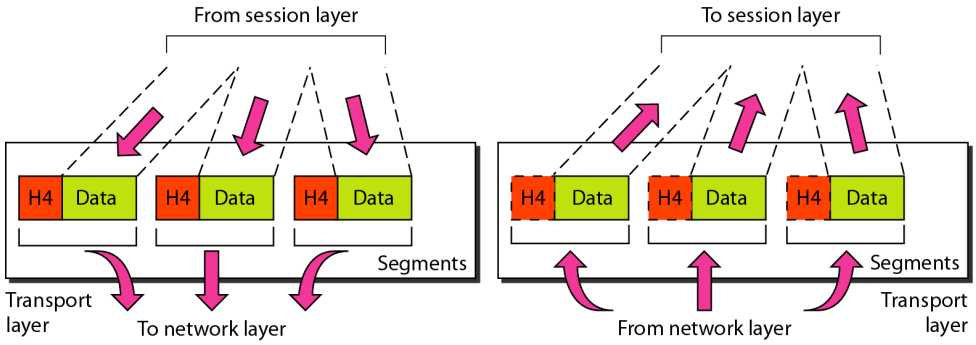
Synchronization. The session layer allows a process to add checkpoints, or synchronization points, to a stream of data.

The Transport Layer (4th Layer of OSI)

The Transport layer segments and reassembles data into a single data stream. Services located at this layer take all the various data received from upper-layer applications, then combine it into the same, concise data stream. These protocols provide end-to-end data transport services and can establish a logical connection between the sending host and destination host on an internetwork.

A pair of well-known protocols called TCP and UDP are integral to this layer.

The PDU is called **Segment.**



The Network Layer (3rd Layer of OSI)

The Network layer, or layer 3, manages device addressing, tracks the location of devices on the network, and determines the best way to move data. This means that it is up to the Network layer to transport traffic between devices that are not locally attached. Routers, which are layer 3 devices, are specified at this layer and provide the routing services within an internetwork.

The network layer is responsible for the delivery of individual packets from the source host to the destination host.

The PDU is called **Packet.**

Protocols – IP, RIP, EIGRP, OSPF

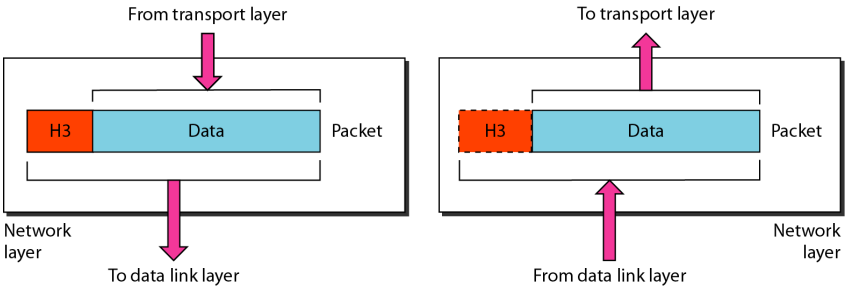
Data Packets These are used to transport user data through the internetwork.

Route Update Packets These packets are used to update neighboring routers about the networks connected to all routers within the internetwork.

Network Addresses Protocol-specific network addresses. **Logical Address** of the source and destination.

Interface The exit interface a packet will take when destined for a specific network.

Metric The distance to the remote network.

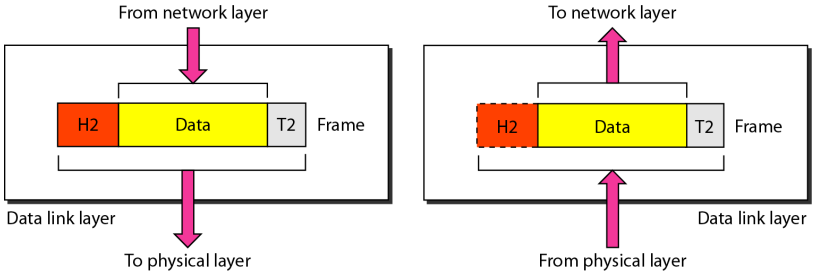


The Data Link Layer (Layer 2 of OSI)

The Data Link layer provides for the physical transmission of data and handles error notification, network topology, and flow control. This means that the Data Link layer will ensure that messages are delivered to the proper device on a LAN using hardware addresses and will translate messages from the Network layer into bits for the Physical layer to transmit.

The Data Link layer formats the messages, each called **a data frame,** and adds a customized header containing the hardware destination and source address.

The data link layer is responsible for moving frames from one hop (node) to the next.



Framing. The data link layer divides the stream of bits received from the network layer into manageable data units called frames.

Physical addressing. The data link layer adds a header to the frame to define the sender and/or receiver of the frame.

Flow control. The data link layer imposes a flow control mechanism to prevent overwhelming the receiver.

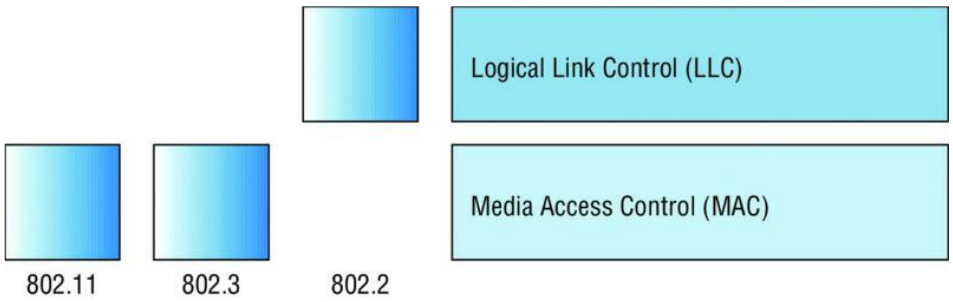
Error control. The data link layer adds reliability to the physical layer by adding mechanisms to detect and retransmit damaged or lost frames.

Access control. When two or more devices are connected to the same link, data link layer protocols are necessary to determine which device has control over the link at any given time.

The IEEE Ethernet Data Link layer has two sublayers:

Media Access Control (MAC) Defines how packets are placed on the media. Contention for media access is "first come/first served" access where everyone shares the same bandwidth— hence the name. Physical addressing is defined here as well as logical topologies.

Logical Link Control (LLC) Responsible for identifying Network layer protocols and then encapsulating them. An LLC header tells the Data Link layer what to do with a packet once a frame is received.



Physical Layer (Layer 1 of OSI)

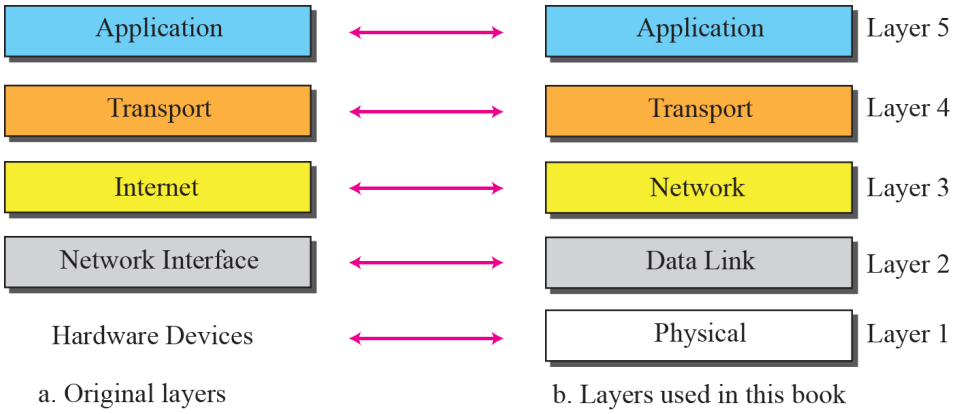
The physical layer coordinates the functions required to carry a bit stream over a physical medium. It deals with the mechanical and electrical specifications of the interface and transmission media.

The physical layer is responsible for moving individual bits from one (node) to the next.

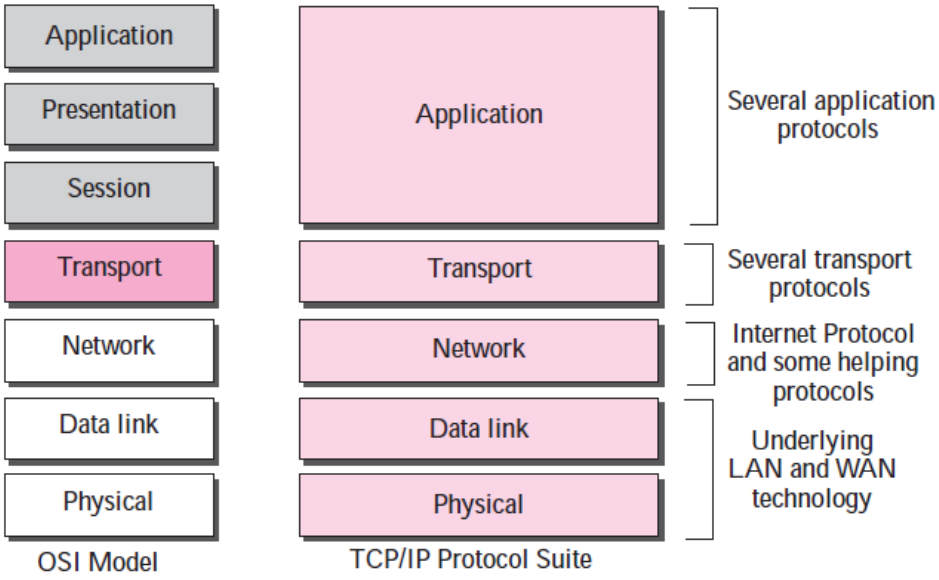
- **Physical characteristics of interfaces and media**. The physical layer defines the characteristics of the interface between the devices and the transmission media.
- **Representation of bits.** The physical layer data consists of a stream of bits (sequence of 0s or 1s) with no interpretation. To be transmitted, bits must be encoded into signals—electrical or optical.
- **Data rate.** The transmission rate—the number of bits sent each second—is also defined by the physical layer.
- **Synchronization of bits.** The sender and receiver must not only use the same bit rate but must also be synchronized at the bit level.
- **Line configuration**. The physical layer is concerned with the connection of devices to the media.
- **Physical topology.** The physical topology defines how devices are connected to make a network. Devices can be connected using a **mesh topology** (every device connected to every other device), a **star topology** (devices are connected through a central device), a **ring topology** (each device is connected to the next, forming a ring), or a **bus topology** (every device on a common link).
- **Transmission mode.** The physical layer also defines the direction of transmission between two devices**: simplex, half-duplex, or full duplex**. In the **simplex mode**, only one device can send; the other can only receive. The simplex mode is a one-way communication. In the **half-duplex mod**e, two devices can send and receive, but not at the same time. In a **full-duplex** (or simply duplex) mode, two devices can send and receive at the same time.

TCP/IP PROTOCOL SUITE

The TCP/IP protocol suite was developed prior to the OSI model. Therefore, the layers in the TCP/IP protocol suite do not match exactly with those in the OSI model. The original TCP/IP protocol suite was defined as four software layers built upon the hardware. Today, however, TCP/IP is thought of as a five-layer model with the layers named similarly to the ones in the OSI model.



Comparison of TCP/IP and OSI model



Physical Layer

TCP/IP does not define any specific protocol for the physical layer. It supports all the standard and proprietary protocols. At this level, the communication is between two hops or nodes, either a computer or router. The unit of communication is **a single bit**.

