LECTURE NOTES

ON

COMPUTER NETWORKS

COURSE CODE: **AIT003**
BRANCH: **IT**

B.Tech **IV** Semester

Mr.N.BHASWANTH
ASSISTANT PROFESSOR
&
Mr. RAHUL
ASSISTANT PROFESSOR

INFORMATION TECHNOLOGY

INSTITUTE OF AERONAUTICAL ENGINEERING
(Autonomous)
DUNDIGAL, HYDERABAD - 500 043

SYLLABUS

UNIT-I

INTRODUCTION TO PHYSICAL LAYER

Introduction: Networks, network types, internet history, standards and administration; Network models: Protocol layering, TCP/IP protocol suite, the OSI model; Introduction to physical layer: Data and signals, transmission impairment, data rate limits, performance; Transmission media: Introduction, guided media,unguided media; Switching: Introduction, circuit switched networks, packet switching.

UNIT-II

INTRODUCTION TO DATA LINK LAYER

Introduction: Link layer addressing; Error detection and correction: Cyclic codes, checksum, forward error correction; Data link control: DLC services, data link layer protocols, HDLC, point to point protocol, media access control: Random access, controlled access, channelization, connecting devices and virtual LAN: Connecting devices, virtual LAN.

UNIT-III

THE NETWORK LAYER

Network layer design issues, routing algorithms, congestion control algorithms, quality of service, and internetworking. The network layer in the internet: IPv4 addresses, IPv6, internet control protocols, OSPF (Open Shortest Path First), BGP (Border Gateway Protocol), IP, (Internet Protocol), ICMP (internet control message protocol.

UNIT-IV

THE TRANSPORT LAYER

The transport service, elements of transport protocols, congestion control; The internet transport protocols: UDP (User Datagram Protocol), TCP (Transport Control Protocol), performance problems in computer networks, network performance measurement.

UNIT-V

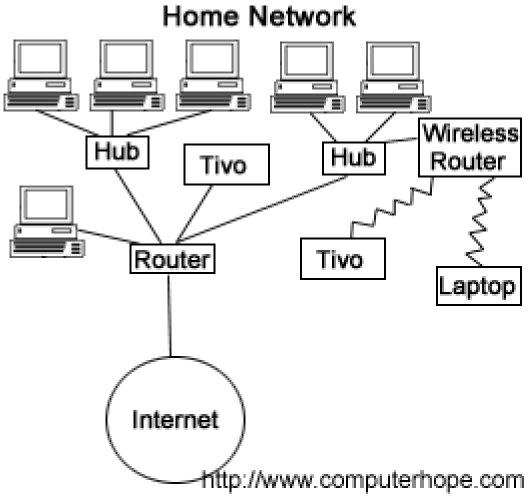
INTRODUCTION TO APPLICATION LAYER

Introduction, client server programming, WWW (World Wide Web) and HTTP (Hyper Text Transfer Protocol), FTP (File Transfer Protocol), E-mail, telnet, secure shell, DNS(Domain Naming System), SNMP (Simple Network Management Protocol).

UNIT-I

INTRODUCTION TO PHYSICAL LAYER

Network: A network is the interconnection of a set of devices capable of communication. In this definition, a device can be a host (or an *end system* as it is sometimes called) such as a large computer, desktop, laptop, workstation, cellular phone, or security system.(or) A network consists of two or more computers that are linked in order to share resources (such as printers and CDs), exchange files, or allow electronic communications. The computers on a network may be linked through cables, telephone lines, radio waves, satellites, or infrared light beams. A device in this definition can also be a connecting device such as a router, which connects the network to other networks, a switch, which connects devices together, a modem (modulator-demodulator), which changes the form of data, and so on. These devices in a network are connected using wired or wireless transmission media such as cable or air. When we connect two computers at home using a plug-and-play router, we have created a network, although very small.



Network Criteria

A network must be able to meet a certain number of criteria. The most important of these are performance, reliability, and security.

Performance

Performance can be measured in many ways, including transit time and response time. Transit time is the amount of time required for a message to travel from one device to another. Response time is the elapsed time between an inquiry and a response. The performance of a network depends on a number of factors, including the number of users, the type of transmission medium, the capabilities of the connected hardware, and the efficiency of the software. Performance is often evaluated by two networking metrics: throughput and delay. We often need more throughputs and less delay. However, these two criteria are often contradictory. If we try to send more data to the network, we may increase throughput but we increase the delay because of traffic congestion in the network.

Reliability

In addition to accuracy of delivery, network reliability is measured by the frequency of failure, the time it takes a link to recover from a failure, and the network's robustness in a catastrophe.

Security
Network security issues include protecting data from unauthorized access, protecting data from damage and development, and implementing policies and procedures for recovery from breaches and data losses.

Physical Structures
Before discussing networks, we need to define some network attributes.

Network Types:
There are several different types of computer networks. Computer networks can be classified by their size as well as their purpose.

The size of a network can be expressed by the geographic area they occupy and the number of computers that are part of the network. Networks can cover anything from a handful of devices within a single room to millions of devices spread across the entire globe.

Some of the different networks based on size are:

- Personal area network, or PAN
- Local area network, or LAN
- Metropolitan area network, or MAN
- Wide area network, or WAN

In terms of purpose, many networks can be considered general purpose, which means they are used for everything from sending files to a printer to accessing the Internet. Some types of networks, however, serve a very particular purpose. Some of the different networks based on their main purpose are:

- Storage area network, or SAN
- Enterprise private network, or EPN
- Virtual private network, or VPN

Personal Area Network:

A **personal area network**, or **PAN**, is a computer network organized around an individual person within a single building. This could be inside a small office or residence. A typical PAN would include one or more computers, telephones, peripheral devices, video game consoles and other personal entertainment devices.

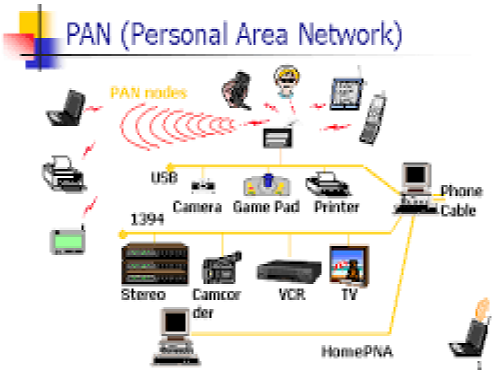
If multiple individuals use the same network within a residence, the network is sometimes referred to as a home area network, or HAN. In a very typical setup, a residence will have a single wired Internet connection connected to a modem. This modem then provides both wired and wireless connections for multiple devices. The network is typically managed from a single computer but can be accessed from any device.

This type of network provides great flexibility. For example, it allows you to:

- Send a document to the printer in the office upstairs while you are sitting on the couch with your laptop.
- Upload a photo from your cell phone to your desktop computer.

- Watch movies from an online streaming service to your TV.

If this sounds familiar to you, you likely have a PAN in your house without having called it by its name.

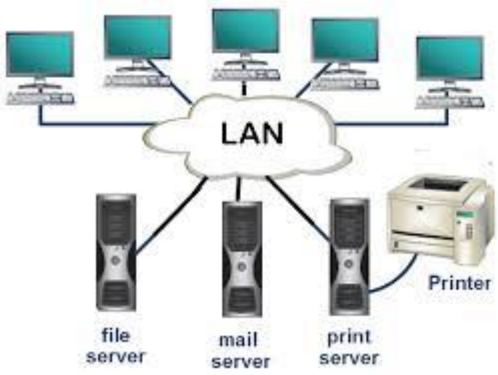


Local Area Network:

A **local area network**, or **LAN**, consists of a computer network at a single site, typically an individual office building. A LAN is very useful for sharing resources, such as data storage and printers. LANs can be built with relatively inexpensive hardware, such as hubs, network adapters and Ethernet cables.

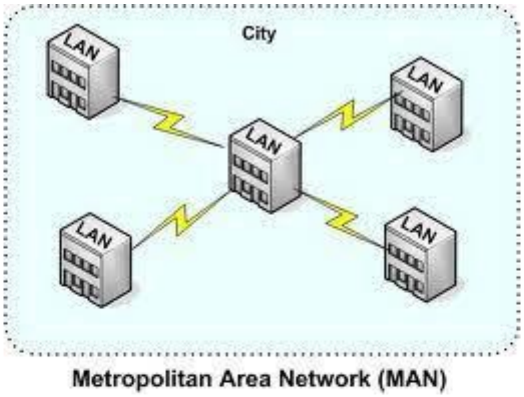
The smallest LAN may only use two computers, while larger LANs can accommodate thousands of computers. A LAN typically relies mostly on wired connections for increased speed and security, but wireless connections can also be part of a LAN. High speed and relatively low cost are the defining characteristics of LANs.

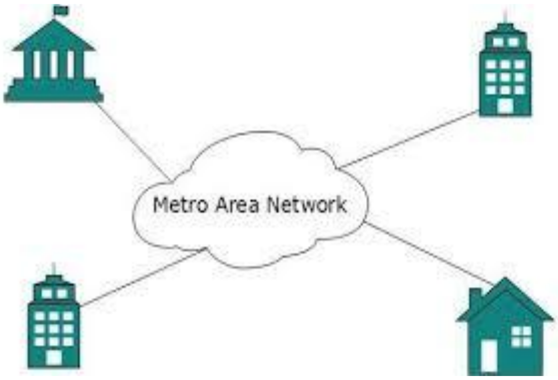
LANs are typically used for single sites where people need to share resources among themselves but not with the rest of the outside world. Think of an office building where everybody should be able to access files on a central server or be able to print a document to one or more central printers. Those tasks should be easy for everybody working in the same office, but you would not want somebody just walking outside to be able to send a document to the printer from their cell phone! If a local area network, or LAN, is entirely wireless, it is referred to as a wireless local area network, or WLAN.



Metropolitan Area Network:

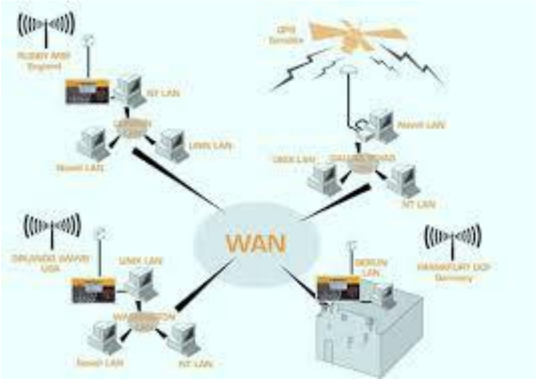
A **metropolitan area network**, or **MAN**, consists of a computer network across an entire city, college campus or small region. A MAN is larger than a LAN, which is typically limited to a single building or site. Depending on the configuration, this type of network can cover an area from several miles to tens of miles. A MAN is often used to connect several LANs together to form a bigger network. When this type of network is specifically designed for a college campus, it is sometimes referred to as a campus area network, or CAN.





Wide Area Network:

A **wide area network**, or **WAN**, occupies a very large area, such as an entire country or the entire world. A WAN can contain multiple smaller networks, such as LANs or MANs. The Internet is the best-known example of a public WAN.

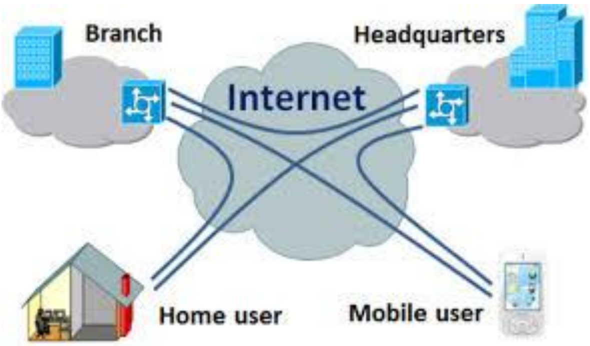


Private Networks:

One of the benefits of networks like PAN and LAN is that they can be kept entirely private by restricting some communications to the connections within the network. This means that those communications never go over the Internet.

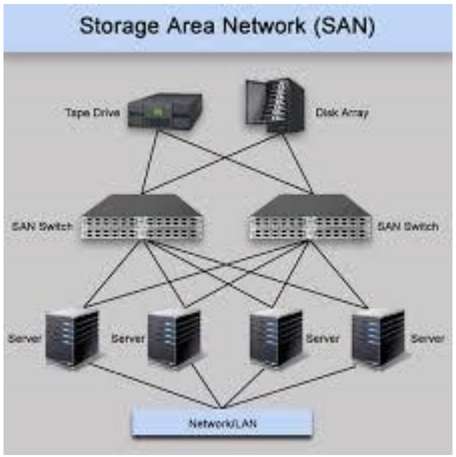
For example, using a LAN, an employee is able to establish a fast and secure connection to a company database without encryption since none of the communications between the employee's computer and the database on the server leave the LAN. But, what happens if the same employee wants to use the database from a remote location? What you need is a private network.

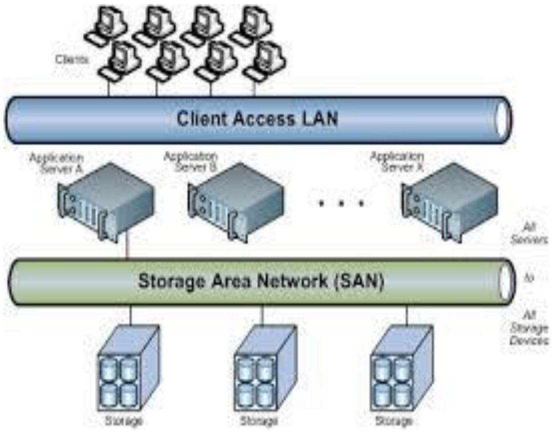
One approach to a private network is to build an **enterprise private network**, or **EPN**. An EPN is a computer network that is entirely controlled by one organization, and it is used to connect multiple locations. Historically, telecommunications companies, like AT&T, operated their own network, separate from the public Internet. EPNs are still fairly common in certain sectors where security is of the highest concern. For example, a number of health facilities may establish their own network between multiple sites to have full control over the confidentiality of patient records.



Storage Area Networks:

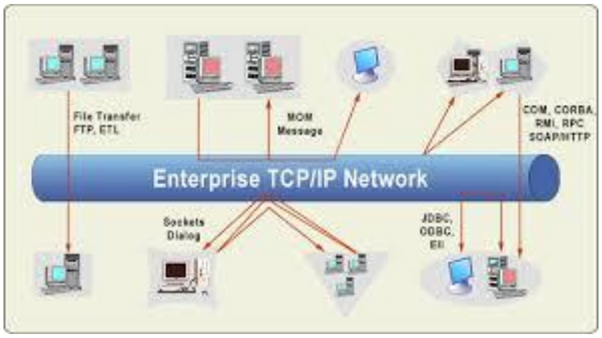
This term is fairly new within the past two decades. It is used to explain a relatively local network that is designed to provide high-speed connection in server-to-server applications (cluster environments), storage area networks (called “SANs” as well) and processor-to-processor applications. The computers connected on a SAN operate as a single system at very high speeds.





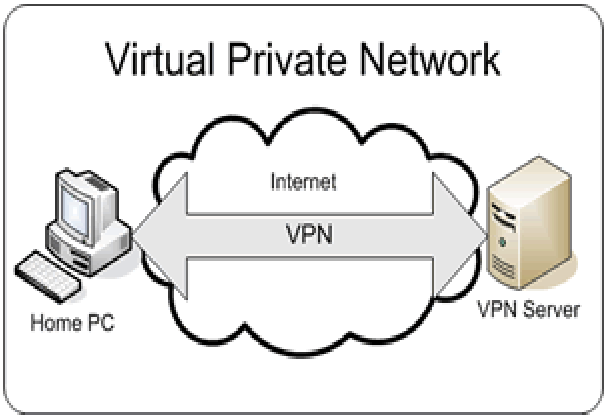
Enterprise Private Network (EPN)

These types of networks are built and owned by businesses that want to securely connect its various locations to share computer resources.



Virtual Private Network (VPN)

By extending a private network across the Internet, a VPN lets its users send and receive data as if their devices were connected to the private network – even if they're not. Through a virtual point-to-point connection, users can access a private network remotely.



INTERNET HISTORY

Now that we have given an overview of the Internet, let us give a brief history of the internet. This brief history makes it clear how the Internet has evolved from a private network to a global one in less than 40 years.

Early History

There were some communication networks, such as telegraph and telephone networks, before 1960. These networks were suitable for constant-rate communication at that time, which means that after a connection was made between two users, the encoded message (telegraphy) or voice (telephony) could be exchanged.

ARPANET

In the mid-1960s, mainframe computers in research organizations were stand-alone devices. Computers from different manufacturers were unable to communicate with one another. The Advanced Research Projects Agency (ARPA) in the Department of Defense (DOD) was interested in finding a way to connect computers so that the researchers they funded could share their findings, thereby reducing costs and eliminating duplication of effort. In 1967, at an Association for Computing Machinery (ACM) meeting, ARPA presented its ideas for the
Advanced Research Projects Agency Network (ARPANET), a small network of connected computers. The idea was that each host computer (not necessarily from the same manufacturer) would be attached to a specialized computer, called an *interface message processor* (IMP). The IMPs, in turn, would be connected to each other. Each IMP had to be able to communicate with other IMPs as well as with its own attached host.

Birth of the Internet

In 1972, Vint Cerf and Bob Kahn, both of whom were part of the core ARPANET group, collaborated on what they called the *Internetting Project. TCPI/P* Cerf and Kahn's landmark 1973 paper outlined the protocols to achieve end-to-end delivery of data. This was a new version of NCP.

This paper on transmission control protocol (TCP) included concepts such as encapsulation, the datagram, and the functions of a gateway. Transmission Control Protocol (TCP) and Internet Protocol (IP). IP would handle datagram routing while TCP would be responsible for higher level functions such as segmentation, reassembly, and error detection. The new combination became known as TCPIIP.

MILNET

In 1983, ARPANET split into two networks: Military Network (MILNET) for military users and ARPANET for nonmilitary users.

CSNET

Another milestone in Internet history was the creation of CSNET in 1981. Computer Science Network (CSNET) was a network sponsored by the National Science Foundation (NSF).

NSFNET

With the success of CSNET, the NSF in 1986 sponsored the National Science Foundation Network (NSFNET), a backbone that connected five supercomputer centers located throughout the United States.

ANSNET

In 1991, the U.S. government decided that NSFNET was not capable of supporting the rapidly increasing Internet traffic. Three companies, IBM, Merit, and Verizon, filled the void by forming a nonprofit organization called Advanced Network & Services (ANS) to build a new, high-speed Internet backbone called Advanced Network Services Network (ANSNET). Internet Today

Today, we witness a rapid growth both in the infrastructure and new applications. The Internet today is a set of pier networks that provide services to the whole world. What has made the internet so popular is the invention of new applications.

World Wide Web

The 1990s saw the explosion of Internet applications due to the emergence of the World Wide Web (WWW). The Web was invented at CERN by Tim Berners-Lee. This invention has added the commercial applications to the Internet.

Multimedia

Recent developments in the multimedia applications such as voice over IP (telephony), video over IP (Skype), view sharing (YouTube), and television over IP (PPLive) has increased the number of users and the amount of time each user spends on the network.

Peer-to-Peer Applications

Peer-to-peer networking is also a new area of communication with a lot of potential.

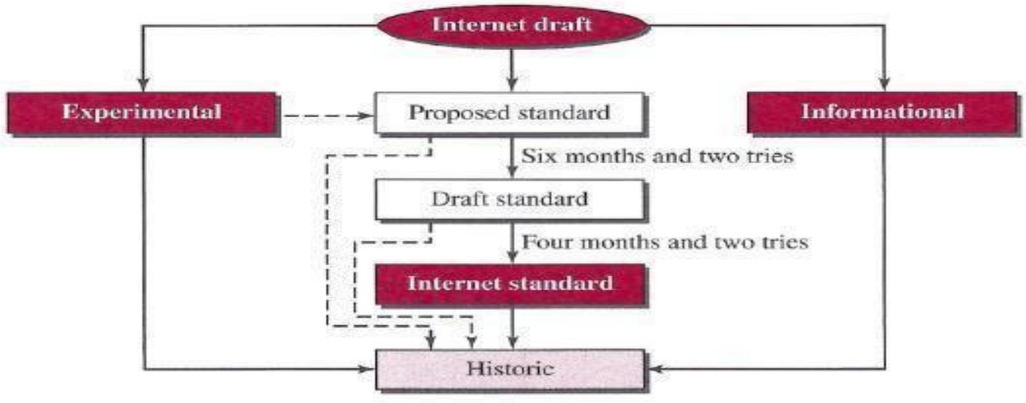
STANDARDS AND ADMINISTRATION

In the discussion of the Internet and its protocol, we often see a reference to a standard or an administration entity. In this section, we introduce these standards and administration entities for those readers that are not familiar with them; the section can be skipped if the reader is familiar with them.

INTERNET STANDARDS

An Internet standard is a thoroughly tested specification that is useful to and adhered to by those who work with the Internet. It is a formalized regulation that must be followed. There is a strict procedure by which a specification attains Internet standard status. A specification begins as an Internet draft. An Internet draft is a working document (a work in progress) with no official status and a six-month lifetime. Upon recommendation from the Internet authorities, a draft 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.

Maturity levels of an RFC



Maturity Levels

An RFC, during its lifetime, falls into one of six *maturity levels:* proposed standard, draft standard, Internet standard, historic, experimental, and informational. *Proposed Standard.* A proposed standard is a specification that is stable, well understood, and of sufficient interest to the Internet community. At this level, the specification is usually tested and implemented by several different groups.

Draft Standard. A proposed standard is elevated to draft standard status after at least two successful independent and interoperable implementations. Barring difficulties, a draft standard, with modifications if specific problems are encountered, normally becomes an Internet standard.

Internet Standard. A draft standard reaches Internet standard status after demonstrations of successful implementation.

Historic The historic RFCs are significant from a historical perspective. They either have been superseded by later specifications or have never passed the necessary maturity levels to become an Internet standard.

Experimental An RFC classified as experimental describes work related to an experimental situation that does not affect the operation of the Internet. Such an RFC should not be implemented in any functional Internet service.

Informational An RFC classified as informational contains general, historical, or tutorial information related to the Internet. It is usually written by someone in a non-Internet organization, such as a vendor.

Requirement Levels

RFCs are classified into five *requirement levels:* required, recommended, elective, limited use, and not recommended.

Required An RFC is labeled *required* if it must be implemented by all Internets systems to achieve minimum conformance. For example, IF and ICMP are required protocols.

Recommended An RFC labeled recommended is not required for minimum conformance; it is recommended because of its usefulness. For example, FTP and TELNET are recommended protocols.

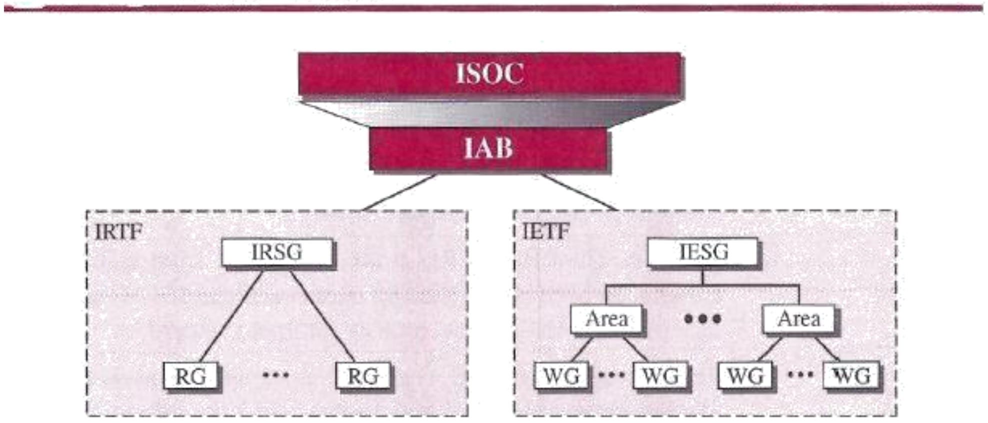
Elective An RFC labeled elective is not required and not recommended. However, a system can use it for its own benefit.

Limited Use An RFC labeled limited use should be used only in limited situations. Most of the experimental RFCs fall under this category.

Not Recommended *An* RFC labeled not recommended is inappropriate for general use. Normally a historic (deprecated) RFC may fall under this category.

INTERNET ADMINISTRATION

The Internet, with its roots primarily in the research domain, has evolved and gained a broader user base with significant commercial activity. Various groups that coordinate Internet issues have guided this growth and development. Appendix G gives the addresses, e-rnail addresses, and telephone numbers for some of these groups. Shows the general organization of Internet administration. E-rnail addresses and telephone numbers for some of these groups. Below figure shows the general organization of Internet administration.


Internet administration

ISOC

The Internet Society (ISOC) is an international, nonprofit organization formed in 1992 to provide support for the Internet standards process. ISOC accomplishes this through maintaining and supporting other Internet administrative bodies such as lAB, IETF,IRTF, and IANA (see the following sections). ISOC also promotes research and other scholarly activities relating to the Internet.

lAB

The Internet Architecture Board (lAB) is the technical advisor to the ISOC. The main purposes of the lAB are to oversee the continuing development of the *TCP/IP* Protocol Suite and to serve in a technical advisory capacity to research members of the Internet community. lAB accomplishes this through its two primary components, the Internet Engineering Task Force (IETF) and the Internet Research Task Force (IRTF). Another responsibility of the lAB is the editorial management of the RFCs, described earlier. lAB is also the external liaison between the Internet and other standards organizations and forums.

JETF

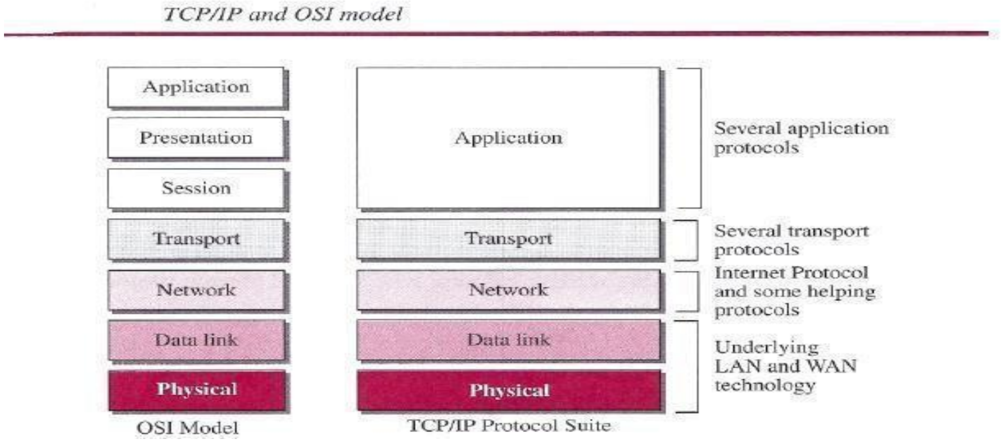
The Internet Engineering Task Force (IETF) is a forum of working groups managed by the Internet Engineering Steering Group (IESG). IETF is responsible for identifying operational problems and proposing solutions to these problems. IETF also develops and reviews specifications intended as Internet standards. The working groups are collected into areas, and each area concentrates on a specific topic. Currently nine areas have been defined. The areas include applications, protocols, routing, network management next generation (lPng), and security.

JRTF

The Internet Research Task Force (IRTF) is a forum of working groups managed by the Internet Research Steering Group (IRSG). IRTF focuses on long-term research topics related to Internet protocols, applications, architecture, and technology.

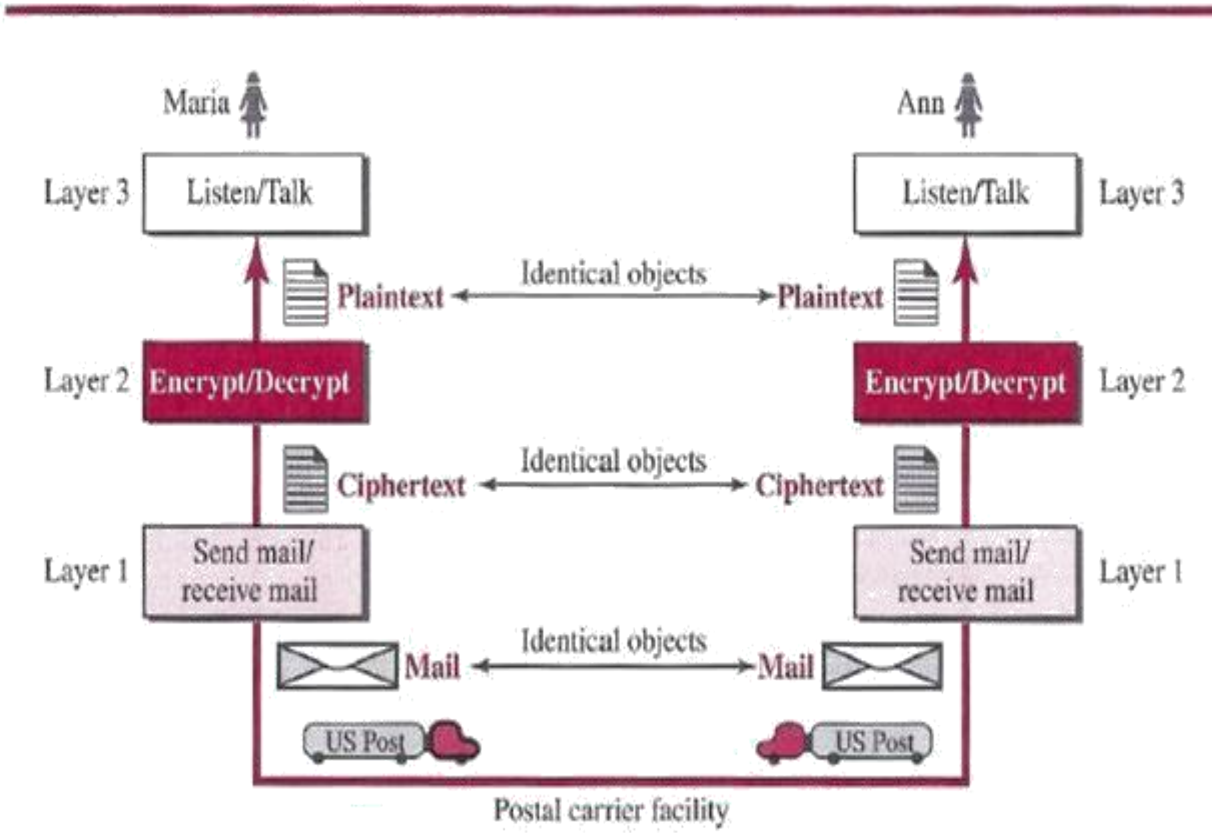
COMPARISION OF OSI AND TCP/IP REFERENCE MODEL

When we compare the two models, we find that two layers, session and presentation, are missing from the *TCP/IP* protocol suite. These two layers were not added to the *TCP/IP* protocol suite after the publication of the OSI model. The application layer in the suite is usually considered to be the combination of three layers in the OSI model.



Two reasons were mentioned for this decision. First, *TCP/IP* has more than one transport-layer protocol. Some of the functionalities of the session layer are available in some of the transport-layer protocols. Second, the application layer is not only one piece of software. Many Applications can be developed at this layer. If some of the functionalities mentioned in the session and presentation layers are needed for a particular application, they can be included in the development of that piece of software.

A three-layer protocol



Network Models:

A *network model* reflects a design or architecture to accomplish communication between different systems. Network models are also referred to as network *stacks* or *protocol suites*. Examples of network models includes TCP/IP, Sequenced Packet Exchange/Internet Packet Exchange (SPX/ IPX) used by Novelle Netware, the Network Basic Input Output System (Net-BIOS), which comprises the building blocks for most Microsoft networking and network applications; and AppleTalk, the network model for Apple Macintosh computers.

A network model usually consists of *layers*. Each layer of a model represents specific functionality. Within the layers of a model, there are usually *protocols* specified to implement specific tasks. You may think of a protocol as a set of rules or a language. Thus, a layer is normally a collection of protocols.

There are a number of different network models. Some of these models relate to a specific implementation, such as the TCP/IP network model. Others simply describe the process of networking, such as the International Organization for Standardization/Open System Interconnection Reference Model (ISO/ OSI-RM, or more simply, OSI-RM).

PROTOCOL LAYERING : In data communication and networking, a protocol defines the rules that both the sender and receiver and all intermediate devices need to follow to be able to communicate effectively. 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.

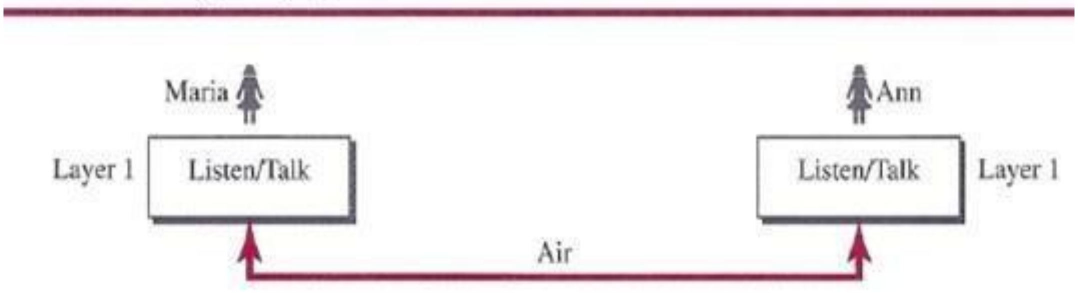
Scenarios

Let us develop two simple scenarios to better understand the need for protocol layering.

First Scenario

In the first scenario, communication is so simple that it can occur in only one layer. Assume Maria and Ann are neighbors with a lot of common ideas. Communication between Maria and Ann takes place in one layer, face to face, in the same language, as shown in Figure

A single-layer protocol

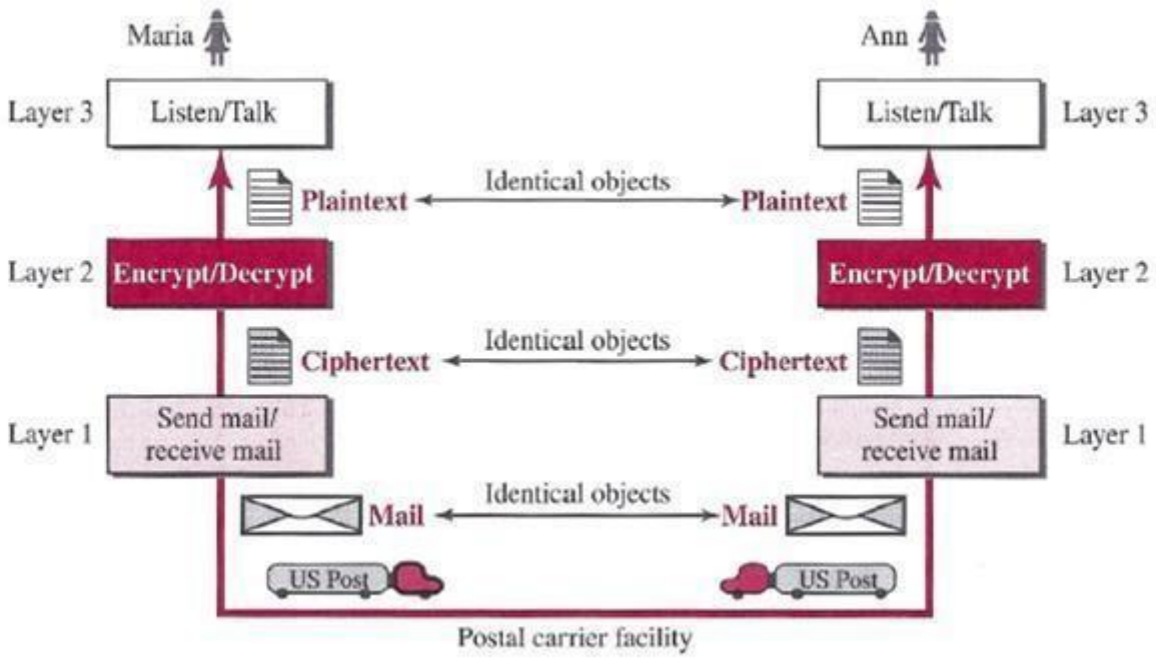


Even in this simple scenario, we can see that a set of rules needs to be followed. First, Maria and Ann know that they should greet each other when they meet. Second, they know that they should confine their vocabulary to the level of their friendship. Third, each party knows that she should refrain from speaking when the other party is speaking. Fourth, each party knows that the conversation should be a dialog, not a monolog: both should have the opportunity to talk about the issue. Fifth, they should exchange some nice words when they leave. We can see that the protocol used by Maria and Ann is different from the communication between a professor and the students in a lecture hall. The communication in the second case is mostly monolog; the professor talks most of the time unless a student has a question, a situation in which the protocol dictates that she should raise her hand and wait for permission to speak. In this case, the communication is normally very formal and limited to the subject being taught.

Second Scenario

In the second scenario, we assume that Ann is offered a higher-level position in her company, but needs to move to another branch located in a city very far from Maria. The two friends still want to continue their communication and exchange ideas because they have come up with an innovative project to start a new business when they both retire. They decide to continue their conversation using regular mail through the post office. However, they do not want their ideas to be revealed by other people if the letters are intercepted. They agree on an encryption/decryption technique. The sender of the letter encrypts it to make it unreadable by an intruder; the receiver of the letter decrypts it to get the original letter. We discuss the encryption/decryption methods in Chapter 31, but for the moment we assume that Maria and Ann use one technique that makes it hard to decrypt the letter if one does not have the key for doing so. Now we can say that the communication between Maria and Ann takes place in three layers, as shown in Figure. We assume that Ann and Maria each have three machines (or robots) that can perform the task at each layer.

A three-layer protocol



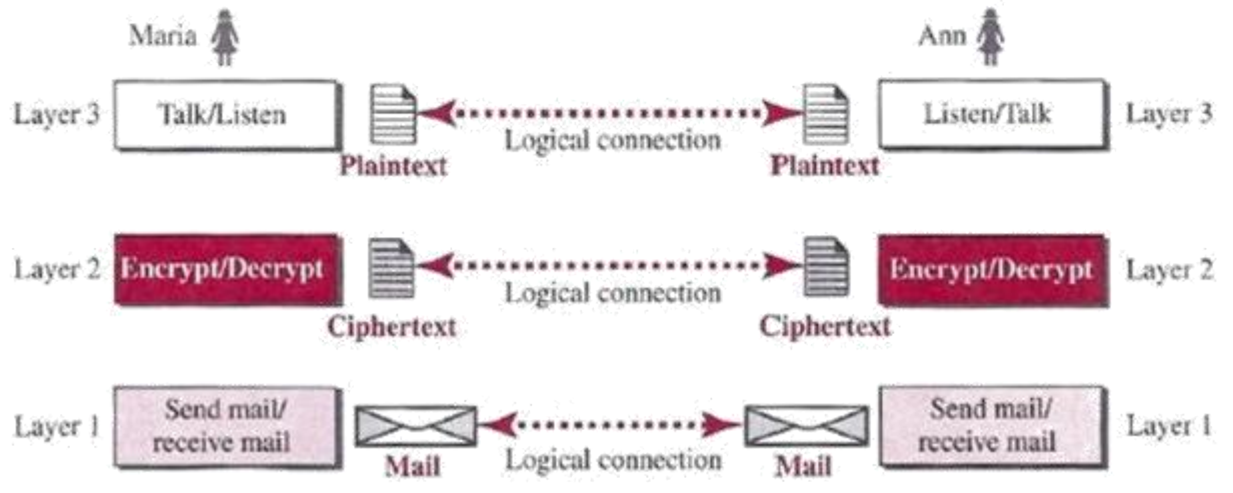
Principles of Protocol Layering
Let us discuss two principles of protocol layering.

First Principle
The first principle dictates that if we want bidirectional communication, we need to make each layer so that it is able to perform two opposite tasks, one in each direction. For example, the third layer task is to listen (in one direction) and *talk* (in the other direction). The second layer needs to be able to encrypt and decrypt. The first layer needs to send and receive mail.

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. For example, the object under layer 3 at both sites should be a plaintext letter. both sites should be a cipher text letter. The object under layer 1 at both sites should be a piece of mail.

Logical Connections
After following the above two principles, we can think about logical connection between each layer as shown in below figure. This means that we have layer-to-layer communication. Maria and Ann can think that there is a logical (imaginary) connection at each layer through which they can send the object created from that layer. We will see that the concept of logical connection will help us better understand the task of layering. We encounter in data communication and networking.