Data Link Layer

TCP/IP does not define any specific protocol for the data link layer either. It supports all of the standard and proprietary protocols. At this level, the communication is also between two hops or nodes. The unit of communication, however, is a packet called **a frame**.

Network Layer

At the network layer (or, more accurately, the internetwork layer), TCP/IP supports the Internet Protocol (IP). The Internet Protocol (IP) is the transmission mechanism used by the TCP/IP protocols. IP transports data in packets called datagrams.

Transport Layer

There is a main difference between the transport layer and the network layer. Although all nodes in a network need to have the network layer, only the two end computers need to have the transport layer. The network layer is responsible for sending individual datagrams from computer A to computer B; the transport layer is responsible for delivering the whole message, which is called a **segment**, a user datagram, or a packet, from A to B.

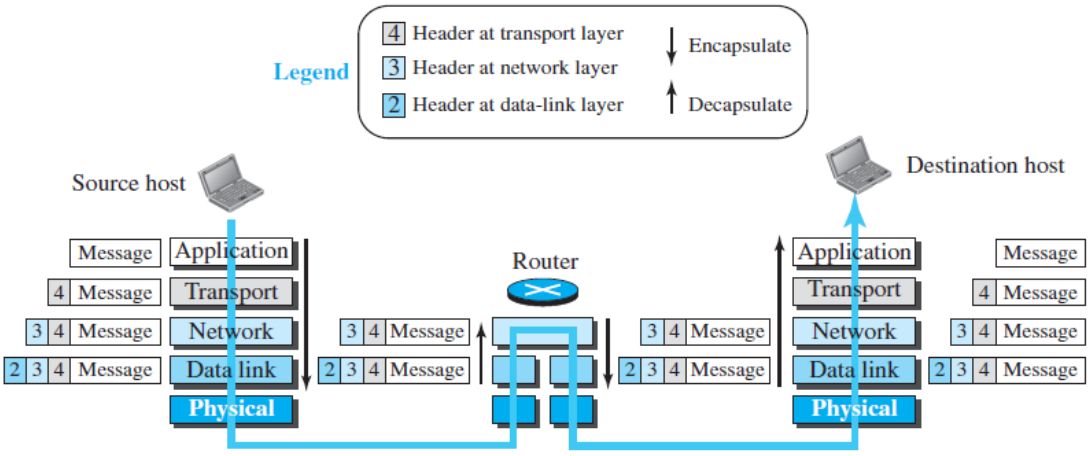
Application Layer

The application layer in TCP/IP is equivalent to the combined session, presentation, and application layers in the OSI model. The application layer allows a user to access the services of our private internet or the global Internet.

Encapsulation and Decapsulation

One of the important concepts in protocol layering in the Internet is encapsulation / decapsulation.

Figure 2.8 *Encapsulation/Decapsulation*



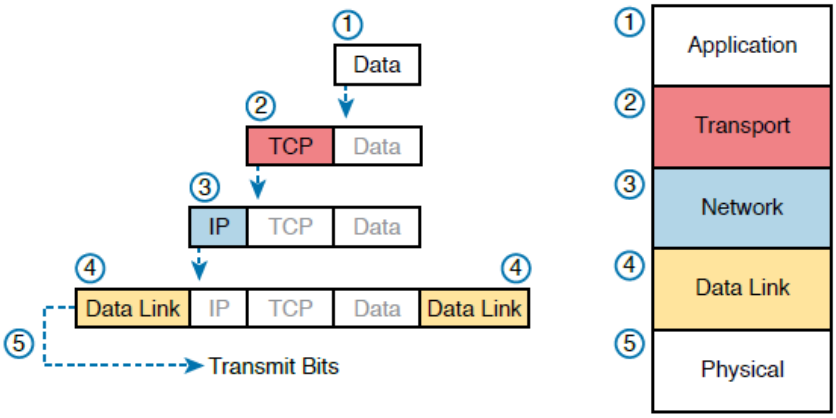


Figure 1-12 *Five Steps of Data Encapsulation: TCP/IP*

Encapsulation at the Source Host

Step 1. Create and encapsulate the application data with any required application layer headers.

Step 2. Encapsulate the data supplied by the application layer inside a transport layer header. For end-user applications, a TCP or UDP header is typically used.

Step 3. Encapsulate the data supplied by the transport layer inside a network layer (IP) header. IP defines the IP addresses that uniquely identify each computer.

Step 4. Encapsulate the data supplied by the network layer inside a data-link layer header and trailer. This layer uses both a header and a trailer.

Step 5. Transmit the bits. The physical layer encodes a signal onto the medium to transmit the frame.

Decapsulation and Encapsulation at the Router

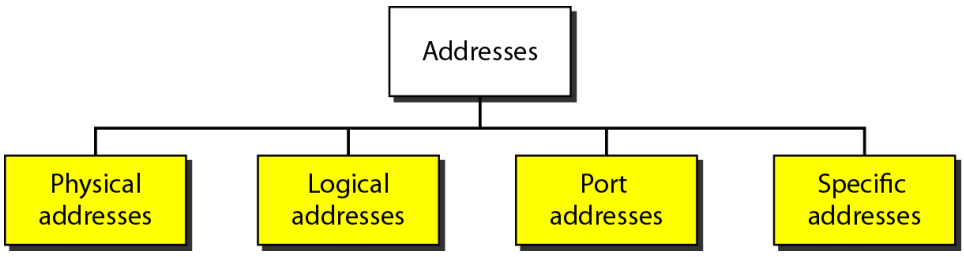
The router decapsulate and encapsulate the received PDUs on corresponding layers and move to next hope.

Decapsulation at the Destination Host

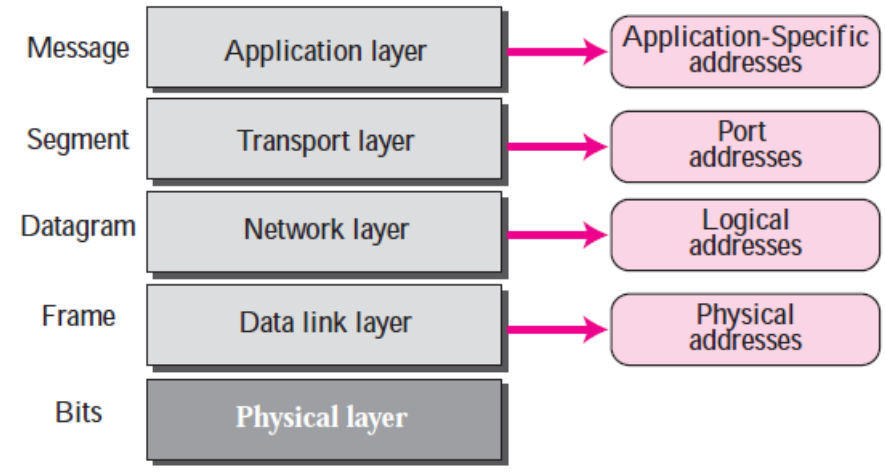
At destination host, each layer decapsulate the received packets and move upper layers.

ADDRESSING

Four levels of addresses are used in an internet employing the TCP/IP protocols: **physical address, logical address, port address, and application-specific address**. Each address is related to a one layer in the TCP/IP architecture.



Addresses in the TCP/IP Protocol Suite



Physical Address

Most local area networks use a 48-bit (6-byte) physical address written as 12 hexadecimal digits; every byte (2 hexadecimal digits) is separated by a colon, as shown below:

07:01:02:01:2C:4B

A 6-byte (12 hexadecimal digits) physical address

Unicast, Multicast, and Broadcast Physical Addresses

Physical addresses can be either unicast (one single recipient), multicast (a group of recipients), or broadcast (to be received by all systems in the network).

The physical addresses will change from hop to hop, but the logical addresses remain the same.

Logical Addresses

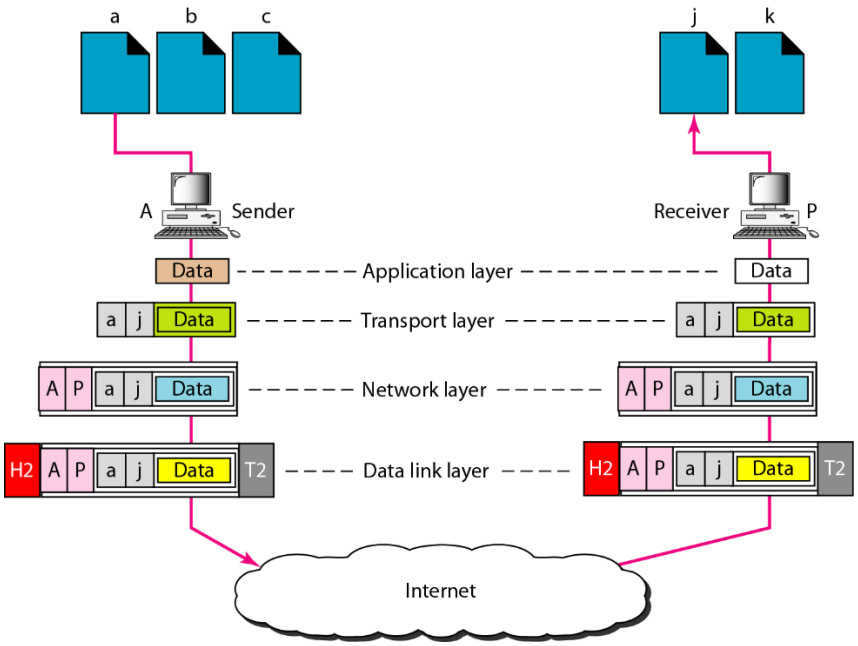
Logical addresses are necessary for universal communications that are independent of underlying physical networks. A universal addressing system is needed in which each host can be identified uniquely, regardless of the underlying physical network.

Unicast, Multicast, and Broadcast Addresses The logical addresses can be either unicast (one single recipient), multicast (a group of recipients), or broadcast (all systems in the network).

Port Addresses

For the multiple processes to receive data simultaneously, we need a method to label the different processes. In other words, they need addresses. In the TCP/IP architecture, the label assigned to a process is called a port address. A port address in TCP/IP is 16 bits in length.

The physical addresses change from hop to hop, but the logical and port addresses usually remain the same.



Application-Specific Addresses

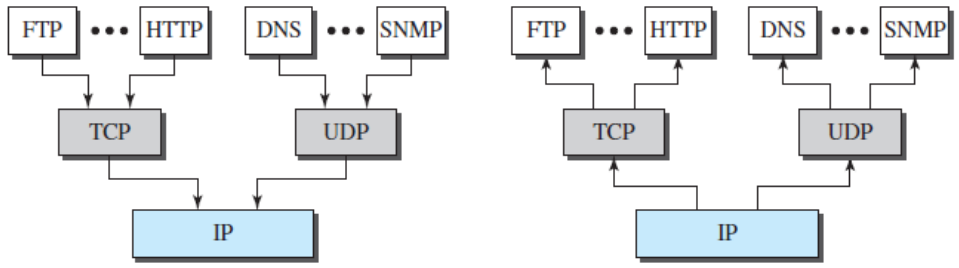
Some applications have user-friendly addresses that are designed for that specific application. Examples include the e-mail address (for example, forouzan@fhda.edu) and the Universal Resource

Locator (URL) (for example, www.mhhe.com). The first defines the recipient of an e-mail; the second is used to find a document on the World Wide Web.

Multiplexing and Demultiplexing

Since the TCP/IP protocol suite uses several protocols at some layers, we can say that we have multiplexing at the source and demultiplexing at the destination. Multiplexing in this case means that a protocol at a layer can encapsulate a packet from several next-higher layer protocols (one at a time); demultiplexing means that a protocol can decapsulate and deliver a packet to several next-higher layer protocols (one at a time).

Figure 2.10 *Multiplexing and demultiplexing*



a. Multiplexing at source b. Demultiplexing at destination

Physical Layer

the Physical layer does two things: it sends bits and receives bits. The Physical layer communicates directly with the various types of actual communication media. Different kinds of media represent these bit values in different ways. Some use *audio tones*, while others employ *state transitions*—changes in voltage from high to low and low to high.

The Physical layer specifies the electrical, mechanical, procedural, and functional requirements for activating, maintaining, and deactivating a physical link between end systems.

This layer is also where you identify the interface between the data terminal equipment (DTE) and the data communication equipment (DCE). The DCE is usually located at the service provider, while the DTE is the attached device.

- **Physical characteristics of interfaces and media**.
- **Representation of bits.**
- **Data rate.**
- **Synchronization of bits**
- **Line configuration**.
- **Physical topology.**
  - mesh topology
  - star topology
  - ring topology
  - bus topology
- **Transmission mode.**
  - simplex mode
  - half-duplex mode,
  - full-duplex.

Analog and Digital Data

Data can be analog or digital. The term **analog data** refers to information that is continuous; **digital data** refers to information that has discrete states.

Physical Layer Core function

Interfaces the electrical signals with the actual physical medium:

For example:

- Binary 1 = Electrical signal of +V volts
- Binary 0 = Electrical signal of -V volts

The receiver must detect the binary 1 or 0 by examining the electrical signal that has arrived even if it has suffered from attenuation and distortion.

Transmission Media

The following are examples of physical media that can be used for the transmission of electrical signals:

- Twisted pair lines
- Coaxial cables
- Optical fiber
- Satellites
- Microwave and radio
- High frequency radio

Datalink Layer

The two main functions of the data link layer are **data link control** and **media access control**. The first, data link control, deals with the design and procedures for communication between two adjacent nodes: node-to-node communication.

Data link control functions include framing, flow and error control, and software implemented protocols that provide smooth and reliable transmission of frames between nodes.

Framing: Layer divides the stream of bits received from the network layer into manageable data units called **frames.**

Physical Addressing: Adds a header to the frame to define the sender and / or receiver of the frame. If the frame outside sender network the receiver address is the address of the connecting device that connects the network to the next one.

- Flow Control
- Error Control
- Media Access Control MAC
- Data Link Control
- Access Control

Framing Methods

Framing in data link layer separates a message from one source to a destination, or from other messages to other destinations by adding a sender address and a destination address. The

destination address defines where the packet is to go. The sender address helps the recipient acknowledge the receipt.

Fixed size framing

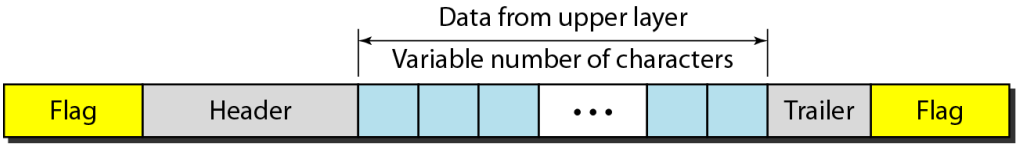
The size is delimiter of the frame the size of the frame is fixed.

Variable size framing

In variable size framing we need to define the end of the frame and the beginning of the next. There are two approaches. a character-oriented approach and a bit-oriented approach.

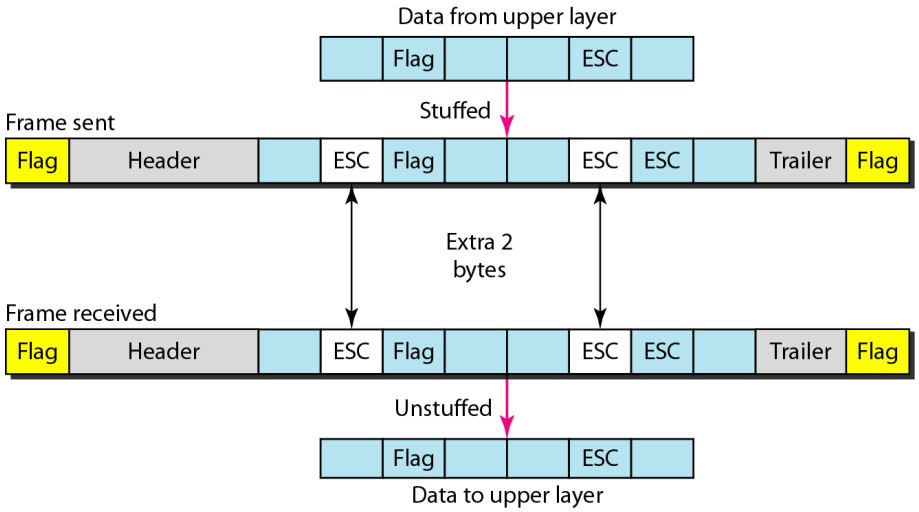
a. **Character Oriented Protocols**

In a character-oriented protocol, data to be carried are 8-bit characters from a coding system such as ASCII. To separate one frame from the next, an 8-bit (I-byte) flag is added at the beginning and the end of a frame. The flag, composed of protocol-dependent special characters,



Byte Stuffing and Unstuffing.

A special byte is added to the data section of the frame. The data source is stuffed with an extra byte. This byte is usually called the escape character.

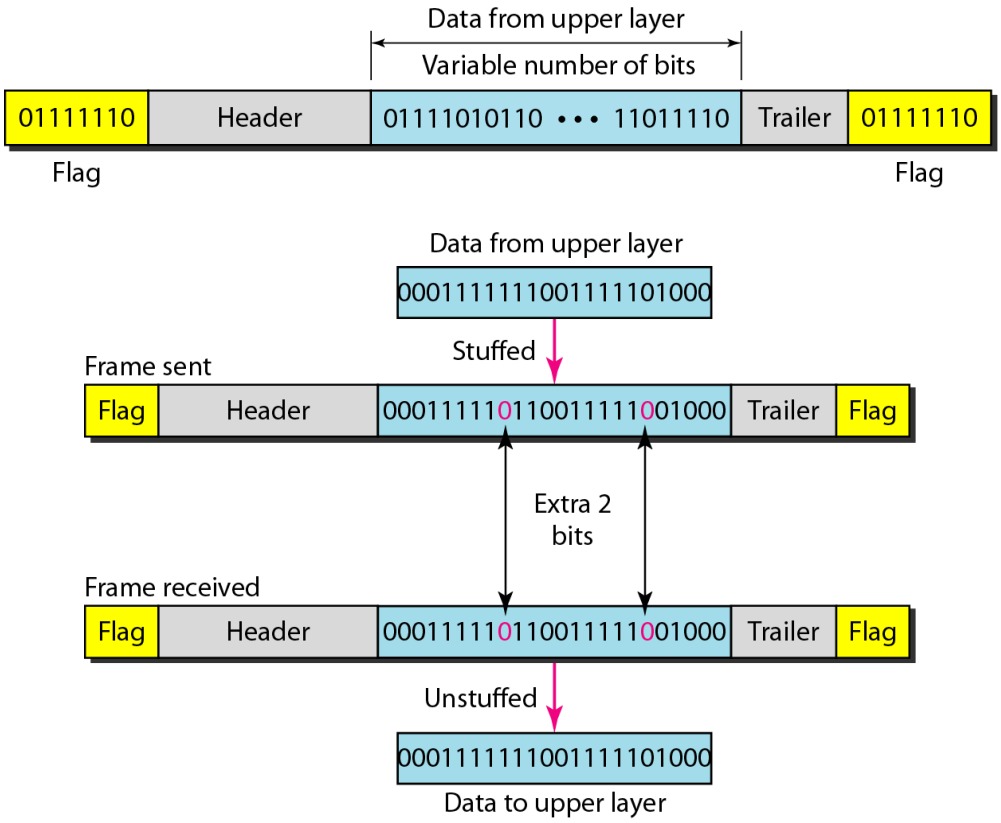


b. **Bit Oriented Protocols**

In a bit-oriented protocol, the data section of a frame is a sequence of bits to be interpreted by the upper layer as text, graphic, audio, video, and so on.

In addition to the header, we need a delimiter to separate one frame from the other. The delimiter to define the beginning and the end of the frame.

Bit Stuffing Bit stuffing is the process of adding one extra 0 whenever five consecutive 1's follows a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

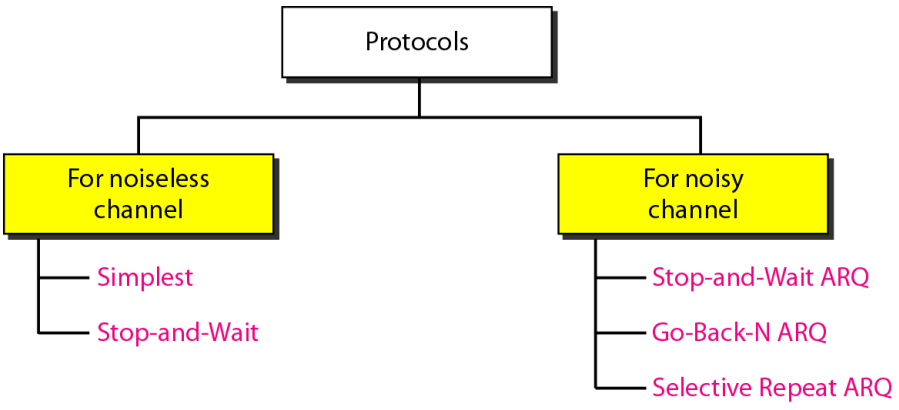


Flow and Error Control

Flow control refers to a set of procedures used to restrict the amount of data that sender can send before waiting for the acknowledgement.

Error Control in the data link layer is based on automatic repeat request which is the re-transmission of the data.

Protocols for the Flow & Error Control

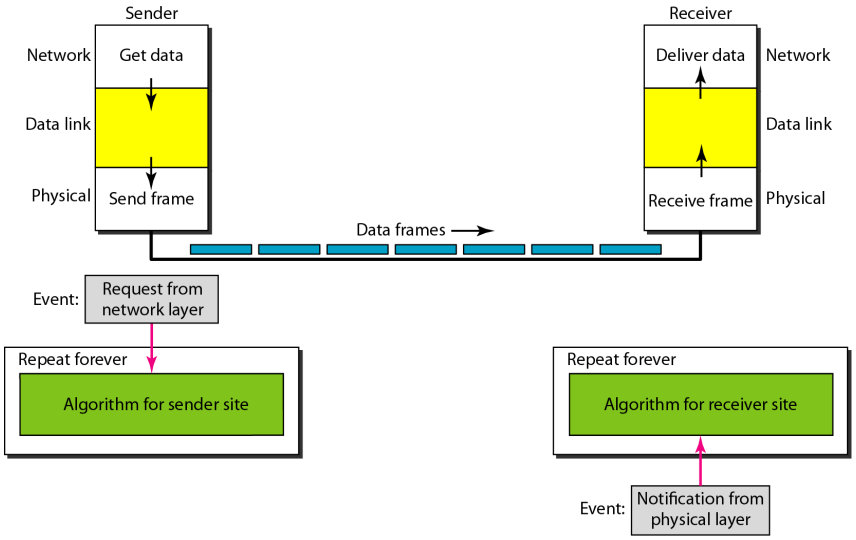


Noiseless channels

In which no frames are lost, duplicated, or corrupted.