Logical connection between peer layers



TCP/IP PROTOCOL SUITE

Now that we know about the concept of protocol layering and the logical communication between layers in our second scenario, we can introduce the *TCP/IP* (Transmission Control Protocol/Internet Protocol). *TCP/IP* is a protocol suite (a set of protocols organized in different layers) used in the Internet today. It is a hierarchical protocol made up of interactive modules, each of which provides a specific functionality. The term *hierarchical* means that each upper level protocol is supported by the services provided by one or more lower level protocols. 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. Following figure shows both configurations.

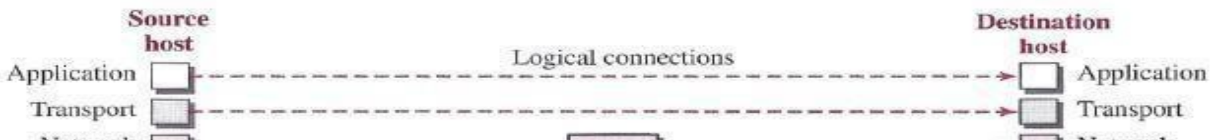
Layered Architecture

To show how the layers in the *TCP/IP* protocol suite are involved in communication between two hosts, we assume that we want to use the suite in a small internet made up of three LANs (links), each with a link-layer switch. We also assume that the links are connected by one router, as shown in below Figure.

Layers in the TCP/IP Protocol Suite

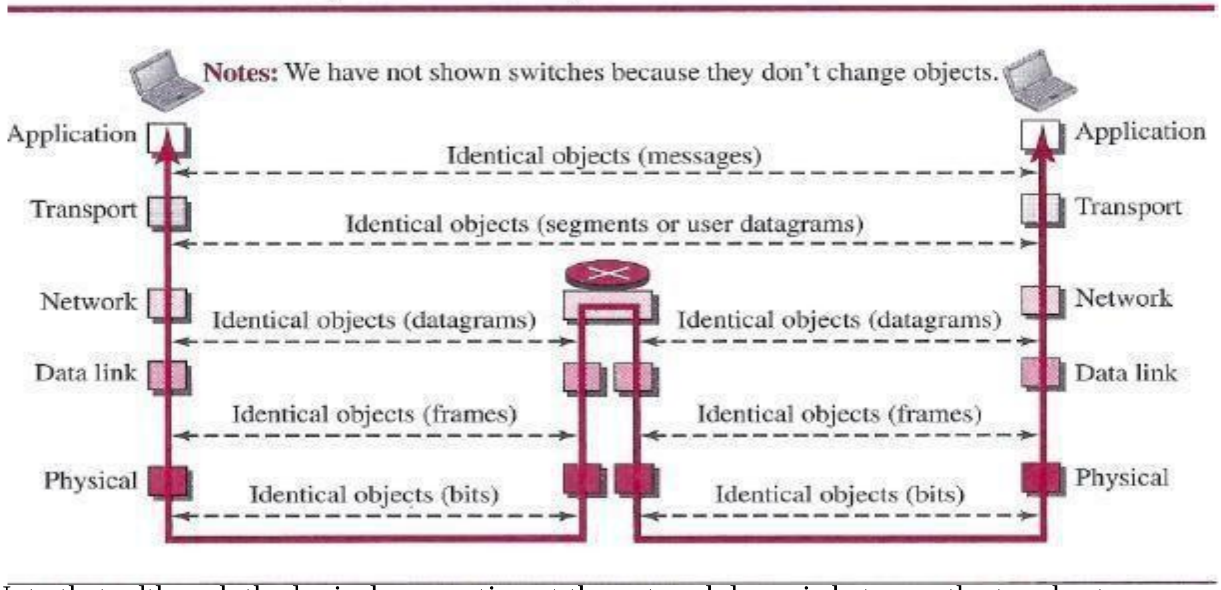
After the above introduction, we briefly discuss the functions and duties of layers in the *TCP/IP* protocol suite. Each layer is discussed in detail in the next five parts of the book. To better understand the duties of each layer, we need to think about the logical connections between layers. Below figure shows logical connections in our simple internet.

Logical connections between layers of the TCP/IP protocol suite



Using logical connections makes it easier for us to think about the duty of each layer. As the figure shows, the duty of the application, transport, and network layers is end-to-end. However, the duty of the data-link and physical layers is hop-to-hop, in which a hop is a host or router. Inother words, the domain of duty of the top three layers is the internet, and the domain of duty of the two lower layers is the link. Another way of thinking of the logical connections is to think about the data unit created from each layer. In the top three layers, the data unit (packets) should not be changed by any router or link-layer switch. In the bottom two layers, the packet created by the host is changed only by the routers, not by the link-layer switches. Below figure shows the second principle discussed previously for protocol layering. We show the identical objects below each layer related to each device.

Identical objects in the TCP/IP protocol suite



Notes: We have not shown switches because they don't change objects.

Application
Identical objects (messages)
Application
Transport
Identical objects (segments or user datagrams)
Transport
Network
Identical objects (datagrams)
Identical objects (datagrams)
Network
Data link
Data link
Identical objects (frames)
Identical objects (frames)
Physical
Identical objects (bits)
Identical objects (bits)
Physical

Note that, although the logical connection at the network layer is between the two hosts, we can only say that identical objects exist between two hops in this case because a router may fragment the packet at the network layer and send more packets than received (see fragmentation in Chapter 19). Note that the link between two hops does not change the object.

Description of Each Layer

After understanding the concept of logical communication, we are ready to briefly discuss the duty of each layer.

Physical Layer

We can say that the physical layer is responsible for carrying individual bits in a frame across the link. Although the physical layer is the lowest level in the TCPIIP protocol suite, the communication between two devices at the physical layer is still a logical communication because there is another, hidden layer, the transmission media, under the physical layer. Two devices are connected by a transmission medium (cable or air). We need to know that the transmission medium does not carry bits; it carries electrical or optical signals. So the bits received in a frame from the data-link layer are transformed and sent through the transmission media, but we can think that the logical unit between two physical layers in two devices is a *bit*. There are several protocols that transform a bit to a signal.

Data-link Layer

We have seen that an internet is made up of several links (LANs and WANs) connected by routers. There may be several overlapping sets of links that a datagram can travel from the host to the destination. The routers are responsible for choosing the *best* links. However, when the next link to travel is determined by the router, the data-link layer is responsible for taking the datagram and moving it across the link. The link can be a wired LAN with a link-layer switch, a wireless LAN, a wired WAN, or a wireless WAN. We can also have different protocols used with any link type. In each case, the data-link layer is responsible for moving the packet through the link. *TCP/IP* does not define any specific protocol for the data-link layer. It supports all
the standard and proprietary protocols. Any protocol that can take the datagram and carry it through the link suffices for the network layer. The data-link layer takes a datagram and encapsulates it in a packet called *«frame.* Each link-layer protocol may provide a different service. Some link-layer protocols provide complete error detection and correction, some provide only error correction.

Network Layer

The network layer is responsible for creating a connection between the source computer and the destination computer. The communication at the network layer is host-to-host. However, since there can be several routers from the source to the destination, the routers in the path are responsible for choosing the best route for each packet. We can say that the network layer is responsible for host-to-host communication and routing the packet through possible routes. Again, we may ask ourselves why we need the network layer. We could have added the routing duty to the transport layer and dropped this layer. One reason, as we said before, is the separation of different tasks between different layers. The second reason is that the routers do not need the application and transport layers.

Transport Layer

The logical connection at the transport layer is also end-to-end. The transport layer at the source host gets the message from the application layer, encapsulates it in a transport layer packet (called a *segment* or a *user datagram* in different protocols) and sends it, through the logical (imaginary) connection, to the transport layer at the destination host. In other words, the transport layer is responsible for giving services to the application layer: to get a message from an application program running on the source host and deliver it to the corresponding application program on the destination host. We may ask why we need an end-to-end transport layer when we already have an end-to-end application layer. The reason is the separation of tasks and duties, which we discussed earlier. The transport layer should be independent of the application layer. In addition, we will see that we have more than one protocol in the transport layer, which means that each application program can use the protocol that best matches its requirement.

Application Layer

The logical connection between the two application layers is end to-end. The two application layers exchange *messages* between each other as though there were a bridge between the two layers. However, we should know that the communication is done through all the layers. Communication at the application layer is between two *processes* (two programs running at this layer). To communicate, a process sends a request to the other process and receives a response. Process-to-process communication is the duty of the application layer. The application layer in the Internet includes many predefined protocols, but a user can also create a pair of processes to be run at the two hosts.

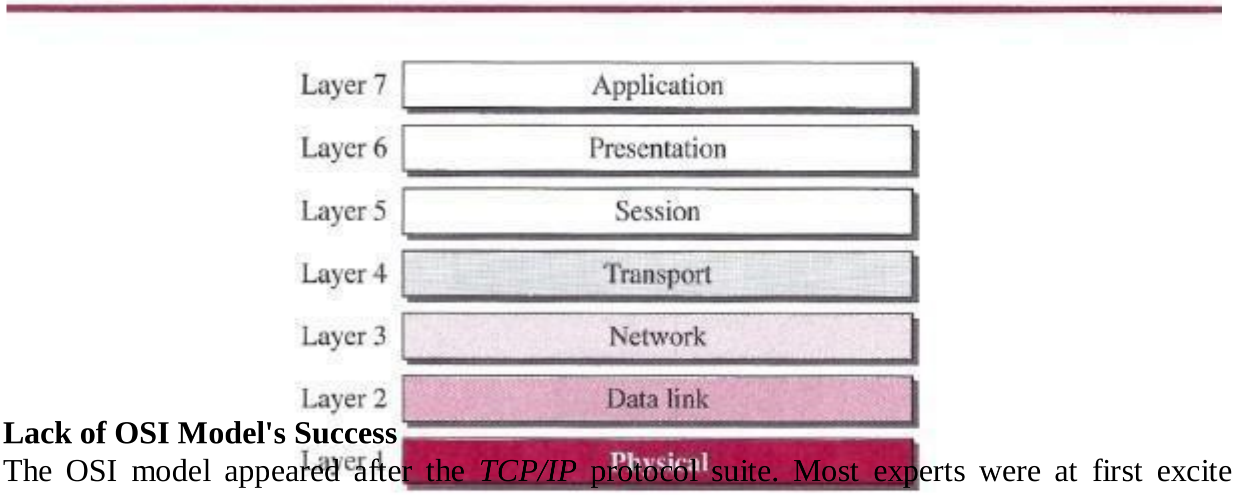
THE OSI MODEL

Although, when speaking of the Internet, everyone talks about the *TCP/IP* protocol suite, this suite is not the only suite of protocols defined. Established in 1947, the International Organization for Standardization (ISO) is a multinational body dedicated to worldwide agreement on international standards. Almost three-fourths of the countries in the world are represented in the ISO. An ISO standard that covers all aspects of network communications is the Open Systems Interconnection (OSI) model. It was first introduced in the late 1970s.

ISO is the organization; OSI is the model

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 model



Lack of OSI Model's Success

The OSI model appeared after the *TCP/IP* protocol suite. Most experts were at first excited and thought that the *TCP/IP* protocol would be fully replaced by the OSI model. This did not happen for several reasons, but we describe only three, which are agreed upon by all experts in the field. First, OSI was completed when *TCP/IP* was fully in place and a lot of time and money had been spent on the suite; changing it would cost a lot. Second, some layers in the OSI model were never fully defined. For example, although the services provided by the presentation and the session layers were listed in the document, actual protocols for these two layers were not fully defined, nor were they fully described, and the corresponding software was not fully developed. Third, when OSI was implemented by an organization in a different application, it did not show a high enough level of performance to entice the Internet authority to switch from the *TCP/IP* protocol suite to the OSI model.

Introduction to Physical Layer:

One of the major functions of the physical layer is to move data in the form of electromagnetic signals across a transmission medium. Physical layer in the OSI model plays the role of interacting with actual hardware and signaling mechanism. Physical layer is the only layer of OSI network model which actually deals with the physical connectivity of two different stations. This layer defines the hardware equipment, cabling, wiring, frequencies, pulses used to represent binary signals etc.Physical layer provides its services to Data-link layer. Data-link layer hands over frames to physical layer. Physical layer converts them to electrical pulses, which represent binary data. The binary data is then sent over the wired or wireless media.

Whether you are collecting numerical statistics from another computer, sending animated pictures from a design workstation, or causing a bell to ring at a distant control center, you are working with the transmission of **data** across network connections. Generally, the data usable to a person or application are not in a form that can be transmitted over a network. For example, a photograph must first be changed to a form that transmission media can accept. Transmission media work by conducting energy along a physical path. For transmission, data needs to be changed to **signals.**

Signals

When data is sent over physical medium, it needs to be first converted into electromagnetic signals. Data itself can be analog such as human voice, or digital such as file on the disk.Both analog and digital data can be represented in digital or analog signals.

- **Digital Signals**:Digital signals are discrete in nature and represent sequence of voltage pulses. Digital signals are used within the circuitry of a computer system.
- **Analog Signals**: Analog signals are in continuous wave form in nature and represented by continuous electromagnetic waves.

Transmission Impairment

When signals travel through the medium they tend to deteriorate. This may have many reasons as given:

- **Attenuation**

  For the receiver to interpret the data accurately, the signal must be sufficiently strong. When the signal passes through the medium, it tends to get weaker.As it covers distance, it loses strength.

- **Dispersion**

  As signal travels through the media, it tends to spread and overlaps. The amount of dispersion depends upon the frequency used.

- **Delay distortion**

  Signals are sent over media with pre-defined speed and frequency. If the signal speed and frequency do not match, there are possibilities that signal reaches destination in arbitrary fashion. In digital media, this is very critical that some bits reach earlier than the previously sent ones.

- **Noise**

  Random disturbance or fluctuation in analog or digital signal is said to be Noise in signal, which may distort the actual information being carried. Noise can be characterized in one of the following class:

  - **Thermal Noise**

    Heat agitates the electronic conductors of a medium which may introduce noise in the media. Up to a certain level, thermal noise is unavoidable.

  - **Intermodulation**

  - When multiple frequencies share a medium, their interference can cause noise in the medium. Intermodulation noise occurs if two different frequencies are sharing a medium and one of them has excessive strength or the component itself is not functioning properly, then the resultant frequency may not be delivered as expected.

  - **Crosstalk**

    This sort of noise happens when a foreign signal enters into the media. This is because signal in one medium affects the signal of second medium.

  - **Impulse**

    This noise is introduced because of irregular disturbances such as lightening, electricity, short-circuit, or faulty components. Digital data is mostly affected by this sort of noise.

Data Rate Limits:

A very important consideration in data communications is how fast we can send data, in bits per second, over a channel. Data rate depends on three factors:

1. The bandwidth available

2. The level of the signals we use

3. The quality of the channel (the level of noise)

Two theoretical formulas were developed to calculate the data rate: one by Nyquist for a noiseless channel, another by Shannon for a noisy channel.

Noiseless Channel: Nyquist

Bit Rate For a noiseless channel, the Nyquist bit rate formula defines the theoretical maximum bit rate

$$\text{BitRate} = 2 \times \text{bandwidth} \times \log_2 L$$

In this formula, bandwidth is the bandwidth of the channel, L is the number of signal levels used to represent data, and BitRate is the bit rate in bits per second. According to the formula, we might think that, given a specific bandwidth, we can have any bit rate we want by increasing the number of signal levels. Although the idea is theoretically correct, practically there is a limit. When we increase the number of signal levels, we impose a burden on the receiver. If the number of levels in a signal is just 2, the receiver can easily distinguish between a 0 and a 1. If the level of a signal is 64, the receiver must be very sophisticated to distinguish between 64 different levels. In other words, increasing the levels of a signal reduces the reliability of the system.

Noisy Channel: Shannon Capacity

In reality, we cannot have a noiseless channel; the channel is always noisy. In 1944, Claude Shannon introduced a formula, called the Shannon capacity, to determine the theoretical highest data rate for a noisy channel:

$$\text{Capacity} = \text{bandwidth} \times \text{log2} (1 + \text{SNR})$$

In this formula, bandwidth is the bandwidth of the channel, SNR is the signal-tonoise ratio, and capacity is the capacity of the channel in bits per second. Note that in the Shannon formula there is no indication of the signal level, which means that no matter how many levels we have, we cannot achieve a data rate higher than the capacity of the channel. In other words, the formula defines a characteristic of the channel, not the method of transmission.

PERFORMANCE :

Up to now, we have discussed the tools of transmitting data (signals) over a network and how the data behave. One important issue in networking is the performance of the network—how good is it?

Bandwidth: One characteristic that measures network performance is bandwidth. However, the term can be used in two different contexts with two different measuring values: bandwidth in hertz and bandwidth in bits per second.

Bandwidth in Hertz: We have discussed this concept. Bandwidth in hertz is the range of frequencies contained in a composite signal or the range of frequencies a channel can pass. For example, we can say the bandwidth of a subscriber telephone line is 4 kHz.

Bandwidth in Bits per Seconds:

The term bandwidth can also refer to the number of bits per second that a channel, a link, or even a network can transmit.

For example, one can say the bandwidth of a Fast Ethernet network (or the links in this network) is a maximum of 100 Mbps. This means that this network can send 100 Mbps.

Relationship:

There is an explicit relationship between the bandwidth in hertz and bandwidth in bits per seconds. Basically, an increase in bandwidth in hertz means an increase in bandwidth in bits per second. The relationship depends on whether we have baseband transmission or transmission with modulation.

Example 3.42 The bandwidth of a subscriber line is 4 kHz for voice or data. The bandwidth of this line for data transmission can be up to 56,000 bps using a sophisticated modem to change the digital signal to analog.

Example 3.43 If the telephone company improves the quality of the line and increases the bandwidth to 8 kHz, we can send 112,000 bps by using the same technology as mentioned in Example 3.42. In networking, we use the term bandwidth in two contexts.

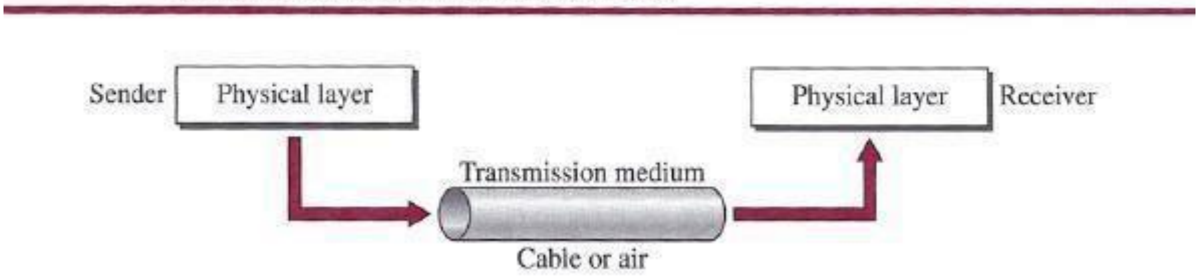
❑ The first, bandwidth in hertz, refers to the range of frequencies in a composite signal or the range of frequencies that a channel can pass.

❑ The second, bandwidth in bits per second, refers to the speed of bit transmission in a channel or link.

TRANSMISSION MEDIA

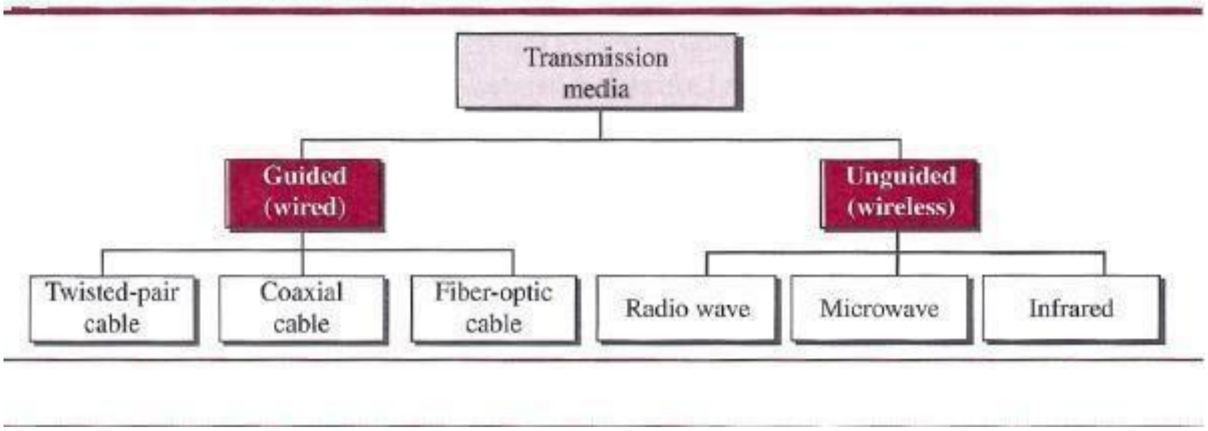
Transmission media are actually located below the physical layer and are directly controlled by the physical layer. We could say that transmission media belong to layer zero. Below figure shows the position of transmission media in relation to the physical layer.

Transmission medium and physical layer



In telecommunications, transmission media can be divided into two broad categories: guided and unguided. Guided media include twisted-pair cable, coaxial cable, and fiber-optic cable. Unguided medium is free space.

Classes of transmission media



GUIDED MEDIA

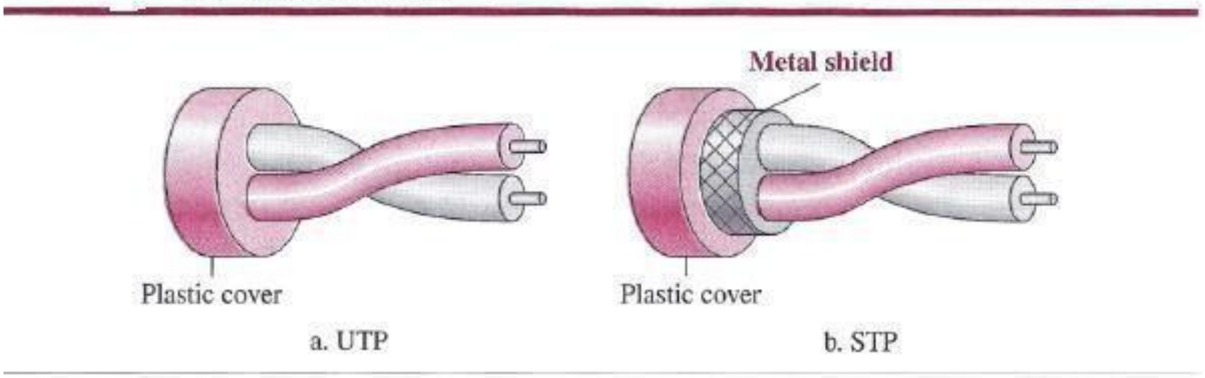
Guided media, which are those that provide a conduit from one device to another, include twisted-pair cable, coaxial cable, and fiber-optic cable. A signal traveling along any of these media is directed and contained by the physical limits of the medium. Twisted-pair and coaxial cable use metallic (copper) conductors that accept and transport signals in the form of electric current. Optical fiber is a cable that accepts and transports signals in the form of light. Twisted-Pair Cable

A twisted pair consists of two conductors (normally copper), each with its own plastic insulation, twisted together, as shown in following figureOne of the wires is used to carry signals to the receiver, and the other is used only as a ground reference. The receiver uses the difference between the two.In addition to the signal sent by the sender on one of the wires, interference (noise) and crosstalk may affect both wires and create unwanted signals. If the two wires are parallel, the effect of these unwanted signals is not the same in both wires because they are at different locations relative to the noise or crosstalk sources (e.g., one is closer and the other is farther). This results in a difference at the receiver. By twisting the pairs, a balance is maintained. For example, suppose in one twist, one wire is closer to the noise source and the other is farther; in the next twist, the reverse is true. Twisting makes it probable that both wires are equally affected by external influences (noise or crosstalk). This means that the receiver, which calculates the difference between the two, receives no unwanted signals. The unwanted signals are mostly canceled out. From the above discussion, it is clear that the number of twists per unit of length (e.g., inch) has some effect on the quality of the cable.

Unshielded Versus Shielded Twisted-Pair Cable

The most common twisted-pair cable used in communications is referred to as *unshielded twisted-pair* (UTP). IBM has also produced a version of twisted-pair cable for its use, called *shielded twisted-pair* (STP). STP cable has a metal foil or braided mesh covering that encases each pair of insulated conductors. Although metal casing improves the quality of cable by preventing the penetration of noise or crosstalk, it is bulkier and more expensive. Below figure

UTP and STP cables



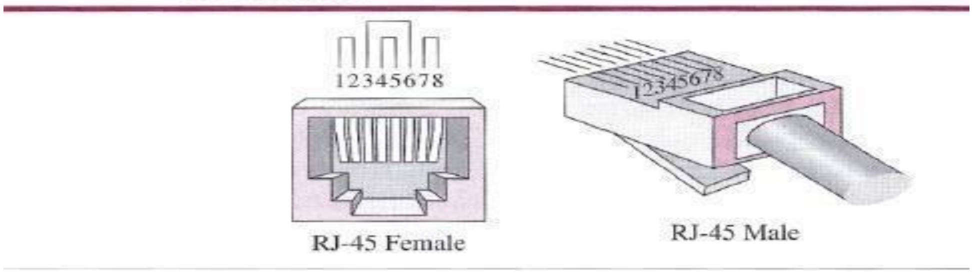
Categories

The Electronic Industries Association (EIA) has developed standards to classify unshielded twisted-pair cable into seven categories. Categories are determined by cable quality, with 1 as the lowest and 7 as the highest. Each EIA category is suitable for specific uses. Table below shows these categories.

Connectors

The most common UTP connector is **RJ45** (RJ stands for registered jack), as shown in below figure. The RJ45 is a keyed connector, meaning the connector can be inserted in only one way.

UTP connector



Performance

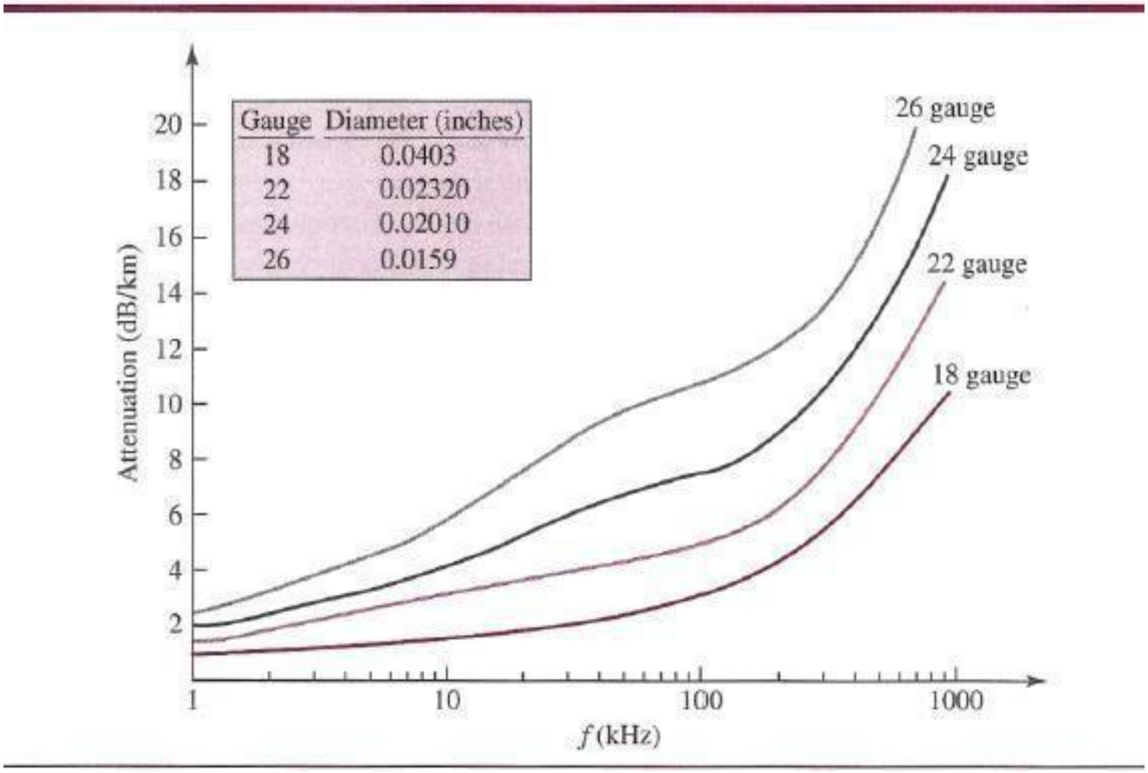
One way to measure the performance of twisted-pair cable is to compare attenuation versus frequency and distance. A twisted-pair cable can pass a wide range of frequencies. However, below figure shows that with increasing frequency, the attenuation, measured in decibels per Kilometer (dB/km), sharply increases with frequencies above 100 kHz. Note that *gauge* is a measure of the thickness of the wire.

Applications

Twisted-pair cables are used in telephone lines to provide voice and data channels. The local loop-the line that connects subscribers to the central telephone office commonly consists of unshielded twisted-pair cables.

The DSL lines that are used by the telephone companies to provide high-data-rate connections also use the high-bandwidth capability of unshielded twisted-pair cables.

UTP performance

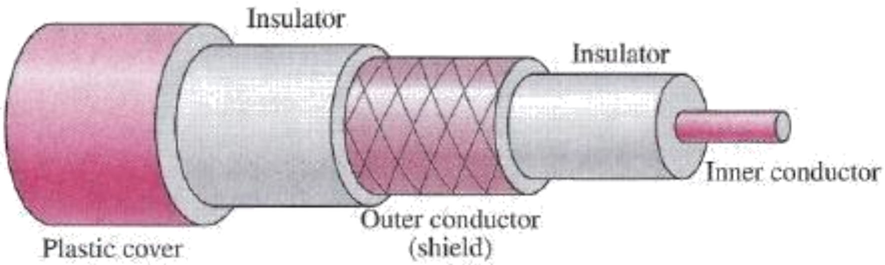


Local-area networks, such as lOBase-T and lOOBase-T, also use twisted-pair cables.

Coaxial Cable

Coaxial cable (or *coax)* carries signals of higher frequency ranges than those in twisted pair cable, in part because the two media are constructed quite differently. Instead of having two wires, coax has a central core conductor of solid or stranded wire (usually copper) enclosed in an insulating sheath, which is, in turn, encased in an outer conductor of metal foil, braid, or a combination of the two. The outer metallic wrapping serves both as a shield against noise and as the second conductor, which completes the circuit. This outer conductor is also enclosed in an insulating sheath, and the whole cable is protected by a plastic cover.

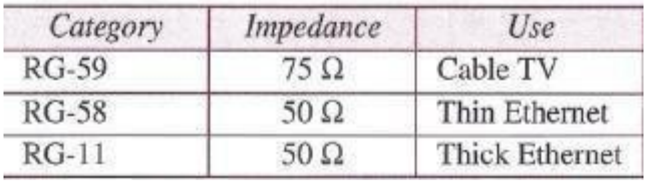
Coaxial cable



Coaxial Cable Standards

Coaxial cables are categorized by their Radio Government (RG) ratings. Each RG number denotes a unique set of physical specifications, including the wire gauge of the inner conductor, the thickness and type of the inner insulator, the construction of the shield, and the size and type of the outer casing. Each cable defined by an RG rating is adapted for a specialized function, as shown in below table.

Categories of coaxial cables

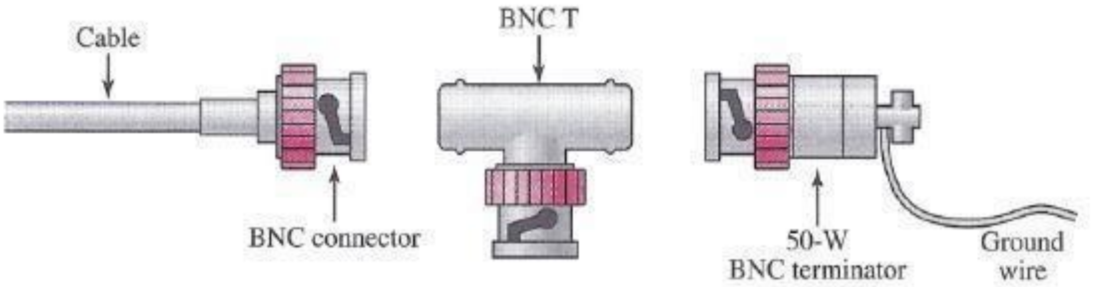


Category	*Impedance*	*Use*
RG-59	75 Ω	Cable TV
RG-58	50 Ω	Thin Ethernet
RG-11	50 Ω	Thick Ethernet

Coaxial Cable Connectors

To connect coaxial cable to devices, we need coaxial connectors. The most common type of connector used today is the Bayonet Neill-Concelman (BNC) connector. Below figure shows three popular types of these connectors: the BNC connector, the BNC T connector, and the BNC terminator.

BNC connectors



The BNC connector is used to connect the end of the cable to a device, such as a TV set. The BNC T connector is used in Ethernet networks (see Chapter 13) to branch out to a connection to a computer or other device. The BNC terminator is used at the end of the cable to prevent the reflection of the signal.

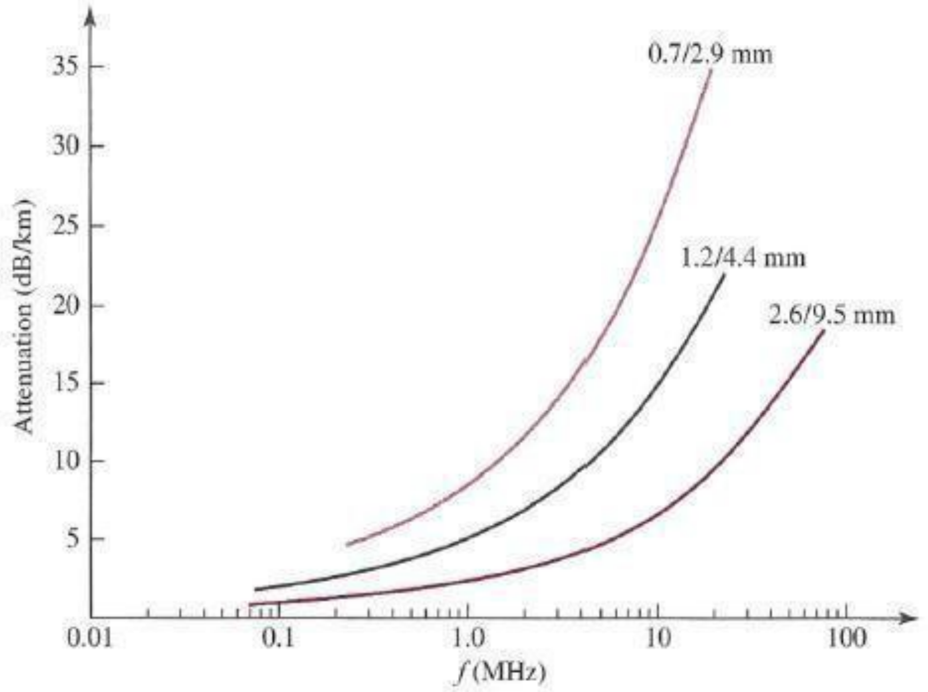
Performance

As we did with twisted-pair cable, we can measure the performance of a coaxial cable. We notice in Figure 7.9 that the attenuation is much higher in coaxial cable than in twisted-pair cable. In other words, although coaxial cable has a much higher bandwidth, the signal weakens rapidly and requires the frequent use of repeaters.

Applications

Coaxial cable was widely used in analog telephone networks where a single coaxial network could carry 10,000 voice signals. Later it was used in digital telephone networks where a single coaxial cable could carry digital data up to 600 Mbps. However, coaxial cable in telephone networks has largely been replaced today with fiber optic cable.

Coaxial cable performance



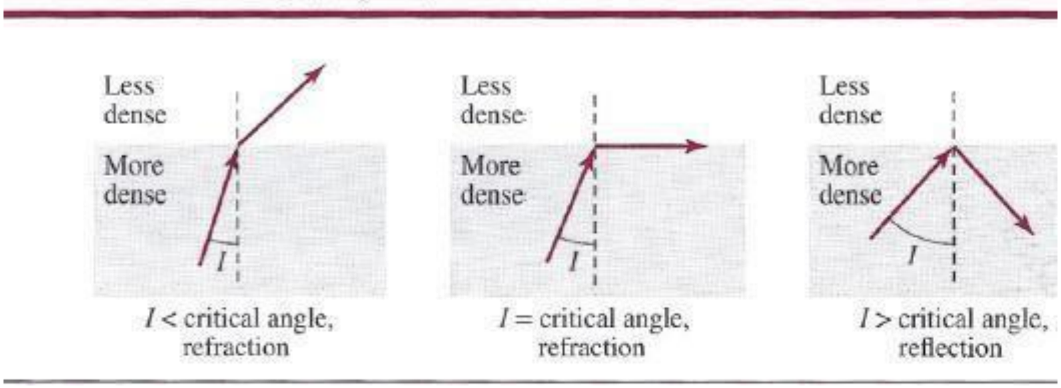
Cable TV networks also use coaxial cables. In the traditional cable TV network, the entire network used coaxial cable. Later, however, cable TV providers replaced most of the media with fiber-optic cable; hybrid networks use coaxial cable only at the network boundaries, near the consumer premises. Cable TV uses RG-59 coaxial cable. Another common application of coaxial cable is in traditional Ethernet LANs (see Because of its high bandwidth, and consequently high data rate, coaxial cable was chosen for digital transmission in early Ethernet LANs. The lOBase-

2, or Thin Ethernet, uses RG-58 coaxial cable with BNC connectors to transmit data at 10 Mbps with a range of 185 m. The lOBase5, or Thick Ethernet, uses RG-ll (thick coaxial cable) to transmit 10 Mbps with a range of 5000 m. Thick Ethernet has specialized connectors.

Fiber-Optic Cable

A fiber-optic cable is made of glass or plastic and transmits signals in the form of light. To understand optical fiber, we first need to explore several aspects of the nature of light. Light travels in a straight line as long as it is moving through a single uniform substance. If a ray oflight traveling through one substance suddenly enters another substance (of a different density), the ray changes direction. Below figure shows how a ray of light changes direction when going from a denser to a less dense substance. As the figure shows, if the angle **of incidence** *I* (the angle the ray makes with the line perpendicular to the interface between the two substances) is less than the **critical angle,** the ray **refracts** and moves closer to the surface. If the angle of incidence is equal to the critical angle, the light bends along the interface. If the angle is greater than the critical angle, the ray **reflects** (makes a turn) and travels again in the denser substance . Note that the critical angle is a property of the substance, and its value differs from one substance to another. Optical fibers use reflection to guide light through a channel. A glass or plastic core is surrounded by a cladding of less dense glass or plastic. The difference in density of the two materials must be such that a beam of light moving through the core is reflected off the cladding instead of being refracted into it. See below figure.