Simplest Protocol

It is the unidirectional protocol in which data frames are travelling in only one direction from the sender to the receiver.



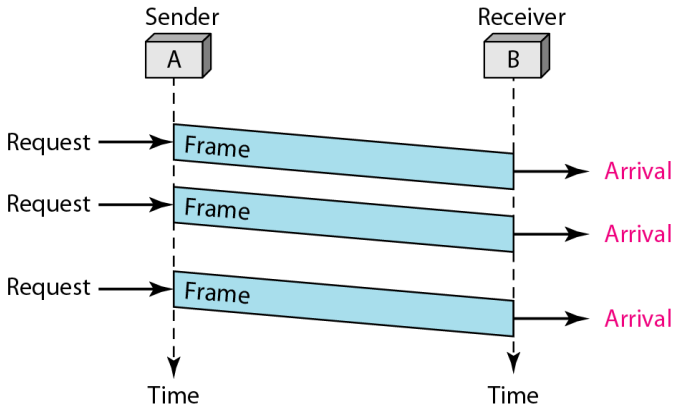
Simplest Protocol Algorithm – Sender Side

```
1  while(true)                          // Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(RequestToSend))           //There is a packet to send
5    {
6       GetData();
7       MakeFrame();
8       SendFrame();                    //Send the frame
9    }
10 }
```

Simplest Protocol Algorithm Receiver Side

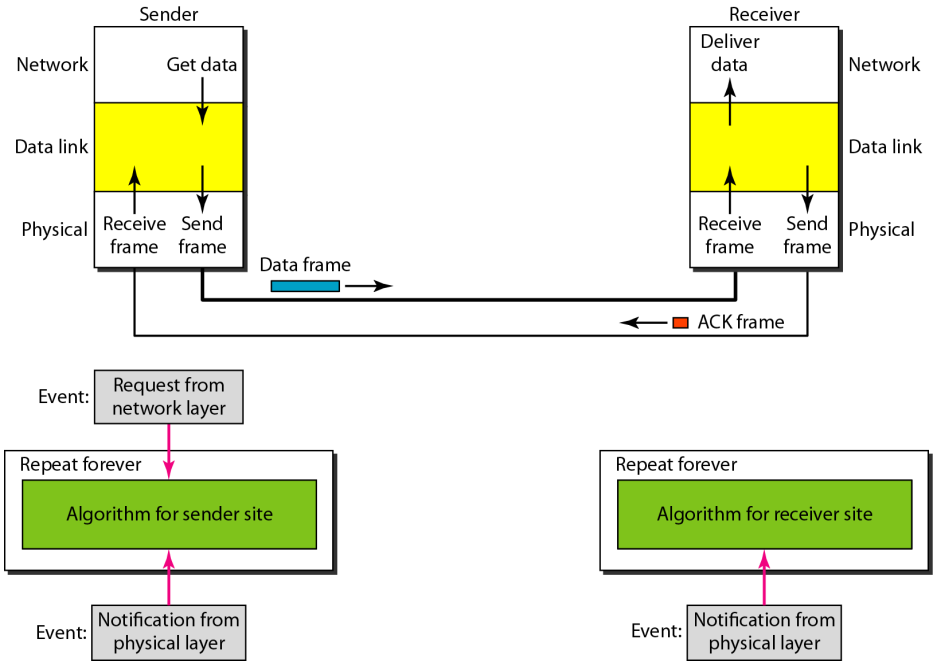
```
1  while(true)                          // Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(ArrivalNotification))     //Data frame arrived
5    {
6       ReceiveFrame();
7       ExtractData();
8       DeliverData();                  //Deliver data to network layer
9    }
10 }
```

Flow Diagram Simplest Protocol



Stop and Wait Protocol

Sender sends a frame stops until it receives confirmation from the receiver (OK to go ahead) and sends next frame after that.



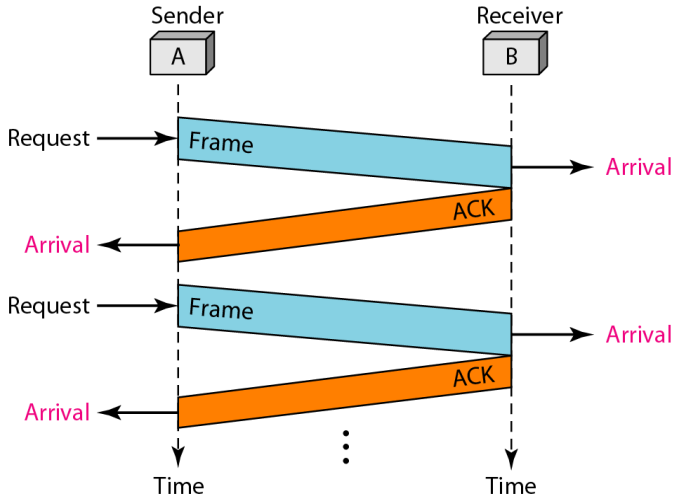
Sender-site algorithm for Stop-and-Wait Protocol

```
1  while(true)                          //Repeat forever
2  canSend = true                       //Allow the first frame to go
3  {
4    WaitForEvent();                    // Sleep until an event occurs
5    if(Event(RequestToSend) AND canSend)
6    {
7       GetData();
8       MakeFrame();
9       SendFrame();                    //Send the data frame
10      canSend = false;                //Cannot send until ACK arrives
11   }
12   WaitForEvent();                    // Sleep until an event occurs
13   if(Event(ArrivalNotification)      // An ACK has arrived
14    {
15      ReceiveFrame();                 //Receive the ACK frame
16      canSend = true;
17    }
18 }
```

Receiver-site algorithm for Stop-and-Wait Protocol

```
1  while(true)                          //Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(ArrivalNotification))     //Data frame arrives
5    {
6       ReceiveFrame();
7       ExtractData();
8       Deliver(data);                  //Deliver data to network layer
9       SendFrame();                    //Send an ACK frame
10   }
11 }
```

Flow Diagram Stop and Wait Protocol



Noisy Channels

Noiseless channels are not available, so we need to add error control to the protocol in order to avoid data loss.

Sliding Window Protocol

In this flow control method, the receiver allocates buffer space for n frames in advance and allows transmission of multiple frames.

- This method allows the sender to transmit n frames without an ACK. A k-bit sequence number is assigned to each frame. The range of sequence number uses modulo-2 arithmetic.
- To keep track of the frames that have been acknowledged, each ACK has a sequence number. The receiver acknowledges a frame by sending an ACK that includes the Sequence number of the next expected frame.
- The sender sends the next n frames starting with the last received sequence number that has been transmitted by the receiver (ACK). Hence, a single ACK can acknowledge multiple frames.

Operation of a Sliding Window

Sending Window & Receiving Window:

In this mechanism we maintain two types of windows (buffer) sending window and receiving windows. Sender is permitted to send frames with sequence numbers in a certain range called **Sending window**. The receiver maintains a **receiving window** depending on the sequence numbers of frames that are accepted.

Stop and Wait ARQ (Automatic Repeat Request)

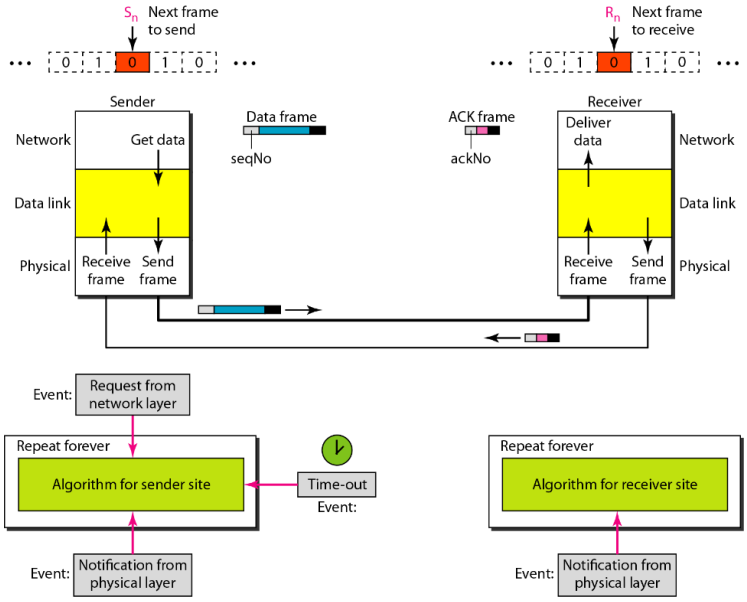
Error correction in Stop-and-Wait ARQ is done by keeping a copy of the sent frame and retransmitting of the frame when the timer expires.

- It is an error control mechanism.
- Keeping a copy of the send frame and re-transmitting of the frame when timer expires.
- Sequencing the frame with sequence number. The sequence numbers are based on modulo 0-2 arithmetic (Modulo 2).
- Acknowledgement number always announces in modulo 0-2 arithmetic the sequence number of the next frame expected.

Efficiency

It is having inefficient bandwidth utilization limitation and single frame pipelining.

Design of the Stop-and-Wait ARQ Protocol



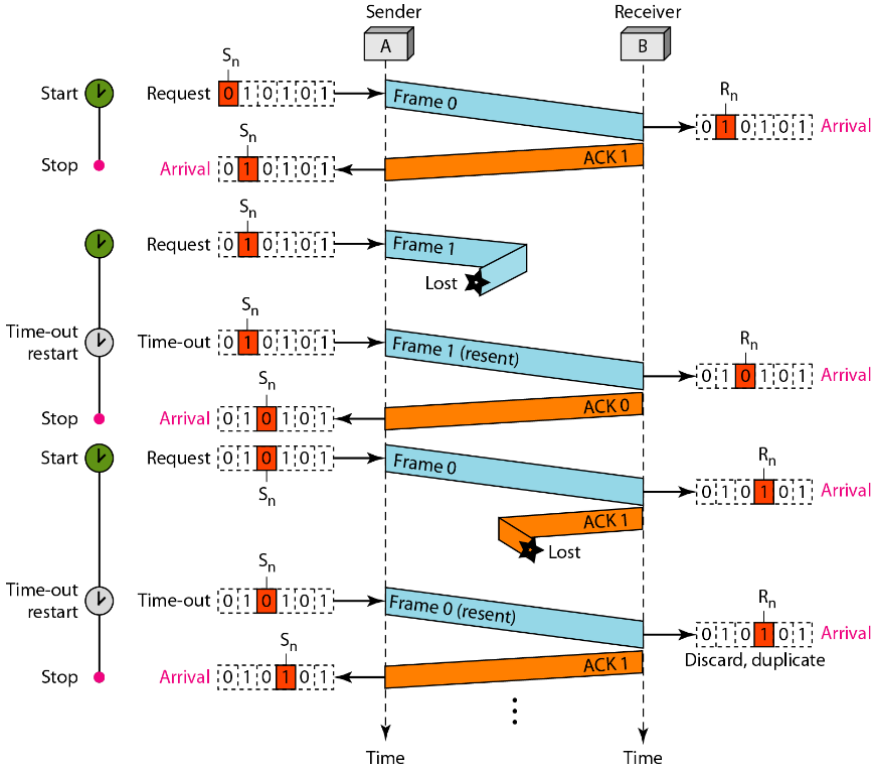
Sender-site algorithm for Stop-and-Wait ARQ

```
1  Sn = 0;                          // Frame 0 should be sent first
2  canSend = true;                  // Allow the first request to go
3  while(true)                      // Repeat forever
4  {
5    WaitForEvent();                // Sleep until an event occurs
6    if(Event(RequestToSend) AND canSend)
7    {
8       GetData();
9       MakeFrame(Sn);                    //The seqNo is Sn
10      StoreFrame(Sn);                   //Keep copy
11      SendFrame(Sn);
12      StartTimer();
13      Sn = Sn + 1;
14      canSend = false;
15   }
16   WaitForEvent();                  // Sleep
17     if(Event(ArrivalNotification)  // An ACK has arrived
18     {
19       ReceiveFrame(ackNo);         //Receive the ACK frame
20       if(not corrupted AND ackNo == Sn) //Valid ACK
21         {
22           Stoptimer();
23           PurgeFrame(Sn-1);              //Copy is not needed
24           canSend = true;
25         }
26      }
27
28     if(Event(TimeOut)              // The timer expired
29     {
30      StartTimer();
31      ResendFrame(Sn-1);                //Resend a copy check
32     }
33 }
```

Receiver-site algorithm for Stop-and-Wait ARQ Protocol

```
1  Rn = 0;                          // Frame 0 expected to arrive first
2  while(true)
3  {
4    WaitForEvent();                // Sleep until an event occurs
5    if(Event(ArrivalNotification))  //Data frame arrives
6    {
7       ReceiveFrame();
8       if(corrupted(frame));
9          sleep();
10      if(seqNo == Rn)              //Valid data frame
11      {
12       ExtractData();
13        DeliverData();             //Deliver data
14        Rn = Rn + 1;
15      }
16       SendFrame(Rn);             //Send an ACK
17   }
18 }
```

Flow diagram for Stop and Wait ARQ

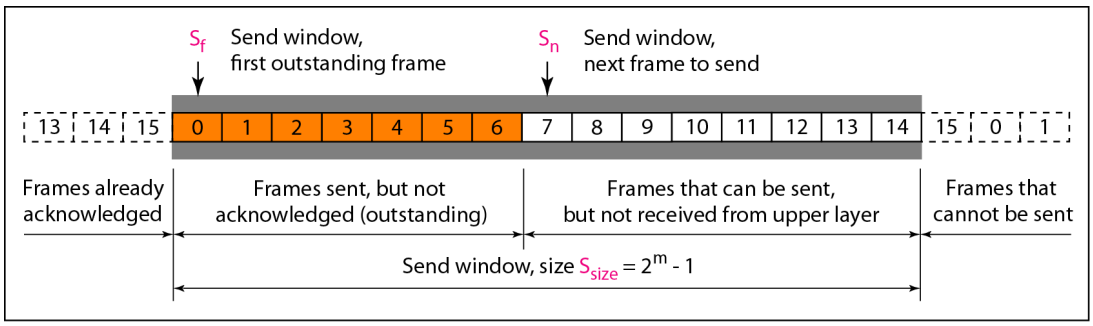


Go-Back-N Automatic Repeat Request

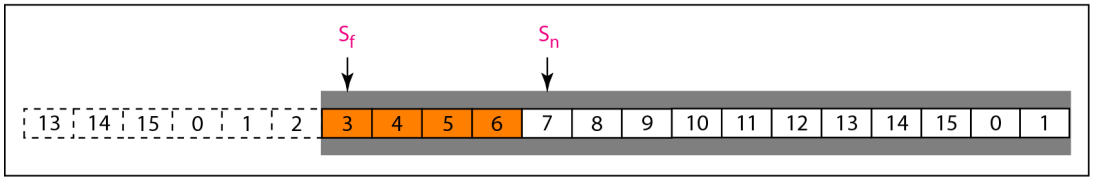
To improve the efficiency of transmission (filling the pipe), multiple frames must be in transition while waiting for acknowledgment.

- We can send several frames before receiving acknowledgements arrive.
- Sequence numbers – $0\text{-}2^m\text{-}1$
- Modulo 2^m where 'm' is the size of the sequence number field in bits.

- **Sliding window** – In this protocol (and the next), the sliding window is an abstract concept that defines the range of sequence numbers that is the concern of the sender and receiver. In other words, the sender and receiver need to deal with only part of the possible sequence numbers.
- Send window can slide one or more when a valid acknowledgement arrives.
- Window size 2^m-1.

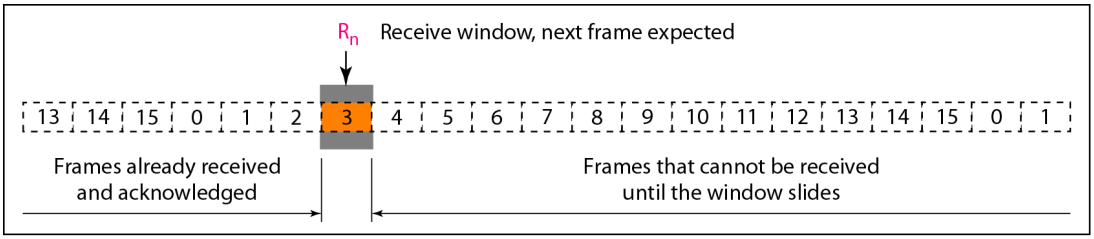


a. Send window before sliding

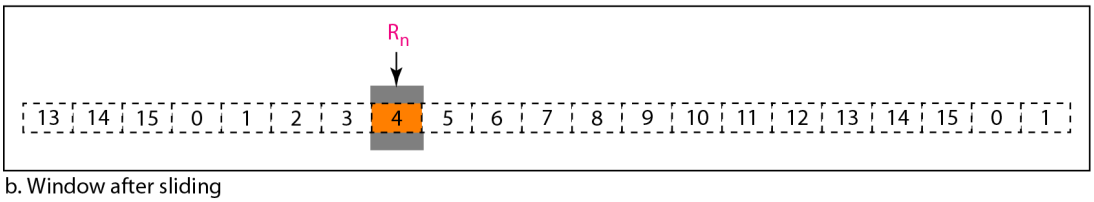


b. Send window after sliding

Receive window for Go-Back-N ARQ



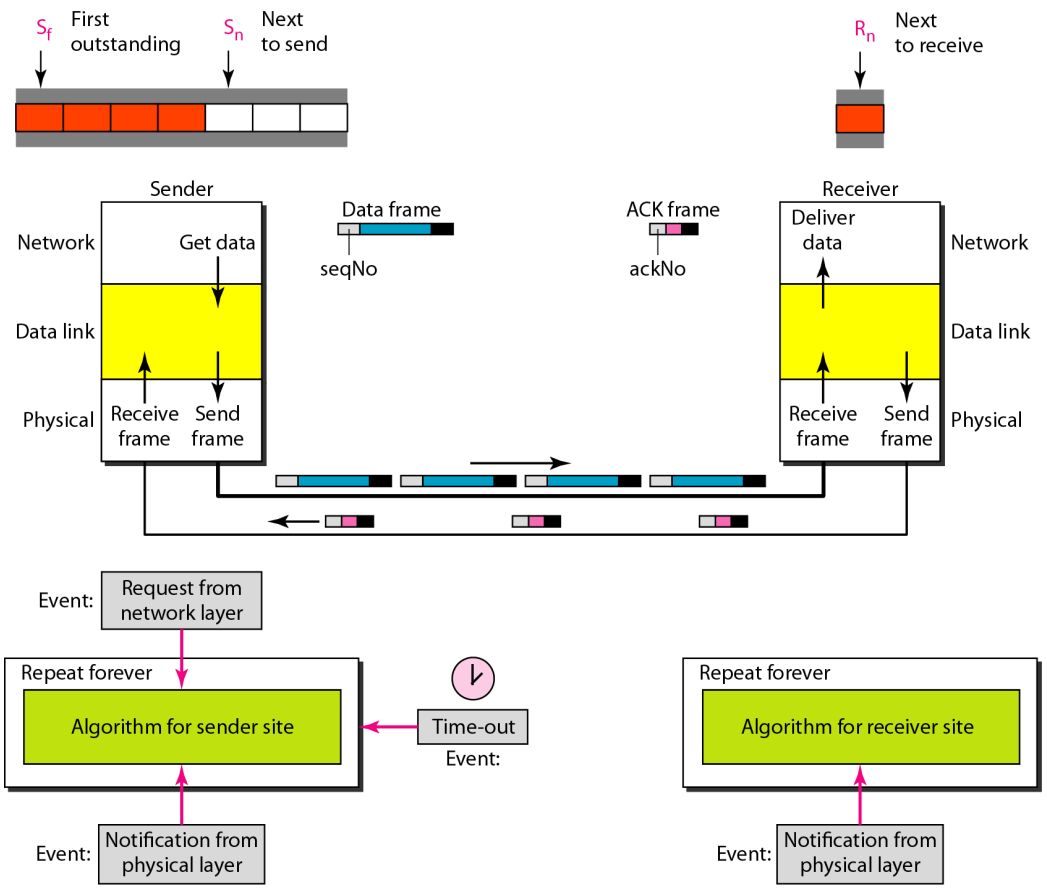
a. Receive window



b. Window after sliding

The receive window is an abstract concept defining an imaginary box of size 1 with one single variable Rn. The window slides when a correct frame has arrived; sliding occurs one slot at a time.

- **Timer**: there can be timer for each frame sent acknowledgement.
- **Acknowledgement**: the receiver sends a positive acknowledgement
- **Repeating a frame**: when the timer expires the sender re-sends all outstanding frames.
- **Size of the Window**: Size of the Send window must be less than 2^m. And the size of the receive window is always 1.