Bending of light ray

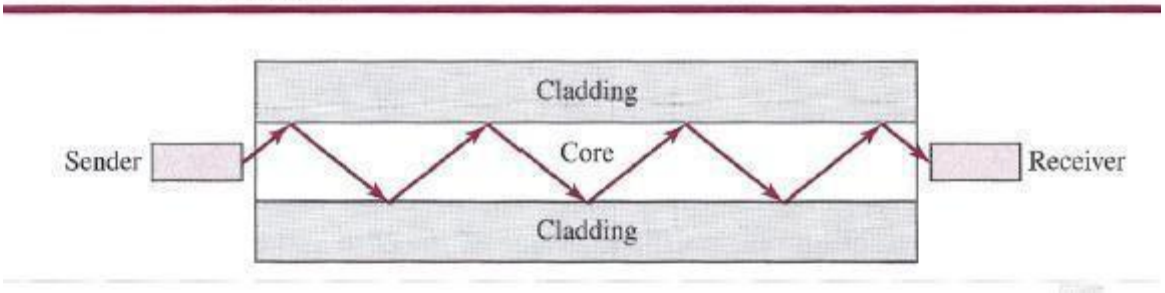


I < critical angle, refraction

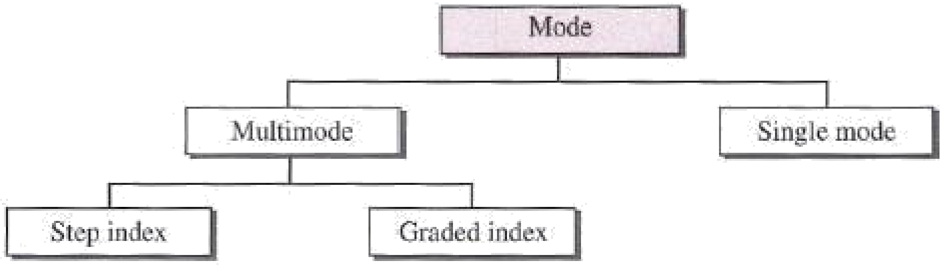
I = critical angle, refraction

I > critical angle, reflection

Optical fiber



Propagation modes



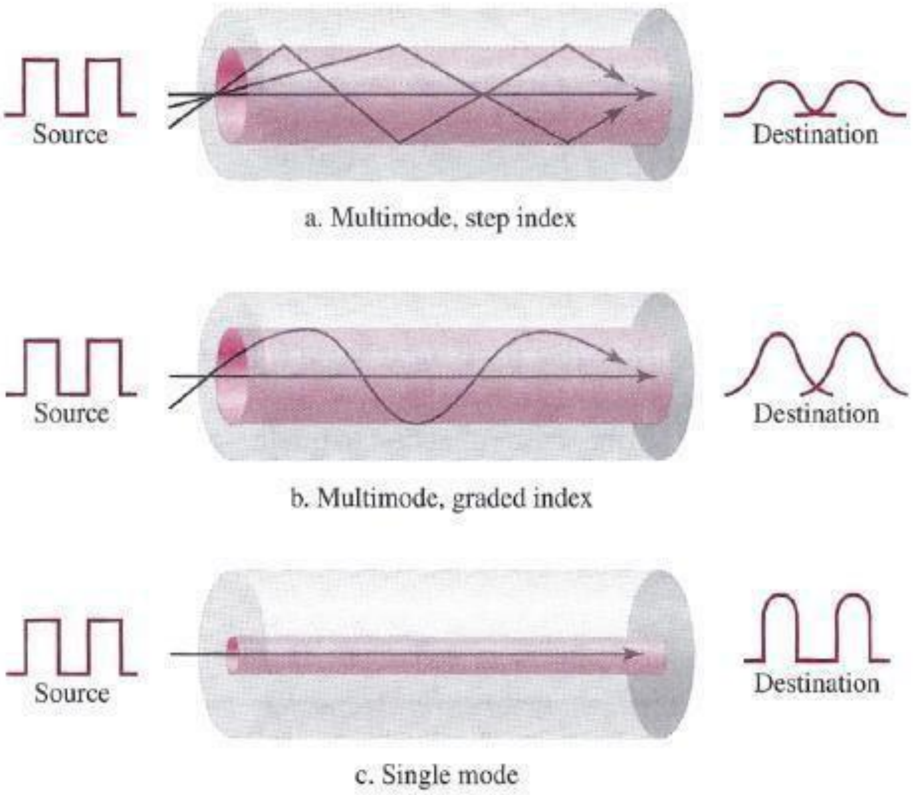
Propagation Modes

Current technology supports two modes (multimode and single mode) for propagating light along optical channels, each requiring fiber with different physical characteristics. Multimode can be implemented in two forms: step-index or graded-index .

Multimode

Multimode is so named because multiple beams from a light source move through the core in different paths. How these beams move within the cable depends on the structure of the core.

Modes



a. Multimode, step index

b. Multimode, graded index

c. Single mode

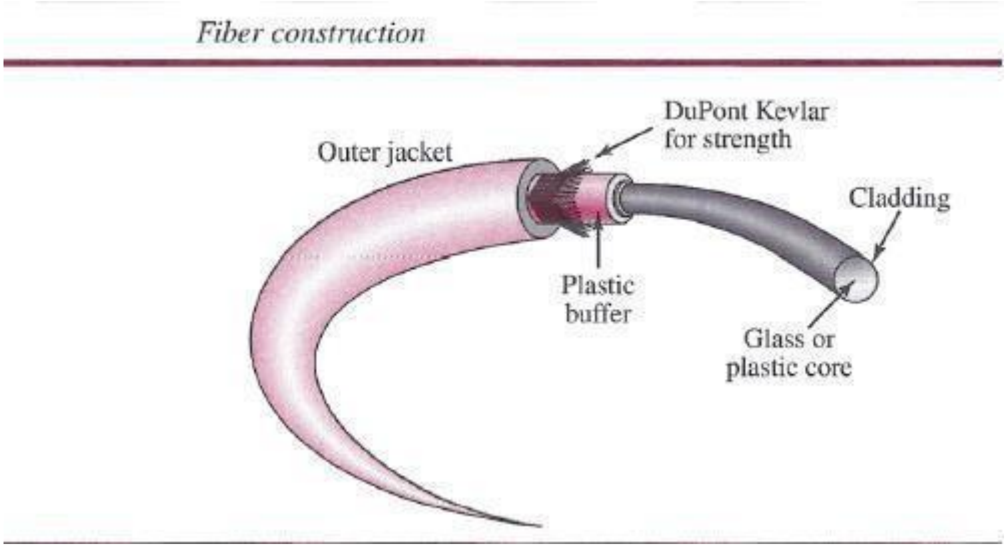
In **multimode step-index fiber,** the density of the core remains constant from the center to the edges. A beam of light moves through this constant density in a straight line until it reaches the interface of the core and the cladding. A second type of fiber, called **multimode graded-index fiber,** decreases this distortion of the signal through the cable. The word *index* here refers to the index of refraction. As we saw above, the index of refraction is related to density. *Single-Mode* Single-mode uses step-index fiber and a highly focused source of light that limits beams to a small range of angles, all close to the horizontal. The **single-mode fiber itself** is manufactured with a much smaller diameter than that of multimode fiber, and with substantially lowers density (index of refraction). The decrease in density results in a critical angle that is close enough to 90° to make the propagation of beams almost horizontal. In this case, propagation of different beams is almost identical, and delays are negligible. All the beams arrive at the destination "together" and can be recombined with little distortion to the signal.

Fiber Sizes

Optical fibers are defined by the ratio of the diameter of their core to the diameter of their cladding, both expressed in micrometers. The common sizes are shown in below table. Note that the last size listed is for single-mode only.

Cable Composition

Following figure shows the composition of a typical fiber-optic cable. The outer jacket is made of either pvc or Teflon. Inside the jacket are Kevlar strands to strengthen the cable. Kevlar is a strong material used in the fabrication of bulletproof vests. Below the Kevlar is another plastic coating to cushion the fiber. The fiber is at the center of the cable, and it consists of cladding and core.



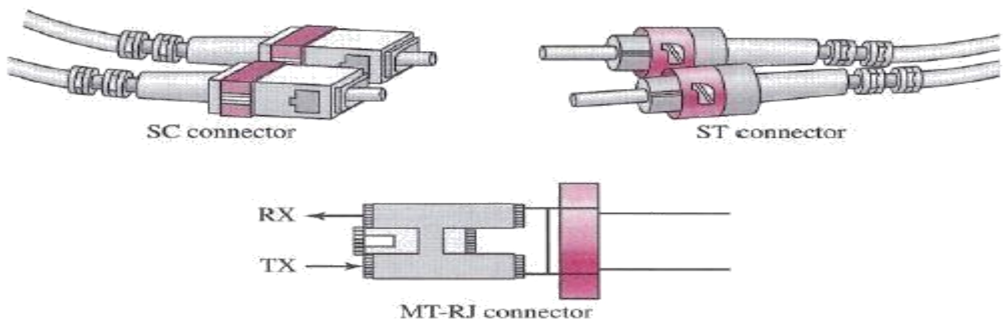
Fiber-Optic Cable Connectors

There are three types of connectors for fiber-optic cables, as shown in below figure. The subscriber channel (SC) connector is used for cable TV. It uses a push/pull locking system. The straight-tip (ST) connector is used for connecting cable to networking devices. It uses a bayonet locking system and is more reliable than sc. MT-RJ is a connector that is the same size as RJ45.

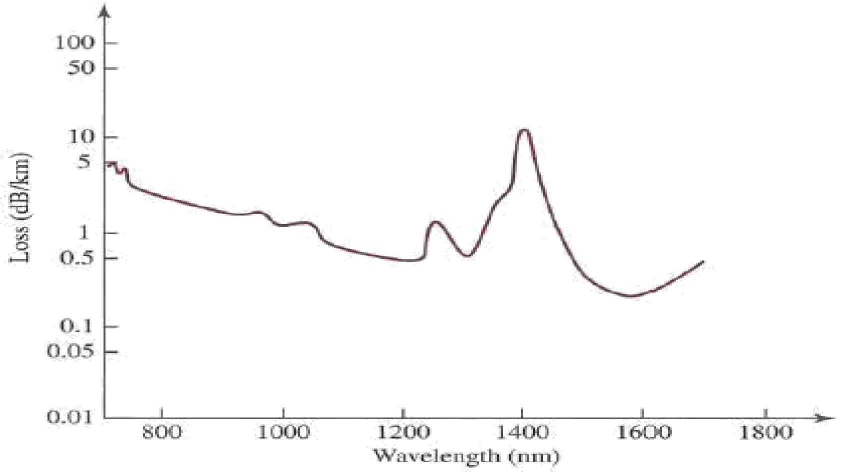
Performance

The plot of attenuation versus wavelength in Figure 7.16 shows a very interesting phenomenon in fiber-optic cable. Attenuation is flatter than in the case of twisted-pair cable and coaxial cable. The performance is such that we need fewer (actually one tenth as many) repeaters when we use fiber-optic cable.

Fiber-optic cable connectors



Optical fiber performance



Applications

Fiber-optic cable is often found in backbone networks because its wide bandwidth is cost-effective. Today, with wavelength-division multiplexing (WDM), we can transfer data at a rate of 1600 Gbps. The SONET network that we discuss in Chapter 14 provides such a backbone.

Some cable TV companies use a combination of optical fiber and coaxial cable, thus creating a hybrid network. Optical fiber provides the backbone structure while coaxial cable provides the connection to the user premises. This is a cost-effective configuration since the narrow bandwidth requirement at the user end does not justify the use of optical fiber. Local-area networks such as 100Base-FX network (Fast Ethernet) and 1000Base-X also use fiber-optic cable.

Advantages and Disadvantages of Optical Fiber

Advantages

Fiber-optic cable has several advantages over metallic cable (twisted-pair or coaxial).

- Higher bandwidth. Fiber-optic cable can support dramatically higher bandwidths (and hence data rates) than either twisted-pair or coaxial cable. Currently, data rates and bandwidth utilization over fiber-optic cable are limited not by the medium but by the signal generation and reception technology available.
- Less signal attenuation. Fiber-optic transmission distance is significantly greater than that of other guided media. A signal can run for 50 km without requiring regeneration. We need repeaters every 5 km for coaxial or twisted-pair cable.
- D Immunity to electromagnetic interference. Electromagnetic noise cannot affect fiber-optic cables.
- D Resistance to corrosive materials. Glass is more resistant to corrosivematerials than copper.
- Light weight. Fiber-optic cables are much lighter than copper cables.
- Greater immunity to tapping. Fiber-optic cables are more immune to tapping than copper cables. Copper cables create antenna effects that can easily be tapped.

Disadvantages

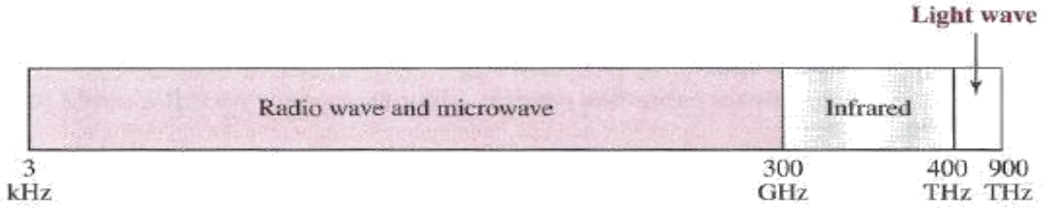
There are some disadvantages in the use of optical fiber.

- Installation and maintenance. Fiber-optic cable is a relatively new technology. Its installation and maintenance require expertise that is not yet available everywhere. o Unidirectional light propagation. Propagation of light is unidirectional. If we need bidirectional communication, two fibers are needed.
- Cost. The cable and the interfaces are relatively more expensive than those of other guided media. If the demand for bandwidth is not high, often the use of optical fiber cannot be justified.

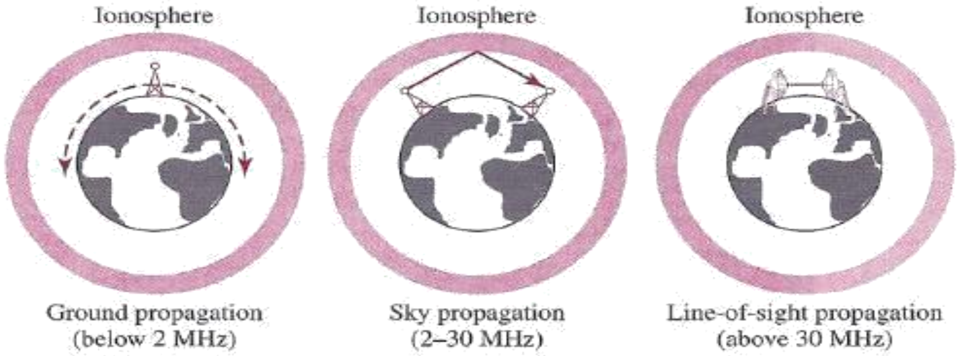
UNGUIDED MEDIA: WIRELESS

Unguided medium transport electromagnetic waves without using a physical conductor. This type of communication is often referred to as *wireless communication.* Signals are normally broadcast through free space and thus are available to anyone who has a device capable of receiving them. Below figure 7.17 shows the part of the electromagnetic spectrum, ranging from 3 kHz to 900 THz, used for wireless communication. Unguided signals can travel from the source to the destination in several ways: ground propagation, sky propagation, and line-of-sight propagation, as shown in below figure.

Electromagnetic spectrum for wireless communication

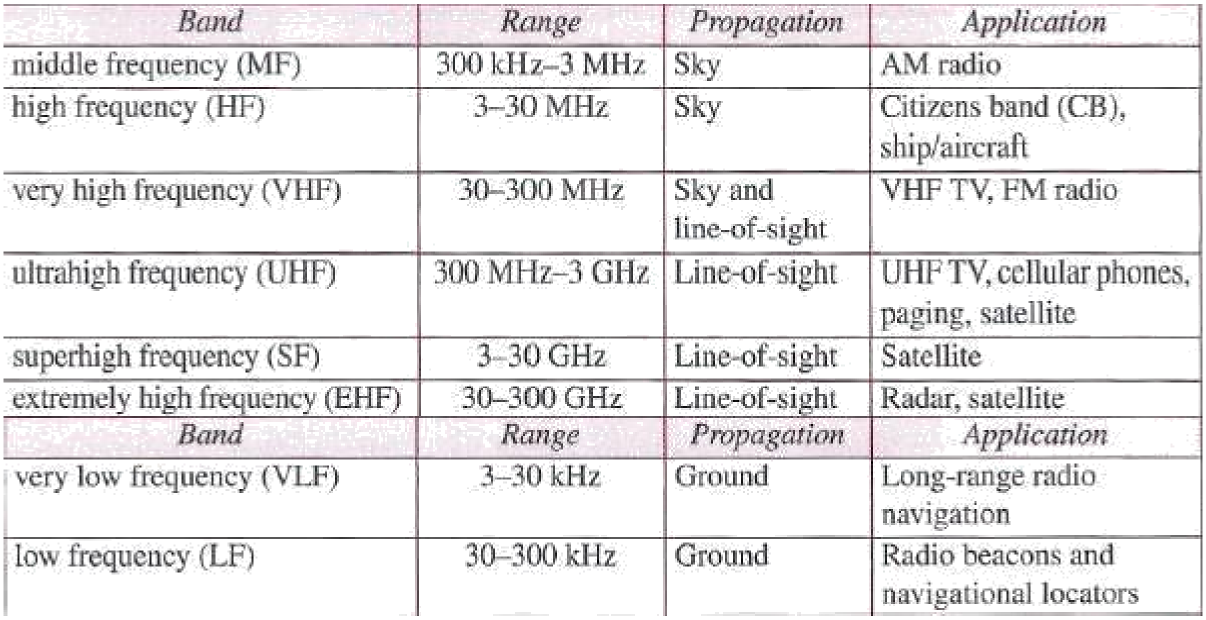


Propagation methods


Ionosphere
Ionosphere
Ionosphere
Ground propagation (below 2 MHz)
Sky propagation (2–30 MHz)
Line-of-sight propagation (above 30 MHz)

In **ground propagation,** radio waves travel through the lowest portion of the atmosphere, hugging the earth. These low-frequency signals emanate in all directions from the transmitting antenna and follow the curvature of the planet. Distance depends on the amount of power in the signal: The greater the power, the greater the distance. **In sky propagation,** higher-frequency radio waves radiate upward into the ionosphere (the layer of atmosphere where particles exist as ions) where they are reflected back to earth. This type of transmission allows for greater distances with lower output power. **In line-of-sight propagation,** very high-frequency signals are transmitted in straight lines directly from antenna to antenna.

The section of the electromagnetic spectrum defined as radio waves and microwaves is divided into eight ranges, called *bands*, each regulated by government authorities. These bands are rated from *very low frequency* (VLF) to *extremely high frequency* (EHF). Below table lists these bands, their ranges, propagation methods, and some applications



Band	Range	Propagation	Application
middle frequency (MF)	300 kHz–3 MHz	Sky	AM radio
high frequency (HF)	3–30 MHz	Sky	Citizens band (CB), ship/aircraft
very high frequency (VHF)	30–300 MHz	Sky and line-of-sight	VHF TV, FM radio
ultrahigh frequency (UHF)	300 MHz–3 GHz	Line-of-sight	UHF TV, cellular phones, paging, satellite
superhigh frequency (SF)	3–30 GHz	Line-of-sight	Satellite
extremely high frequency (EHF)	30–300 GHz	Line-of-sight	Radar, satellite
very low frequency (VLF)	3–30 kHz	Ground	Long-range radio navigation
low frequency (LF)	30–300 kHz	Ground	Radio beacons and navigational locators

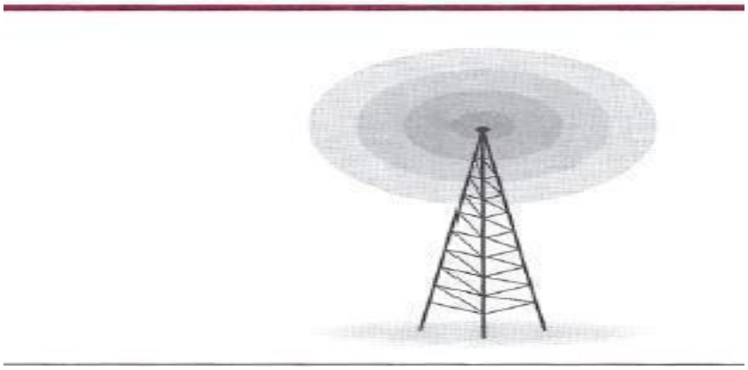
Radio Waves

Although there is no clear-cut demarcation between radio waves and microwaves, electromagnetic waves ranging in frequencies between 3 kHz and 1 GHz are normally called radio waves; waves ranging in frequencies between I and 300 GHz are called microwaves. However, the behavior of the waves, rather than the frequencies, is a better criterion for classification. Radio waves, for the most part, are omnidirectional. When an antenna transmits radio waves, they are propagated in all directions. This means that the sending and receiving antennas do not have to be aligned. A sending antenna sends waves that can be received by any receiving antenna. The omnidirectional property has a disadvantage, too. The radio waves transmitted by one antenna are susceptible to interference by another antenna that may send signals using the same frequency or band. Radio waves, particularly those waves that propagate in the sky mode, can travel long distances. This makes radio waves a good candidate for long-distance broadcasting such as AM radio.

Omni directional Antenna

Radio waves use omni directional antennas that send out signals in all directions. Based on the wavelength, strength, and the purpose of transmission, we can have several types of antennas. Below Figure shows an omni directional antenna.

Omnidirectional antenna



Applications

The omnidirectional characteristics of radio waves make them useful for multicasting, in which there is one sender but many receivers. AM and FM radio, television, maritime radio, cordless phones, and paging are examples of multicasting.

**Radio waves are used for multicast communications,
such as radio and television, and paging systems.**

Microwaves

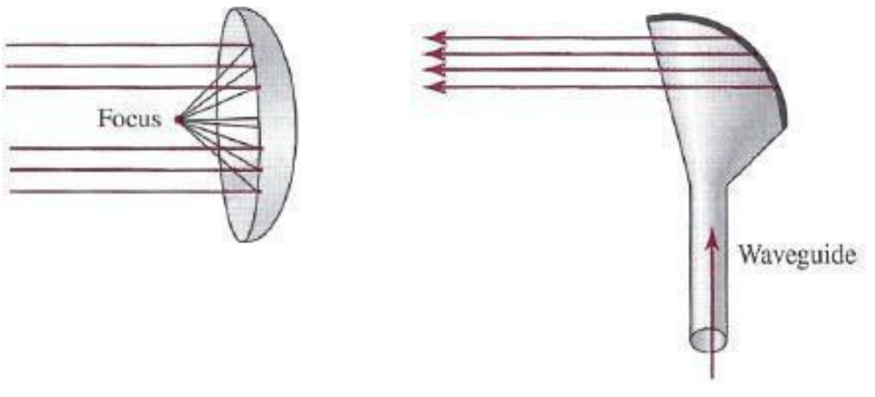
Electromagnetic waves having frequencies between 1 and 300 GHz are called microwaves. Microwaves are unidirectional. When an antenna transmits microwaves, they can be narrowly focused. This means that the sending and receiving antennas need to be aligned. The unidirectional property has an obvious advantage. A pair of antennas can be aligned without interfering with another pair of aligned antennas.

The following describes some characteristics of microwave propagation:

- Microwave propagation is line-of-sight. Since the towers with the mounted antennas need to be in direct sight of each other, towers that are far apart need to be very tall. The curvature of the earth as well as other blocking obstacles does not allow two short towers to communicate by using microwaves. Repeaters are often needed for long distance communication.
- Very high-frequency microwaves cannot penetrate walls. This characteristic can be a disadvantage if receivers are inside buildings.
- The microwave band is relatively wide, almost 299 GHz. Therefore wider subbands can be assigned, and a high data rate is possible.
- Use of certain portions of the band requires permission fromauthorities.

Unidirectional Antenna

Microwaves need unidirectional antennas that send out signals in one direction. Two types of antennas are used for microwave communications: the parabolic dish and the horn .



Unidirectional antennas

a. Parabolic dish antenna

b. Horn antenna

A parabolic dish antenna is based on the geometry of a parabola: Every line parallel to the line of symmetry (line of sight) reflects off the curve at angles such that all the lines intersect in a common point called the focus. The parabolic dish works as a funnel, catching a wide range of waves and directing them to a common point. In this way, more of the signal is recovered than would be possible with a single-point receiver.

Outgoing transmissions are broadcast through a horn aimed at the dish. The microwaves hit the dish and are deflected outward in a reversal of the receipt path. A horn antenna looks like a gigantic scoop. Outgoing transmissions are broadcast up a stem (resembling a handle) and deflected outward in a series of narrow parallel beams by the curved head. Received transmissions are collected by the scooped shape of the horn, in a manner similar to the parabolic dish, and are deflected down into the stem.

Applications

Microwaves, due to their unidirectional properties, are very useful when unicast (one to- one) communication is needed between the sender and the receiver. They are used in cellular phone, satellite networks, and wireless LANs

Microwaves are used for unicast communication such as cellular telephones, satellite networks, and wireless LANs.

Infrared

Infrared waves, with frequencies from 300 GHz to 400 THz (wavelengths from 1 mm to 770 nrn), can be used for short-range communication. Infrared waves, having high frequencies, cannot penetrate walls. This advantageous characteristic prevents interference between one system and another; a short-range communication system in one room cannot be affected by another system in the next room. When we use our infrared remote control, we do not interfere with the use of the remote by our neighbors. However, this same characteristic makes infrared signals useless for long-range communication. In addition, we cannot use infrared waves outside a building because the sun's rays contain infrared waves that can interfere with the communication.

Applications

The infrared band, almost 400 THz, has an excellent potential for data transmission. Such a wide bandwidth can be used to transmit digital data with a very high data rate. The *Infrared Data Association* elrDA), an association for sponsoring the use of infrared waves, has established standards for using these signals for communication between devices such as keyboards, mice, PCs, and printers. For example, some manufacturers provide a special port called the IrDA port that allows a wireless keyboard to communicate with a PC. The standard originally defined a data rate of 75 kbps for a distance up to 8 m. The recent standard defines a data rate of 4 Mbps.

Infrared signals defined by IrDA transmit through line of sight; the IrDA port on the keyboard needs to point to the PC for transmission to occur. Infrared signals can be used for short-range communication in a closed area using line-of-sight propagation.

Switching:

Switching is process to forward packets coming in from one port to a port leading towards the destination. When data comes on a port it is called ingress, and when data leaves a port or goes out it is called egress. A communication system may include number of switches and nodes. At broad level, switching can be divided into two major categories:

- **Connectionless:** The data is forwarded on behalf of forwarding tables. No previous handshaking is required and acknowledgements are optional.

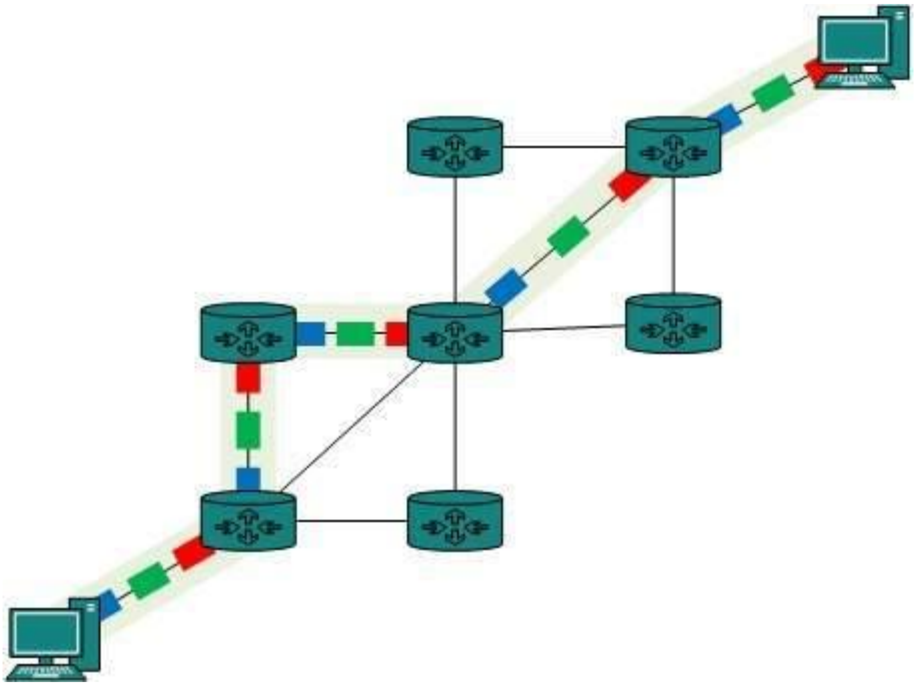
- **Connection Oriented:** Before switching data to be forwarded to destination, there is a need to pre-establish circuit along the path between both endpoints. Data is then forwarded on that circuit. After the transfer is completed, circuits can be kept for future use or can be turned down immediately.

Circuit Switching

When two nodes communicate with each other over a dedicated communication path, it is called circuit switching.There 'is a need of pre-specified route from which data will travels and no other data is permitted.In circuit switching, to transfer the data, circuit must be established so that the data transfer can take place.

Circuits can be permanent or temporary. Applications which use circuit switching may have to go through three phases:

- Establish a circuit
- Transfer the data
- Disconnect the circuit

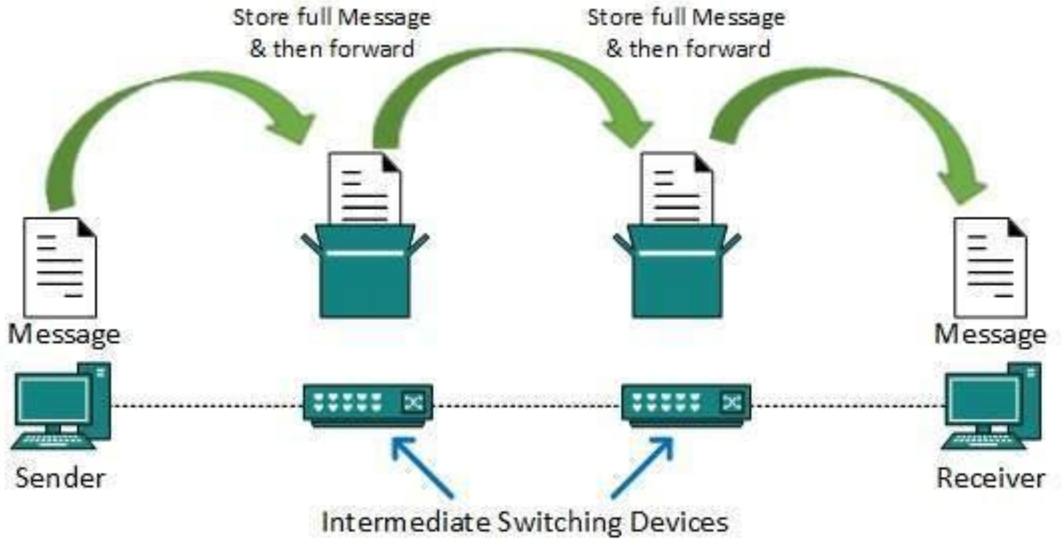
- 

Circuit switching was designed for voice applications. Telephone is the best suitable example of circuit switching. Before a user can make a call, a virtual path between caller and callee is established over the network.

Message Switching:

This technique was somewhere in middle of circuit switching and packet switching. In message switching, the whole message is treated as a data unit and is switching / transferred in its entirety.

A switch working on message switching, first receives the whole message and buffers it until there are resources available to transfer it to the next hop. If the next hop is not having enough resource to accommodate large size message, the message is stored and switch waits.


Store full Message & then forward
Store full Message & then forward
Message
Message
Sender
Receiver
Intermediate Switching Devices

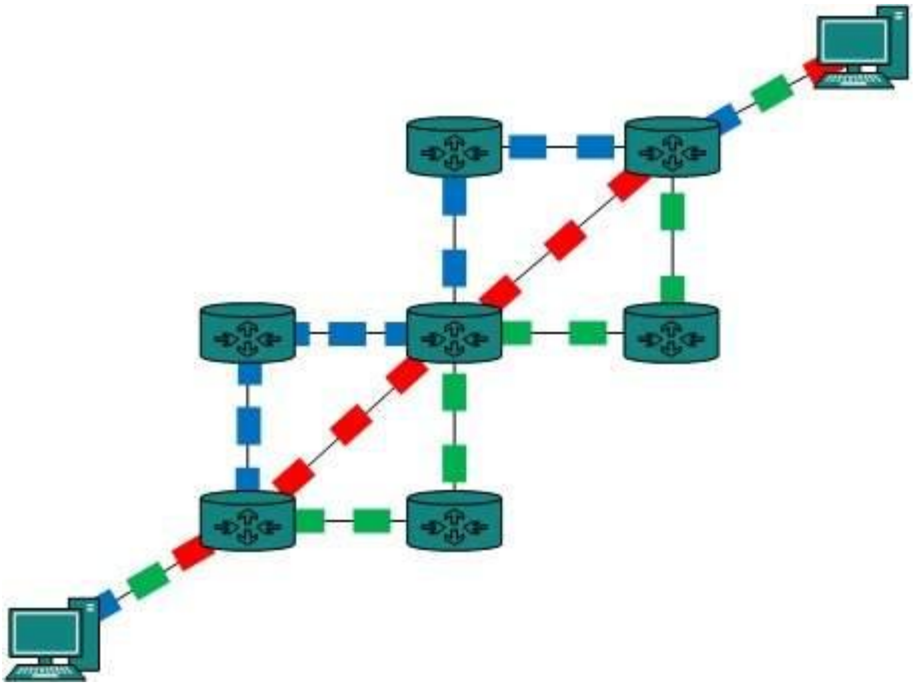
This technique was considered substitute to circuit switching. As in circuit switching the whole path is blocked for two entities only. Message switching is replaced by packet switching. Message switching has the following drawbacks:

- Every switch in transit path needs enough storage to accommodate entire message.
- Because of store-and-forward technique and waits included until resources are available, message switching is very slow.
- Message switching was not a solution for streaming media and real-time applications.

Packet Switching:

Shortcomings of message switching gave birth to an idea of packet switching. The entire message is broken down into smaller chunks called packets. The switching information is added in the header of each packet and transmitted independently.

It is easier for intermediate networking devices to store small size packets and they do not take much resources either on carrier path or in the internal memory of switches.



Packet switching enhances line efficiency as packets from multiple applications can be multiplexed over the carrier. The internet uses packet switching technique. Packet switching enables the user to differentiate data streams based on priorities. Packets are stored and forwarded according to their priority to provide quality of service.

Circuit Switched Networks:

A circuit-switched communication system involves three phases: circuit establishment (setting up dedicated links between the source and destination); data transfer (transmitting the data between the source and destination); and circuit disconnect (removing the dedicated links).

The important thing to look for in transmitting information over such a complex network is the path or circuit. In circuit-switching, this path is decided upon before the data transmission starts. The system decides on which route to follow, based on a resource-optimizing algorithm, and transmission goes according to the path. For example, a call from Los Angeles to Chicago will obviously take a different path or circuit than a call from Los Angeles to Miami.

It's important to note the difference between a circuit switched network and a packet switched network. A packet is an internet protocol (IP) that breaks data into chunks and takes those chunks and bundles them into packets. In packet switching, there is no predetermined path.

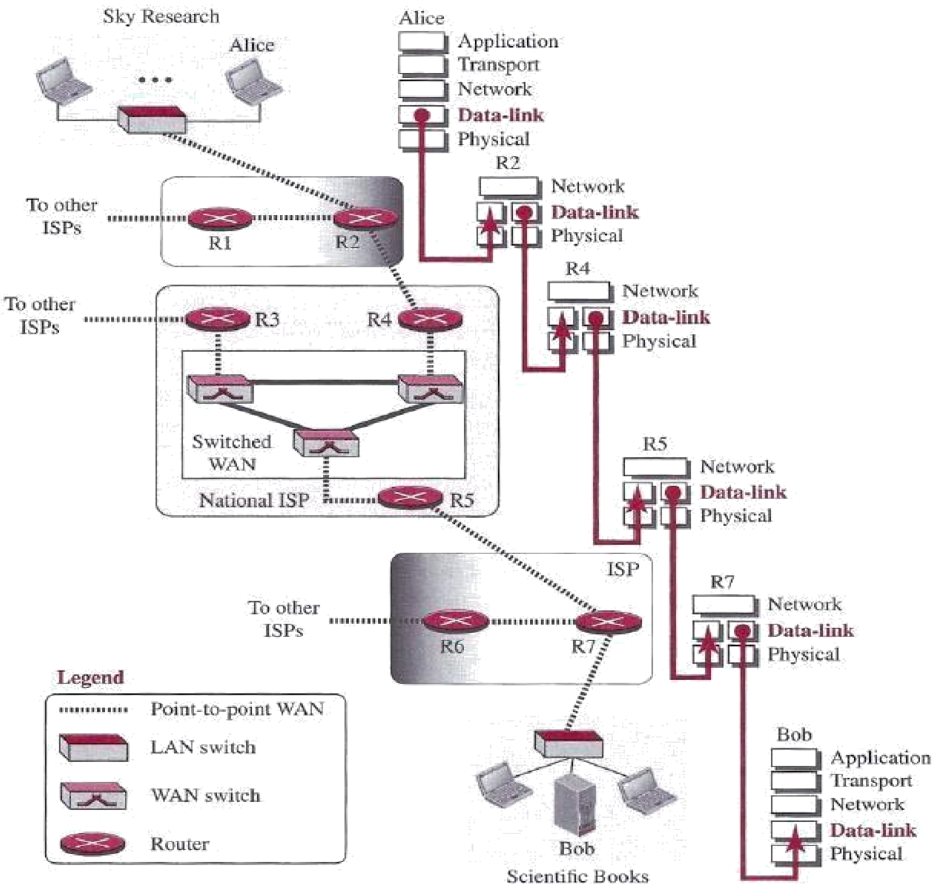
In modern circuit-switched networks, electronic signals pass through several switches before a connection is established. And during a call, no other network traffic can use those switches. The route for the whole length of the communication session between the two communicating bodies is dedicated, exclusive, and released only when the session terminates.Circuit-switching is reliable because when you have a circuit dedicated for a session, you are sure to get all information across.

UNIT II
INTRODUCTION TO DATA LINK LAYER

DATA-LINK LAYER

The Internet is a combination of networks glued together by connecting devices (routers or switches). If a packet is to travel from a host to another host, it needs to pass through these networks. Below figure shows the same scenario. Communication at the data-link layer is made up of five separate logical connections between the data-link layers in the path.

Communication at the data-link layer



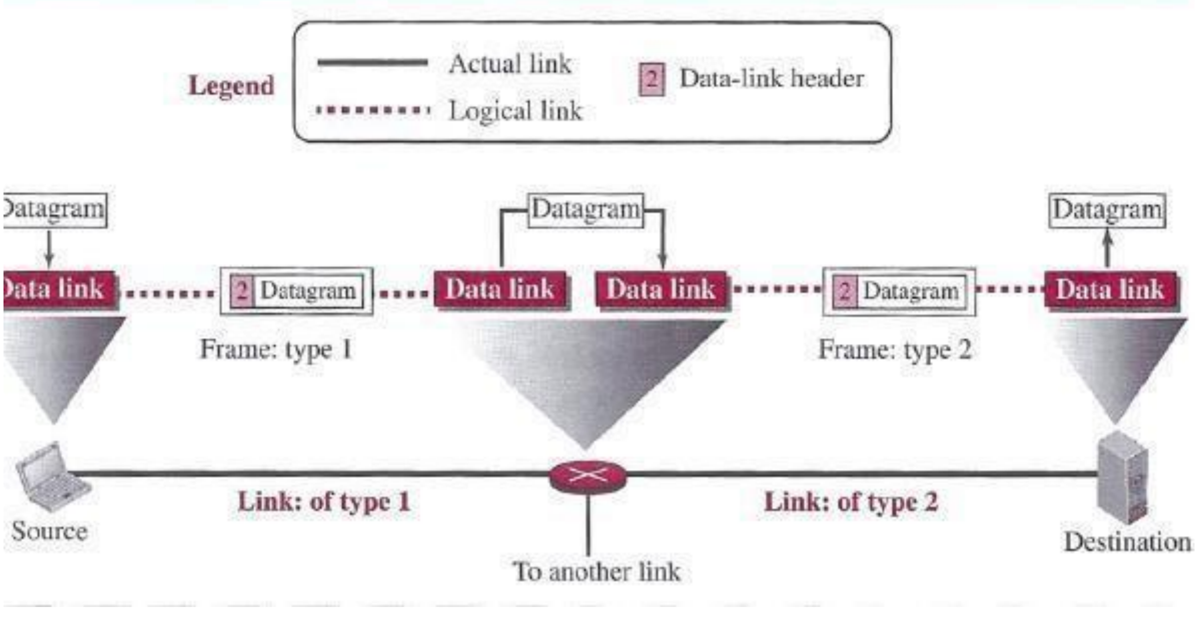
The data-link layer at Alice's computer communicates with the data-link layer at router R2. The data-link layer at router R2 communicates with the data-link layer at router R4

Design Issues:
The data-link layer is located between the physical and the network layers. The data link layer provides services to the network layer; it receives services from the physical layer. Let us discuss services provided by the data-link layer. The duty scope of the data-link layer is node-to-node. When a packet is travelling in the Internet, the data-link layer of a node (host or router) is responsible for delivering a datagram to the next node in the path.

For this purpose, the data-link layer of the sending node needs to encapsulate the datagram received from the network in a frame, and the data-link layer of the receiving node needs to decapsulate the datagram from the frame. In other words, the data-link layer of the source host needs only to encapsulate, the data-link layer of the destination host needs to decapsulate, but each intermediate node needs to both encapsulate and decapsulate. One may ask why we need encapsulation and decapsulation at each intermediate node. The reason is that each link may be using a different protocol with a different frame format. Even if one link and the next are using the same protocol, encapsulation and decapsulation are needed because the link-layer addresses are normally different. An analogy may help in this case. Assume a person needs to travel from her home to her friend's home in another city.

The traveller can use three transportation tools. She can take a taxi to go to the train station in her own city, then travel on the train from her own city to the city where her friend lives, and finallyreach her friend's home using another taxi. Here we have a source node, a destination node, and two intermediate nodes. The traveller needs to get into the taxi at the source node, get out of the taxi and get into the train at the first intermediate node (train station in the city where she lives), get out of the train and get into another taxi at the second intermediate node (train station in the city where her friend lives), and finally get out of the taxi when she arrives at her destination. A kind of encapsulation occurs at the source node, encapsulation and decapsulation occur at the intermediate nodes, and decapsulation occurs at the destination node. For simplicity, we have assumed that we have only one router between the source and destination. The datagram received by the data-link layer of the source host is encapsulated in a frame. The frame is logically transported from the source host to the router. The frame is decapsulated at the data-link layer of the router and encapsulated at another frame. The new frame is logically transported from the router to the destination host. Note that, although we have shown only two data-link layers at the router, the router actually has three data-link layers because it is connected to three physical links.