Go-Back-N sender algorithm.

```
1  Sw = 2^m - 1;
2  Sf = 0;
3  Sn = 0;
4
5  while (true)                        //Repeat forever
6  {
7   WaitForEvent();
8    if(Event(RequestToSend))          //A packet to send
9    {
10      if(Sn-Sf >= Sw)                //If window is full
11            Sleep();
12      GetData();
13      MakeFrame(Sn);
14      StoreFrame(Sn);
15      SendFrame(Sn);
16      Sn = Sn + 1;
17      if(timer not running)
18           StartTimer();
19   }
20
```

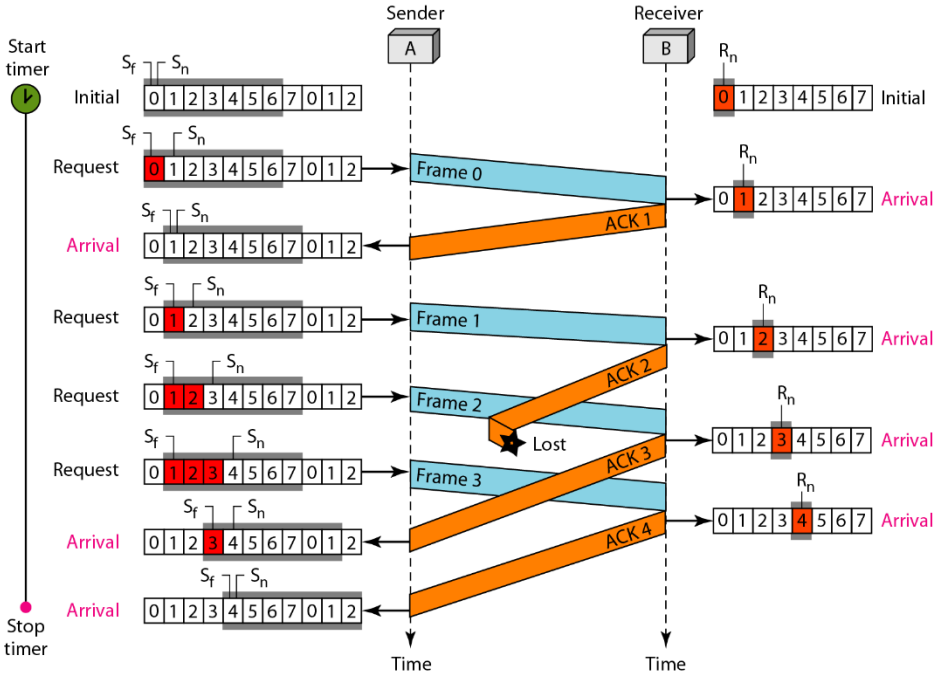
```
21   if(Event(ArrivalNotification))  //ACK arrives
22   {
23      Receive(ACK);
24      if(corrupted(ACK))
25           Sleep();
26      if((ackNo>Sf)&&(ackNo<=Sn))  //If a valid ACK
27      While(Sf <= ackNo)
28       {
29        PurgeFrame(Sf);
30        Sf = Sf + 1;
31       }
32       StopTimer();
33   }
34
35   if(Event(TimeOut))               //The timer expires
36   {
37    StartTimer();
38    Temp = Sf;
39    while(Temp < Sn);
40     {
41      SendFrame(Sf);
42      Sf = Sf + 1;
43     }
44   }
45 }
```

Go-Back-N receiver algorithm.

```
1  Rn = 0;
2
3  while (true)                     //Repeat forever
4  {
5    WaitForEvent();
6
7    if(Event(ArrivalNotification)) /Data frame arrives
8    {
9       Receive(Frame);
10      if(corrupted(Frame))
11           Sleep();
12      if(seqNo == Rn)              //If expected frame
13      {
14        DeliverData();             //Deliver data
15        Rn = Rn + 1;               //Slide window
16        SendACK(Rn);
17      }
18   }
19 }
```

Stop-and-Wait ARQ is a special case of Go-Back-N ARQ in which the size of the send window is 1.

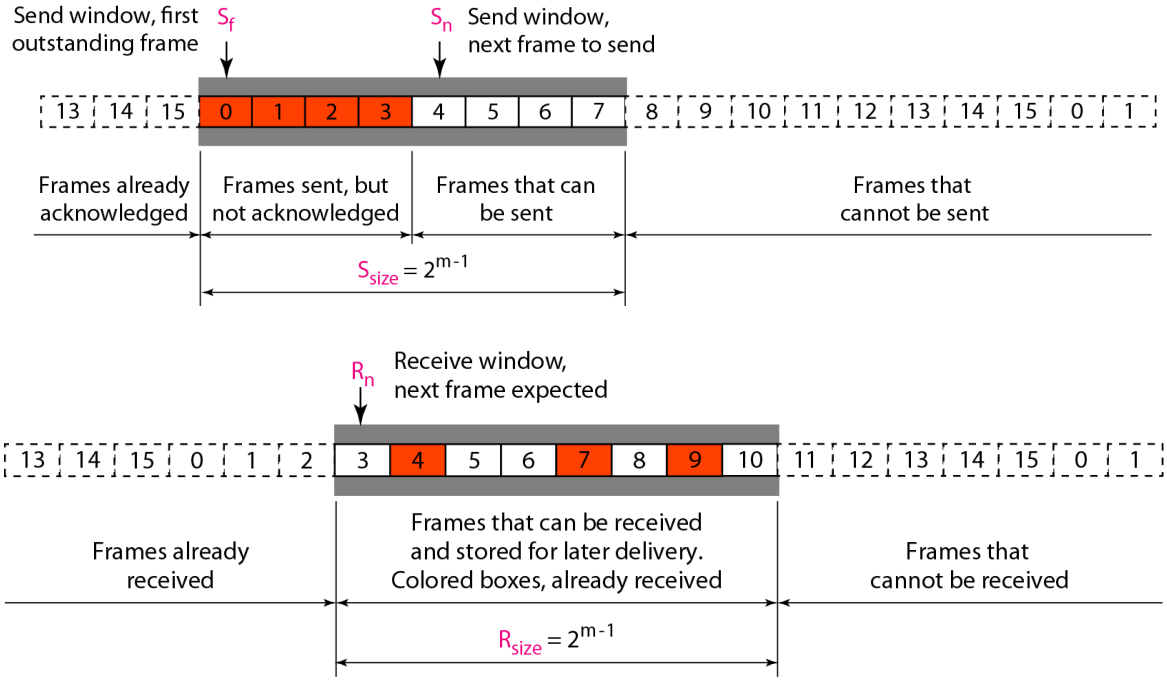
Flow diagram for Go Back N ARQ



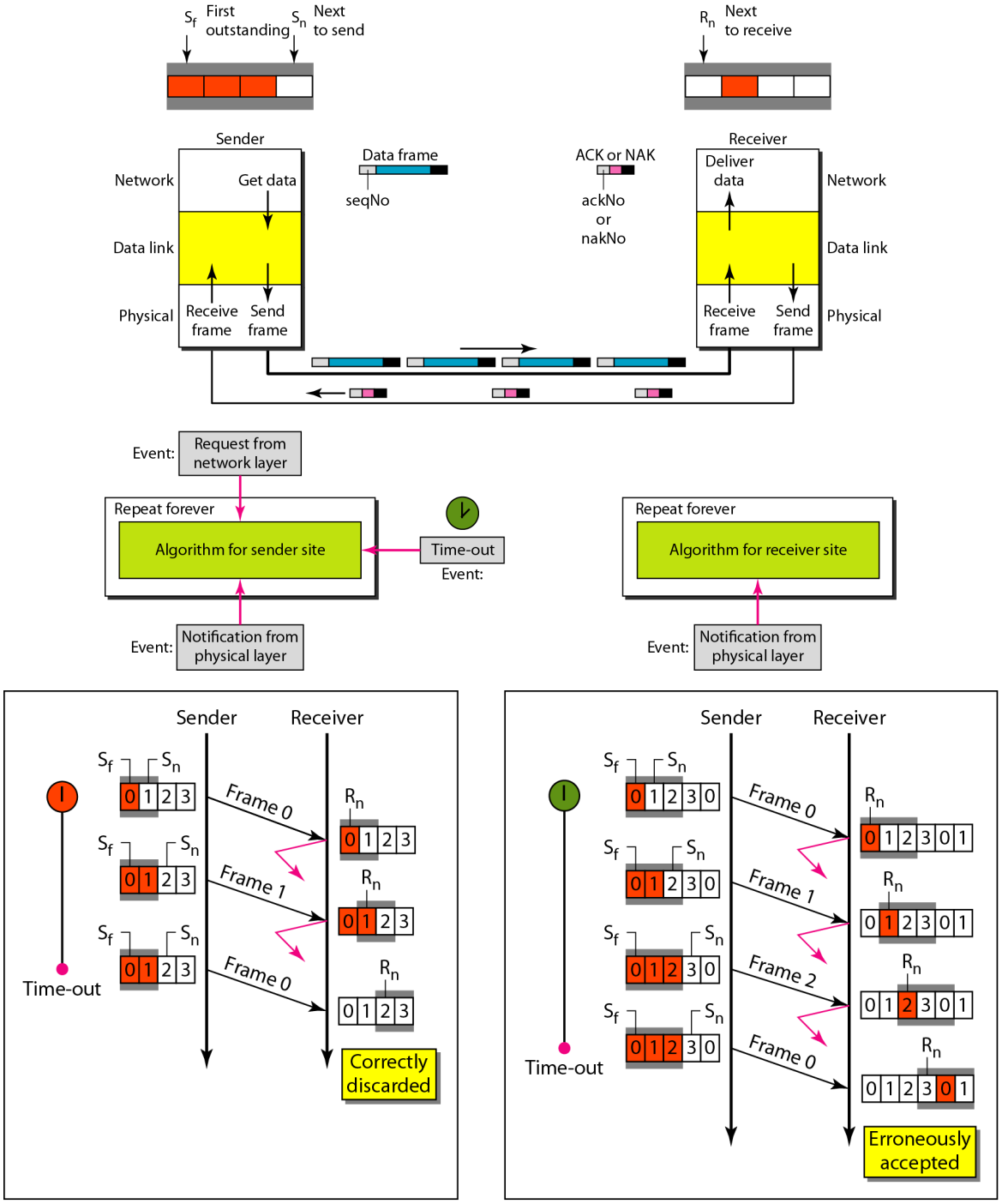
Selective Repeat Automatic Repeat Request

In a noisy link a frame has a higher probability of damage. Which means resending of multiple frames.

- Selective repeat ARQ does not respond N frames, only damaged frame is resent.
- It is more efficient in noisy links.
- Selective repeat protocol also uses two windows.
- Send window size 2^m-1
- Receive window size is also 2^m-1
- The window if m is 4 then the window size is 8 (0 – 7)



Design of Selective Repeat ARQ

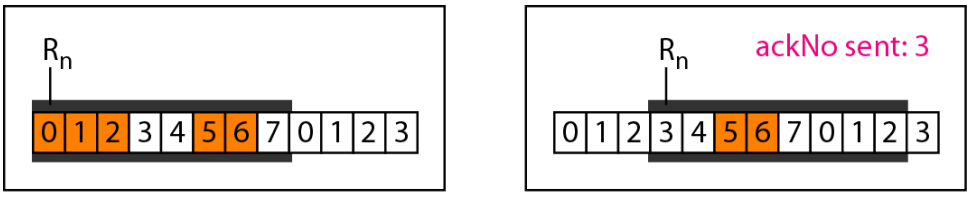


a. Window size = 2^{m-1}

b. Window size > 2^{m-1}

In Selective Repeat ARQ, the size of the sender and receiver window must be at most one-half of 2^m.

Delivery of data in Selective Repeat ARQ

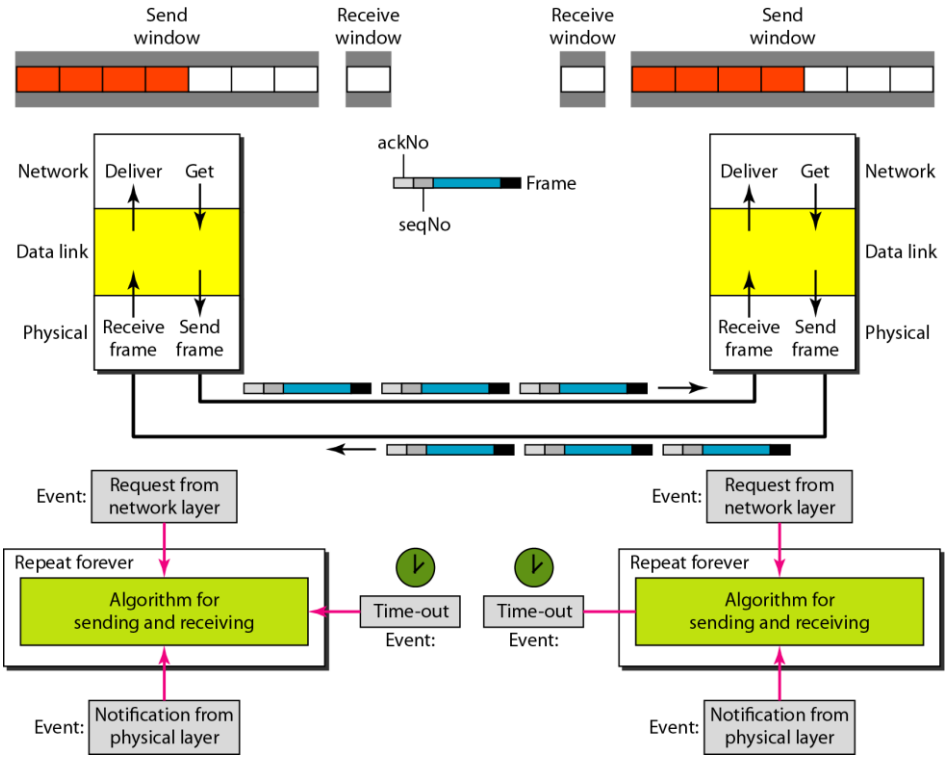


a. Before delivery

b. After delivery

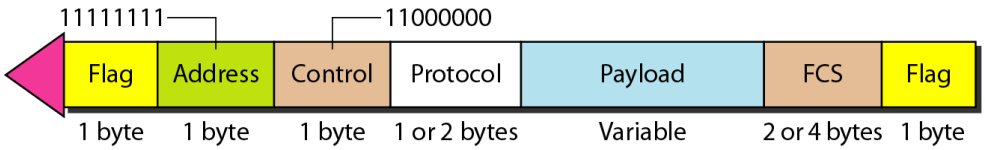
Piggy Backing

A technique used to improve the efficiency of the bi-directional protocols, when a frame is carrying data from A to B, it can also carry control info about arrived or lost frames from B and vice versa.



POINT-TO-POINT PROTOCOL

Although HDLC is a general protocol that can be used for both point-to-point and multipoint configurations, one of the most common protocols for point-to-point access is the Point-to-Point Protocol (PPP). PPP is a byte-oriented protocol.



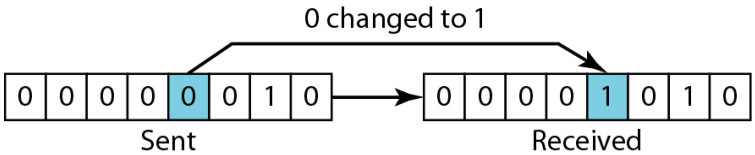
Error Detection and Correction

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

Types of Errors

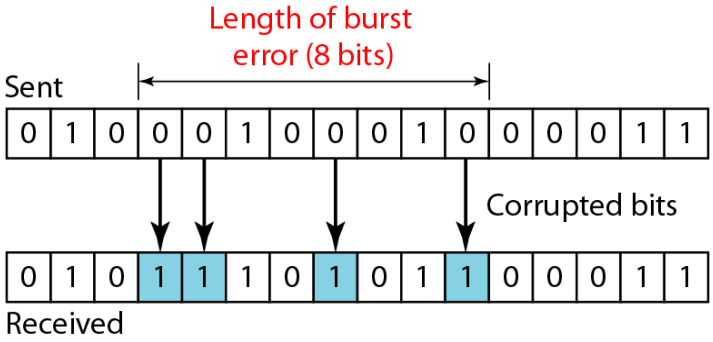
Whenever bits flow from one point to another, they are subject to unpredictable changes because of interferences.

a. **Single Bit Errors**
   Only one bit of given data unit is changed from 1 to 0 or 0 to 1.



b. Burst Error

It means two or more data bits have changed.



Redundancy

The central concept in detecting or correcting errors is redundancy. To be able to detect or correct errors we need to send some extra bits with our data.

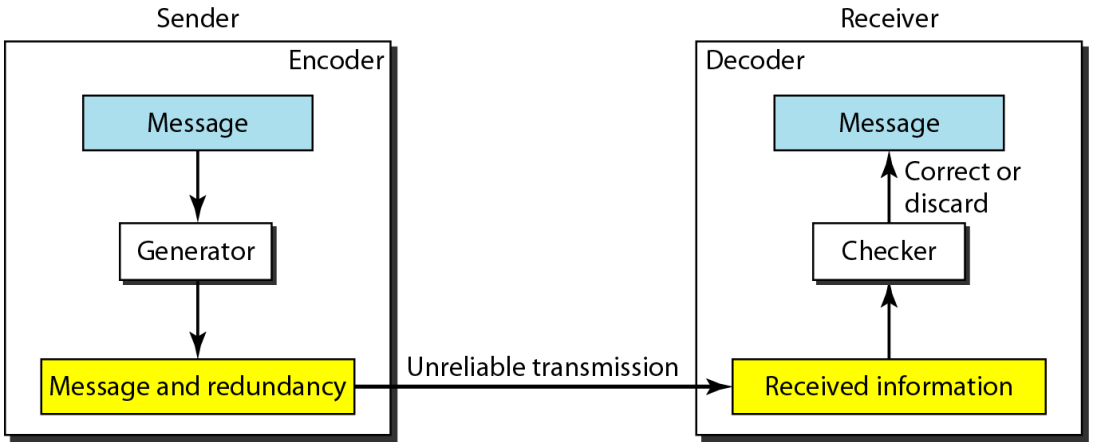
Detection versus Correction.

There are two main methods of error correction. **Error correction** is the process in which the receiver tries to guess the message by using redundant bits. This is possible, as we see later, if the number of errors is small. **Correction by retransmission** is a technique in which the receiver detects the occurrence of an error and asks the sender to resend the message. Resending is repeated until a message arrives that the receiver believes is error-free.

Coding

Redundancy is achieved through various coding schemas. The sender adds redundant bits through a process that creates relationship between the redundant bits and the actual bits.

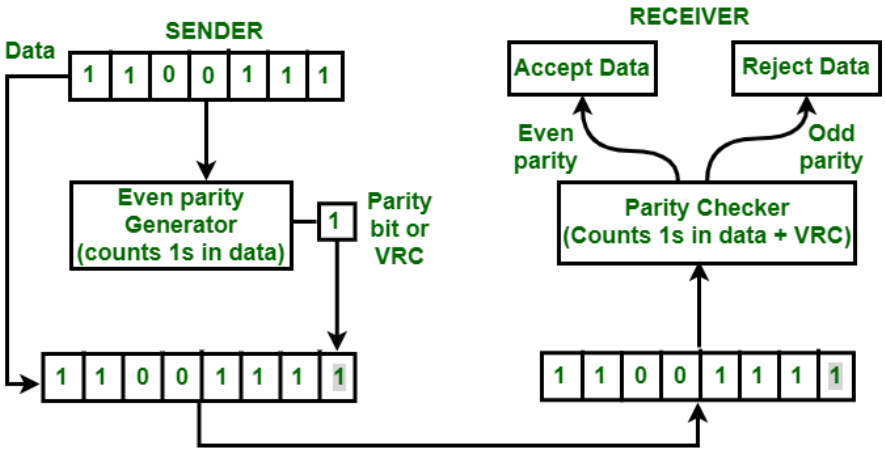
The structure of encoding and decoding.



Error Correction Methods

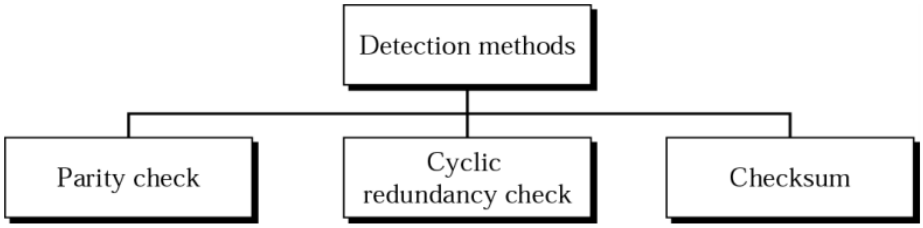
1. VRC – Vertical Redundancy Check (Parity Check)
2. LRC- Longitude Redundancy Check (longitude check)
3. Checksum
4. CRC – Cyclic Redundancy Check

Parity Check (VRC – Vertical Parity Check)



Parity bit

- One additional bit per character
- Even parity
- Odd parity
- One parity bit can detect one odd number error.



Single Parity Check

- One extra bit called as parity bit is sent along with the original data bits.
- Parity bit helps to check if any error occurred in the data during the transmission.

Step1.

a. The total number of 1's in the data unit to be transmitted is counted.
b. Total number of 1's in the data unit is made even in case of even parity or made odd in case of odd parity.

Step2. Newly formed word (original data + parity bit is transmitted to the receiver.

Step3.

- Receiver receives the transmitted code. The total number of 1's in the received code word is counted.
- If total no of 1's is even if even parity is used, then no error
- If total of 1's is odd if even parity is used, then there is error.

Sequence of 7 bits	Even Parity	Odd Parity
0100010	01000100	01000101
1000000	10000001	10000000

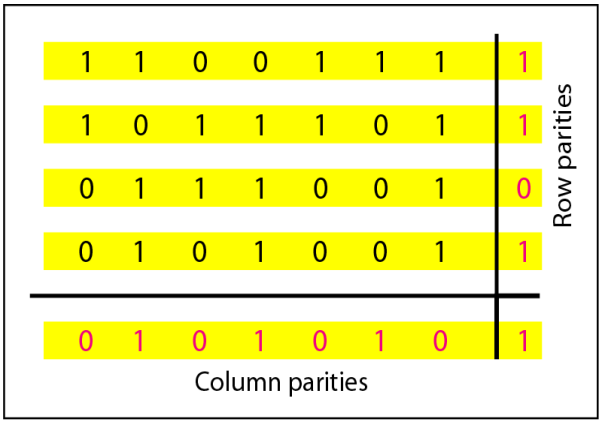
Drawbacks of parity check

- It can detect only single bit error.
- It can detect only burst error if the no of errors is odd.

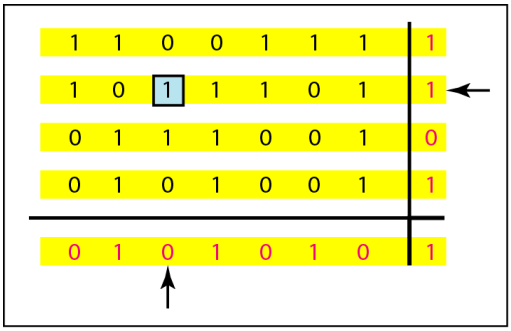
Eg. 11100001 → 1010001 Error | 10100101 – no error.

Longitudinal (2D parity check) and Its performance

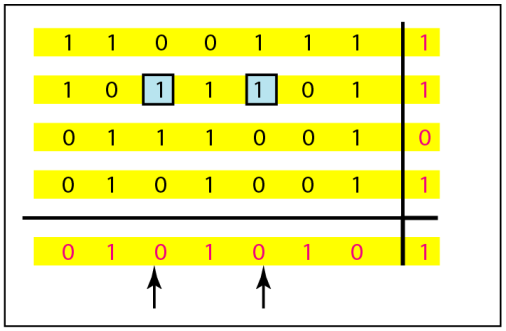
Longitudinal redundancy checks also known as 2-d parity check, which the user wants to send is organized into tables or rows and columns.