A communication with only three nodes



Framing

Definitely, the first service provided by the data-link layer is framing. The data-link layer at each node needs to encapsulate the datagram (packet received from the network layer) in a frame before sending it to the next node. The node also needs to decapsulate the datagram from the frame received on the logical channel. Although we have shown only a header for a frame, we will see in future chapters that a frame may have both a header and a trailer. Different data-link layers have different formats for framing. A packet at the data-link layer is normally called *a frame.*

Flow Control

Whenever we have a producer and a consumer, we need to think about flow control. If the producer produces items that cannot be consumed, accumulation of items occurs. The sending data-link layer at the end of a link is a producer of frames; the receiving data-link layer at the other end of a link is a consumer. If the rate of produced frames is higher than the rate of consumed frames, frames at the receiving end need to be buffered while waiting to be consumed (processed). Definitely, we cannot have an unlimited buffer size at the receiving side. We have two choices. The first choice is to let the receiving data-link layer drop the frames if its buffer is full. The second choice is to let the receiving data-link layer send a feedback to the sending data-link layer to ask it to stop or slow down. Different data-link-layer protocols use different strategies for flow control. Since flow control also occurs at the transport layer, with a higher degree of importance, we discuss this issue in Chapter 23 when we talk about the transport layer.

Error Control

At the sending node, a frame in a data-link layer needs to be changed to bits, transformed to electromagnetic signals, and transmitted through the transmission media. At the receiving node, electromagnetic signals are received, transformed to bits, and put together to create a frame. Since electromagnetic signals are susceptible to error, a frame is susceptible to error. The error needs first to be detected. After detection, it needs to be either corrected at the receiver node or discarded and retransmitted by the sending node. Since error detection and correction is an issue in every layer (node-to node or host-to-host).

Congestion Control

Although a link may be congested with frames, which may result in frame loss, most data-link-layer protocols do not directly use a congestion control to alleviate congestion, although some wide-area networks do. In general, congestion control is considered an issue in the network layer or the transport layer because of its end-to-end nature.

Link layer addressing: MAC,ARP, LAN Addressing

Nodes that is hosts and routers have link-layer addresses. Now you might find this surprising, recalling from "The Network Layer" that nodes have network-layer addresses as well. You might be thinking, why in the world do we need to have addresses at both the network and link layers? As well as describing the syntax and function of the link-layer addresses, in this section we hope to shed some light on why the two layers of addresses are useful and, in fact, essential. Well also cover the Address Resolution Protocol (ARP), which provides a mechanism to translate IP addresses to link-layer addresses.

MAC Addresses

In fact, it is not a node (that is, host or router) that has a link-layer address but instead a nodes adapter that has a link-layer address. This is shown in Figure 1. A link-layer address is variously called a LAN address, a physical address, or a MAC address. Because MAC address seems to be the most popular term, well henceforth refer to link-layer addresses as MAC address. For most LANs (including Ethernet and 802.11 wireless LANs) the MAC address is 6 bytes long, giving 248 possible MAC addresses. As shown in Figure 1, these 6-byte addresses are usually expressed in hexadecimal notation, with each byte of the address expressed as a pair of hexadecimal numbers. Although MAC addresses were designed to be permanent, it is now possible to change an adapters MAC address via software. For the rest of this section, however, well assume that an adapters MAC address is fixed. One interesting property of MAC addresses is that no two adapters have the same address. This might seem surprising given that adapters are manufactured in many countries by many companies. How does a company manufacturing adapters in Taiwan make sure that it is using different addresses from a company manufacturing adapters in Belgium? The answer is that the IEEE manages the MAC address space. Particularly, when a company wants to manufacture adapters, it purchases a chunk of the address space consisting of 224 addresses for a nominal fee. IEEE allocates the chunk of 224 addresses by fixing the first 24 bits of a MAC address and letting the company create unique combinations of the last 24 bits for each adapter.

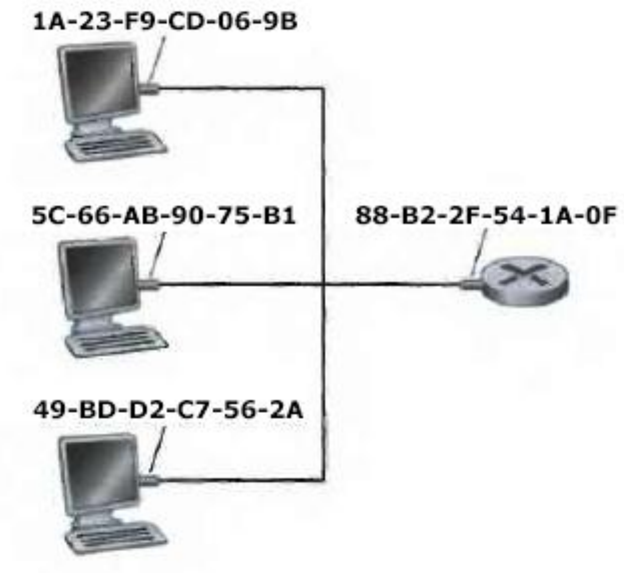


Figure 1. Each adapter connected to a LAN has a unique MAC address.

An adapters MAC address has a flat structure (as opposed to a hierarchical structure) and doesnt change no matter where the adapter goes. A portable computer with an Ethernet card always has the same MAC address, no matter where the computer goes. A PDA with an 802.11 interface always has the same MAC address, no matter where the PDA goes. Remember that, in contrast, IP address have a hierarchical structure (that is, a network part and a host part), and a nodes IP address needs to be changed when the host moves, i.e, changes the network to which it is attached. An adapters MAC address is similar to a person's social security number, which also has a flat addressing structure and which doesn't change no matter where the person goes. An IP address is similar to a person's postal address, which is hierarchical and which must be changed whenever a person moves. Just as a person may find it useful to have both a postal address and a social security number, it is useful for a node to have both a network-layer address and a MAC address.

Error detection and Error Correction:

Error detection and correction has great practical importance in maintaining data (information) integrity across noisy Communication Networks channels and less than- reliable storage media.

Error Correction: Send additional information so incorrect data can be corrected and accepted. Error correction is the additional ability to reconstruct the original, error-free data.

There are two basic ways to design the channel code and protocol for an error correcting system :

• **Automatic Repeat-Request (ARQ) :** The transmitter sends the data and also an error detection code, which the receiver uses to check for errors, and request retransmission of erroneous data. In many cases, the request is implicit; the receiver sends an acknowledgement (ACK) of correctly received data, and the transmitter re-sends anything not acknowledged within a reasonable period of time.

• **Forward Error Correction (FEC) :** The transmitter encodes the data with an error-correcting code (ECC) and sends the coded message. The receiver never sends any messages back to the transmitter. The receiver decodes what it receives into the "most likely" data. The codes are designed so that it would take an "unreasonable" amount of noise to trick the receiver into misinterpreting the data.

Error Detection: Send additional information so incorrect data can be detected and rejected. Error detection is the ability to detect the presence of errors caused by noise or other impairments during transmission from the transmitter to the receiver.

Error Detection Schemes : In telecommunication, a redundancy check is extra data added to a message for the purposes of error detection. Several schemes exist to achieve error detection, and are generally quite simple. All error detection codes transmit more bits than were in the original data. Most codes are "systematic": the transmitter sends a fixed number of original data bits, followed by fixed number of check bits usually referred to as redundancy which are derived from the data bits by some deterministic algorithm.

The receiver applies the same algorithm to the received data bits and compares its output to the received check bits; if the values do not match, an error has occurred at some point during the transmission. In a system that uses a "non-systematic" code, such as some raptor codes, data bits are transformed into at least as many code bits, and the transmitter sends only the code bits.

Repetition Schemes : Variations on this theme exist. Given a stream of data that is to be sent, the data is broken up into blocks of bits, and in sending, each block is sent some predetermined number of times. For example, if we want to send "1011", we may repeat this block three times each. Suppose we send "1011 1011 1011", and this is received as "1010 1011 1011".

As one group is not the same as the other two, we can determine that an error has occurred. This scheme is not very efficient, and can be susceptible to problems if the error occurs in exactly the same place for each group e.g. "1010 1010 1010" in the example above will be detected as correct in this scheme. The scheme however is extremely simple, and is in fact used in some transmissions of numbers stations.

Parity Schemes : A parity bit is an error detection mechanism . A *parity bit* is an extra bit transmitted with a data item, chose to give the resulting bitseven or odd parity. *Parity* refers to the number of bits set to 1 in the data item. There are 2 types of parity

- ***Even parity*** - an even number of bits are 1 Even parity - data: 10010001, parity bit 1
- ***Odd parity*** - an odd number of bits are 1 Odd parity - data: 10010111, parity bit 0

The stream of data is broken up into blocks of bits, and the number of 1 bits is counted. Then, a "parity bit" is set (or cleared) if the number of one bits is odd (or even).This scheme is called even parity; odd parity can also be used. There is a limitation to parity schemes. A parity bit is only guaranteed to detect an odd number of bit errors (one, three, five, and so on). If an even number of bits (two, four, six and so on) are flipped, the parity bit appears to be correct, even though the data is corrupt. For example

- Original data and parity: 10010001+1 (even parity)
- Incorrect data: 10110011+1 (even parity!)

Parity usually used to catch one-bit errors

Checksum: A checksum of a message is an arithmetic sum of message code words of a certain word length, for example byte values, and their carry value. The sum is negated by means of ones-complement, and stored or transferred as an extra code word extending the message. On the receiver side, a new checksum may be calculated, from the extended message.

If the new checksum is not 0, error is detected.Checksum schemes include parity bits, check digits, and longitudinal redundancy check. Suppose we have a fairly long message, which can reasonably be divided into shorter words (a 128 byte message, for instance). We can introduce an accumulator with the same width as a word (one byte, for instance), and as each word comes in, add it to the accumulator.

When the last word has been added, the contents of the accumulator are appended to the message (as a 129th byte, in this case). The added word is called a *checksum*. Now, the receiver performs the same operation, and checks the checksum. If the checksums agree, we assume the message was sent without error.

Hamming Distance Based Checks : If we want to detect d bit errors in an n bit word we can map every n bit word into a bigger n+d+1 bit word so that the minimum Hamming distance between each valid mapping is d+1. This way, if one receives n+d+1 bit word that doesn't match any word in the mapping (with a Hamming distance x <= d+1 from any word in the mapping) it can successfully detect it as an errored word. Even more, d or fewer errors will never transform a valid word into another, because the Hamming distance between each valid word is at least d+1, and such errors only lead to invalid words that are detected correctly.

Given a stream of m*n bits, we can detect x <= d bit errors successfully using the above method on every n bit word. In fact, we can detect a maximum of m*d errors if every n word is transmitted with maximum d errors. The *Hamming distance* between two bit strings is the number of bits you have to change to convert one to the other. The basic idea of an error correcting code is to use extra bits to increase the dimensionality of the hypercube, and make sure the Hamming distance between any two valid points is greater than one.

- If the Hamming distance between valid strings is only one, a single bit error results in another valid string. This means we can't detect an error.
- If it's two, then changing one bit results in an invalid string, and can be detected as an error. Unfortunately, changing just one more bit can result in another valid string, which means we can't detect which bit was wrong; so we can detect an error but not correct it.
- If the Hamming distance between valid strings is three, then changing one bit leaves us only one bit away from the original error, but two bits away from any other valid string. This means if we have a one-bit error, we can figure out which bit is the error; but if we have a two-bit error, it looks like one bit from the other direction. So we can have single bit correction, but that's all.
- Finally, if the Hamming distance is four, then we can correct a single-bit error and detect a double-bit error. This is frequently referred to as a SECDED (Single Error Correct, Double Error Detect) scheme.

Cyclic Redundancy Checks : For CRC following some of Peterson & Brown's notation here . . .

- ***k*** is the length of the message we want to send, *i.e.,* the number of information bits.
- ***n*** is the total length of the message we will end up sending the information bits followed by the check bits. Peterson and Brown call this a *code polynomial*.
- ***n-k*** is the number of check bits. It is also the degree of the generating polynomial. The basic (mathematical) idea is that we're going to pick the n-k check digits in such a way that the code polynomial is divisible by the generating polynomial. Then we send the data, and at the other end we look to see whether it's still divisible by the generating polynomial; if it's not then we know we have an error, if it is, we hope there was no error. The way we calculate a CRC is we establish some predefined n-k+1 bit number P (called the Polynomial, for reasons relating to the fact that modulo-2 arithmetic is a special case of polynomial arithmetic). Now we append n-k 0's to our message, and divide the result by P using modulo-2 arithmetic. The remainder is called the Frame Check Sequence. Now we ship off the message with the remainder appended in place of the 0's. The receiver can either re compute the FCS or see if it gets the same answer, or it can just divide the whole message (including the FCS) by P and see if it gets a remainder of 0. As an example, let's set a 5-bit polynomial of 11001, and compute the CRC of a 16 bit message:

Error Detection and Correction

INTRODUCTION

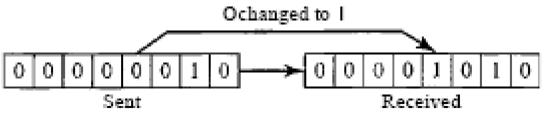
Types of Errors

Whenever bits flow from one point to another, they are subject to unpredictable changes because of interference. This interference can change the shape of the signal. In a single-bit error, a 0 is changed to a 1 or a 1 to a O. In a burst error, multiple bits are changed. For example, a 11100 s burst of impulse noise on a transmission with a data rate of 1200 bps might change all or some of the12 bits of information.

Single-Bit Error

The term *single-bit error* means that only 1 bit of a given data unit (such as a byte, character, or packet) is changed from 1 to 0 or from 0 to 1. Figure 10.1 shows the effect of a single-bit error on a data unit. To understand the impact of the change, imagine that each group of 8 bits is an ASCII character with a 0 bit added to the left. In Figure 10.1,00000010 (ASCII *STX)* was sent, meaning *start of text,* but 00001010 (ASCII *LF)* was received, meaning *line feed.*

:Figure 10.1 *Single-bit error*



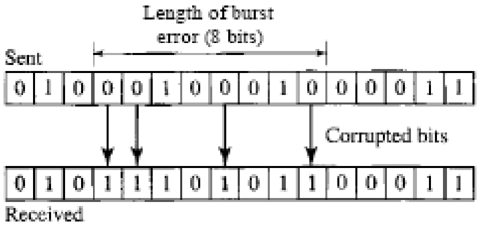
Single-bit errors are the least likely type of error in serial data transmission. To understand why, imagine data sent at 1 Mbps. This means that each bit lasts only 1/1,000,000 s, or 1)ls. For a single-bit error to occur, the noise must have a duration of only 1)ls, which is very rare; noise normally lasts much longer than this.

Burst Error

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

0 to 1. Figure 10.2 shows the effect of a burst error on a data unit. In this case, 0100010001000011 was sent, but 0101110101100011 was received. Note that a burst error does not necessarily mean that the errors occur in consecutive bits. The length of the burst is measured from the first corrupted bit to the last corrupted bit. Some bits in between may not have been corrupted.

Figure 10.2 *Burst error oflength 8*



A burst error is more likely to occur than a single-bit error. The duration of noise is normally longer than the duration of 1 bit, which means that when noise affects data, it affects a set of bits. The number of bits affected depends on the data rate and duration of noise. For example, if we are sending data at I kbps, a noise of 11100 s can affect 10 bits; if we are sending data at I Mbps, the same noise can affect 10,000 bits.

Redundancy

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

Detection Versus Correction

The correction of errors is more difficult than the detection. In error detection, we are looking only to see if any error has occurred. The answer is a simple yes or no. We are not even interested in the number of errors. A single-bit error is the same for us as a burst error.

In error correction, we need to know the exact number of bits that are corrupted and more importantly, their location in the message. The number of the errors and the size of the message are important factors. If we need to correct one single error in an 8-bit data unit, we need to consider eight possible error locations; if we need to correct two errors in a data unit of the same size, we need to consider 28 possibilities. You can imagine the receiver's difficulty in finding 10 errors in a data unit of 1000 bits.

Forward Error Correction Versus Retransmission

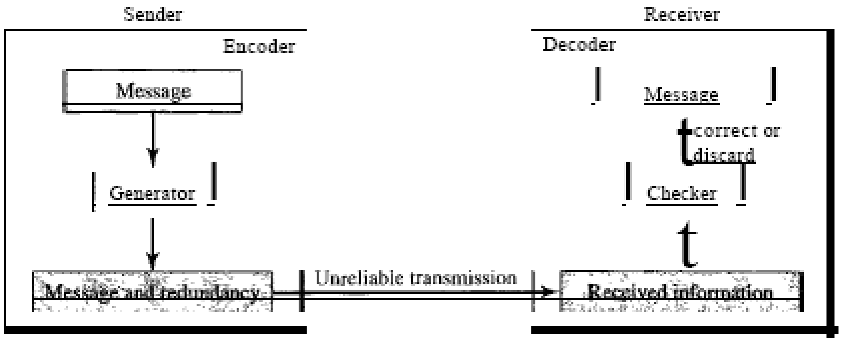
There are two main methods of error correction. Forward error correction is the process in which the receiver tries to guess the message by using redundant bits. This is possible, as we see later, if the number of errors is small. Correction by retransmission is a technique in which the receiver detects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

Coding

Redundancy is achieved through various coding schemes. The sender adds redundant bits through a process that creates a relationship between the redundant bits and the actual data bits. The receiver checks the relationships between the two sets of bits to detect or correct the errors. The ratio of redundant bits to the data bits and the robustness of the process are important factors in any coding scheme. Figure 10.3 shows the general idea of coding.

We can divide coding schemes into two broad categories: block coding and convolution coding..

Figure 10.3 *The structure ofencoder and decoder*

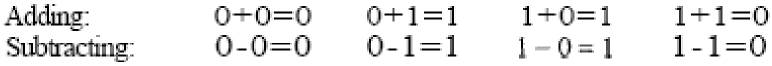


Modular Arithmetic

In modular arithmetic, use only a limited range of integers. An upper limit, called a modulus *N*
then use only the integers 0 to *N* - I, inclusive. This is *modulo-N* arithmetic.

For example, if the modulus is 12, we use only the integers 0 to 11, inclusive. An example of modulo arithmetic is our clock system. It is based on modulo-12 arithmetic, substituting the number 12 for O. In a *modulo-N* system, if a number is greater than *N*, it is divided by *N* and the remainder is the result. If it is negative, as many *Ns* as needed are added to make it positive.

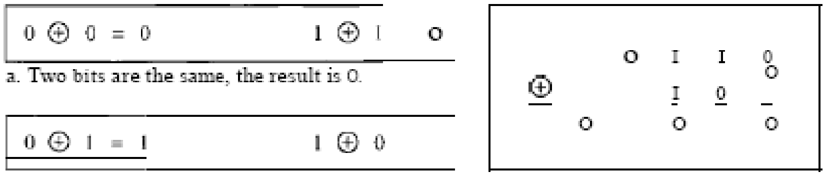
Modulo-2 Arithmetic
Of particular interest is modulo-2 arithmetic. In this arithmetic, the modulus *N* is 2. We can use only 0 and 1. Operations in this



Adding:	0+0=0	0+1=1	1+0=1	1+1=0
Subtracting:	0-0=0	0-1=1	1 - 0 = 1	1-1=0

Notice particularly that addition and subtraction give the same results. In this arithmetic we use the XOR (exclusive OR) operation for both addition and subtraction. The result of an XOR operation is 0 if two bits are the same; the result is I if two bits are different. Figure 10.4 shows this operation.

Figure 10.4 *XORing oftwo single bits or two words*



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

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

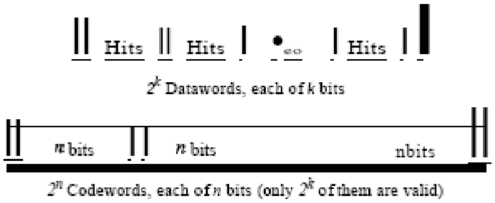
c. Result of XORing two patterns

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. How the extra *r* bits is chosen or calculated is something we will discuss later. For the moment, it is important to know that we have a set of datawords, each of size *k*, and a set of codewords, each of size of *n*. With *k* bits, we can create a combination of *2k* datawords; with *n* bits, we can create a combination of *2n* codewords. Since *n* > *k*, the number of possible codewords is larger than the number of possible datawords.

The block coding process is one-to-one; the same dataword is always encoded as the same codeword. This means that we have *2n* - *2k* codewords that are not used. We call these codewords invalid or illegal. Figure 10.5 shows the situation.

Figure 10.5 *Datawords and codewords in block coding*


Hits
Hits
Hits
2^k Datawords, each of k bits
n bits
n bits
nbits
2^n Codewords, each of n bits (only 2^k of them are valid)

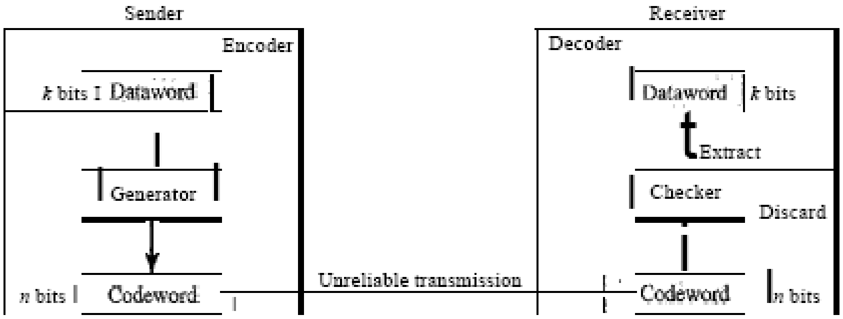
Error Detection

How can errors 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 codewords.
2. The original codeword has changed to an invalid one.

Figure 10.6 shows the role of block coding in error detection.

Figure 10.6 *Process of error detection in block coding*



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

Example 10.2

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.

Table 10.1 *A code for error detection (Example 10.2)*

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

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

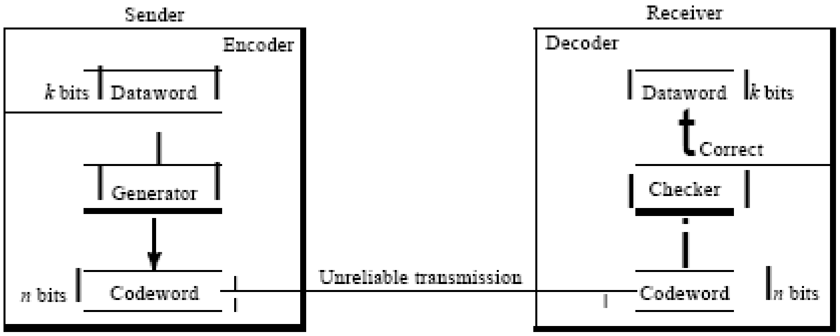
1. The receiver receives 01l. It is a valid codeword. The receiver extracts the dataword 01 from it.
2. The codeword is corrupted during transmission, and 111 is received (the leftmost bit is corrupted).

This is not a valid codeword and is discarded.

Error Correction

As we said before, error correction is much more difficult than error detection. In error detection, the receiver needs to know only that the received codeword is invalid; in error correction the receiver needs to find (or guess) the original codeword sent. We can say that we need more redundant bits for error correction than for error detection. Figure 10.7 shows the role of block coding in error correction. We can see that the idea is the same as error detection but the checker functions are much more complex.

Figure 10.7 *Structure of encoder and decoder in error correction*

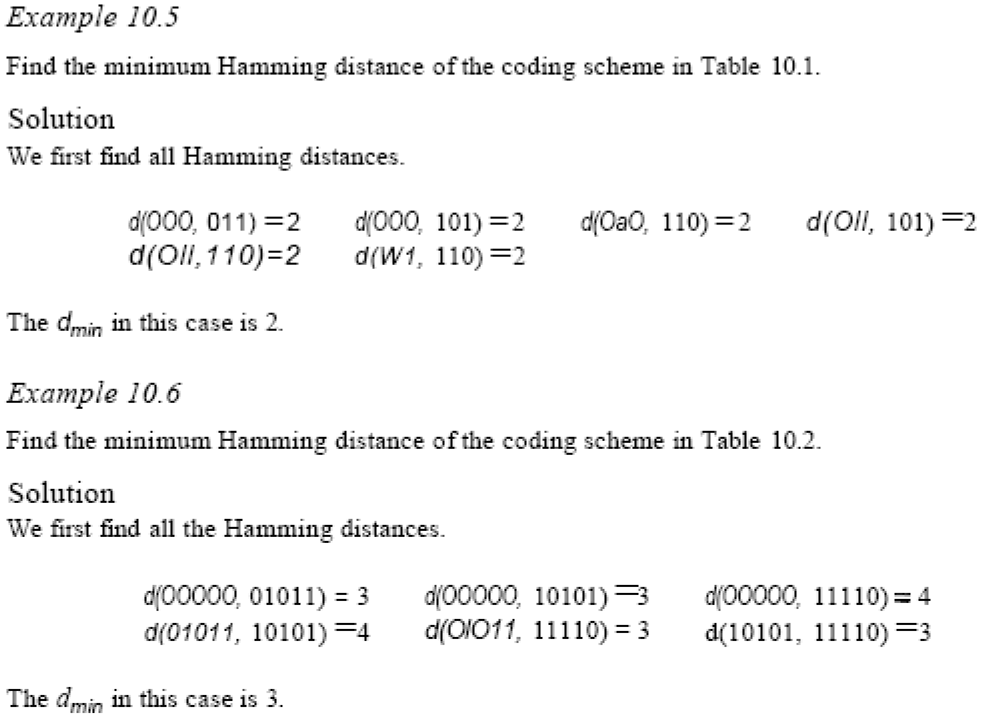


Hamming Distance

One of the central concepts in coding for error control is the idea of the Hamming distance. The Hamming distance between two words (of the same size) is the number of differences between the corresponding bits. We show the Hamming distance between two words *x* and *y* as *d(x, y).* The Hamming distance can easily be found if wc apply the XOR operation (ffi) on the two words and count the number of Is in the result. Note that the Hamming distance is a value greater than zero.

Minimum Hamming Distance

Although the concept of the Hamming distance is the central point in dealing with error detection and correction codes, the measurement that is used for designing a code is the minimum Hamming distance. In a set of words, the minimum Hamming distance is the smallest Hamming distance between all possible pairs. We use dmin to define the minimum Hamming distance in a coding scheme. To find this value, we find the Hamming distances between all words and select the smallest one.



Example 10.5

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

Solution

We first find all Hamming distances.

$d(OOO, 011) = 2$ $d(OOO, 101) = 2$ $d(OaO, 110) = 2$ $d(OII, 101) = 2$
$d(OII, 110) = 2$ $d(W1, 110) = 2$

The d_{min} in this case is 2.

Example 10.6

Find the minimum Hamming distance of the coding scheme in Table 10.2.

Solution

We first find all the Hamming distances.

$d(OOOOO, 01011) = 3$ $d(OOOOO, 10101) = 3$ $d(OOOOO, 11110) = 4$
$d(01011, 10101) = 4$ $d(OIO11, 11110) = 3$ $d(10101, 11110) = 3$

The d_{min} in this case is 3.

Three Parameters

Before we continue with our discussion, we need to mention that any coding scheme needs to have at least three parameters: the codeword size *n,* the dataword size *k,* and the minimum Hamming distance *dmin.* A coding scheme C is written as *C(n, k)* with a separate expression for *dmin-* For example, we can call our first coding scheme *C(3, 2)* with dmin =2 and our second coding scheme *C(5,* 2) with dmin ::= 3.

Hamming Distance and Error

Before we explore the criteria for error detection or correction, let us discuss the relationship between the Hamming distance and errors occurring during transmission. When a codeword is corrupted during transmission, the Hamming distance between the sent and received codewords is the number of bits affected by the error. In other words, the Hamming distance between the received codeword and the sent codeword is the number ofbits that are corrupted during transmission. For example, if the codeword 00000 is sent and 01101 is received, 3 bits are in error and the Hamming distance between the two is *d(OOOOO,* 01101) =3.

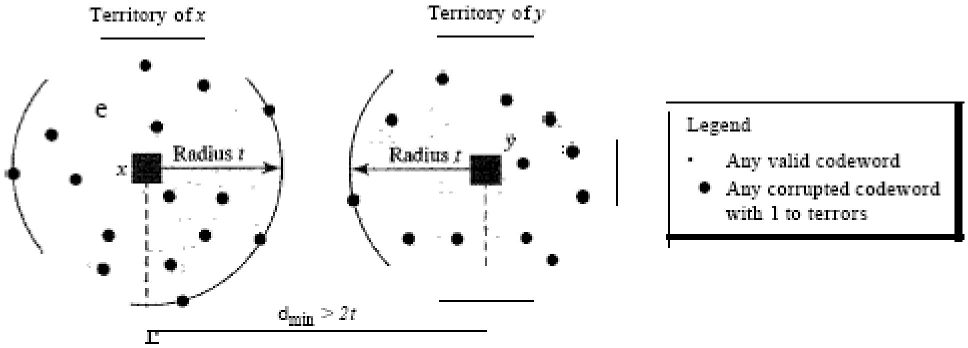
Minimum Distance for Error Detection

Now let us find the minimum Hamming distance in a code if we want to be able to detect up to s errors. If s errors occur during transmission, the Hamming distance between the sent codeword and received codeword is s. If our code is to detect up to s errors, the minimum distance between the valid codes must be s + 1, so that the received codeword does not match a valid codeword. In other words, if the minimum distance between all valid codewords is s + 1, the received codeword cannot be erroneously mistaken for another codeword. The distances are not enough (s
+ 1) for the receiver to accept it as valid. The error will be detected. We need to clarify a point here: Although a code with *dmin* =s + 1

Minimum Distance for Error Correction

Error correction is more complex than error detection; a decision is involved. When a received codeword is not a valid codeword, the receiver needs to decide which valid codeword was actually sent. The decision is based on the concept of territory, an exclusive area surrounding the codeword. Each valid codeword has its own territory. We use a geometric approach to define each territory. We assume that each valid codeword has a circular territory with a radius of *t* and that the valid codeword is at the center. For example, suppose a codeword *x* is corrupted by *t* bits or less. Then this corrupted codeword is located either inside or on the perimeter of this circle. If the receiver receives a codeword that belongs to this territory, it decides that the original codeword is the one at the center. Note that we assume that only up to *t* errors have occurred; otherwise, the decision is wrong. Figure 10.9 shows this geometric interpretation. Some texts use a sphere to show the distance between all valid block codes.

Figure 10.9 *Geometric conceptfor finding d_{min} in error correction*



In Figure 10.9, $d_{min} > 2t$; since the next integer increment is 1, we can say that $d_{min} = 2t + 1$.

CYCLIC CODES

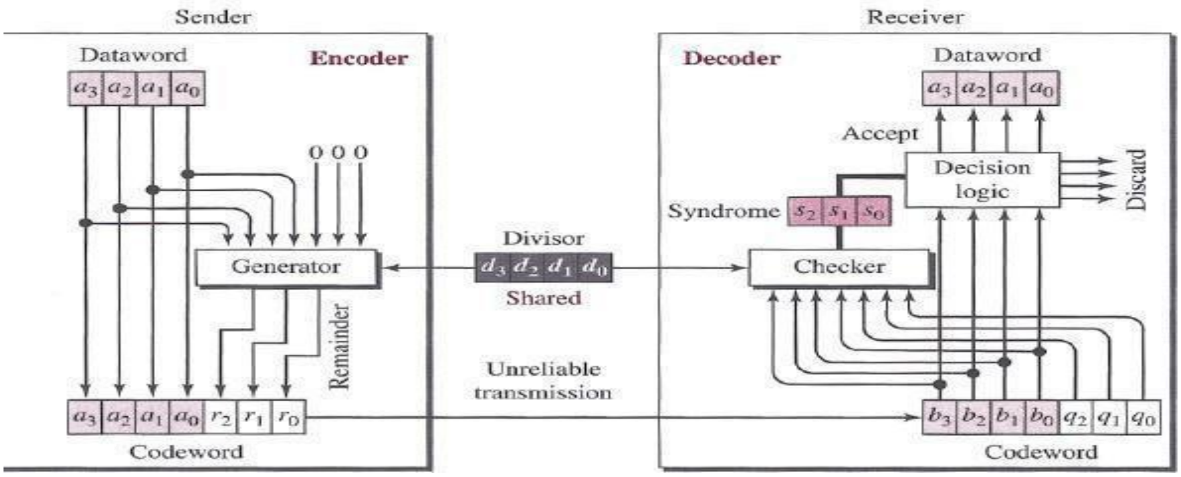
Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword. For example, if 1011000 is a codeword and we cyclically left-shift, then 0110001 is also a codeword. In this case, if we call the bits in the first word *ao* to *a6,* and the bits in the second word *bo* to *b6,* we can shift the bits by using the following: In the rightmost equation, the last bit of the first word is wrapped around and becomes the first bit of the second word.

Cyclic Redundancy Check

We can create cyclic codes to correct errors. However, the theoretical background required is beyond the scope of this book. In this section, we simply discuss a subset of cyclic codes called the cyclic redundancy check (CRC), which is used in networks such as LANs and WANs. Table below shows an example of a CRC code. We can see both the linear and cyclic properties of this code.

Figure below shows one possible design for the encoder and decoder.

CRC encoder and decoder

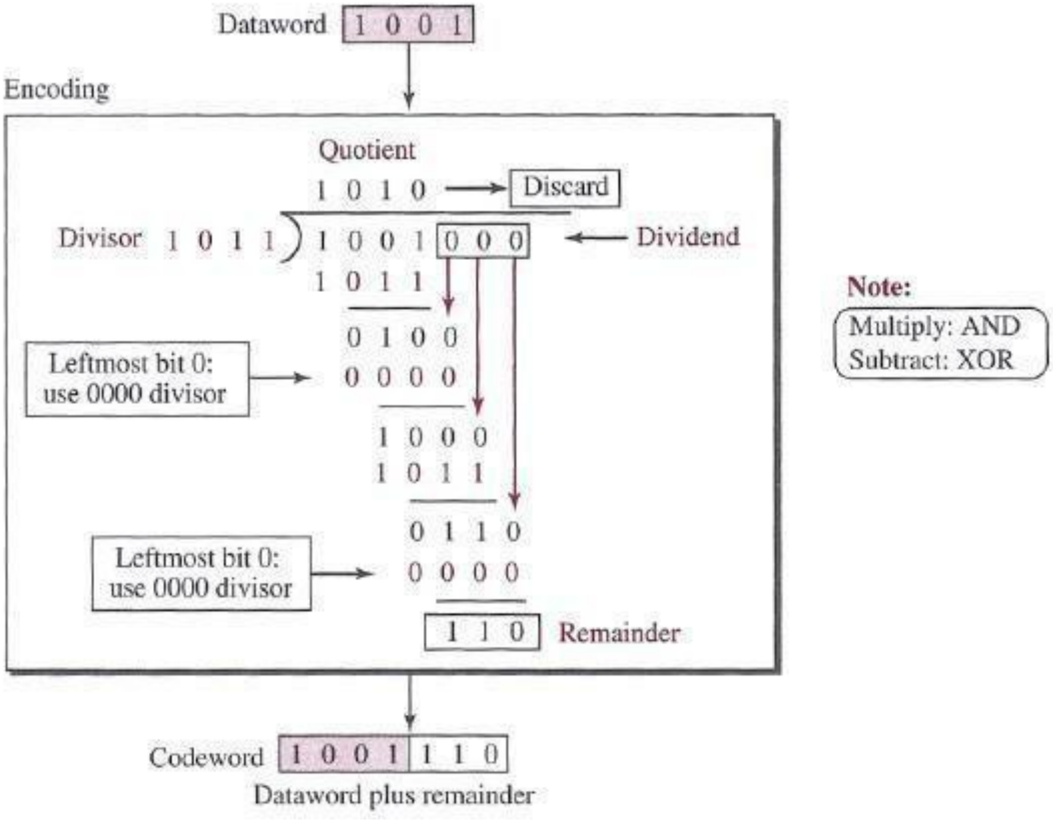


In the encoder, the dataword has *k* bits (4 here); the codeword has *n* bits (7 here). The size of the dataword is augmented by adding *n* - *k* (3 here) Os to the right-hand side of the word. The *n-bit* result is fed into the generator. The generator uses a divisor of size *n* - *k* + 1 (4 here), predefined and agreed upon. The generator divides the augmented dataword by the divisor (modulo-2 division). The quotient of the division is discarded; the remainder *(r2rlrO)* is appended to the dataword to create the codeword. The decoder receives the codeword (possibly corrupted in transition). A copy of all *n* bits is fed to the checker, which is a replica of the generator. The remainder produced by the checker is a syndrome of *n* - *k* (3 here) bits, which is fed to the decision logic analyzer. The analyzer has a simple function. If the syndrome bits are all Os, the 4 leftmost bits of the codeword are accepted as the dataword (interpreted as no error); otherwise, the 4 bits are discarded (error).

Encoder

Let us take a closer look at the encoder. The encoder takes a dataword and augments it with *n* - *k* number of Os. It then divides the augmented dataword by the divisor, as shown in below figure.

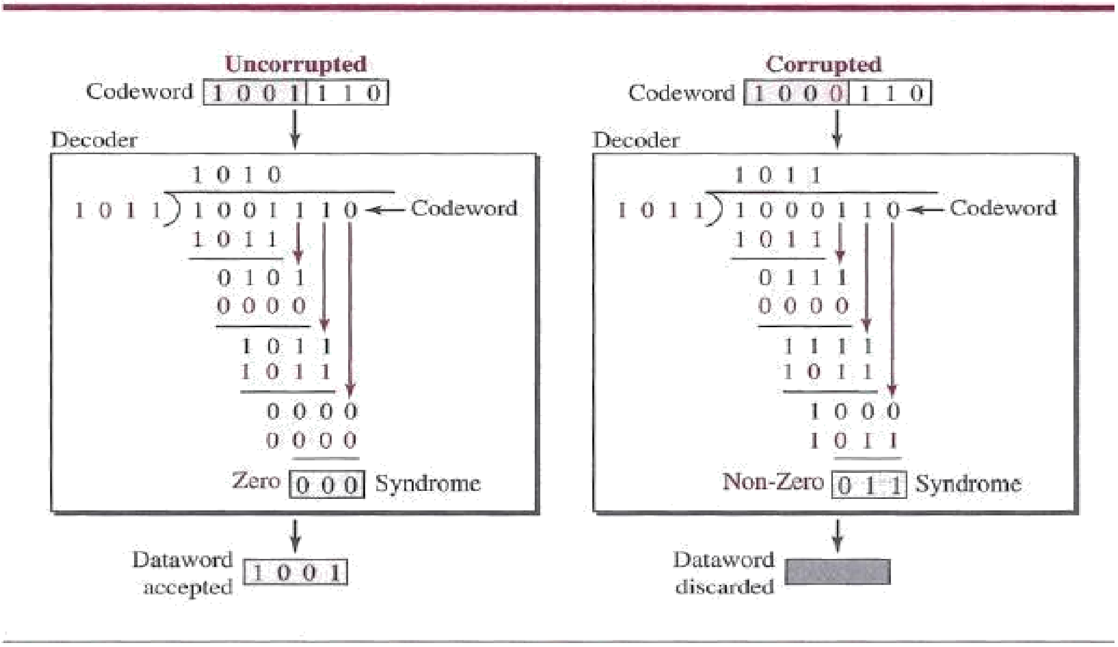
Division in CRC encoder



Decoder

The codeword can change during transmission. The decoder does the same division process as the encoder. The remainder of the division is the syndrome. If the syndrome is all Os, there is no error with a high probability; the dataword is separated from the received codeword and accepted. Otherwise, everything is discarded. Figure 10.7 shows two cases: The left-hand figure shows the value of the syndrome when no error has occurred; the syndrome is 000. The right-hand part of the figure shows the case in which there is a single error. The syndrome is not all Os (it is 011).