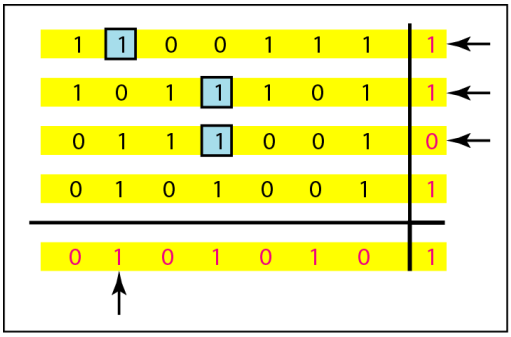
a. Design of row and column parities



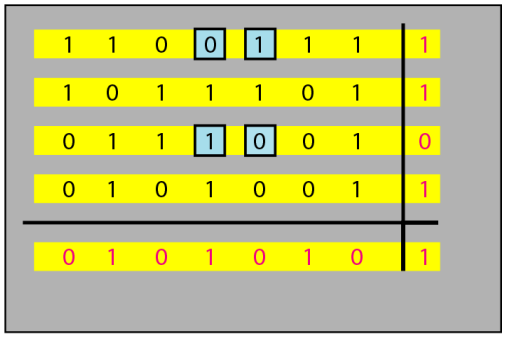
b. One error affects two parities



c. Two errors affect two parities



d. Three errors affect four parities



e. Four errors cannot be detected

Advantage

Used to detect burst errors.

Disadvantages

It is not able to detect errors if two bits in data unit are damaged. And two bits is exactly same position in the data unit are also damaged.

Cyclic Redundancy Code (CRC)

The most commonly used method for detecting burst error in the data stream is Cyclic Redundancy Check Method. This method is based on the use of polynomial codes. Polynomial codes are based on representing bit strings as polynomials with coefficients as 0 and 1 only.

For example, the bit string 1110011 can be represented by the following polynomial:

$$x6 + x5 + x4 + x1 + 1.$$

- The polynomial is manipulated using modulo 2 arithmetic (which is equivalent to Exclusive OR or XOR).
- Depending on the content of the frame a set of check digits is computed for each frame that is to be transmitted. Then the receiver performs the same arithmetic as the sender on the frame and check the digits. If the result after computation is the same, then the data received is error free.
- A different answer after the computation by the receiver indicates that, some error is present in the data.
- The computed check digits are called the frame check sequence (FCS) or the cyclic redundancy check (CRC).

Steps for generating CRC at sender.

1. Find the length of the divisor 'len'
2. Add len-1 message to the original message.
3. Do modulo binary division (polynomial division) by XORing to the original message.
4. Calculate the reminder for the division, it will be the CRC Code to be added to original message.

CRC should be of len-1 bits.

Cyclic codes

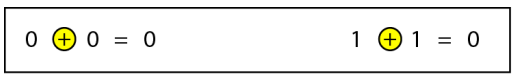
Are special linear block codes with one extra property in a cycle code if a code word is cyclically shifted (Rotated). The result is another codeword.

CRC We can use cyclic codes to correct errors.

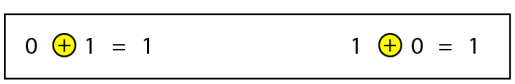
1. Divider – 100100 (Original Message) Divisor 1101 (Key) the length of the key is 4, so we need to add len-1 bits to the original which is original message + 3 bits.

   The polynomial of divisor is $x^3 + x^2 + 1$.

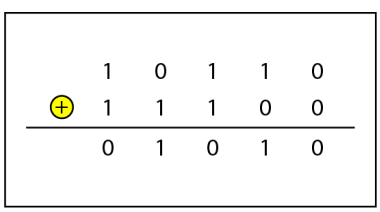
2. The code generator polynomial consists of 4 bits so a string of 3 bits are added to the bit stream to be transmitted (Zeros)
3. The resulting bit Stream is 100 100 000
4. Then perform the binary division



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

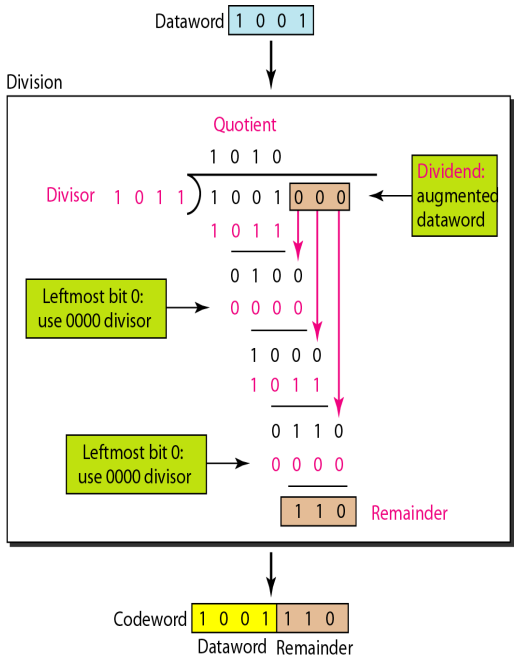
Which is called modulo 2 binary division. Like binary division instead of subtraction we use XOR.

CRC calculation Example

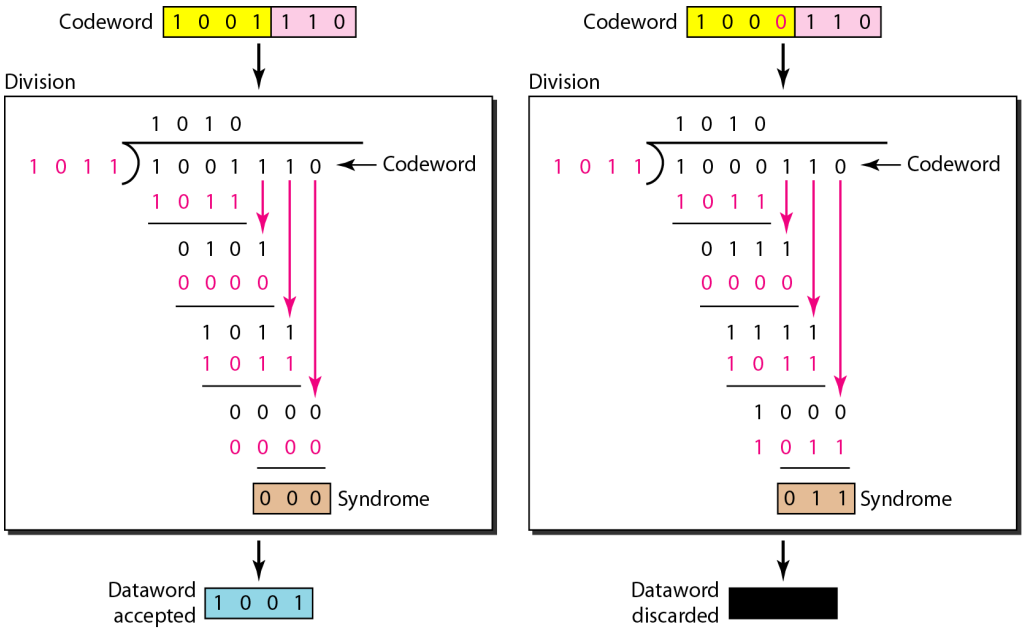
CRC Encoder

Steps.

1. In each step copy the divisor (or data) is XORed with the k bits of the dividend
2. The result of the XOR operation (reminder) is (n-1) bit which is used for the next step after 1 extra bit is pulled down to made it n bits long.
3. When there is no bits to pull down we have a result the n-1 bit reminder which is appended at the sender side.



CRC Decoder



Example1. Data is 100100, Polynomial is X^3+X^2+1

- Ans: X3 + X2 + 1 is 1101 the dividend – Key
- We have 4 bits in dividend key, so we need to add 3 zeros to the original data = 100100 000
- Then divide it with 1101 (XOR Division)

Sender Side

```
          1  1  1  1  0  1
       ------------------
1101 | 1  0  0  1  0  0  0  0  0
       1  1  0  1
          1  0  0  0
          1  1  0  1
             1  0  1  0
             1  1  0  1
                1  1  1  0
                1  1  0  1
                   1  1  0  0
                   1  1  0  1
                ------------
                   0  0  1
```

Receiver Side

```
          1  1  1  1  0  1
       ------------------
1101 | 1  0  0  1  0  0  0  0  1
       1  1  0  1
          1  0  0  0
          1  1  0  1
             1  0  1  0
             1  1  0  1
                1  1  1  0
                1  1  0  1
                   1  1  0  1
                   1  1  0  1
                ------------
                   0  0  0
```

Add 001 to the original code word replace the extra added 0s

The CRC will be 100 100 001

Example 2. CRC for 1110 010 101, Key polynomial is x3 + x2 + 1

Ans from the key polynomial we will get 1101 as the key for calculating CRC. The length is 4 so we need to add 3 zeros to the original code word = 1110 010 101 000 will be the code word

Calculation

Sender Side

```
          1  0  0  0  0  0  1  0
       ------------------------
1101 | 1  1  1  0  0  1  0  1  0  1  0  0  0
       1  1  0  1
          1  1  0  1
          1  1  0  1
                   1  0  1  0
                   1  1  0  1
                      1  1  1  0
                      1  1  0  1
                      ---------
                         1  1  0
```

Receiver Side

```
          1  0  0  0  0  0  1  0
       ------------------------
1101 | 1  1  1  0  0  1  0  1  0  1  1  1  0
       1  1  0  1
          1  1  0  1
          1  1  0  1
                   1  0  1  1
                   1  1  0  1
                      1  1  0  1
                      1  1  0  1
                      ---------
                         0  0  0
```

The code word is 1110 010 101 110

Example 3. CRC Word is 1100 1001 and the polynomial is X3 +1

Ans from the polynomial we can get 1001 as the key word, and the length is 4, so we need to add 3 zeros to the original code word. = 1100 1001 000

Calculation

Sender Side

```
          1  1  0  0  1  0
      ┌──────────────────────
1001  │ 1  1  0  0  1  0  0  1  0  0  0
      │ 1  0  0  1
      │    1  0  1  1
      │    1  0  0  1
      │          1  0  0  0
      │          1  0  0  1
      │                1  1  0  0
      │                1  0  0  1
      │                   1  0  1  0
      │                   1  0  0  1
      ──────────────────────────────────
                         0  1  1
```

Receiver Side

```
          1  1  0  0  1  0
      ┌──────────────────────
1001  │ 1  1  0  0  1  0  0  1  0  1  1
      │ 1  0  0  1
      │    1  0  1  1
      │    1  0  0  1
      │          1  0  0  0
      │          1  0  0  1
      │                1  1  0  1
      │                1  0  0  1
      │                   1  0  0  1
      │                   1  0  0  1
      ──────────────────────────────────
                         0  0  0
```

The CRC code word is 1100 1001 011

Checksum

Checksum is based on redundancy check. This is a block code method where a check sum is created based on the data values in the data block to be transmitted using some algorithm and appended to the data when the receiver gets this data a new checksum is calculated and compared with the existing checksum. A non-match indicates error.