Division in the CRC decoder for two cases



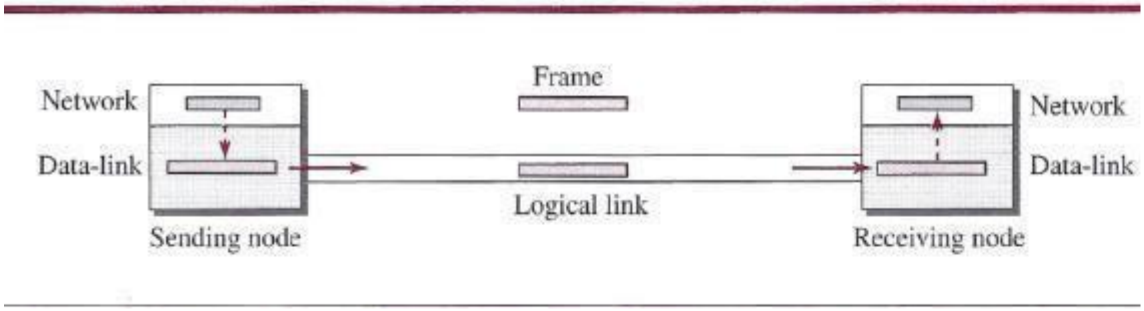
ELEMENT DATA LINK PROTOCOLS AND SLIDING WINDOW PROTOCOL

Traditionally four protocols have been defined for the data-link layer to deal with flow and error control: Simple, Stop-and-Wait, Go-Back-N, and Selective-Repeat. Although the first two protocols still are used at the data-link layer, the last two have disappeared.

Simple Protocol

Our first protocol is a simple protocol with neither flow nor error control. We assume that the receiver can immediately handle any frame it receives. In other words, the receiver can never be overwhelmed with incoming frames. Below figure shows the layout for this protocol.

Simple protocol

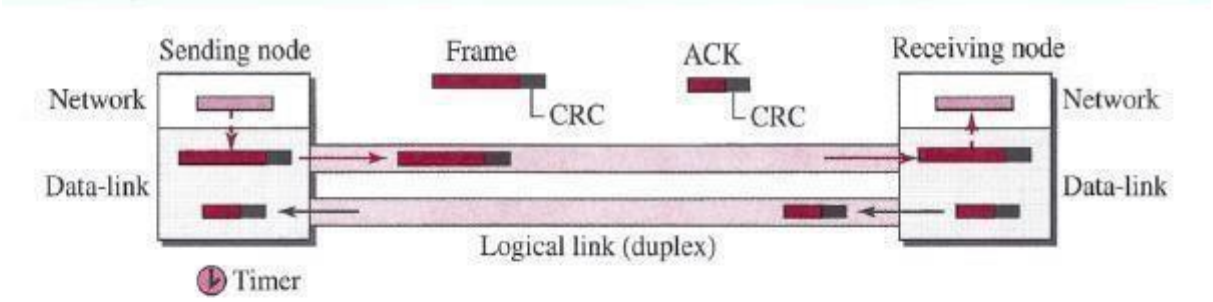


The data-link layer at the sender gets a packet from its network layer, makes a frame out of it, and sends the frame. The data-link layer at the receiver receives a frame from the link, extracts the packet from the frame, and delivers the packet to its network layer. The data-link layers of the sender and receiver provide transmission services for their network layers.

Stop-and- Wait Protocol

Our second protocol is called the Stop-and- Wait protocol, which uses both flow and error control. We show a primitive version of this protocol here, but we discuss the more sophisticated version in Chapter 23 when we have learned about sliding windows. In this protocol, the sender sends one frame at a time and waits for an acknowledgment before sending the next one. To detect corrupted frames, we need to add a CRC to each data frame. When a frame arrives at the receiver site, it is checked. If its CRC is incorrect, the frame is corrupted and silently discarded. The silence of the receiver is a signal for the sender that a frame was either corrupted or lost. Every time the sender sends a frame, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next frame (if it has one to send). If the timer expires, the sender resends the previous frame, assuming that the frame was either lost or corrupted. This means that the sender needs to keep a copy of the frame until its acknowledgment arrives. When the corresponding acknowledgment arrives, the sender discards the copy and sends the next frame if it is ready. Below figure shows the outline for the Stop-and-Wait protocol. Note that only one frame and one acknowledgment can be in the channels at any time.

Stop-and-Wait protocol



HDLC

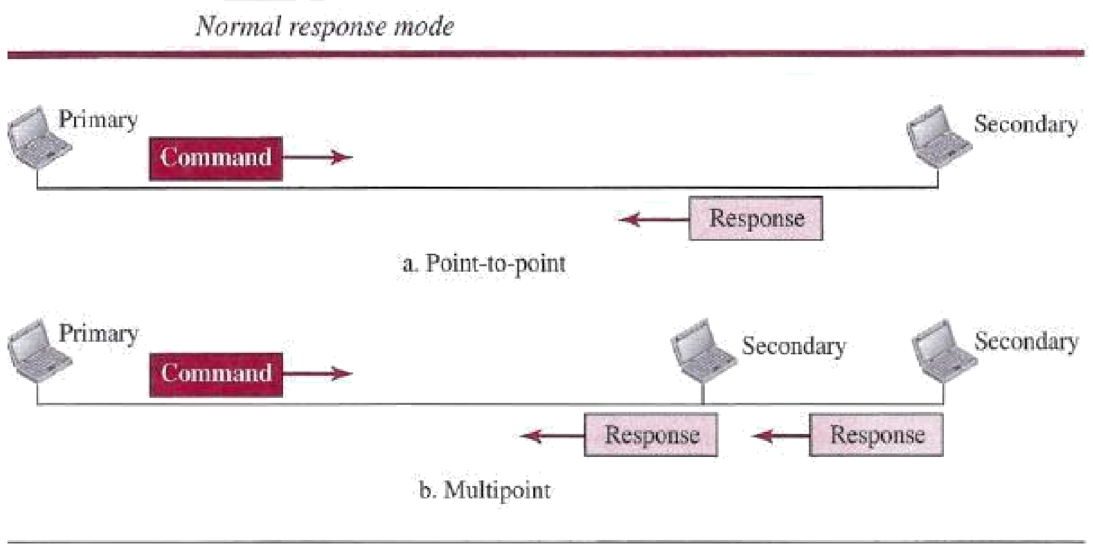
High-level Data Link Control (HDLC) is a bit-oriented protocol for communication over point - to-point and multipoint links. It implements the Stop-and- Wait protocol we discussed earlier. Although this protocol is more a theoretical issue than practical, most of the concept defined in this protocol is the basis for other practical protocols such as PPP, which we discuss next, or the Ethernet protocol.

Configurations and Transfer Modes

HDLC provides two common transfer modes that can be used in different configurations:

Normal response mode (NRM)* and *Asynchronous balanced mode (ABM)

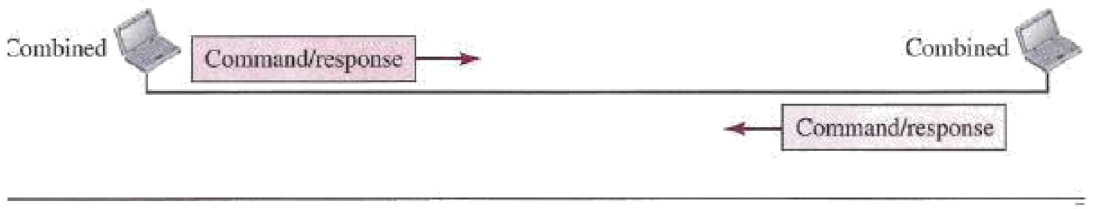
In *normal response mode (NRM),* the station configuration is unbalanced. We have one primary station and multiple secondary stations. A *primary station* can send commands; a *secondary station* can only respond. The NRM is used for both point-to-point and multipoint links, as shown in below Figure. In ABM, the configuration is balanced. The link is point-to-point, and each station can function as a primary and a secondary (acting as peers) this is the common mode today.



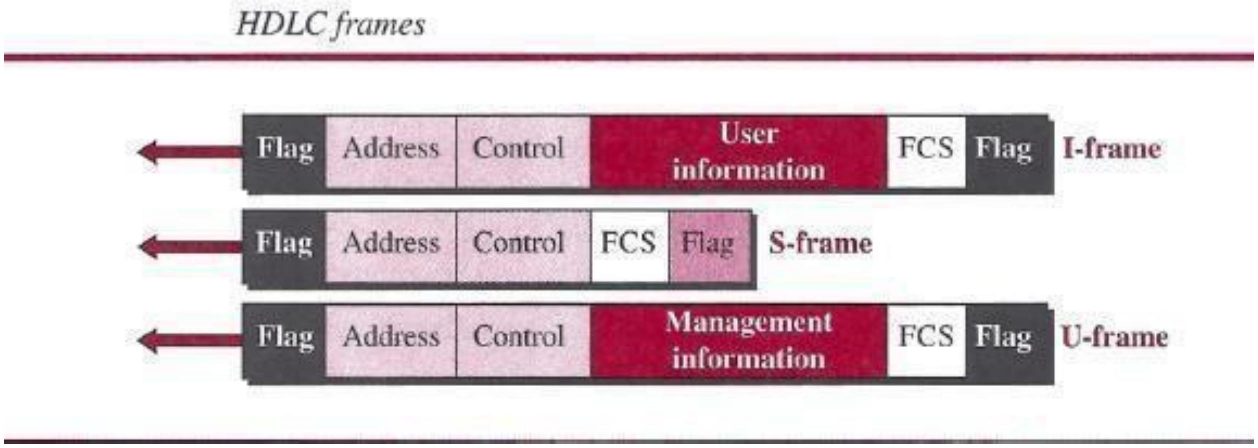
a. Point-to-point

b. Multipoint

Figure 11.15 *Asynchronous balanced mode*



link itself. Each frame in HDLC may contain up to six fields: a beginning flag field, an address field, a control field, an information field, a frame check sequence (FCS) field, and an ending flag field. In multiple-frame transmissions, the ending flag of one frame can serve as the beginning flag of the next frame.

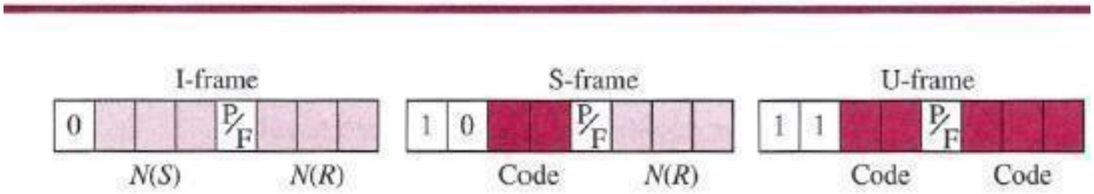


Let us now discuss the fields and their use in different frame types.

- ➢ D *Flag field.* This field contains synchronization pattern 01111110, which identifies both the beginning and the end of a frame.
- ➢ D *Address field.* This field contains the address of the secondary station. If a primary station created the frame, it contains a *to* address. If a secondary station creates the frame, it contains *a from* address. The address field can be one byte or several bytes long, depending on the needs of the network.
    - *Control field.* The control field is one or two bytes used for flow and error control.
    - *Information field.* The information field contains the user's data from the network layer or management information. Its length can vary from one network to another.
    - *FCS field.* The frame check sequence (FCS) is the HDLC error detection field. It can contain either a 2- or 4-byte CRC.

The control field determines the type of frame and defines its functionality. So let us discuss the format of this field in detail. The format is specific for the type of frame, as shown in below Figure.

Control field format for the different frame types



Control Fieldfor I-Frames

I- frames are designed to carry user data from the network layer. In addition, they can include flow- and error-control information (piggybacking). The subfields in the control field are used to define these functions. The first bit defines the type. If the first bit of the control field is 0, this means the frame is an I-frame. The next 3 bits, called *N(S),* define the sequence number of the frame. Note that with 3 bits, we can define a sequence number between 0 and 7. The last 3 bits, called *N(R),* correspond to the acknowledgment number when piggybacking is used. The single bit between *N(S)* and *N(R)* is called the *PIF* bit. The *PIF* field is a single bit with a dual purpose. It has meaning only when it is set (bit = 1) and can mean poll or final. It means *poll* when the frame is sent by a primary station to a secondary (when the address field contains the address of the receiver). It means *final* when the frame is sent by a secondary to a primary (when the address field contains the address of the sender).

Control Field for S-Frames

Supervisory frames are used for flow and error control whenever piggybacking is either impossible or inappropriate. S-frames do not have information fields. If the first 2 bits of the control field are 10, this means the frame is an S-frame. The last 3 bits, called *N(R)*, correspond to the acknowledgment number (ACK) or negative acknowledgment number (NAK), depending on the type of S-frame. The 2 bits called *code* are used to define the type of S-frame itself. With 2 bits, we can have four types of S-frames, as described below:

- ***Receive ready (RR)*** If the value of the code subfield is 00, it is an RR S-frame. This kind of frame acknowledges the receipt of a safe and sound frame or group of frames. In this case, the value of the *N(R)* field defines the acknowledgment number.
- ***Receive not ready (RNR)*** If the value of the code subfield is 10, it is an RNR S frame. This kind of frame is an RR frame with additional functions. It acknowledges the receipt of a frame or group of frames, and it announces that the receiver is busy and cannot receive more frames. It acts as a kind of congestion-control mechanism by asking the sender to slow down. The value of *N(R)* is the acknowledgment number.
- ***Reject (REJ)*** If the value of the code subfield is 01, it is an REJ S-frame. This is a NAK frame, but not like the one used for Selective Repeat ARQ. It is a NAK that can be used in *Go-Back-N* ARQ to improve the efficiency of the process by informing the sender, before the sender timer expires, that the last frame is lost or damaged. The value of *N(R)* is the negative acknowledgment number.
- ***Selective reject (SREJ)*** If the value of the code subfield is 11, it is an SREJ S frame. This is a NAK frame used in Selective Repeat ARQ. Note that the HDLC Protocol uses the term *selective reject* instead of *selective repeat.* The value of *N(R)* is the negative acknowledgment number.

Control Field or V-Frames

Unnumbered frames are used to exchange session management and control information between connected devices. Unlike S-frames, U-frames contain an information field, but one used for system management information, not user data. As with S-frames, however, much of the information carried by If-frames is contained in codes included in

the control field. If-frame codes are divided into two sections: a 2-bit prefix before the *PI* F bit and a 3-bit suffix after the *PIP* bit. Together, these two segments (5 bits) can be used to create up to 32 different types of U-frames.

Control Fieldfor V-Frames

Unnumbered frames are used to exchange session management and control information between connected devices. Unlike S-frames, U-frames contain an information field, but one used for system management information, not user data. As with S-frames, however, much of the information carried by U-frames is contained in codes included in the control field. U-frame codes are divided into two sections: a 2-bit prefix before the *PIP* bit and a 3-bit suffix after the *P/F* bit. Together, these two segments (5 bits) can be used to create up to 32 different types of If-frames.

POINT-TO-POINT PROTOCOL (PPP)

One of the most common protocols for point-to-point access is the **Point-to-Point Protocol (PPP).** Today, millions of Internet users who need to connect their home computers to the server of an Internet service provider use PPP. The majority of these users have a traditional modem; they are connected to the Internet through a telephone line, which provides the services of the physical layer. But to control and manage the transfer of data, there is a need for a point-to-point protocol at the data-link layer. PPP is by far the most common.

Services

The designers of PPP have included several services to make it suitable for a point-to point protocol, but have ignored some traditional services to make it simple.

Services Provided by PPP

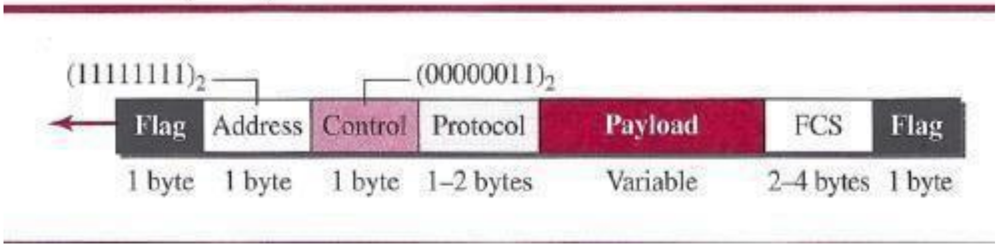
PPP defines the format of the frame to be exchanged between devices. It also defines how two devices can negotiate the establishment of the link and the exchange of data. PPP is designed to accept payloads from several network layers (not only IP). Authentication is also provided in the protocol, but it is optional. The new version of PPP, called *Multilink PPP,* provides connections over multiple links. One interesting feature of PPP is that it provides network address configuration. This is particularly useful when a home user needs a temporary network address to connect to the Internet.

Framing

PPP uses a character-oriented (or byte-oriented) frame. Below figure shows the format of a **PPP** frame. The description of each field follows:

- ***Flag A*** PPP frame starts and ends with a l-byte flag with the bit pattern 01111110.

PPP frame format



- ***Address*** The address field in this protocol is a constant value and set to 11111111 (broadcast address).
- **D *Control*** *This* field is set to the constant value 00000011 (imitating unnumbered frames in HDLC). As we will discuss later, PPP does not provide any flow control. Error control is also limited to error detection.
- ***Protocol*** The protocol field defines what is being carried in the data field: either user data or other information. This field is by default 2 bytes long, but the two parties can agree to use only 1 byte.
- ***Payload field*** This field carries either the user data or other information that we will discuss shortly. The data field is a sequence of bytes with the default of a maximum of 1500 bytes; but this can be changed during negotiation. The data field is byte-stuffed if the flag byte pattern appears in this field. Because there is no field defining the size of the data field, padding is needed if the size is less than the maximum default value or the maximum negotiated value D *FCS.* The frame check sequence (FCS) is simply a 2-byte or 4-byte standard CRC.

Byte Stuffing

Since PPP is a byte-oriented protocol, the flag in PPP is a byte that needs to be escaped whenever it appears in the data section of the frame. The escape byte is 01111101, which means that every time the flag like pattern appears in the data, this extra byte is stuffed to tell the receiver that the next byte is not a flag. Obviously, the escape byte itself should be stuffed with another escape byte.

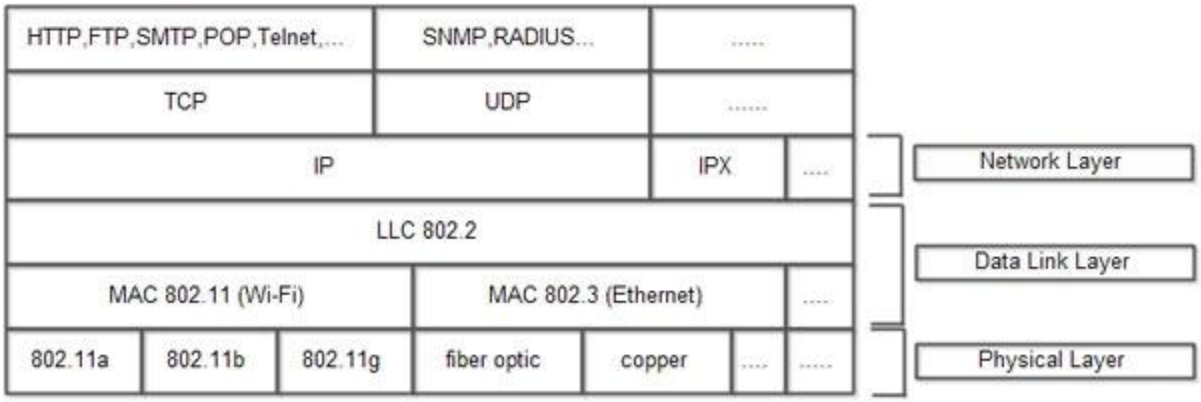
Media Access Control:

The Media Access Control (MAC) data communication Networks protocol sub-layer, also known as the Medium Access Control, is a sub-layer of the data link layer specified in the seven-layer OSI model. The medium access layer was made necessary by systems that share a common communications medium. Typically these are local area networks. The MAC layer is the "low" part of the second OSI layer, the layer of the "data link". In fact, the IEEE divided this layer into two layers "above" is the control layer the logical connection (Logical Link Control, LLC) and "down" the control layer The medium access (MAC).

The LLC layer is standardized by the IEEE as the 802.2 since the beginning 1980 Its purpose is to allow level 3 network protocols (for eg IP) to be based on a single layer (the LLC layer) regardless underlying protocol used, including WiFi, Ethernet or Token Ring, for example. All WiFi data packets so carry a pack LLC, which contains itself packets from the upper network layers. The header of a packet LLC indicates the type of layer 3 protocol in it: most of the time, it is IP protocol, but it could be another protocol, such as IPX (Internet Packet Exchange) for example. Thanks to the LLC layer, it is possible to have at the same time, on the same network, multiple Layer 3 protocols.

In LAN nodes uses the same communication channel for transmission. The MAC sub-layer has two primary responsibilities:

Data encapsulation, including frame assembly before transmission, and frame parsing/error detection during and after reception. Media access control, including initiation of frame transmission and recovery from transmission failure.



Network layers.

Following Protocols are used by Medium Access Layer :

ALOHA : ALOHA is a system for coordinating and arbitrating access to a shared communication channel. It was developed in the 1970s at the University of Hawaii. The original system used terrestrial radio broadcasting, but the system has been implemented in satellite communication systems. A shared communication system like ALOHA requires a method of handling collisions that occur when two or more systems attempt to transmit on the channel at the same time.

In the ALOHA system, a node transmits whenever data is available to send. If another node transmits at the same time, a collision occurs, and the frames that were transmitted are lost. However, a node can listen to broadcasts on the medium, even its own, and determine whether the frames were transmitted.

Carrier Sensed Multiple Access (CSMA) : CSMA is a network access method used on shared network topologies such as Ethernet to control access to the network. Devices attached to the network cable listen (carrier sense) before transmitting. If the channel is in use, devices wait before transmitting. MA (Multiple Access) indicates that many devices can connect to and share the same network. All devices have equal access to use the network when it is clear.

Even though devices attempt to sense whether the network is in use, there is a good chance that two stations will attempt to access it at the same time. On large networks, the transmission time between one end of the cable and another is enough that one station may access the cable even though another has already just accessed it. There are two methods for avoiding these so-called collisions, listed here :

CSMA/CD (Carrier Sense Multiple Access/Collision Detection) : CD (collision detection) defines what happenswhen two devices sense a clear channel, then attempt totransmit at the same time. A collision occurs, and bothdevices stop transmission, wait for a random amount oftime, and then retransmit. This is the technique used to access the 802.3 Ethernet network channel.

This method handles collisions as they occur, but if the bus is constantly busy, collisions can occur so often that performance drops drastically. It is estimated that network traffic must be less than 40 percent of the bus capacity for the network to operate efficiently. If distances are long, time lags occur that may result in inappropriate carrier sensing, and hence collisions.

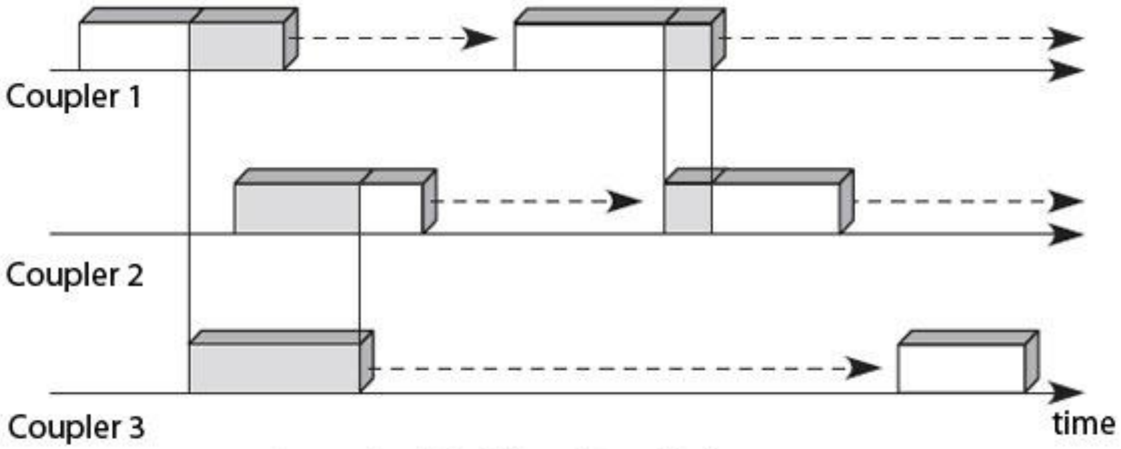
CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) : In CA collision avoidance), collisions areavoided because each node signals its intent to transmitbefore actually doing so. This method is not popular becauseit requires excessive overhead that reduces performance.

- **Ethernet : IEEE 802.3 Local Area Network (LAN) Protocols :** Ethernet protocols refer to the family of local-area network (LAN)covered by the IEEE 802.3. In the Ethernet standard, there are twomodes of operation: half-duplex and full-duplex modes. In the halfduplex mode, data are transmitted using the popular Carrier-SenseMultiple Access/Collision Detection (CSMA/CD) protocol on ashared medium.
- The main disadvantages of the half-duplex are theefficiency and distance limitation, in which the link distance islimited by the minimum MAC frame size. This restriction reducesthe efficiency drastically for high-rate transmission. Therefore, thecarrier extension technique is used to ensure the minimum framesize of 512 bytes in Gigabit Ethernet to achieve a reasonable linkdistance.Four data rates are currently defined for operation over opticalfiber and twisted-pair cables :

- 10 Mbps - 10Base-T Ethernet (IEEE 802.3)
- 100 Mbps - Fast Ethernet (IEEE 802.3u)
- 1000 Mbps - Gigabit Ethernet (IEEE 802.3z)
- 10-Gigabit - 10 Gbps Ethernet (IEEE 802.3ae).
-

Random Access control:

Random Access, which is to issue a completely random time, relies on the Aloha method. The latter takes its name from an experiment performed on a network connecting the various islands of the Hawaiian Archipelago early 1970. In this method, when a coupler has information to transmit, it sends it without worry about other users. If there is a collision, that is to say superposition of two signals or more users, the signals become indecipherable and are lost. They are subsequently transmitted, as shown in Figure, in which the couplers 1, 2 and 3 collide. The coupler 1 transmits its field first because he shot the smallest timer. Then, the module 2 emits, and its signals collide with the coupler 1. Both derive a random time of retransmission. The coupler 3 is listening while the couplers 1 and 2 are silent, so that the frame of the coupler 3 passes successfully. Technical aloha is the origin of all the random access methods.



Operating Principle of Pure Aloha

In addition to its extreme simplicity, aloha has the advantage of not requiring any synchronization and be completely decentralized. Its main drawback is the loss of information resulting from a collision and its lack of efficiency, since the transmission of colliding frames is not interrupted.

The flow rate of such a system becomes very small when the number of couplers increases. It can be shown mathematically that if the number of stations goes to infinity, the flow becomes zero. From a certain moment, the system is more stable. To reduce the likelihood of conflict between users, various improvements of this technique have been proposed.

Slotted aloha, aloha or sliced

Improved technical aloha was to cut the time into time slots, or slots, and to authorize the issuance of frames that slice first, the transmission time of a frame requiring exactly a slice of time. In this way, there is no collision if a single frame transmitted at the beginning of slice. However, if several frames start transmitting in the beginning of slice, the frames emissions are superimposed along the slot. In the latter case, there has retransmission after a random time.

This method improves the throughput during the start-up period but remains unstable. In addition, there is an additional cost from a complication of the devices, since all emissions must be synchronized.

CSMA, or listen with random access carrier

Technical CSMA (Carrier Sense Multiple Access) is to listen to the channel before transmitting. If the module detects a signal on the line, it differs his show at a later date. This significantly reduces the risk of collision, but does not eliminate them completely. If during the propagation time between the couple of the more remote stations (vulnerability period), a coupler does not detect the transmission of a frame, and there may be signal superposition. Therefore, it is necessary to subsequently retransmit lost frames.

Numerous variations of this technique have been proposed, which differ by three Features:

• The strategy followed by the module after detecting the channel status.

• The way collisions are detected.

• The message retransmission after collision policy.

Its main variants are:

• **Non-persistent CSMA**. The coupler the listening channel when a frame is ready to be sent. If the channel is free, the module emits. Otherwise, it starts the same process after a random delay.

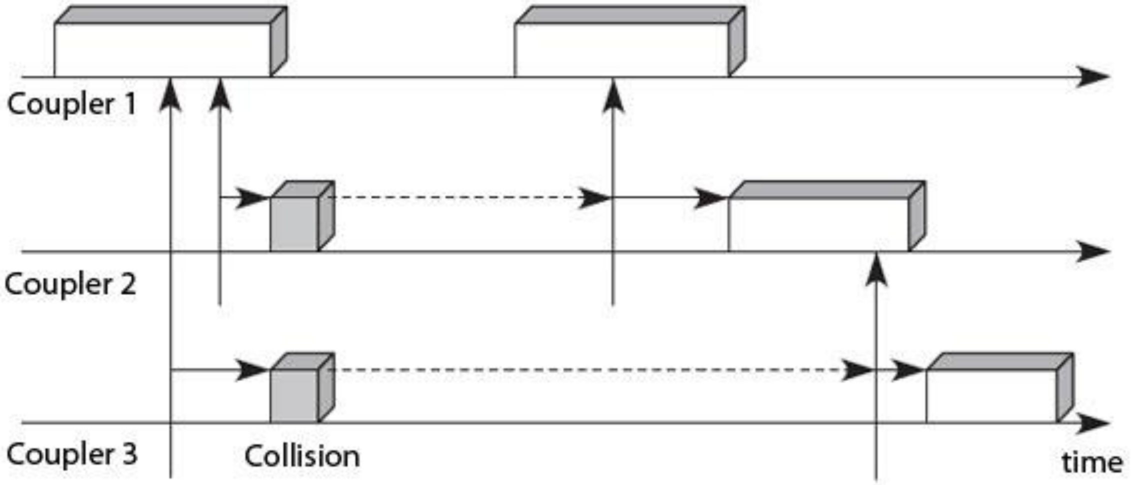
• **Persistent CSMA** - A loan coupler to transmit the channel and previously listening forwards if it is free. If it detects the occupation of carrier, it continues to listen until the channel is clear and transmits at that time. This technique allows lose less time than in the previous case, but it has the disadvantage increase the likelihood of collision, since the frames that accumulate during the busy time are all transmitted simultaneously.

• P-persistent CSMA - The algorithm is the same as before, but when the

Channel becomes free; the module transmits with probability p. In other words, the coupler differs his show with probability 1 - p. This algorithm reduces the likelihood of collision. Assuming both terminals simply making the collision is inevitable in the standard case. With the new algorithm, there is a probability 1 - p that each terminal does not transmit, thereby avoiding the collision. However, it increases the time before transmission, since a terminal may choose not to transmit, with a probability 1 - p, while the channel is free.

• **CSMA / CD (Carrier Sense Multiple Access / Collision Detection)** - This technique normalized random access by the IEEE 802.3 working group is currently the longer used. At a preliminary listening to the network is added listening during transmission. Coupler to issue a loan that detected free channel transmits and continues to listen the channel. The coupler continues to listen, which is sometimes indicated by the CSMA / CD persistent acronym. If there is a collision, it interrupts its transmission as soon as possible and sends special signals, called padding bits so that all couplers are notified of the collision. He tries again his show later using an algorithm that we present later.

Figure shows the CSMA/CD. In this example, the couplers 2 and 3 attempt broadcasting for the coupler 1 transmits its own frame. The couplers 2 and 3 begin to listen and transmit at the same time, the propagation delay around, from the end of the Ethernet frame transmitted by the coupler 1. A collision ensues. Like the couplers 2 and 3 continue to listen to the physical media, they realize the collision, stop their transmission and draw a random time to start retransmission process.



Operating Principle of CSMA / CD

The CSMA/CD create an efficiency gain compared to other techniques random access because there are immediate collision detection and interruption of current transmission. Issuers couplers recognize a collision by comparing the transmitted signal with the passing on the line. The collisions are no longer recognized by absence of acknowledgment but by detecting interference. This conflict detection method is relatively simple, but it requires sufficient performance coding techniques to easily recognize a superposition signal. It is generally used for this differential coding technique, such as differential Manchester code.

• **CSMA / CA** - Less known than the CSMA / CD access CSMA / CA (Carrier Sense Multiple Access / Collision Avoidance) starts to be heavily used in Wi-Fi networks, that is to say, the wireless Ethernet IEEE 802.11. This is a variation of the CSMA / CD, which allows the CSMA method run when collision detection is not possible, as in the radio. Its operating principle is to resolve contention before the data are transmitted using acknowledgments and timers.

The couplers are testing wishing to transmit the channel several times to ensure that no activity is detected. Every message received shall be immediately paid by the receiver. Sending new messages takes place only after a certain period, so as to ensure a transport without loss of information. The non-return of an acknowledgment, after a predetermined time interval, to detect if there was a collision. This strategy not only makes it possible to implement an acknowledgment mechanism in frame but has the advantage of being simple and economic, since it does not require collision detection circuit, unlike the CSMA/ CD.

There are various techniques of CSMA with collision resolution, including the CSMA / CR (Carrier Sense Multiple Access / Collision Resolution). Some variants use the CSMA also priority mechanisms that may come under this term, that avoid collisions by separate priority levels associated with different stations connected to the network.

Controlled Access Protocols

Controlled access:

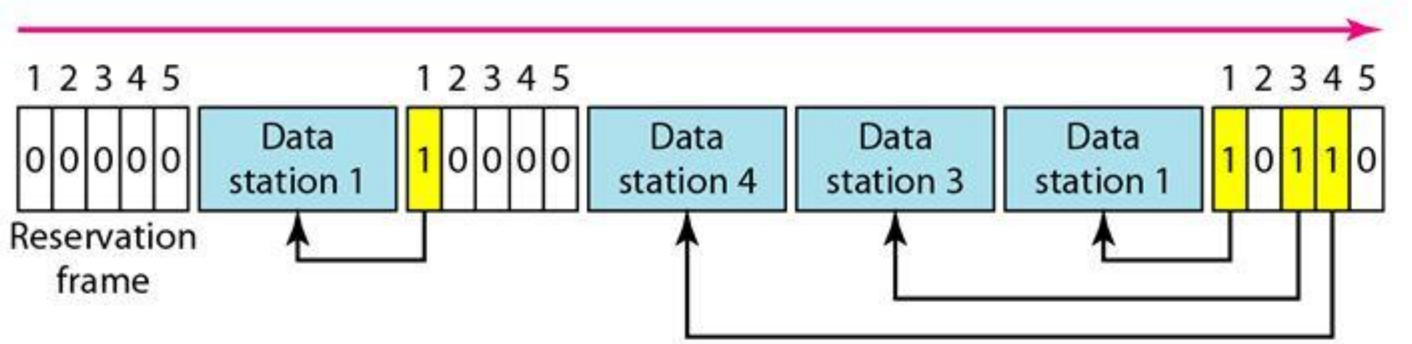
In controlled access, the stations consult one another to find which station has the right to send. A station cannot send unless it has been authorized by other stations. The three popular controlled-access methods are as follows.

1. Reservation:

In the reservation method, a station needs to make a reservation before sending data. Time is divided into intervals. In each interval, a reservation frame precedes the data frames sent in that interval.

If there are N stations in the system, there are exactly N reservation minislots in the reservation frame. Each minislot belongs to a station. When a station needs to send a data frame, it makes a reservation in its own minislot. The stations that have made reservations can send their data frames after the reservation frame.

The following figure shows a situation with five stations and a five-minislot reservation frame. In the first interval, only stations 1, 3, and 4 have made reservations. In the second interval, only station 1 has made a reservation.



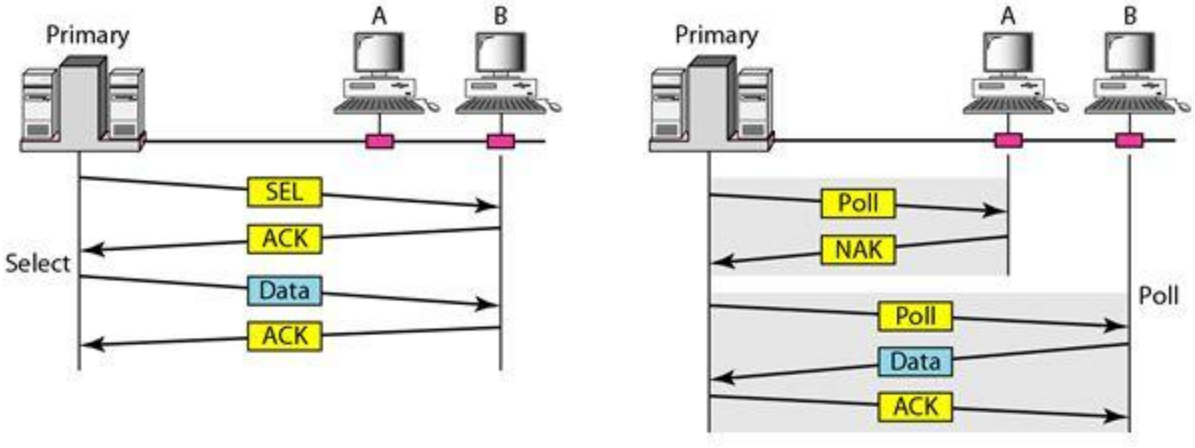
2. Polling:

Polling works with topologies in which one device is designated as a primary station and the other devices are secondary stations. All data exchanges must be made through the primary device even when the ultimate destination is a secondary device.

- Random access Protocols
- Aloha Protocols
- Carrier Sense Multiple Access Protocol

- Carrier Sense Multiple Access with Collision Detection

The primary device controls the link; the secondary devices follow its instructions. It is up to the primary device to determine which device is allowed to use the channel at a given time. The primary device, therefore, is always the initiator of a session. Consider the following figure.



If the primary wants to receive data, it asks the secondaries if they have anything to send, this is called poll function. If the primary wants to send data, it tells the secondary to get ready to receive; this is called select function.

Select:

The select function is used whenever the primary device has something to send. If it has something to send, the primary device sends it. It has to know whether the target device is prepared to receive or not. So the primary must alert the secondary to the upcoming transmission and wait for an acknowledgment of the secondary's ready status. Before sending data, the primary creates and transmits a select (SEL) frame, one field of which includes the address of the intended secondary.

Poll:

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

3. Token Passing:

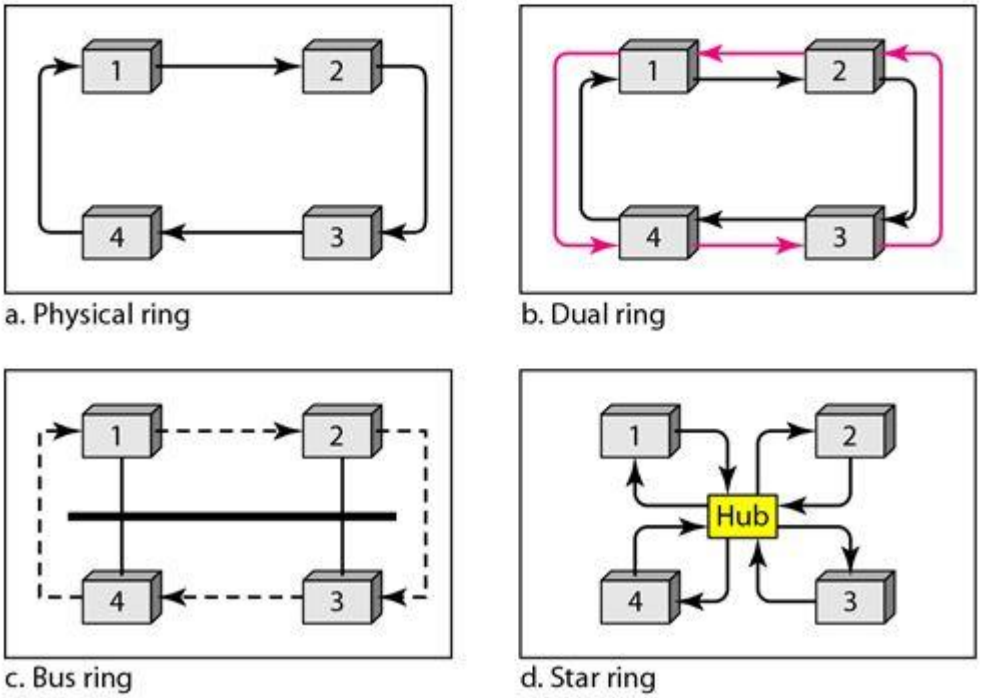
In the token-passing method, the stations in a network are organized in a logical ring. In other words, for each station, there is a predecessor and a successor. The predecessor is the station which is logically before the station in the ring; the successor is the station which is after the station in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

In this method, a special packet called a token circulates through the ring. The possession of the token gives the station the right to access the channel and send its data. When a station has some data to send, it waits until it receives the token from its predecessor. It then holds the token and sends its data. When the station has no more data to send, it releases the token, passing it to the next logical station in the ring. The station cannot send data until it receives the token again in the next round.

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

Logical Ring:

In a token-passing network, stations do not have to be physically connected in a ring; the ring can be a logical one. The following figure show four different physical topologies that can create a logical ring.



a. Physical ring

b. Dual ring

c. Bus ring

d. Star ring

- In the physical ring topology, when a station sends the token to its successor, the token cannot be seen by other stations; the successor is the next one in line. This means that the token does not have to have the address of the next successor. The problem with this topology is that if one of the links-the medium between two adjacent stations fails, the whole system fails.

- The dual ring topology uses a second (auxiliary) ring which operates in the reverse direction compared with the main ring. The second ring is for emergencies only. If one of the links in the main ring fails, the system automatically combines the two rings to form a temporary ring. After the failed link is restored, the auxiliary ring becomes idle again.

- In the bus ring topology, also called a token bus, the stations are connected to a single cable called a bus. They, however, make a logical ring, because each station knows the address of its successor (and also predecessor for token management purposes). When a station has finished sending its data, it releases the token and inserts the address of its successor in the token. Only the station with the address matching the destination address of the token gets the token to access the shared media. The Token Bus LAN, standardized by IEEE, uses this topology.

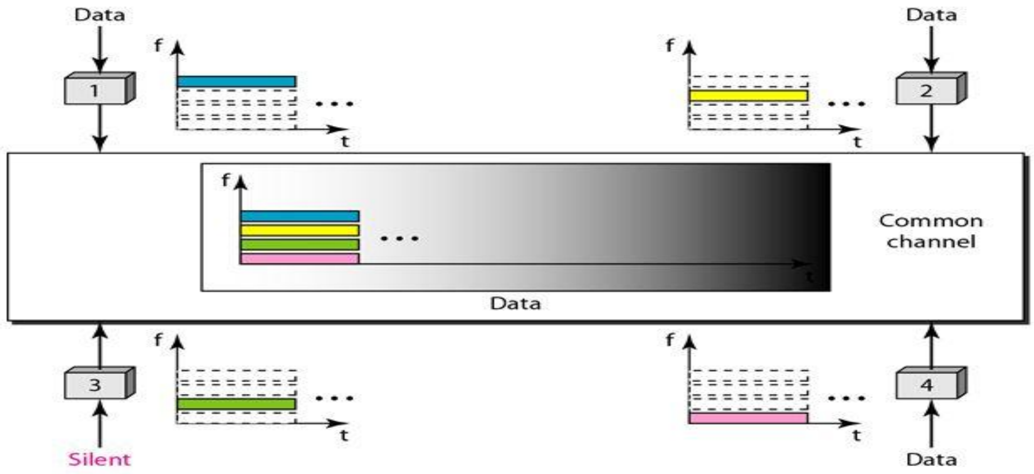
- In a star ring topology, the physical topology is a star. There is a hub, however, that acts as the connector. The wiring inside the hub makes the ring; the stations are connected to this ring through the two wire connections. This topology makes the network less prone to failure because if a link goes down, it will be bypassed by the hub and the rest of the stations can operate. Also adding and removing stations from the ring is easier. This topology is still used in the Token Ring LAN designed by IBM.

Channelization:

Channelization is a multiple-access method in which the available bandwidth of a link is shared in time, frequency, or through code, between different stations. The three channelization protocols are FDMA, TDMA, and CDMA.

The Frequency-Division Multiple Access (FDMA):

In frequency-division multiple access (FDMA), the available bandwidth is divided into frequency bands. Each station is allocated a band to send its data. In other words, each band is reserved for a specific station, and it belongs to the station all the time. Each station also uses a bandpass filter to confine the transmitter frequencies. To prevent station interferences, the allocated bands are separated from one another by small guard bands. The following figure shows the idea of FDMA.



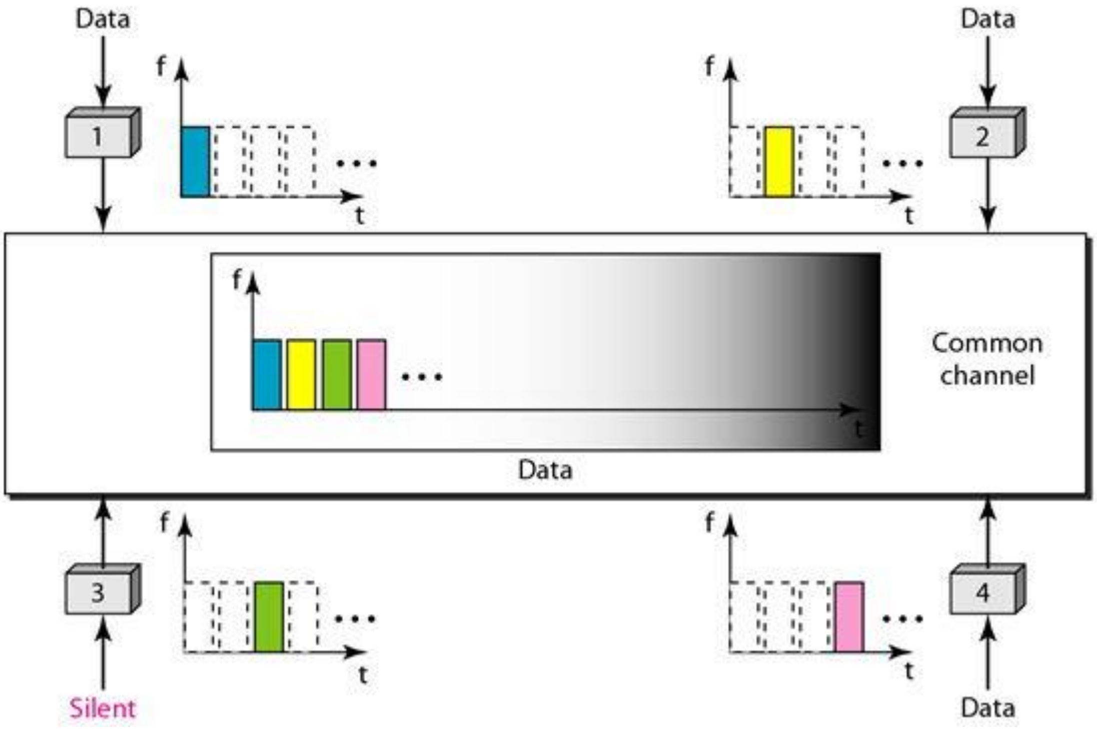
The differences between FDM and FDMA are as follows:

FDM, is a physical layer technique that combines the loads from low-bandwidth channels and transmits them by using a high-bandwidth channel. The channels that are combined are low-pass. The multiplexer modulates the signals, combines them, and creates a bandpass signal. The bandwidth of each channel is shifted by the multiplexer.

FDMA, on the other hand, is an access method in the data link layer. The data link layer in each station tells its physical layer to make a bandpass signal from the data passed to it. The signal must be created in the allocated band. There is no physical multiplexer at the physical layer. The signals created at each station are automatically bandpass-filtered. They are mixed when they are sent to the common channel

Time-Division Multiple Access (TDMA):

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



The main problem with TDMA lies in achieving synchronization between the different stations. Each station needs to know the beginning of its slot and the location of its slot. This may be difficult because of propagation delays introduced in the system if the stations are spread over a large area. To compensate for the delays, we can insert guard times. Synchronization is normally accomplished by having some synchronization bits at the beginning of each slot.

The differences between TDMA and TDM are :

• TDM is a physical layer technique that combines the data from slower channels and transmits them by using a faster channel. The process uses a physical multiplexer that interleaves data units from each channel.

• TDMA, on the other hand, is an access method in the data link layer. The data link layer in each station tells its physical layer to use the allocated time slot. There is no physical multiplexer at the physical layer.

Code-Division Multiple Access (CDMA):

CDMA simply means communication with different codes. CDMA differs from FDMA because only one channel occupies the entire bandwidth of the link. It differs from TDMA because all stations can send data simultaneously; there is no timesharing.

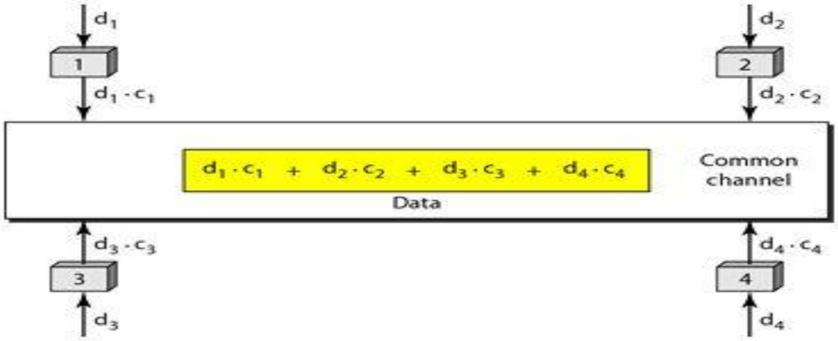
Implementation:

Let us assume we have four stations 1, 2, 3, and 4 connected to the same channel. The data from station 1 are d1 , from station 2 are d2, and so on. The code assigned to the first station is c1, to the second is c2, and so on. We assume that the assigned codes have two properties.

1. If we multiply each code by another, we get 0.

2. If we multiply each code by itself, we get 4 (the number of stations).

With these two properties in mind, how the above four stations can send data using the same common channel, as shown in the following figure.



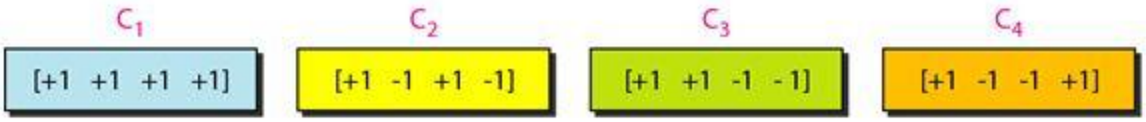
Station 1 multiplies (a special kind of multiplication, as we will see) its data by its code to get d1.c1. Station 2 multiplies its data by its code to get d2.c2. And so on. The data that go on the channel are the sum of all these terms, as shown in the box.

Any station that wants to receive data from one of the other three multiplies the data on the channel by the code of the sender. For example, suppose stations 1 and 2 are talking to each other. Station 2 wants to hear what station 1 is saying. It multiplies the data on the channel by c1 the code of station1.
Because (c1.c1) is 4, but (c2 . c1), (c3. c1), and (c4 .c1) are all 0s, station 2 divides the result by 4 to get the data from station1.

data =(d1.c1+d2.c2+d3.c3+d4.c4).c1
= c1. d1. c1+ c1. d2. c2+ c1. d3. c3+ c1. d4. c4= 4d1

Chips:

CDMA is based on coding theory. Each station is assigned a code, which is a sequence of numbers called chips, as shown in the following figure. The codes are for the previous example.



We need to know that we did not choose the sequences randomly; they were carefully selected. They are called orthogonal sequences and have the following properties:

1. Each sequence is made of N elements, where N is the number of stations.

2. If we multiply a sequence by a number, every element in the sequence is multiplied by that element. This is called multiplication of a sequence by a scalar. For example,

2. [+1 +1-1-1] = [+2+2-2-2]

3. If we multiply two equal sequences, element by element, and add the results, we get N, where N is the number of elements in the each sequence. This is called the inner product of two equal sequences. For example,

[+1 +1-1 -1]. [+1 +1 -1 -1] = 1 + 1 + 1 + 1 = 4

4. If we multiply two different sequences, element by element, and add the results, we get 0. This is called inner product of two different sequences. For example,

[+1 +1 -1-1] • [+1 +1 +1 +1] = 1 + 1 - 1 - 1 = 0

5. Adding two sequences means adding the corresponding elements. The result is another sequence. For example,

[+1+1-1-1]+[+1+1+1+1]=[+2+2 +0 +0].

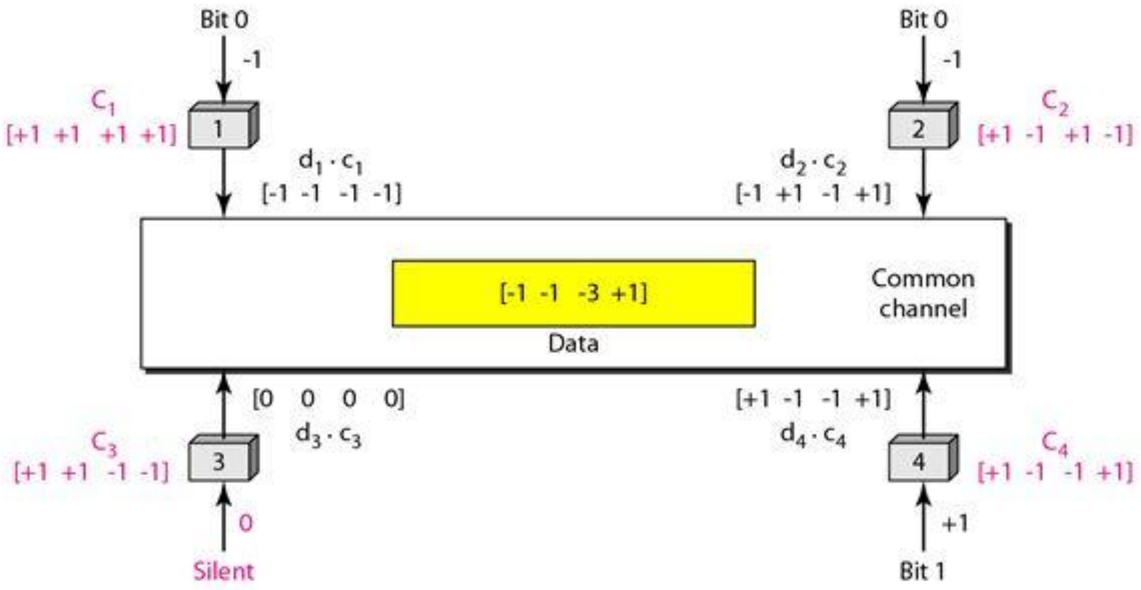
Data Representation:

We follow the following rules for encoding: If a station needs to send a 0 bit, it encodes it as -1, if it needs to send a 1 bit, it encodes it as +1. When a station is idle, it sends no signal, which is interpreted as a 0.

Encoding and Decoding:

As a simple example, we show how four stations share the link during a 1-bit interval. The procedure can easily be repeated for additional intervals. We assume that stations 1 and 2 are sending a 0 bit and channel 4 is sending a 1 bit. Station 3 is silent.

The data at the sender site are translated to -1, -1, 0, and +1. Each station multiplies the corresponding number by its chip (its orthogonal sequence), which is unique for each station. The result is a new sequence which is sent to the channel. For simplicity, we assume that all stations send the resulting sequences at the same time. The sequence on the channel is the sum of all four sequences as defined before. The following figure shows the situation.

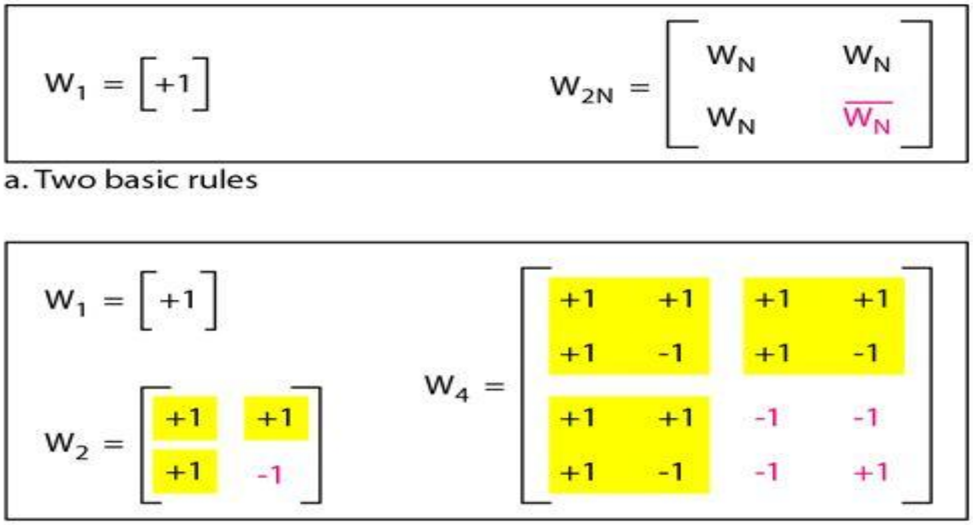


Now imagine station 3, which we said is silent, is listening to station 2. Station 3 multiplies the total data on the channel by the code for station 2, which is [+1 -1 +1-1], to get

[-1-1-3 +1]• [+1-1 +1-1] =-4/4 =-1 ---- --> bit 1

Sequence Generation:

To generate chip sequences, we use a Walsh table, which is a two-dimensional table with an equal number of rows and columns, as shown in the following figure.



a. Two basic rules

b. Generation of W_1, W_2, and W_4

In the Walsh table, each row is a sequence of chips. W1 for a one-chip sequence has one row and one column. We can choose –1 or +1 for the chip for this trivial table (we chose +1).

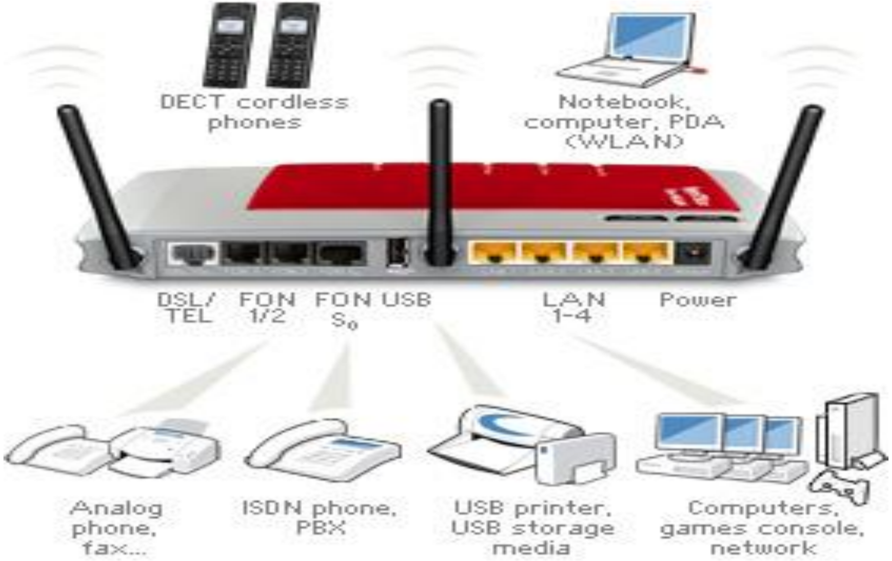
According to Walsh, if we know the table for N sequences WN we can create the table for 2N sequences W2N, as shown in Figure. The WN with the overbar WN stands for the complement of WN where each +1 is changed to -1 and vice versa.

The above figure also shows how we can create W2 and W4 from W1. After we select W1, W2 can be made from four W1 's, with the last one the complement of W1 After W2 is generated, W4 can be made of four W2's, with the last one the complement of W2. Of course, W8 is composed of four W4's, and so on. Note that after WN is made, each station is assigned a chip corresponding to a row.

Something we need to emphasize is that the number of sequences N needs to be a power of 2. In other words, we need to have N = 2^m.

Connecting devices virtual LAN:

Connecting Devices - Hub, Repeater, Switch, Bridge, Router, Gateway


DECT cordless phones
Notebook, computer, PDA (WLAN)
DSL/ TEL
FON 1/2
FON S₀
USB
LAN 1-4
Power
Analog phone, fax...
ISDN phone, PBX
USB printer, USB storage media
Computers, games console, network

To understand what connecting devices are, it is important to know about Backbone Networks. Backbone Network is a means of connecting 2 LAN’s. It provides a transmission channel for packets from being transmitted from one LAN to the other. The individual LAN’s are connected to the Backbone Network by using some types of devices such as Hubs, Repeaters, Switches, Bridges, Routers and Gateways.

Although these terms sound familiar, not many of us know the purpose of using these devices difference between these devices. Hence, it is very important to know the basic function of these devices in order to decide upon the device that is to be used for a particular purpose.

Hub

A hub works in the physical layer of the OSI model. It is basically a non-intelligent device, and has no decision making capability. What a Hub basically does is take the input data from one of the ports and broadcast the information to all the other ports connected to the network.

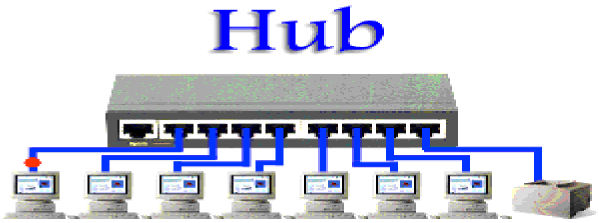


Fig 1: port network

To demonstrate its working, consider a 4 port network as shown in Fig 1. There are 4 computers connected to the 4 ports. Suppose, if Computer A wants to send some data to Computer B using a Hub, then, Computer A broadcasts the data on the network, and Computer B, being connected to the network, has access to the data. But, in this case all the other ports connected to the network has access to the data that is being transmitted by Computer A. This happens because, the Hub works in the Physical Layer and hence it does not know about the MAC addresses of the ports connected to the network. So, there is a lack of security in the Hub.

Fig 2: USB Hub

The picture shows a USB Hub, wherein the data is fed into the input port and is broadcasted to all the other 4 ports. The Network Hubs are outdated and are out of the market.

Repeater:

A repeater is a device similar to the Hub, but has additional features. It also works in the Physical layer. The repeaters are used in places where amplification of input signal is necessary. But, the kind of amplification done by the repeater is different from the regular amplification by amplifiers. The regular amplifies everything fed into it. That means, if the input signal has noise induced into it, both the desired signal and noise signal are together amplified. But, in the case of a repeater, it regenerates the input signal, and amplifies only the desirable ignal. Hence, the noise component of the signal is eliminated.

Fig 3: Repeater

The repeaters are necessary since, during the transmission of the signals over long distances, the signal has attenuation, delay distortions and noise, which lead in loss of data. Hence, in order to prevent this, the regenerative repeaters are used. Hence, the repeater regenerates the faded signal. In addition, it has all the features of a Hub. One common problem between the repeaters and the Hubs are that only one transmission can take place on the network at a particular time. If multiple devices transmit data simultaneously, there will be data collision.

Switch

A switch is an intelligent device that works in the data link layer. The term intelligent refers to the decision making capacity of the Switch. Since it works in the Data link layer, it has knowledge of the MAC addresses of the ports in the network.

Fig 4: Switch

Hence, in the Fig 1, if data has to be sent from Computer A to Computer B, then, the data is transferred to the Computer B only, and not to any other computers connected on the network. Hence, it establishes a link between the sender and the receiver based on the MAC addresses. This also means that when data is being sent from A to B, Computer C can establish a link with Computer D and communication can take place between them. So, simultaneous data transfer is possible in a switch. Also, Hub divides bandwidth, but a Switch does not.

It is also to be noted that a switch is a secure device, because it sends information only to the desired destinations, and also certain security features such as firewalls can be implemented in the Switches.

Bridge

A bridge is also a device which works in the Data Link Layer, but is more primitive when compared to a switch. Initial bridges were used to connect only 2 LAN's, but the most recent ones perform similar operation as the switches. It also works on the principle of transfer of information using the MAC addresses of the ports.

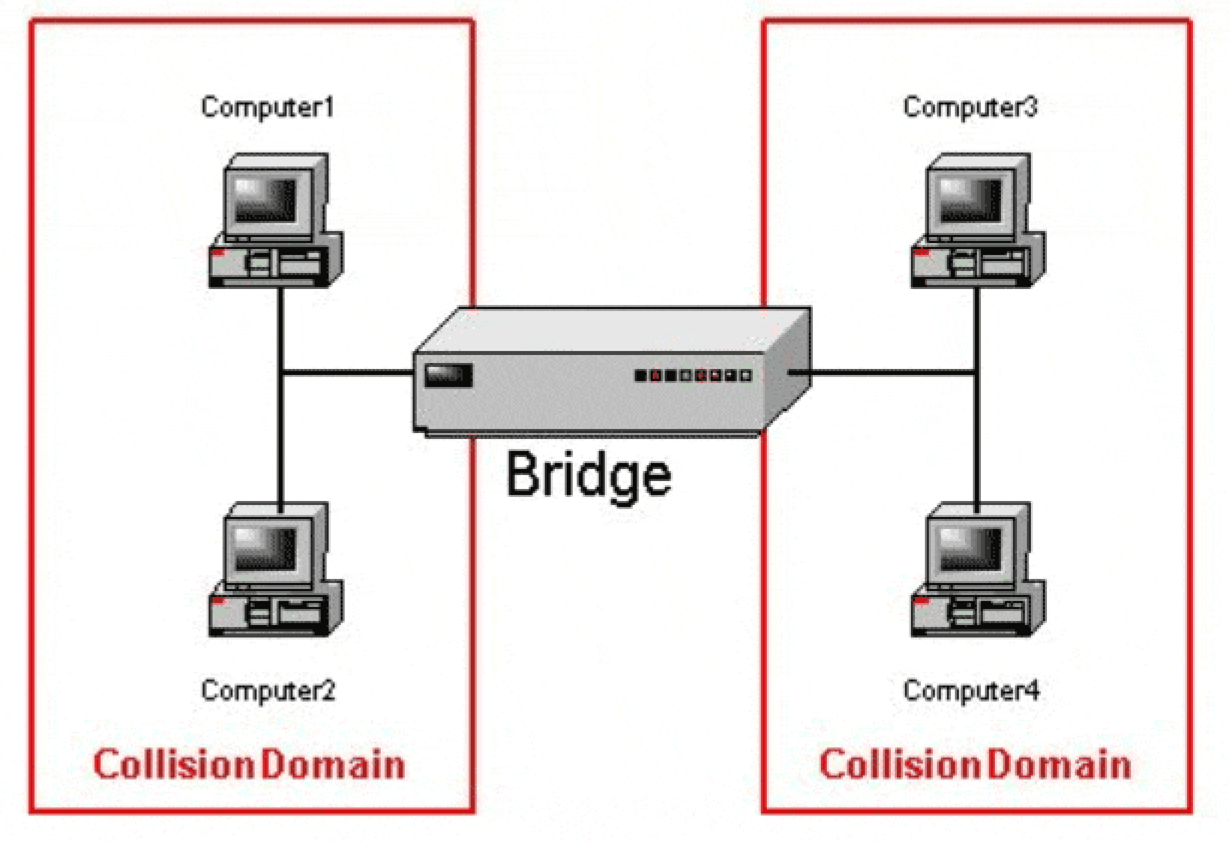


Fig 5: Bridge

It can be noted is that the normal ADSL modem can be connected via bridging also. The only difference is that, when bridging is used, each time the device has to be connected to the internet, it has to dial to the internet and establish a connection. Also, a bridge alone cannot be used to connect to the internet, because, the bridge works in the Data Link Layer, and has no knowledge of the IP Addresses, which are used in the Internet.

Router

Any computer can be connected to the internet via MODEM, which performs the MODulation and the DEModulation operations. But, when there are more than one computer at home or in an organization, and you have a single internet connection, you need a Router. Router is a device which is used when multiple devices need to connect to the Internet using the same IP.

Any Internet Service Provider (ISP) provides a single IP, and especially for personal use, the IP address is assigned dynamically. This is done because, suppose, an ISP has 1000 IP addresses, it does not mean that it has 1000 customers. An ISP assumes that not all devices will be connected to the internet at the same time. Hence, when a user wants to access the internet, any IP address from the pool of IP addresses from the ISP will be assigned to connect the user to the internet.

Fig 6: Router

Hence, the router does the job of connecting multiple devices in a LAN to the internet using the same IP address. Since the router works in the Network Layer, it does forwarding on the basis of IP addresses.

The WiFi routers that are commonly used now are the IEEE 802.11 b/g standard router, which is explained below.

IEEE 802.11

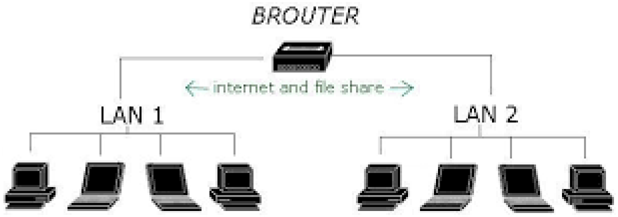
IEEE 802.11 is a standard for WiFi. There are several different technologies/ generations that have been implemented. As mentioned, the recent modems are IEEE 802.11 b/g modems. The word b/g has the meaning as follows:

An IEEE 802.11 b standard uses 2.4GHz band and has a maximum transfer rate of 11 Mbps, while the IEEE 802.11 g standard uses 2.4 GHz band and has maximum transfer rate of 54 Mbps. Thus the b/g modem refers to a dual bandwidth modem, which is compatible with both the b and g standards. The standards are mainly differentiated based on the distance and speed of data transfer.

The more recent IEEE 802.11 N standard has the capability to provide speeds of over 100 Mbps. It basically uses multiple wireless signals and antennas, and has increased signal intensity in order to be able to provide network for greater distances. It employs MIMO technology, wherein spatial encoding is used. The spatial pre-coding is done at the transmitter and the post-coding is done at the receiver. Recently, Reliance Communications was in news for implementing MIMO technology to improve its 3G data transfer speeds.

Brouter

Brouter (Bridging Router) is a device which has two functions. Brouter acts as a router for known protocols (known by the router and those on the network) and hence works in the network layer. For data packets with unknown protocols, it acts as a bridge by connecting two different networks which is the function of a bridge - and this works in the data-link layer.



Gateway

The Gateway devices work in the Transport layer and above, where the different network technologies are implemented. A gateway is necessary when there are different technologies implemented by the different LAN's which are to be connected together.

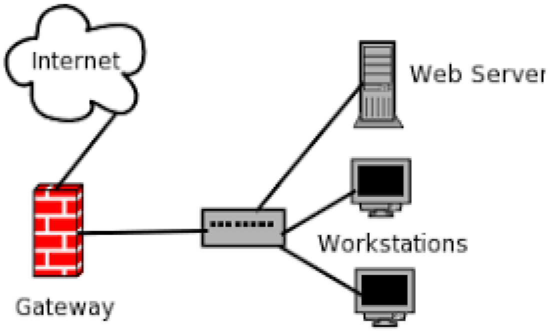


Fig 7: Gateway

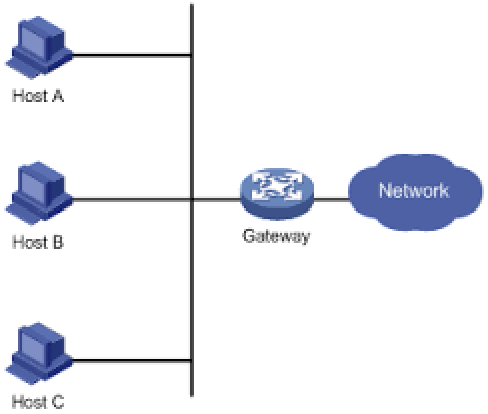
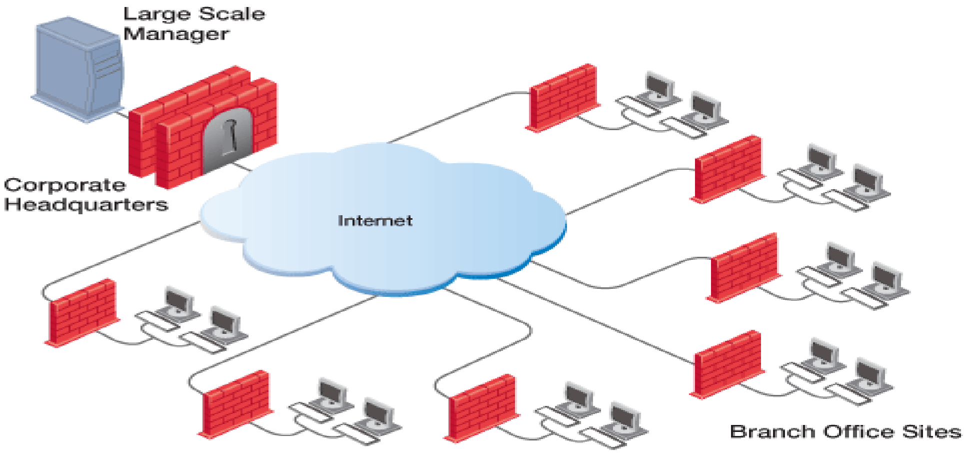


Fig 8: Gateway



The Fig 7 shows the working of a gateway. Consider 2 networks, say in New York, and a network in London. If data has to be sent from one place to another, we need to ensure that the network technologies that are being used by both the networks are the same. If not, we need to use a Gateway.

In the more common example, we use a telephone network and internet networks, which works on different technologies. The telephone network follows the ISDN, and the Internet follows the IP. Here, 2 different technologies are being used. In this case, the router fails to work, since the router cannot understand the functionalities of both the networks. Hence, we require a Gateway, which acts as a translator in communicating between the 2 networks.

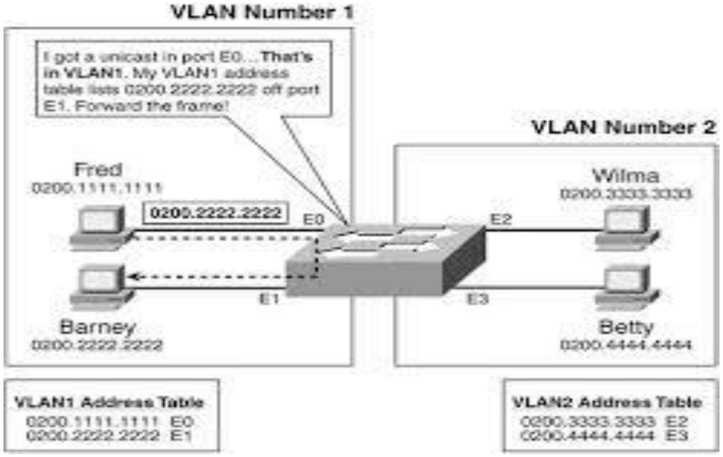
Connecting Cables

While connecting different networks, we come across different connecting cables, which are as follows:

1. RJ45/ RJ 11 Connectors: The RJ45 (Registered Jack 45) cable or the Cat 5 cable, is used to connect the two different LAN's together. This is normally confused with the RJ11 cable, which is used in the interconnections in the telephone network.
2. Crossover cables: Crossover cables are generally used when 2 different computers are to be connected together. They get the name because, in these cables, a crossover is made between the Transmitter and Receiver ports, i.e., Transmitter of one end of the cable is connected to the Receiver port at the other end and vice versa.
3. Null Modem Cables: The null modem cables are also those which are used in connecting 2 different computers to form a network. They also have a crossover, but generally, the term null modem cables are used for RS232 standard cables.
4. Optical Fibres: The optical fibres are used when gigabit Ethernet is used, and very high rates of data transmission is necessary.

Virtual LANS:

VLANs allow network administrators to partition their networks to match the functional and security requirements of their systems without having to run new cables or make major changes in their current network infrastructure. IEEE 802.1Q is the standard defining VLANs; the VLAN identifier or tag consists of 12 bits in the Ethernet frame, creating an inherent limit of 4,096 VLANs on a LAN.



Why use VLAN's: **VLAN's offer a number of advantages over traditional LAN's. They are:**

1) Performance

In networks where traffic consists of a high percentage of broadcasts and multicasts, VLAN's can reduce the need to send such traffic to unnecessary destinations. For example, in a broadcast domain consisting of 10 users, if the broadcast traffic is intended only for 5 of the users, then placing those 5 users on a separate VLAN can reduce traffic Compared to switches, routers require more processing of incoming traffic. As the volume of traffic passing through the routers increases, so does the latency in the routers, which results in reduced performance. The use of VLAN's reduces the number of routers needed, since VLAN's create broadcast domains using switches instead of routers.

2) Formation of Virtual Workgroups

Nowadays, it is common to find cross-functional product development teams with members from different departments such as marketing, sales, accounting, and research. These workgroups are usually formed for a short period of time. During this period, communication between members of the workgroup will be high. To contain broadcasts and multicasts within the workgroup, a VLAN can be set up for them. With VLAN's it is easier to place members of a workgroup together. Without VLAN's, the only way this would be possible is to physically move all the members of the workgroup closer together.

However, virtual workgroups do not come without problems. Consider the situation where one user of the workgroup is on the fourth floor of a building, and the other workgroup members are on the second floor. Resources such as a printer would be located on the second floor, which would be inconvenient for the lone fourth floor user.

Another problem with setting up virtual workgroups is the implementation of centralized server farms, which are essentially collections of servers and major resources for operating a network at a central location. The advantages here are numerous, since it is more efficient and cost-effective to provide better security, uninterrupted power supply, consolidated backup, and a proper operating environment in a single area than if the major resources were scattered in a building. Centralized server farms can cause problems when setting up virtual workgroups if servers cannot be placed on more than one VLAN. In such a case, the server would be placed on a single VLAN and all other VLAN's trying to access the server would have to go through a router; this can reduce performance [Netreference Inc. article].

3) Simplified Administration

Seventy percent of network costs are a result of adds, moves, and changes of users in the network .Every time a user is moved in a LAN, recabling, new station addressing, and reconfiguration of hubs and routers becomes necessary. Some of these tasks can be simplified with the use of VLAN's. If a user is moved within a VLAN, reconfiguration of routers is unnecessary. In addition, depending on the type of VLAN, other administrative work can be reduced or eliminated .However the full power of VLAN's will only really be felt when good management tools are created which can allow network managers to drag and drop users into different VLAN's or to set up aliases.

Despite this saving, VLAN's add a layer of administrative complexity, since it now becomes necessary to manage virtual workgroups .

4) Reduced Cost

VLAN's can be used to create broadcast domains which eliminate the need for expensive routers.

5) Security

Periodically, sensitive data may be broadcast on a network. In such cases, placing only those users who can have access to that data on a VLAN can reduce the chances of an outsider gaining access to the data. VLAN's can also be used to control broadcast domains, set up firewalls, restrict access, and inform the network manager of an intrusion

4.0 How VLAN's work

When a LAN bridge receives data from a workstation, it tags the data with a VLAN identifier indicating the VLAN from which the data came. This is called explicit tagging. It is also possible to determine to which VLAN the data received belongs using implicit tagging. In implicit tagging the data is not tagged, but the VLAN from which the data came is determined based on other information like the port on which the data arrived. Tagging can be based on the port from which it came, the source Media Access Control (MAC) field, the source network address, or some other field or combination of fields. VLAN's are classified based on the method used. To be able to do the tagging of data using any of the methods, the bridge would have to keep an updated database containing a mapping between VLAN's and whichever field is used for tagging. For example, if tagging is by port, the database should indicate which ports belong to which VLAN. This database is called a filtering database. Bridges would have to be able to maintain this database and also to make sure that all the bridges on the LAN have the same information in each of their databases. The bridge determines where the data is to go next based on normal LAN operations. Once the bridge determines where the data is to go, it now needs to determine whether the VLAN identifier should be added to the data and sent. If the data is to go to a device that knows about VLAN implementation (VLAN-aware), the VLAN identifier is added to the data. If it is to go to a device that has no knowledge of VLAN implementation (VLAN-unaware), the bridge sends the data without the VLAN identifier.

In order to understand how VLAN's work, we need to look at the types of VLAN's, the types of connections between devices on VLAN's, the filtering database which is used to send traffic to the correct VLAN, and tagging, a process used to identify the VLAN originating the data.

VLAN Standard: IEEE 802.1Q Draft Standard

There has been a recent move towards building a set of standards for VLAN products. The Institute of Electrical and Electronic Engineers (IEEE) is currently working on a draft standard 802.1Q for VLAN's. Up to this point, products have been proprietary, implying that anyone wanting to install VLAN's would have to purchase all products from the same vendor. Once the standards have been written and vendors create products based on these standards, users will no longer be confined to purchasing products from a single vendor. The major vendors have supported these standards and are planning on releasing products based on them. It is anticipated that these standards will be ratified later this year.

Types of VLAN's

VLAN membership can be classified by port, MAC address, and protocol type.

1) Layer 1 VLAN: Membership by Port

Membership in a VLAN can be defined based on the ports that belong to the VLAN. For example, in a bridge with four ports, ports 1, 2, and 4 belong to VLAN 1 and port 3 belongs to VLAN 2 (see *Figure*3).

Port	VLAN
1	1
2	1
3	2
4	1

*Figure*3: Assignment of ports to different VLAN's.

The main disadvantage of this method is that it does not allow for user mobility. If a user moves to a different location away from the assigned bridge, the network manager must reconfigure the VLAN.

2) Layer 2 VLAN: Membership by MAC Address

Here, membership in a VLAN is based on the MAC address of the workstation. The switch tracks the MAC addresses which belong to each VLAN (see *Figure*4). Since MAC addresses form a part of the workstation's network interface card, when a workstation is moved, no reconfiguration is needed to allow the workstation to remain in the same VLAN. This is unlike Layer 1 VLAN's where membership tables must be reconfigured.

MAC Address	VLAN
1212354145121	1
2389234873743	2
3045834758445	2
5483573475843	1

*Figure*4: Assignment of MAC addresses to different VLAN's.

The main problem with this method is that VLAN membership must be assigned initially. In networks with thousands of users, this is no easy task. Also, in environments where notebook PC's are used, the MAC address is associated with the docking station and not with the notebook PC. Consequently, when a notebook PC is moved to a different docking station, its VLAN membership must be reconfigured.

3) Layer 2 VLAN: Membership by Protocol Type

VLAN membership for Layer 2 VLAN's can also be based on the protocol type field found in the Layer 2 header (see *Figure*5).

Protocol	VLAN
IP	1
IPX	2

*Figure*5: Assignment of protocols to different VLAN's.

4) Layer 3 VLAN: Membership by IP Subnet Address

Membership is based on the Layer 3 header. The network IP subnet address can be used to classify VLAN membership (see *Figure 6*).

IP Subnet	VLAN
23.2.24	1
26.21.35	2

*Figure*6: Assignment of IP subnet addresses to different VLAN's.

Although VLAN membership is based on Layer 3 information, this has nothing to do with network routing and should not be confused with router functions. In this method, IP addresses are used only as a mapping to determine membership in VLAN's. No other processing of IP addresses is done.

In Layer 3 VLAN's, users can move their workstations without reconfiguring their network addresses. The only problem is that it generally takes longer to forward packets using Layer 3 information than using MAC addresses.

5) Higher Layer VLAN's

It is also possible to define VLAN membership based on applications or service, or any combination thereof. For example, file transfer protocol (FTP) applications can be executed on one VLAN and telnet applications on another VLAN.

The 802.1Q draft standard defines Layer 1 and Layer 2 VLAN's only. Protocol type based VLAN's and higher layer VLAN's have been allowed for, but are not defined in this standard. As a result, these VLAN's will remain proprietary.

Types of Connections

Devices on a VLAN can be connected in three ways based on whether the connected devices are VLAN-aware or VLAN-unaware. Recall that a VLAN-aware device is one which understands VLAN memberships (i.e. which users belong to a VLAN) and VLAN formats.

1) Trunk Link

All the devices connected to a trunk link, including workstations, must be VLAN-aware. All frames on a trunk link must have a special header attached. These special frames are called tagged frames (see *Figure*7).

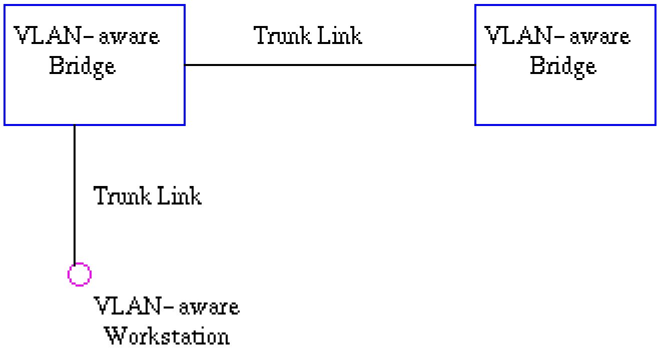


*Figure*7: Trunk link between two VLAN-aware bridges.

2) Access Link

An access link connects a VLAN-unaware device to the port of a VLAN-aware bridge. All frames on access links must be implicitly tagged (untagged) (see *Figure*8). The VLAN-unaware device can be a LAN segment with VLAN-unaware workstations or it can be a number of LAN segments containing VLAN-unaware devices (legacy LAN).

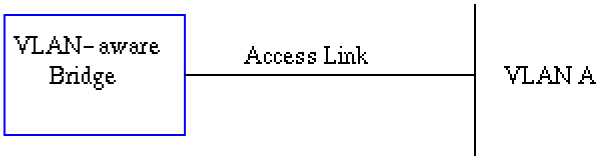


Figure 8: Access link between a VLAN-aware bridge and a VLAN-unaware device.

3) Hybrid Link

This is a combination of the previous two links. This is a link where both VLAN-aware and VLAN-unaware devices are attached (see *Figure*9). A hybrid link can have both tagged and untagged frames, but *all* the frames for a specific VLAN must be either tagged or untagged.

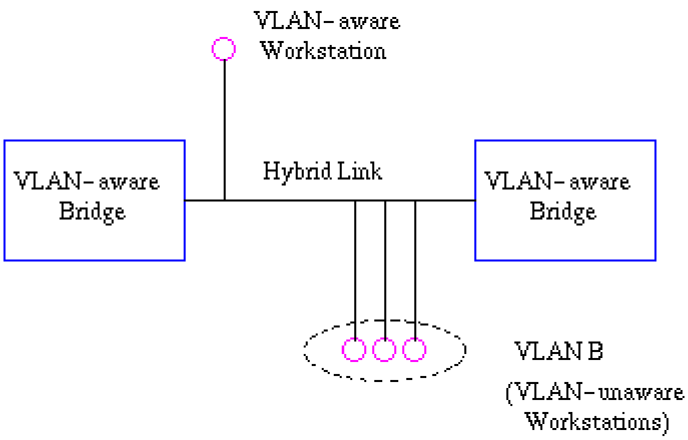


*Figure*9: Hybrid link containing both VLAN-aware and VLAN-unaware devices.

It must also be noted that the network can have a combination of all three types of links.

UNIT – III
THE NETWORK LAYER

Introduction: Layer-3 in the OSI model is called Network layer. Network layer manages options pertaining to host and network addressing, managing sub-networks, and internetworking.

Network layer takes the responsibility for routing packets from source to destination within or outside a subnet. Two different subnet may have different addressing schemes or non-compatible addressing types. Same with protocols, two different subnet may be operating on different protocols which are not compatible with each other. Network layer has the responsibility to route the packets from source to destination, mapping different addressing schemes and protocols.

NETWORK LAYER DESIGN ISSUES

In the following sections, we will give an introduction to some of the issues that the designers of the network layer must grapple with. These issues include the service provided to the transport layer and the internal design of the network.

Network Layer Functionalities:

Devices which work on Network Layer mainly focus on routing. Routing may include various tasks aimed to achieve a single goal. These can be:

- Addressing devices and networks.
- Populating routing tables or static routes.
- Queuing incoming and outgoing data and then forwarding them according to quality of service constraints set for those packets.
- Internetworking between two different subnets.
- Delivering packets to destination with best efforts.
- Provides connection oriented and connection less mechanism.

Network Layer Features

With its standard functionalities, Layer 3 can provide various features as:

- Quality of service management

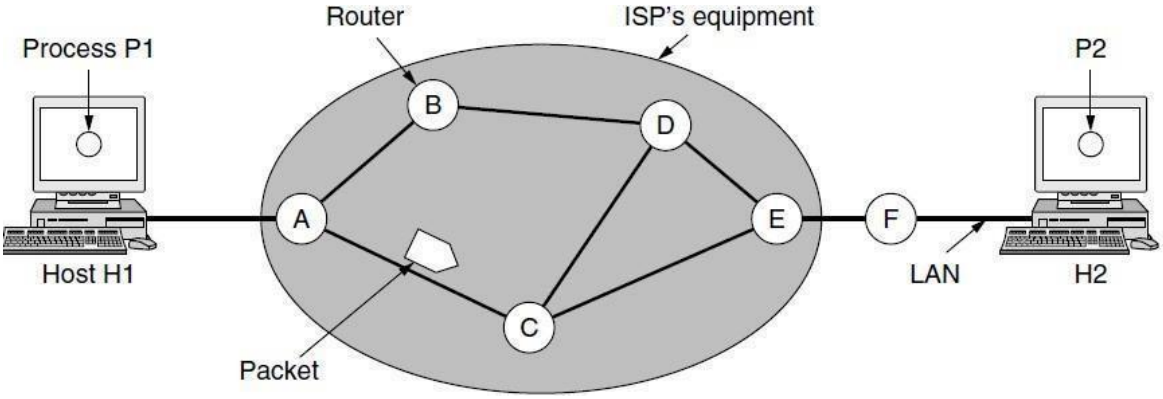
- Load balancing and link management
- Security
- Interrelation of different protocols and subnets with different schema.
- Different logical network design over the physical network design.
- L3 VPN and tunnels can be used to provide end to end dedicated connectivity.

NETWORK LAYER DESIGN ISSUES

In the following sections, we will give an introduction to some of the issues that the designers of the network layer must grapple with. These issues include the service provided to the transport layer and the internal design of the network.

Store-and-Forward Packet Switching

Before starting to explain the details of the network layer, it is worth restating the context in which the network layer protocols operate. This context can be seen in. The major components of the network are the ISP's equipment (routers connected by transmission lines), shown inside the shaded oval, and the customers' equipment, shown outside the oval. Host *H1* is directly connected to one of the ISP's routers, *A*, perhaps as a home computer that is plugged into a DSL modem. In contrast, *H2* is on a LAN, which might be an office Ethernet, with a router, *F*, owned and operated by the customer. This router has a leased line to the ISP's equipment. We have shown *F* as being outside the oval because it does not belong to the ISP. For the purposes of this chapter, however, routers on customer premises are considered part of the ISP network because they run the same algorithms as the ISP's routers (and our main concern here is algorithms).



The environment of the network layer protocols.

This equipment is used as follows. A host with a packet to send transmits it to the nearest router, either on its own LAN or over a point-to-point link to the ISP. The packet is stored there until it has fully arrived and the link has finished its processing by verifying the checksum. Then it is forwarded to the next router along the path until it reaches the destination host, where it is delivered. This mechanism is store-and-forward packet switching.

Services Provided to the Transport Layer: The network layer provides services to the transport layer at the network layer/transport layer interface. An important question is precisely what kind of services the network layer provides to the transport layer. The services need to be carefully designed with the following goals in mind:

1. The services should be independent of the router technology.
2. The transport layer should be shielded from the number, type, and topology of the routers present.
3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

Given these goals, the designers of the network layer have a lot of freedom in writing detailed specifications of the services to be offered to the transport layer.

This freedom often degenerates into a raging battle between two warring factions. The discussion centers on whether the network layer should provide connection-oriented service or connectionless service.

One camp (represented by the Internet community) argues that the routers' job is moving packets around and nothing else. In this view (based on 40 years of experience with a real computer network), the network is inherently unreliable, no matter how it is designed. Therefore, the hosts should accept this fact and do error control (i.e., error detection and correction) and flow control themselves. This viewpoint leads to the conclusion that the network service should be connectionless, with primitives SEND PACKET and RECEIVE PACKET and little else. In particular, no packet ordering and flow control should be done, because the hosts are going to do that anyway and there is usually little to be gained by doing it twice. This reasoning is an example of the **end-to-end argument**, a design principle that has been very influential in shaping the Internet (Saltzer et al., 1984). Furthermore, each packet must carry the full destination address, because each packet sent is carried independently of its predecessors, if any.

The **other camp** (represented by the telephone companies) argues that the network should provide a reliable, connection-oriented service. They claim that 100 years of successful experience with the worldwide telephone system is an excellent guide. In this view, quality of service is the dominant factor, and without connections in the network, quality of service is very difficult to achieve, especially for real-time traffic such as voice and video. Even after several decades, this controversy is still very much alive. Early, widely used data networks, such as X.25 in the 1970s and its successor Frame Relay in the 1980s, were connection-oriented. However, since the days of the ARPANET and the early Internet, connectionless network layers have grown tremendously in popularity. The IP protocol is now an ever-present symbol of success. It was undeterred by a connection-oriented technology called ATM that was developed to overthrow it in the 1980s; instead, it is ATM that is now found in niche uses and IP that is taking over telephone networks. Under the covers, however, the Internet is evolving connection-oriented features as quality of service becomes more important. Two examples of connection-oriented technologies are MPLS (Multi Protocol Label Switching and VLANs, which we saw in. Both technologies are widely used.

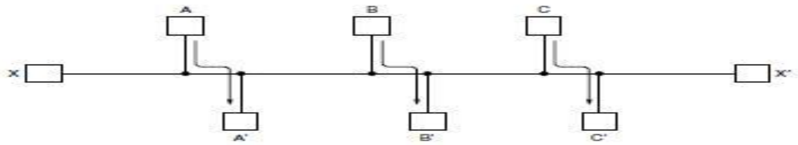
ROUTING ALGORITHMS

The main function of the network layer is routing packets from the source machine to the destination machine. In most networks, packets will require multiple hops to make the journey. The only notable exception is for broadcast networks, but even here routing is an issue if the source and destination are not on the same network segment. The algorithms that choose the routes and the data structures that they use are a major area of network layer design.

The **routing algorithm** is that part of the network layer software responsible for deciding which output line an incoming packet should be transmitted on. If the network uses datagrams internally, this decision must be made anew for every arriving data packet since the best route may have changed since last time. If the network uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, data packets just follow the already established route. The latter case is sometimes called **session routing** because
a route remains in force for an entire session (e.g., while logged in over a VPN). It is sometimes useful to make a distinction between routing, which is making the decision which routes to use, and forwarding, which is what happens when a packet arrives. One can think of a router as having two processes inside it. One of them handles each packet as it arrives, looking up the outgoing line to use for it in the routing tables. This process is **forwarding**. The other process is responsible for filling in and updating the routing tables. That is where the routing algorithm comes into play.

Regardless of whether routes are chosen independently for each packet sent or only when new connections are established, certain properties are desirable in a routing algorithm: correctness, simplicity, robustness, stability, fairness, and efficiency. Correctness and simplicity hardly require comment, but the need for robustness may be less obvious at first. Once a major network comes on the air, it may be expected to run continuously for years without system-wide failures. During that period there will be hardware and software failures of all kinds. Hosts, routers, and lines will fail repeatedly, and the topology will change many times. The routing algorithm should be able to cope with changes in the topology and traffic without requiring all jobs in all hosts to be aborted. Imagine the havoc if the network needed to be rebooted every time some router crashed! Stability is also an important goal for the routing algorithm. There exist routing algorithms that never converge to a fixed set of paths, no matter how long they run. A stable algorithm reaches equilibrium and stays there. It should converge quickly too, since communication may be disrupted until the routing algorithm has reached equilibrium. Fairness and efficiency may sound obvious—surely no reasonable person would oppose them—but as it turns out, they are often contradictory goals. As a simple example of this conflict, look at Fig. Suppose that there is enough traffic between *A* and *A′*, between *B* and *B′*, and between *C* and *C′* to saturate the horizontal links. To maximize the total flow, the *X* to *X′* traffic should be shut off altogether. Unfortunately, *X* and *X′* may not see it that way. Evidently, some compromise between global efficiency and fairness to individual connections is needed. Before we can even attempt to find trade-offs between fairness and efficiency, we must decide what it is we seek to optimize. Minimizing the mean packet delay is an obvious candidate to send traffic through the network effectively, but so is maximizing total network throughput. Furthermore, these two goals are also in conflict, since operating any queuing system near capacity implies a long queuing delay. As a compromise, many networks attempt to minimize the distance a packet must travel, or simply reduce the number of hops a packet must make. Either choice tends to improve the delay and also reduce the amount of bandwidth consumed per packet, which tends to improve the overall network throughput as well.

Routing algorithms can be grouped into two major classes: non adaptive and adaptive. **Non adaptive algorithms** do not base their routing decisions on any measurements or estimates of the current topology



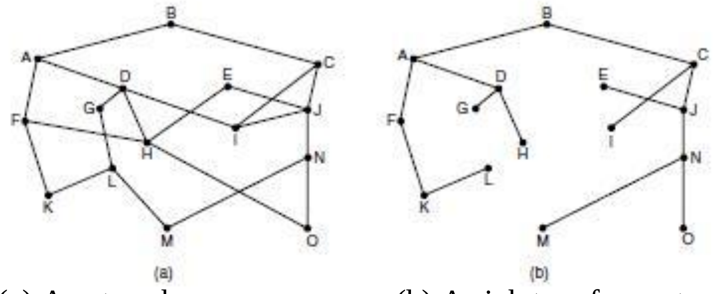
Network with a conflict between fairness and efficiency.

and traffic. Instead, the choice of the route to use to get from *I* to *J* (for all *I* and *J*) is computed in advance, offline, and downloaded to the routers when the network is booted. This procedure is sometimes called **static routing**. Because it does not respond to failures, static routing is mostly useful for situations in which the routing choice is clear. For example, router *F* in Fig. should send packets headed into the network to router *E* regardless of the ultimate destination. **Adaptive algorithms**, in contrast, change their routing decisions to reflect changes in the topology, and sometimes changes in the traffic as well.

These **dynamic routing** algorithms differ in where they get their information (e.g., locally, from adjacent routers, or from all routers), when they change the routes(e.g., when the topology changes or every T seconds as the load changes) and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time).In the following sections, we will discuss a variety of routing algorithms. The algorithms cover delivery models besides sending a packet from a source to a destination. Sometimes the goal is to send the packet to multiple, all, or one of a set of destinations. All of the routing algorithms we describe here make decisions based on the topology; we defer the possibility of decisions based on the traffic levels to Sec.

The Optimality Principle

Before we get into specific algorithms, it may be helpful to note that one can make a general statement about optimal routes without regard to network topology or traffic. This statement is known as the **optimality principle** (Bellman, 1957). It states that if router *J* is on the optimal path from router *I* to router *K*, then the optimal path from *J* to *K* also falls along the same route. To see this, call the part of the route from *I* to *J* *r*1 and the rest of the route *r* 2. If a route better than *r* 2 existed from *J* to *K*, it could be concatenated with *r* 1 to improve the route from *I* to *K*, contradicting our statement that *r* 1*r* 2 is optimal. As a direct consequence of the optimality principle, we can see that the set of optimal routes from all sources to a given destination form a tree rooted at the destination. Such a tree is called a **sink tree** and is illustrated in Fig. where the distance metric is the number of hops. The goal of all routing algorithms is to discover and use the sink trees for all routers.



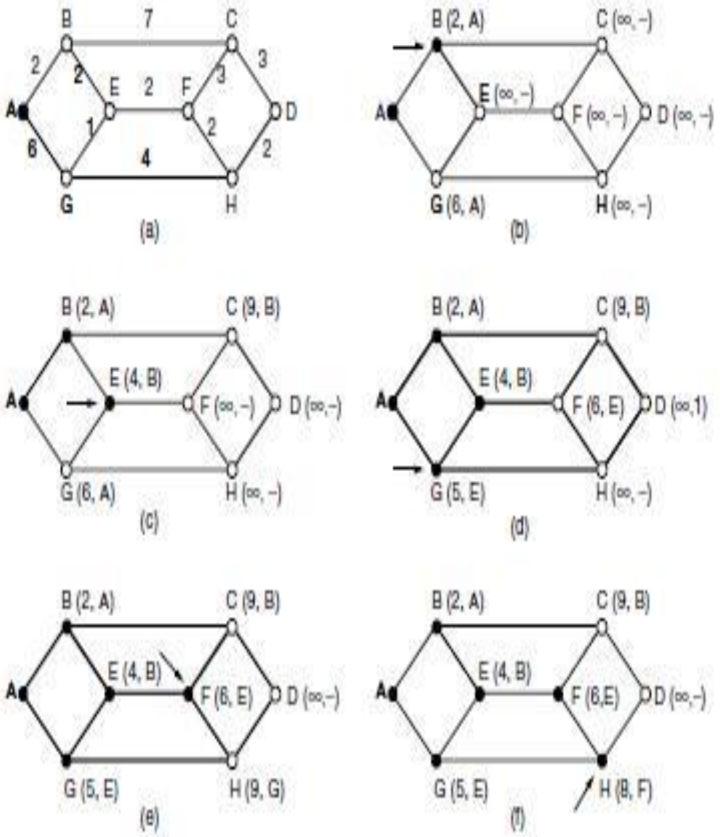
(a) A network. (b) A sink tree for router *B*.

Note that a sink tree is not necessarily unique; other trees with the same path lengths may exist. If we allow all of the possible paths to be chosen, the tree becomes a more general structure called a **DAG** (**Directed Acyclic Graph**). DAGs have no loops. We will use sink trees as convenient shorthand for both cases. Both cases also depend on the technical assumption that the paths do not interfere with each other so, for example, a traffic jam on one path will not cause another path to divert. Since a sink tree is indeed a tree, it does not contain any loops, so each packet will be delivered within a finite and bounded number of hops. In practice, life is not quite this easy. Links and routers can go down and come back up during operation, so different routers may have different ideas about the current topology. Also, we have quietly finessed the issue of whether each router has to individually acquire the information on which to base its sink tree computation or whether this information is collected by some other means. We will come back to these issues shortly. Nevertheless, the optimality principle and the sink tree provide a benchmark against which other routing algorithms can be measured

Shortest Path Algorithm

Let us begin our study of routing algorithms with a simple technique for computing optimal paths given a complete picture of the network. These paths are the ones that we want a distributed routing algorithm to find, even though not all routers may know all of the details of the network. The idea is to build a graph of the network, with each node of the graph representing a router and each edge of the graph representing a communication line, or link. To choose a route between a given pair of routers, the algorithm just finds the shortest path between them on the graph.

The concept of a **shortest path** deserves some explanation. One way of measuring path length is the number of hops. Using this metric, the paths *ABC* and *ABE* in Fig. are equally long. Another metric is the geographic distance in kilometers, in which case *ABC* is clearly much longer than *ABE* (assuming the figure is drawn to scale).



The first six steps used in computing the shortest path from *A* to *D*. The arrows indicate the working node.

However, many other metrics besides hops and physical distance are also possible. For example, each edge could be labeled with the mean delay of a standard test packet, as measured by hourly runs. With this graph labeling, the shortest path is the fastest path rather than the path with the fewest edges or kilometers. In the general case, the labels on the edges could be computed as a function of the distance, bandwidth, average traffic, communication cost, measured delay, and other factors. By changing the weighting function, the algorithm would then compute the ‘‘shortest’’ path measured according to any one of a number of criteria or to a combination of criteria. Several algorithms for computing the shortest path between two nodes of a graph are known. This one is due to Dijkstra (1959) and finds the shortest paths between a source and all destinations in the network. Each node is labeled (in parentheses) with its distance from the source node along the best known path. The distances must be non-negative, as they will be if they are based on real quantities like bandwidth and delay. Initially, no paths are known, so all nodes are labeled with infinity. As the algorithm proceeds and paths are found, the labels may change, reflecting better paths. A label may be either tentative or permanent. Initially, all labels are tentative. When it is discovered that a label represents the shortest possible path from the source to that node, it is made permanent and never changed thereafter. To illustrate how the labeling algorithm works, look at the weighted, undirected graph of Fig., where the weights represent, for example, distance. We want to find the shortest path from *A* to *D*. We start out by marking node *A* as permanent, indicated by a filled-in circle. Then we examine, in turn, each of the nodes adjacent to *A* (the working node), relabeling each one with the distance to *A*. Whenever a node is relabeled, we also label it with the node from which the probe was made so that we can reconstruct the final path later. If the network had more than one shortest path from *A* to *D* and we wanted to find all of them, we would need to remember all of the probe nodes that could reach a node with the same distance. Having examined each of the nodes adjacent to *A*, we examine all the tentatively labeled nodes in the whole graph and make the one with the smallest label permanent, as shown in Fig. (b). this one becomes the new working node. We now start at *B* and examine all nodes adjacent to it. If the sum of the label on *B* and the distance from *B* to the node being considered is less than the label on that node, we have a shorter path, so the node is relabeled. After all the nodes adjacent to the working node have been inspected and the tentative labels changed if possible, the entire graph is searched for the tentatively labeled node with the smallest value. This node is made permanent and becomes the working node for the next round. Figure shows the first six steps of the algorithm. To see why the algorithm works, look at Fig. (c). At this point we have just made *E* permanent. Suppose that there were a shorter path than *ABE*, say *AXYZE* (for some *X* and *Y*). There are two possibilities: either node *Z* has already been made permanent, or it has not been. If it has, then *E* has already been probed (on the round following the one when *Z* was made permanent), so the *AXYZE* path has not escaped our attention and thus cannot be a shorter path. Now consider the case where *Z* is still tentatively labeled. If the label at *Z* is greater than or equal to that at *E*, then *AXYZE* cannot be a shorter path than *ABE*. If the label is less than that of *E*, then *Z* and not *E* will become permanent first, allowing *E* to be probed from *Z*. This algorithm is given in Fig. The global variables *n* and *dist* describe the graph and are initialized before *shortest path* is called. The only difference between the program and the algorithm described above is that in Fig., we compute the shortest path starting at the terminal node, *t*, rather than at the source node, *s*. Since the shortest paths from *t* to *s* in an undirected graph are the same as the shortest paths from *s* to *t*, it does not matter at which end we begin. The reason for searching backward is that each node is labeled with its predecessor rather than its successor. When the final path is copied into the output variable, *path*, the path is thus reversed. The two reversal effects cancel, and the answer is produced in the correct order.

Flooding

When a routing algorithm is implemented, each router must make decisions based on local knowledge, not the complete picture of the network. A simple local technique is **flooding**, in which every incoming packet is sent out on every outgoing line except the one it arrived on. Flooding obviously generates vast numbers of duplicate packets, in fact, an infinite number unless some measures are taken to damp the process. One such measure is to have a hop counter contained in the header of each packet that is decremented at each hop, with the packet being discarded when the counter reaches zero. Ideally, the hop counter should be initialized to the length of the path from source to destination. If the sender does not know how long the path is, it can initialize the counter to the worst case, namely, the full diameter of the network.

Flooding with a hop count can produce an exponential number of duplicate packets as the hop count grows and routers duplicate packets they have seen before. A better technique for damming the flood is to have routers keep track of which packets have been flooded, to avoid sending them out a second time. One way to achieve this goal is to have the source router put a sequence number in each packet it receives from its hosts. Each router then needs a list per source router telling which sequence numbers originating at that source have already been seen. If an incoming packet is on the list, it is not flooded.

```
#define MAX NODES 1024 /* maximum number of nodes */
#define INFINITY 1000000000 /* a number larger than every maximum path */
int n, dist[MAX NODES][MAX NODES]; /* dist[i][j] is the distance from i to j
*/ void shortest path(int s, int t, int path[])
{ struct state { /* the path being worked on */
int predecessor; /* previous node */
int length; /* length from source to this node */
enum {permanent, tentative} label; /* label state */
} state[MAX NODES];
int i, k, min;
struct state *p;
for (p = &state[0]; p < &state[n]; p++) { /* initialize state */
p->predecessor = −1;
p->length = INFINITY;
p->label = tentative;
}
state[t].length = 0; state[t].label = permanent;
k = t; /* k is the initial working node */
do { /* Is there a better path from k? */
for (i = 0; i < n; i++) /* this graph has n nodes */
if (dist[k][i] != 0 && state[i].label == tentative) {
```

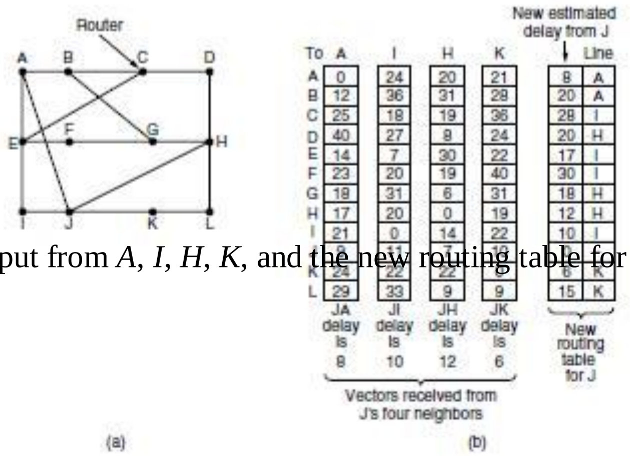
```
if (state[k].length + dist[k][i] < state[i].length) {
state[i].predecessor = k;
state[i].length = state[k].length + dist[k][i];
}
}
/* Find the tentatively labeled node with the smallest label. */
k = 0; min = INFINITY;
for (i = 0; i < n; i++)
if (state[i].label == tentative && state[i].length < min) {
min = state[i].length;
k = i;
}
state[k].label = permanent;
} while (k != s);
/* Copy the path into the output array. */
i = 0; k = s;
do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

Dijkstra's algorithm to compute the shortest path through a graph

To prevent the list from growing without bound, each list should be augmented by a counter, *k*, meaning that all sequence numbers through *k* have been seen. When a packet comes in, it is easy to check if the packet has already been flooded (by comparing its sequence number to *k*; if so, it is discarded. Furthermore, the full list below *k* is not needed, since *k* effectively summarizes it. Flooding is not practical for sending most packets, but it does have some important uses. First, it ensures that a packet is delivered to every node in the network. This may be wasteful if there is a single destination that needs the packet, but it is effective for broadcasting information. In wireless networks, all messages transmitted by a station can be received by all other stations within its radio range, which is, in fact, flooding, and some algorithms utilize this property. Second, flooding is tremendously robust. Even if large numbers of routers are blown to bits (e.g., in a military network located in a war zone), flooding will find a path if one exists, to get a packet to its destination. Flooding also requires little in the way of setup. The routers only need to know their neighbors. This means that flooding can be used as a building block for other routing algorithms that are
more efficient but need more in the way of setup. Flooding can also be used as a metric against which other routing algorithms can be compared. Flooding always chooses the shortest path because it chooses every possible path in parallel. Consequently, no other algorithm can produce a shorter delay (if we ignore the overhead generated by the flooding process itself).

Distance Vector Routing:

Computer networks generally use dynamic routing algorithms that are more complex than flooding, but more efficient because they find shortest paths for the current topology. Two dynamic algorithms in particular, distance vector routing and link state routing, are the most popular. In this section, we will look at the former algorithm. In the following section, we will study the latter algorithm. A **distance vector routing** algorithm operates by having each router maintain a table (i.e., a vector) giving the best known distance to each destination and which link to use to get there. These tables are updated by exchanging information with the neighbors. Eventually, every router knows the best link to reach each destination. The distance vector routing algorithm is sometimes called by other names, most commonly the distributed **Bellman-Ford** routing algorithm, after the researchers who developed it (Bellman, 1957; and Ford and Fulkerson, 1962). It was the original ARPANET routing algorithm and was also used in the Internet under the name RIP. In distance vector routing, each router maintains a routing table indexed by, and containing one entry for each router in the network. This entry has two parts: the preferred outgoing line to use for that destination and an estimate of the distance to that destination. The distance might be measured as the number of hops or using another metric, as we discussed for computing shortest paths. The router is assumed to know the ‘‘distance’’ to each of its neighbors. If the metric is hops, the distance is just one hop. If the metric is propagation delay, the router can measure it directly with special ECHO packets that the receiver just timestamps and sends back as fast as it can. As an example, assume that delay is used as a metric and that the router knows the delay to each of its neighbors. Once every *T* m sec, each router sends to each neighbor a list of its estimated delays to each destination. It also receives a similar list from each neighbor. Imagine that one of these tables has just come in from neighbor *X*, with *Xi* being *X*'s estimate of how long it takes to get to router *i*.If the router knows that the delay to *X* is *m* m sec, it also knows that it can reach router *i* via *X* in *Xi m* msec.□By performing this calculation for each neighbor, a router can find out which estimate seems the best and use that estimate and the corresponding link in its new routing table. Note that the old routing table is not used in the calculation. This updating process is illustrated in Fig. 5-9. Part (a) shows a network. The first four columns of part (b) show the delay vectors received from the neighbors of router *J*. *A* claims to have a 12-msec delay to *B*, a 25-msec delay to *C*, a 40-msec delay to *D*, etc. Suppose that *J* has measured or estimated its delay to its neighbors, *A*, *I*, *H*, and *K*, as 8, 10, 12, and 6 m sec, respectively.



To	A	I	H	K		Line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	[illegible]
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA delay is 8	JI delay is 10	JH delay is 12	JK delay is 6	New routing table for J	

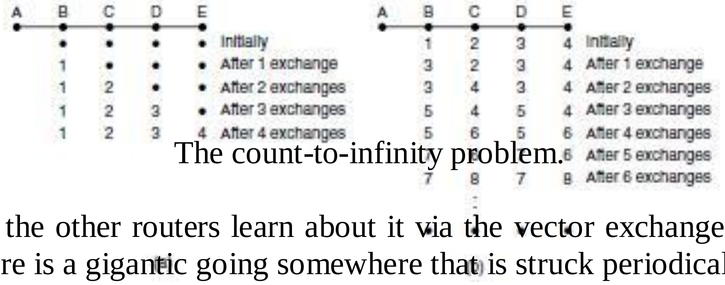
Vectors received from J's four neighbors

(a) A network. (b) Input from *A*, *I*, *H*, *K*, and the new routing table for *J*.

Consider how *J* computes its new route to router *G*. It knows that it can get to *A* in 8 m sec, and furthermore *A* claims to be able to get to *G* in 18 m sec, so *J* knows it can count on a delay of 26 m sec to *G* if it forwards packets bound for *G* to *A*. Similarly, it computes the delay to *G* via *I*, *H*, and *K* as 41 (31 + 10), 18 (6 + 12), and 37 (31 + 6) m sec, respectively. The best of these values is 18, so it makes an entry in its routing table that the delay to *G* is 18 m sec and that the route to use is via *H*. The same calculation is performed for all the other destinations, with the new routing table shown in the last column of the figure.

The Count-to-Infinity Problem

The settling of routes to best paths across the network is called **convergence**. Distance vector routing is useful as a simple technique by which routers can collectively compute shortest paths, but it has a serious drawback in practice: although it converges to the correct answer, it may do so slowly. In particular, it reacts rapidly to good news, but leisurely to bad news. Consider a router whose best route to destination *X* is long. If, on the next exchange, neighbor *A* suddenly reports a short delay to *X*, the router just switches over to using the line to *A* to send traffic to *X*. In one vector exchange, the good news is processed. To see how fast good news propagates, consider the five-node (linear) network of Fig. 5-10, where the delay metric is the number of hops. Suppose *A* is down initially and all the other routers know this. In other words, they have all recorded the delay to *A* as infinity.



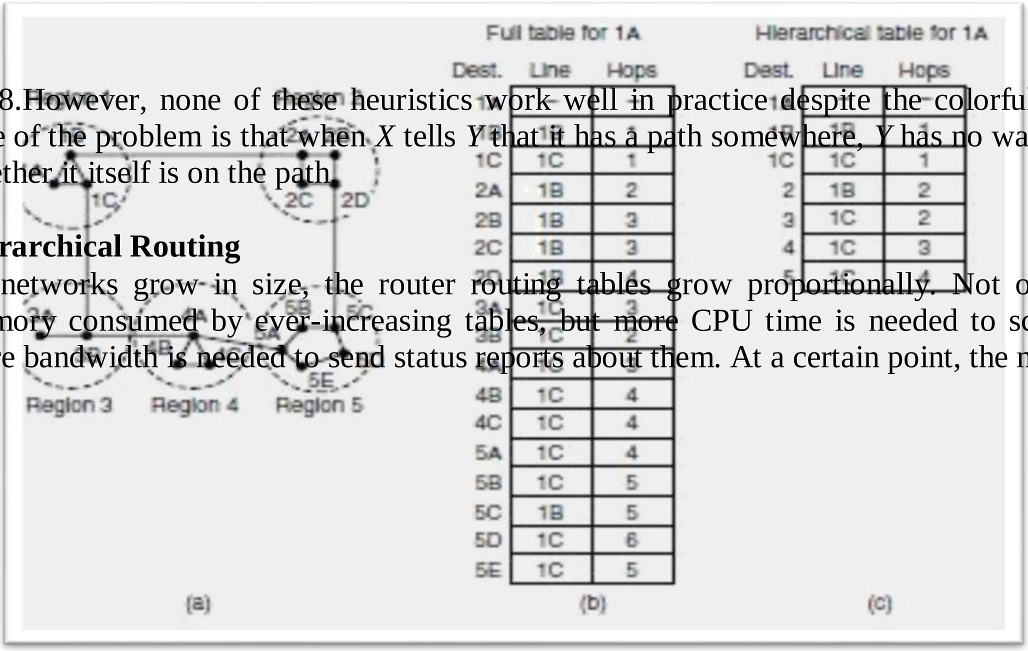
B	C	D	E	
•	•	•	•	Initially
1	•	•	•	After 1 exchange
1	2	•	•	After 2 exchanges
1	2	3	•	After 3 exchanges
1	2	3	4	After 4 exchanges

B	C	D	E	
1	2	3	4	Initially
3	2	3	4	After 1 exchange
3	4	3	4	After 2 exchanges
5	4	5	4	After 3 exchanges
5	6	5	6	After 4 exchanges
7	6	[illegible]	6	After 5 exchanges
7	8	7	8	After 6 exchanges
	⋮			
•	•	•	•	

The count-to-infinity problem.

When *A* comes up, the other routers learn about it via the vector exchanges. For simplicity, we will assume that there is a gigantic going somewhere that is struck periodically to initiate a vector exchange at all routers simultaneously. At the time of the first exchange, *B* learns that its left-hand neighbor has zero delay to *A*. *B* now makes an entry in its routing table indicating that *A* is one hop away to the left. All the other routers still think that *A* is down. At this point, the routing table entries for *A* are as shown in the second row of Fig. (a). On the next exchange, *C* learns that *B* has a path of length 1 to *A*, so it updates its routing table to indicate a path of length 2, but *D* and *E* do not hear the good news until later. Clearly, the good news is spreading at the rate of one hop per exchange. In a network whose longest path is of length *N* hops, within *N* exchanges everyone will know about newly revived links and routers. Now let us consider the situation of (b), in which all the links and routers are initially up. Routers *B*, *C*, *D*, and *E* have distances to *A* of 1, 2, 3, and 4 hops, respectively. Suddenly, either *A* goes down or the link between *A* and *B* is cut (which is effectively the same thing from *B*'s point of view).At the first packet exchange, *B* does not hear anything from *A*. Fortunately, *C* says ''Do not worry; I have a path to *A* of length 2.'' Little does *B* suspect that *C*'s path runs through *B* itself . For all *B* knows, *C* might have ten links all with separate paths to *A* of length 2. As a result, *B* thinks it can reach *A* via *C*, with a path length of 3. *D* and *E* do not update their entries for *A* on the first exchange.

On the second exchange, *C* notices that each of its neighbors claims to have a path to *A* of length 3. It picks one of them at random and makes its new distance to *A* 4, as shown in the third row of Fig. 5-10(b). Subsequent exchanges produce the history shown in the rest of Fig. 5-10(b).From this figure, it should be clear why bad news travels Slowly: no router ever has a value more than one higher than the minimum of all its neighbors. Gradually, all routers work their way up to infinity, but the number of exchanges required depends on the numerical value used for infinity. For this reason, it is wise to set infinity to the longest path plus 1.Not entirely surprisingly, this problem is known as the **count-to-infinity** problem. There have been many attempts to solve it, for example, preventing routers from advertising their best paths back to the neighbors from which they heard them with the split horizon with poisoned reverse rule discussed in RFC



Full table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

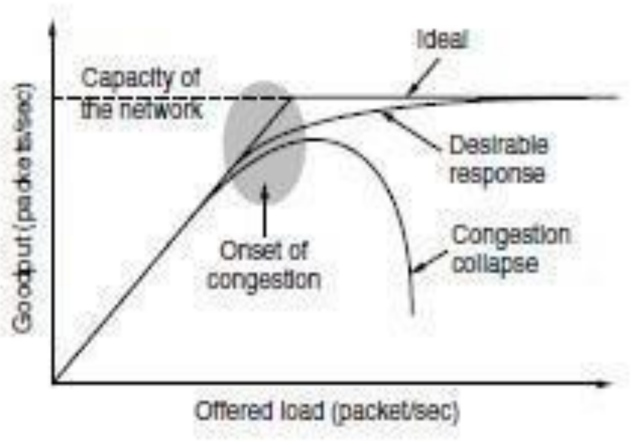
1058.However, none of these heuristics work well in practice despite the colorful names. The core of the problem is that when *X* tells *Y* that it has a path somewhere, *Y* has no way of knowing whether it itself is on the path.

Hierarchical Routing

As networks grow in size, the router routing tables grow proportionally. Not only is router memory consumed by ever-increasing tables, but more CPU time is needed to scan them and more bandwidth is needed to send status reports about them. At a certain point, the network may grow to the point where it is no longer feasible for every router to have an entry for every other router, so the routing will have to be done hierarchically, as it is in the telephone network. When hierarchical routing is used, the routers are divided into what we will call **regions**. Each router knows all the details about how to route packets to destinations within its own region but knows nothing about the internal structure of other regions. When different networks are interconnected, it is natural to regard each one as a separate region to free the routers in one network from having to know the topological structure of the other ones. For huge networks, a two-level hierarchy may be insufficient; it may be necessary to group the regions into clusters, the clusters into zones, the zones into groups, and so on, until we run out of names for aggregations. As an example of a multilevel hierarchy, consider how a packet might be routed from Berkeley, California, to Malindi, Kenya. The Berkeley router would know the detailed topology within California but would send all out-of-state traffic to the Los Angeles router. The Los Angeles router would be able to route traffic directly to other domestic routers but would send all foreign traffic to New York. The New York router would be programmed to direct all traffic to the router in the destination country responsible for handling foreign traffic, say, in Nairobi. Finally, the packet would work its way down the tree in Kenya until it got to Malindi. Figure gives a quantitative example of routing in a two-level hierarchy with five regions. The full routing table for router *1A* has 17 entries, as shown in Fig. (b). When routing is done hierarchically, as in Fig. 5-14(c), there are entries for all the local routers, as before, but all other regions are condensed into a single router, so all traffic for region 2 goes via the *1B*-*2A* line, but the rest of the remote traffic goes via the *1C-3B* line. Hierarchical routing has reduced the table from 17 to 7 entries. As the ratio of the number of regions to the number of routers per region grows, the savings in table space increase. Unfortunately, these gains in space are not free. There is a penalty to be paid: increased path length. For example, the best route from *1A* to *5C* is via region 2,but with hierarchical routing all traffic to region 5 goes via region 3, because that is better for most destinations in region 5.When a single network becomes very large, an interesting question is ‘‘how many levels should the hierarchy have?’’ For example, consider a network with 720 routers. If there is no hierarchy, each router needs 720 routing table entries. If the network is partitioned into 24 regions of 30 routers each, each router needs 30 local entries plus 23 remote entries for a total of 53 entries. If a three-level hierarchy is chosen, with 8 clusters each containing 9 regions of 10 routers, each router needs 10 entries for local routers, 8 entries for routing to other regions within its own cluster, and 7 entries for distant clusters, for a total of 25 entries.Kamoun and Kleinrock (1979) discovered that the optimal number of levels for an *N* router network is ln *N*, requiring a total of *e* ln *N* entries per router. They have also shown that the increase in effective mean path length caused by hierarchical routing is sufficiently small that it is usually acceptable.

CONGESTION CONTROL ALGORITHMS

Too many packets present in (a part of) the network causes packet delay and loss that degrades performance. This situation is called **congestion**. The network and transport layers share the responsibility for handling congestion. Since congestion occurs within the network, it is the network layer that directly experiences it and must ultimately determine what to do with the excess packets. However, the most effective way to control congestion is to reduce the load that the transport layer is placing on the network. This requires the network and transport layers to work together. In this chapter we will look at the network aspects of congestion. In Chap. 6, we will complete the topic by covering the transport aspects of congestion. Figure depicts the onset of congestion. When the number of packets hosts send into the network is well within its carrying capacity, the number delivered is proportional to the number sent. If twice as many are sent, twice as many are delivered. However, as the offered load approaches the carrying capacity, bursts of traffic occasionally fill up the buffers inside routers and some packets are lost. These lost packets consume some of the capacity, so the number of delivered packets falls below the ideal curve. The network is now congested.



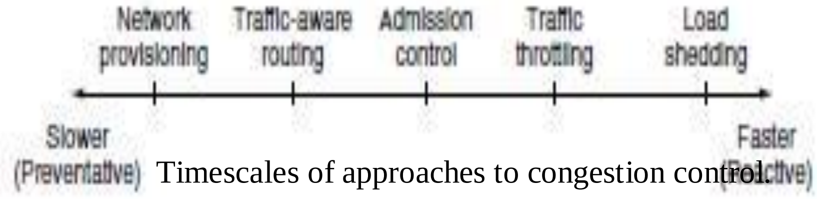
With too much traffic, performance drops sharply.

Unless the network is well designed, it may experience a **congestion collapse**, in which performance plummets as the offered load increases beyond the capacity. This can happenbecause packets can be sufficiently delayed inside the network that they are no longer useful when they leave the network. For example, in the early Internet, the time a packet spent waiting for a backlog of packets ahead of it to be sent over a slow 56-kbps link could reach the maximum time it was allowed to remain in the network. It then had to be thrown away. A different failure mode occurs when senders retransmit packets that are greatly delayed, thinking that they have been lost. In this case, copies of the same packet will be delivered by the network, again wasting its capacity. To capture these factors, the y-axis of Fig. is given as **good put**, which is the rate at which *useful* packets are delivered by the network. We would like to design networks that avoid congestion where possible and do not suffer from congestion collapse if they do become congested. Unfortunately, congestion cannot wholly be avoided. If all of a sudden, streams of packets begin arriving on three or four input lines and all need the same output line, a queue will build up. If there is insufficient memory to hold all of them, packets will be lost. Adding more memory may help up to a point, but Nagle (1987) realized that if routers have an infinite amount of memory, congestion gets worse, not better. This is because by the time packets get to the front of the queue, they have already timed out (repeatedly) and duplicates have been sent. This makes matters worse, not better—it leads to congestion collapse. Low-bandwidth links or routers that process packets more slowly than the line rate can also become congested. In this case, the situation can be improved by directing some of the traffic away from the bottleneck to other parts of the network. Eventually, however, all regions of the network will be congested. In this situation, there is no alternative but to shed load or build a faster network. It is worth pointing out the difference between congestion control and flow control, as the relationship is a very subtle one. Congestion control has to do with making sure the network is able to carry the offered traffic. It is a global issue, involving the behavior of all the hosts and routers. Flow control, in contrast, relates to the traffic between a particular sender and a particular receiver. Its job is to make sure that a fast sender cannot continually transmit data faster than the receiver is able to absorb it. To see the difference between these two concepts, consider a network made up of 100-Gbps fiber optic links on which a supercomputer is trying to force feed a large file to a personal computer that is capable of handling only 1 Gbps. Although there is no congestion (the network itself is not in trouble), flow control is needed to force the supercomputer to stop frequently to give the personal computer chance to breathe. At the other extreme, consider a network with 1-Mbps lines and 1000 large computers, half of which are trying to transfer files at 100 kbps to the other half. Here, the problem is not that of fast senders overpowering slow receivers, but that the total offered traffic exceeds what the network can handle.

The reason congestion control and flow control are often confused is that the best way to handle both problems is to get the host to slow down. Thus, a host can get a ‘‘slow down’’ message either because the receiver cannot handle the load or because the network cannot handle it. We will come back to this point in Chap. 6. We will start our study of congestion control by looking at the approaches that can be used at different time scales. Then we will look at approaches to preventing congestion from occurring in the first place, followed by approaches for coping with it once it has set in.

Approaches to Congestion Control

The presence of congestion means that the load is (temporarily) greater than the resources (in a part of the network) can handle. Two solutions come to mind: increase the resources or decrease the load. As shown in Fig., these solutions are usually applied on different time scales to either prevent congestion or react to it once it has occurred.



Timescales of approaches to congestion control.

The most basic way to avoid congestion is to build a network that is well matched to the traffic that it carries. If there is a low-bandwidth link on the path along which most traffic is directed, congestion is likely. Sometimes resources on spare routers or enabling lines that are normally used only as backups (to make the system fault tolerant) or purchasing bandwidth on the open market. More often, links and routers that are regularly heavily utilized are upgraded at the earliest opportunity. This is called **provisioning** and happens on a time scale of months, driven by long-term traffic trends. To make the most of the existing network capacity, routes can be tailored to traffic patterns that change during the day as network user's wake and sleep in different time zones. For example, routes may be changed to shift traffic away from heavily used paths by changing the shortest path weights. Some local radio stations have helicopters flying around their cities to report on road congestion to make it possible for their mobile listeners to route their packets (cars) around hotspots. This is called **traffic-aware routing**. Splitting traffic across multiple paths is also helpful. However, sometimes it is not possible to increase capacity. The only way then to beat back the congestion is to decrease the load. In a virtual-circuit network, new connections can be refused if they would cause the network to become congested. This is called **admission control**. At a finer granularity, when congestion is imminent the network can deliver feedback to the sources whose traffic flows are responsible for the problem. The network can request these sources to throttle their traffic, or it can slow down the traffic itself. Two difficulties with this approach are how to identify the onset of congestion, and how to inform the source that needs to slow down. To tackle the first issue, routers can monitor the average load, queuing delay, or packet loss. In all cases, rising numbers indicate growing congestion. To tackle the second issue, routers must participate in a feedback loop with the sources. For a scheme to work correctly, the time scale must be adjusted carefully. If every time two packets arrive in a row, a router yells STOP and every time a router is idle for 20 sec, it yells GO, the system will oscillate wildly and never converge. On the other hand, if it waits 30 minutes to make sure before saying anything, the congestion-control mechanism will react too sluggishly to be of any use. Delivering timely feedback is a nontrivial matter. An added concern is having routers send more messages when the network is already congested.

Finally, when all else fails, the network is forced to discard packets that it cannot deliver. The general name for this is **load shedding**. A good policy for choosing which packets to discard can help to prevent congestion collapse.

Traffic-Aware Routing

The first approach we will examine is traffic-aware routing. The routing schemes we looked at in Sec used fixed link weights. These schemes adapted to changes in topology, but not to changes in load. The goal in taking load into account when computing routes is to shift traffic away fromhotspots that will be the first places in the network to experience congestion. The most direct way to do this is to set the link weight to be a function of the (fixed) link bandwidth and propagation delay plus the (variable) measured load or average queuing delay. Least-weight paths will then favor paths that are more lightly loaded, all else being equal. Traffic-aware routing was used in the early Internet according to this model (Khanna and Zinky, 1989). However, there is a peril. Consider the network of Fig., which is divided into two parts, East and West, connected by two links, *CF* and *EI*. Suppose that most of the traffic between East and West is using link *CF*, and, as a result, this link is heavily loaded with long delays. Including queuing delay in the weight used for the shortest path calculation will make *EI* more attractive. After the new routing tables have been installed, most of the East-West traffic will now go over *EI*, loading this link. Consequently, in the next update, *CF* will appear to be the shortest path. As a result, the routing tables may oscillate wildly, leading to erratic routing and many potential problems.

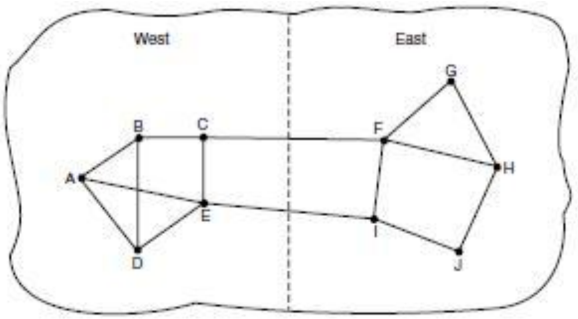
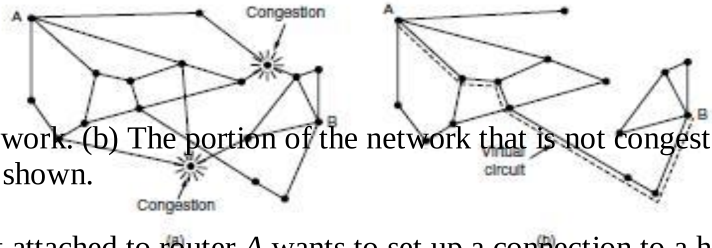


Figure : A network in which the East and West parts are connected by two links.

If load is ignored and only bandwidth and propagation delay are considered, this problem does not occur. Attempts to include load but change weights within a narrow range only slow down routing oscillations. Two techniques can contribute to a successful solution. The first is multipath routing, in which there can be multiple paths from a source to a destination. In our example this means that the traffic can be spread across both of the East to West links. The second one is for the routing scheme to shift traffic across routes slowly enough that it is able to converge, as in the scheme of Gallagher (1977).Given these difficulties, in the Internet routing protocols do not generally adjust their routes depending on the load. Instead, adjustments are made outside the routing protocol by slowly changing its inputs. This is called **traffic engineering**.

Admission Control:

One technique that is widely used in virtual-circuit networks to keep congestion at bay is **admission control**. The idea is simple: do not set up a new virtual circuit unless the network can carry the added traffic without becoming congested. Thus, attempts to set up a virtual circuit may fail. This is better than the alternative, as letting more people in when the network is busy just makes matters worse. By analogy, in the telephone system, when a switch gets overloaded it practices admission control by not giving dial tones. The trick with this approach is working out when a new virtual circuit will lead to congestion. The task is straightforward in the telephone network because of the fixed bandwidth of calls (64 kbps for uncompressed audio). However, virtual circuits in computer networks come in all shapes and sizes. Thus, the circuit must come with some characterization of its traffic if we are to apply admission control. Traffic is often described in terms of its rate and shape. The problem of how to describe it in a simple yet meaningful way is difficult because traffic is typically bursty—the average rate is only half the story. For example, traffic that varies while browsing the Web is more difficult to handle than a streaming movie with the same long-term throughput because the bursts of Web traffic are more likely to congest routers in the network. A commonly used descriptor that captures this effect is the **leaky bucket** or **token bucket**. A leaky bucket has two parameters that bound the average rate and the instantaneous burst size of traffic. Since leaky buckets are widely used for quality of service, we will go over them in detail in Sec. Armed with traffic descriptions, the network can decide whether to admit the new virtual circuit. One possibility is for the network to reserve enough capacity along the paths of each of its virtual circuits that congestion will not occur. In this case, the traffic description is a service agreement for what the network will guarantee its users. We have prevented congestion but veered into the related topic of quality of service a little too early; we will return to it in the next section. Even without making guarantees, the network can use traffic descriptions for admission control. The task is then to estimate how many circuits will fit within the carrying capacity of the network without congestion. Suppose that virtual circuits that may blast traffic at rates up to 10 Mbps all pass through the same 100-Mbps physical link. How many circuits should be admitted? Clearly, 10 circuits can be admitted without risking congestion, but this is wasteful in the normal case since it may rarely happen that all 10 are transmitting full blast at the same time. In real networks, measurements of past behavior that capture the statistics of transmissions can be used to estimate the number of circuits to admit, to trade better performance for acceptable risk. Admission control can also be combined with traffic-aware routing by considering routes around traffic hotspots as part of the setup procedure. For example, consider the network illustrated in Fig (a), in which two routers are congested, as indicated.



(a) congested network. (b) The portion of the network that is not congested. A virtual circuit from *A* to *B* is also shown.

Suppose that a host attached to router *A* wants to set up a connection to a host attached to router Normally, this connection would pass through one of the congested routers. To avoid this situation, we can redraw the network as shown in Fig. 5-24(b), omitting the congested routers and all of their lines. The dashed line shows a possible route for the virtual circuit that avoids the congested routers. Shaikh et al. (1999) give a design for this kind of load-sensitive routing.

Quality of Service:

The notion of quality of service, or QoS, concerns certain characteristics of a network connection under the sole of the network service provider liability.

A QoS value applies to the whole of a network connection. It must be identical at both ends of the connection, even if it is supported by several interconnected subnetworks each offering different services.

QoS is described by parameters. Defining a QoS parameter indicates how to measure or determine its value, mentioning if necessary the events specified by the network service primitives.

Two types of QoS parameters have been defined:

- Those whose values are transmitted peer users via the Network service during the establishment phase of the network connection. During this transmission, a tripartite negotiation can take place between users and the network service provider to define a value for the QoS parameters.
- Those whose values are transmitted or negotiated between users and network service provider. For these QoS parameters, it is possible to obtain, by local means, information on the value to the supplier and values to each user of the network service.

The main QoS parameters are:

- **Time of establishment of the network connection**. Is the time that elapses between a network connection request and confirmation of the connection? This QoS parameter indicates the maximum time acceptable to the user.
- **Probability of failure of the establishment of the network connection**. This probability is established from the applications which have not been met in the normal time limit for establishing the connection.
- **Flow data transfer**. The flow rate defines the number of bytes transported over a network connection in a reasonably long time (a few minutes, a few hours or days). The difficulty in determining the speed of a connection network comes from the asynchronous transport packets. To obtain a value acceptable, observe the network on a sequence of several packages and consider number of bytes of data transported taking into account the elapsed time since the application or the data transfer indication.
- **Transit time when transferring data**. The transit time corresponds to elapsed time between a data transfer request and indicating transfer of data. This transit time is difficult to calculate because of the geographical distribution ends. The satisfaction of a quality service on the transit time may moreover contradict flow control.
- **Residual error rate**. Is calculated from the number of packets that arrive erroneous, lost or duplicated on the total number of transmitted packets. It is a rate Error packet. Also denotes the probability that a packet does not arrive correctly to the receiver.
- **Transfer Probability incident**. Is obtained by the ratio of the number of incidents listed on the total number of transfer taken. To have a correct estimate of this probability, just consider the number of network disconnection relative to the number of transfer taken.

• **Probability of failure of the network connection**. Is calculated from the number of release and resetting of a network connection based on the number of transfer made.

• **Release time the network connection**. This is the maximum acceptable delay between a disconnection request and the actual release.

• Probability of failure upon release of the network connection. The number Liberation of failure required by the total number requested release.

The following three additional parameters used to characterize the quality of Service:

• **Protection of the network connection**. Determines the probability that the network connection be in working order throughout the period when it is opened by the user. There is ways to protect a connection by duplicating or having a Backup connection ready to be opened in case of failure. The value for a telephone network is 99.999%, the so-called five nines, equivalent to a few minutes of downtime per year. The protection is much lower for an IP network, with a value of the order of 99.9%, three or nine. This value arises besides problem for IP telephony, which requires stronger protection telephone connections.

• **Priority of the network connection**. Determines priority of access to a connection network, the holding priority of a network connection and priority of data connection.

• **Maximum acceptable cost**. Determines if the network connection is tolerable or not. The definition of the cost is quite complex since it depends on the use of resources for the establishment, maintenance and release of the connection network.

Flow Characteristics

Traditionally, four types of characteristics are attributed to a flow: reliability, delay, jitter and bandwidth.

Reliability

- Reliability is an important characteristic of flow.
- Lack of reliability means losing a packet or acknowledgement which then requires retransmission.
- However, the sensitivity of application programs to reliability is not the same. For example, it is more important that electronic mail, file transfer, and internet access have reliable transmissions than audio conferencing or telephony.

Delay

- Source to destination delay is another flow characteristic.
- Applications can tolerate delay in different degrees.
- In this case, telephony, audio conferencing, video conferencing and remote log in need minimum delay while delay in file transfer or e-mail is less important.

Jitter

- Jitter is defined as the variation in delay for packets belonging to the same flow.

- High Jitter means the difference between delays is large and low jitter means the variation is small.

- For example, if four packets depart at times 0, 1,2,3 and arrive at 20, 21,22, 23, all have same delay, 20 units of time. On the other hand, if the above four packets arrive at 21,23,21, and 28 they will have different delays of21, 22, 19 and 24.

Bandwidth

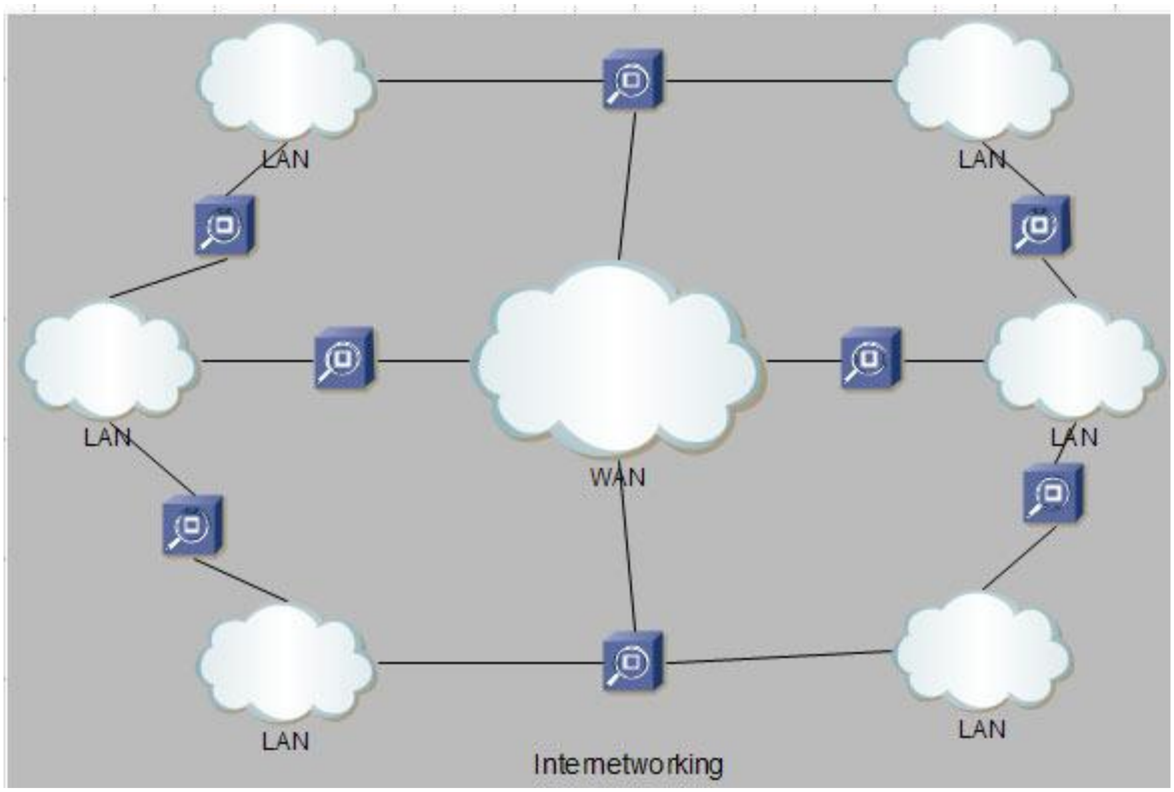
- Different applications need different bandwidths.

- In video conferencing we need to send million of bits per second to refresh a colour screen while the total number of bits in an email may not reach even a million.

Internetworking :

Internetworking started as a way to connect disparate types of computernetworking technology. Computer network term is used to describe two or more computers that are linked to each other. When two or more computer networks or computer network segments are connected using devices such as a router then it is called as **computer internetworking**.

Internetworking is a term used by Cisco. Any interconnection among or between public, private, commercial, industrial, or governmental computer networks may also be defined as an internetwork or **Internetworking**.



In modern practice, the interconnected computer networks or **Internetworking** use the Internet Protocol. Two architectural models are commonly used to describe the protocols and methods used in **internetworking**. The standard reference model for **internetworking** is Open Systems Interconnection (**OSI)**.

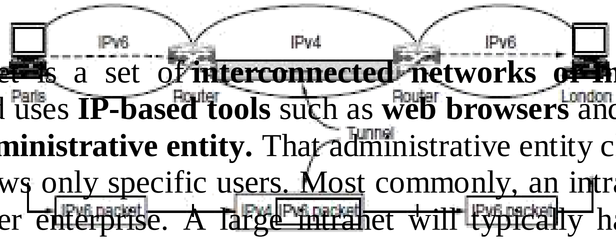
Internetworking is implemented in Layer 3 (Network Layer) of this model The most notable example of internetworking is the Internet (capitalized). There are three variants of internetwork or **Internetworking**, depending on who administers and who participates in them :

- **Extranet**
- **Intranet**
- **Internet**

Intranets and extranets may or may not have connections to the Internet. If connected to the Internet, the intranet or extranet is normally protected from being accessed from the Internet without proper authorization. The Internet is not considered to be a part of the intranet or extranet, although it may serve as a portal for access to portions of an extranet.

Extranet : An extranet is a **network of internetwork or Internetworking** that is limited in scope to a **single organisation or entity** but which also has **limited connections** to the networks of one or more other usually, but not necessarily, trusted organizations or entities .Technically, an **extranet may also be categorized as a MAN, WAN**, or other type of network, although, by definition, an extranet cannot consist of a single LAN; it must have at least one connection with an external network.

Intranet : An intranet is a set of **interconnected networks or Internetworking**, using the **Internet Protocol** and uses **IP-based tools** such as **web browsers** and **ftp tools**, that is under the control of a **single administrative entity.** That administrative entity closes the intranet to the rest of the world, and allows only specific users. Most commonly, an intranet is the internal network of a company or other enterprise. A large intranet will typically have its **own web server** to provide users with browseable information.



Internet : A specific **Internetworking**, consisting of a **worldwide interconnection** of governmental, academic, public, and private networks based upon the Advanced Research Projects Agency Network (**ARPANET)** developed by ARPA of the U.S. **Department of Defense** also **home**to the **World Wide Web (WWW)** and referred to as the '**Internet**' with a capital 'I' to distinguish it from other generic internetworks. Participants in the Internet, or their service providers, use IP Addresses obtained from address registries that control assignments.

Tunneling

Handling the general case of making two different networks interwork is exceedingly difficult. However, there is a common special case that is manageable even for different network protocols. This case is where the source and destination hosts are on the same type of network, but there is a different network in between. As an example, think of an international bank with an IPv6 network in Paris, an IPv6 network in London and connectivity between the offices via the IPv4 Internet. This situation is shown in Fig.

Figure: Tunneling a packet from Paris to London.

The solution to this problem is a technique called **tunneling**. To send an IP packet to a host in the London office, a host in the Paris office constructs the packet containing an IPv6 address in London, and sends it to the multiprotocol router that connects the Paris IPv6 network to the IPv4 Internet. When this router gets the IPv6 packet, it encapsulates the packet with an IPv4 header addressed to the IPv4 side of the multiprotocol router that connects to the London IPv6 network.

That is, the router puts a (IPv6) packet inside a (IPv4) packet. When this wrapped packet arrives, the London router removes the original IPv6 packet and sends it onward to the destination host.

The path through the IPv4 Internet can be seen as a big tunnel extending from one multiprotocol router to the other. The IPv6 packet just travels from one end of the tunnel to the other, snug in its nice box. It does not have to worry about dealing with IPv4 at all. Neither do the hosts in Paris or London. Only the multiprotocol routers have to understand both IPv4 and IPv6 packets. In effect, the entire trip from one multiprotocol router to the other is like a hop over a single link.

An analogy may make tunneling clearer. Consider a person driving her car from Paris to London. Within France, the car moves under its own power, but when it hits the English Channel, it is loaded onto a high-speed train and transported to England through the Chunnel (cars are not permitted to drive through the Chunnel). Effectively, the car is being carried as freight, as depicted in Fig. At the far end, the car is let loose on the English roads and once again continues to move under its own power. Tunneling of packets through a foreign network works the same way. Tunneling is widely used to connect isolated hosts and networks using other networks. The network that results is called an **overlay** since it has effectively been overlaid on the base network. Deployment of a network protocol with a new feature is a common reason, as our ‘‘IPv6 over IPv4’’ example shows. The disadvantage of tunneling is that none of the hosts on the network that are tunneled over can be reached because the packets cannot escape in the middle of the tunnel.

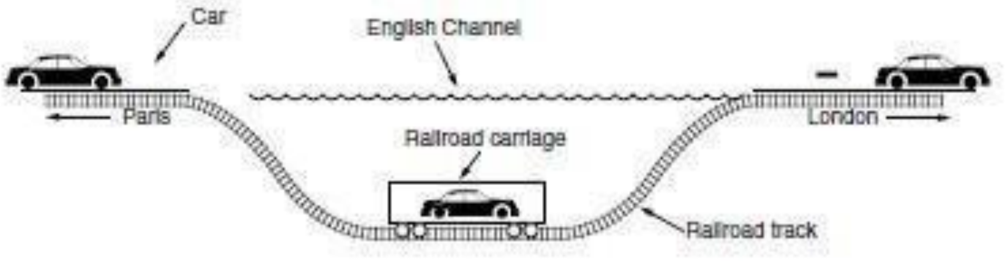


Figure: Tunneling a car from France to England.

However, this limitation of tunnels is turned into an advantage with **VPNs** (**Virtual Private Networks**). A VPN is simply an overlay that is used to provide a measure of security.

Internetwork Routing

Routing through an internet poses the same basic problem as routing within a single network, but with some added complications. To start, the networks may internally use different routing algorithms. For example, one network may use link state routing and another distance vector routing. Since link state algorithms need to know the topology but distance vector algorithms do not, this difference alone would make it unclear how to find the shortest paths across the internet. Networks run by different operators lead to bigger problems. First, the operators may have different ideas about what is a good path through the network. One operator may want the route with the least delay, while another may want the most inexpensive route. This will lead the operators to use different quantities to set the shortest-path costs (e.g., milliseconds of delay vs. monetary cost). The weights will not be comparable across networks, so shortest paths on the internet will not be well defined. Worse yet, one operator may not want another operator to even know the details of the paths in its network, perhaps because the weights and paths may reflect sensitive information (such as the monetary cost) that represents a competitive business advantage. Finally, the internet may be much larger than any of the networks that comprise it. It may therefore require routing algorithms that scale well by using a hierarchy; even if none of the individual networks need to use a hierarchy. All of these considerations lead to a two-level routing algorithm. Within each network, an **intra domain** or **interior gateway protocol** is used for routing.(‘‘Gateway’’ is an older term for ‘‘router.’’) It might be a link state protocol of the kind we have already described. Across the networks that make up the internet, an **inter domain** or **exterior gateway protocol** is used. The networks may all use different intra domain protocols, but they must use the same inter domain protocol. In the Internet, the inter domain routing protocol is called **BGP** (**Border Gateway Protocol**).

We will there is one more important term to introduce. Since each network is operated independently of all the others, it is often referred to as an **AS** (**Autonomous System**). A good mental model for an AS is an ISP network. In fact, an ISP network may be comprised of more than one AS, if it is managed, or, has been acquired, as multiple networks. But the difference is usually not significant. The two levels are usually not strictly hierarchical, as highly suboptimal paths might result if a large international network and a small regional network were both abstracted to be a single network. However, relatively little information about routes within the networks is exposed to find routes across the internetwork.

This helps to address all of the complications. It improves scaling and lets operators freely select routes within their own networks using a protocol of their choosing. It also does not require weights to be compared across networks or expose sensitive information outside of networks. However, we have said little so far about how the routes across the networks of the internet are determined. In the Internet, a large determining factor is the business arrangements between ISPs. Each ISP may charge or receive money from the other ISPs for carrying traffic. Another factor is that if internetwork routing requires crossing international boundaries, various laws may suddenly come into play, such as Sweden’s strict privacy laws about exporting personal data about Swedish citizens from Sweden. All of these nontechnical factors are wrapped up in the concept of a **routing policy** that governs the way autonomous networks select the routes that they use. We will return to routing policies when we describe BGP.

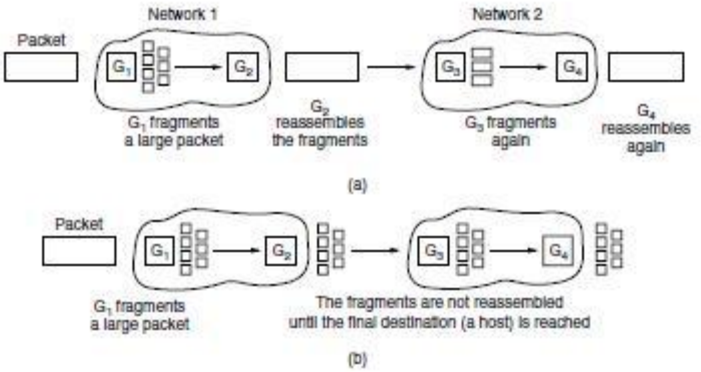
Packet Fragmentation

Each network or link imposes some maximum size on its packets. These limits have various causes, among them

1. Hardware (e.g., the size of an Ethernet frame).
2. Operating system (e.g., all buffers are 512 bytes).
3. Protocols (e.g., the number of bits in the packet length field).
4. Compliance with some (inter)national standard.
5. Desire to reduce error-induced retransmissions to some level.
6. Desire to prevent one packet from occupying the channel too long.

The result of all these factors is that the network designers are not free to choose any old maximum packet size they wish. Maximum payloads for some common technologies are 1500 bytes for Ethernet and 2272 bytes for 802.11. IP is more generous, allows for packets as big as 65,515 bytes. Hosts usually prefer to transmit large packets because this reduces packet overheads such as bandwidth wasted on header bytes. An obvious internetworking problem appears when a large packet wants to travel through a network whose maximum packet size is too small. This nuisance has been a persistent issue, and solutions to it have evolved along with much experience gained on the Internet. One solution is to make sure the problem does not occur in the first place. However, this is easier said than done. A source does not usually know the path a packet will take through the network to a destination, so it certainly does not know how small packets must be to get there. This packet size is called the **Path MTU** (**Path Maximum Transmission Unit**). Even if the source did know the path MTU, packets are routed independently in a connectionless network such as the Internet. This routing means that paths may suddenly change, which can unexpectedly change the path MTU. The alternative solution to the problem is to allow routers to break up packets into **fragments**, sending each fragment as a separate network layer packet. However, as every parent of a small child knows, converting a large object into small fragments is considerably easier than the reverse process. (Physicists have even given this effect a name: the second law of thermodynamics.) Packet-switching networks, too, have trouble putting the fragments back together again. Two opposing strategies exist for recombining the fragments back into the original packet. The first strategy is to make fragmentation caused by a ‘‘small packet’’ network transparent to any subsequent networks through which the packet must pass on its way to the ultimate destination. This option is shown in Fig (a). In this approach, when an oversized packet arrives at *G1*, the router breaks it up into fragments. Each fragment is addressed to the same exit router, *G2*, where the pieces are recombined. In this way, passage through the small-packet network is made transparent. Subsequent networks are not even aware that fragmentation has occurred.

Transparent fragmentation is straightforward but has some problems. For one thing, the exit router must know when it has received all the pieces, so either a count field or an ‘‘end of packet’’ bit must be provided. Also, because all packets must exit via the same router so that they can be reassembled, the routes are constrained. By not allowing some fragments to follow one route to the ultimate destination and other fragments a disjoint route, some performance may be lost. More significant is the amount of work that the router may have to do. It may need to buffer the fragments as they arrive, and decide when to throw them away if not all of the fragments arrive. Some of this work may be wasteful, too, as the packet may pass through a series of small packet networks and need to be repeatedly fragmented and reassembled. The other fragmentation strategy is to refrain from recombining fragments at any intermediate routers. Once a packet has been fragmented, each fragment is

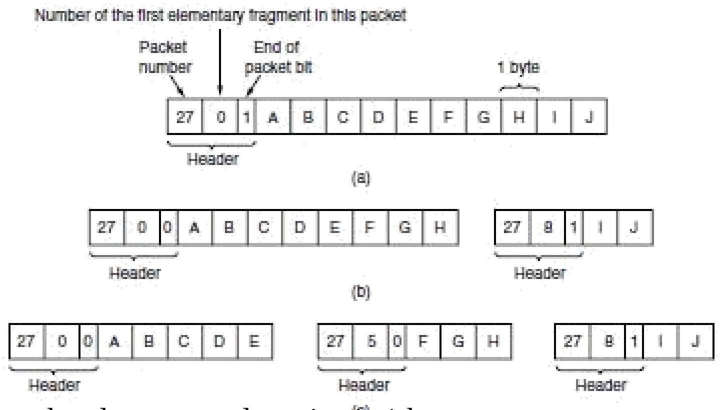

Network 1
Network 2
Packet
G_1
G_2
G_3
G_4
G_1 fragments a large packet
G_2 reassembles the fragments
G_3 fragments again
G_4 reassembles again
(a)
Packet
G_1
G_2
G_3
G_4
G_1 fragments a large packet
The fragments are not reassembled until the final destination (a host) is reached
(b)

(a) Transparent fragmentation. (b) Nontransparentfragmentation.

treated as though it were an original packet. The routers pass the fragments, as shown in Fig. (b), and reassembly is performed only at the destination host. The main advantage of nontransparent fragmentation is that it requires routers to do less work. IP works this way. A complete design requires that the fragments be numbered in such a way that the original data stream can be reconstructed.

The design used by IP is to give every fragment a packet number (carried on all packets), an absolute byte offset within the packet, and a flag indicating whether it is the end of the packet. An example is shown in Fig. While simple, this design has some attractive properties. Fragments can be placed in a buffer at the destination in the right place for reassembly, even if they arrive out of order.

Fragments can also be fragmented if they pass over a network with a yet smaller MTU. This is shown in Fig. (c). Retransmissions of the packet (if all fragments were not received) can be fragmented into different pieces. Finally, fragments can be of arbitrary size, down to a single byte plus the packet header. In all cases, the destination simply uses the packet number and fragment offset to place the data in the right position, and the end-of-packet flag to determine when it has the complete packet. Unfortunately, this design still has problems. The overhead can be higher than with transparent fragmentation because fragment headers are now carried over some links where they may not be needed. But the real problem is the existence of fragments in the first place. Kent and Mogul (1987) argued that fragmentation is detrimental to performance because, as well as the header overheads, a whole packet is lost if any of its fragments are lost and because fragmentation is more of a burden for hosts than was originallyrealized.



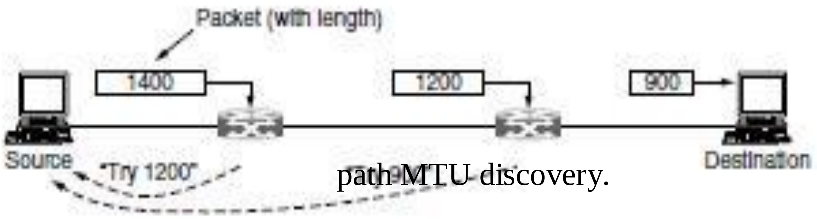
Fragmentation when the elementary data size is 1 byte.

(a) Original packet, containing 10 data bytes.

(b) Fragments after passing through a network with maximum packet size of 8 payload bytes plus header.

(c) Fragments after passing through a size 5 gateway.

This leads us back to the original solution of getting rid of fragmentation in the network, the strategy used in the modern Internet. The process is called **path MTU discovery** (Mogul and Deering, 1990). It works as follows. Each IP packet is sent with its header bits set to indicate that no fragmentation is allowed to be performed. If a router receives a packet that is too large, it generates an error packet, returns it to the source, and drops the packet. This is shown in Fig. When the source receives the error packet, it uses the information inside to refragment the packet into pieces that are small enough for the router to handle. If a router further down the path has an even smaller MTU, the process is repeated.

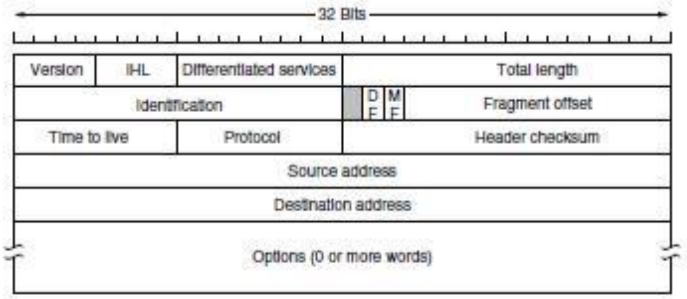


path MTU discovery.

The advantage of path MTU discovery is that the source now knows what length packet to send. If the routes and path MTU change, new error packets will be triggered and the source will adapt to the new path. However, fragmentation is still needed between the source and the destination unless the higher layers learn the path MTU and pass the right amount of data to IP. TCP and IP are typically implemented together (as ‘‘TCP/IP’’) to be able to pass this sort of information. Even if this is not done for other protocols, fragmentation has still been moved out of the network and into the hosts. The disadvantage of path MTU discovery is that there may be added startup delays simply to send a packet. More than one round-trip delay may be needed to probe the path and find the MTU before any data is delivered to the destination. This begs the question of whether there are better designs. The answer is probably ‘‘Yes.’’ Consider the design in which each router simply truncates packets that exceed its MTU. This would ensure that the destination learns the MTU as rapidly as possible (from the amount of data that was delivered) and receives some of the data.

The IP Version 4 Protocol

An appropriate place to start our study of the network layer in the Internet is with the format of the IP datagrams themselves. An IPv4 datagram consists of a header part and a body or payload part. The header has a 20-byte fixed part and a variable-length optional part. The header format is shown in Fig. 5-46. The bits are transmitted from left to right and top to bottom, with the high-order bit of the *Version* field going first. (This is a ‘‘big-endian’’ network byte order. On little endian machines, such as Intel x86 computers, a software conversion is required on both transmission and reception.) In retrospect, little endian would have been a better choice, but at the time IP was designed, no one knew it would come to dominate computing.



The IPv4 (Internet Protocol) header.

The *Version* field keeps track of which version of the protocol the datagram belongs to. Version 4 dominates the Internet today, and that is where we have started our discussion. By including the version at the start of each datagram, it becomes possible to have a transition between versions over a long period of time. In fact, IPv6, the next version of IP, was defined more than a decade ago, yet is only just beginning to be deployed. We will describe it later in this section. Its use will eventually be forced when each of China’s almost 231 people has a desktop PC, a laptop, and an IP phone. As an aside on numbering, IPv5 was an experimental real-time stream protocol that was never widely used.

Since the header length is not constant, a field in the header, *IHL*, is provided to tell how long the header is, in 32-bit words. The minimum value is 5, which applies when no options are present. The maximum value of this 4-bit field is 15, which limits the header to 60 bytes, and thus the *Options* field to 40 bytes. For some options, such as one that records the route a packet has taken, 40 bytes is far too small, making those options useless. The *Differentiated services* field is one of the few fields that has changed its meaning (slightly) over the years. Originally, it was called the *Type of service* field. It was and still is intended to distinguish between different classes of service.

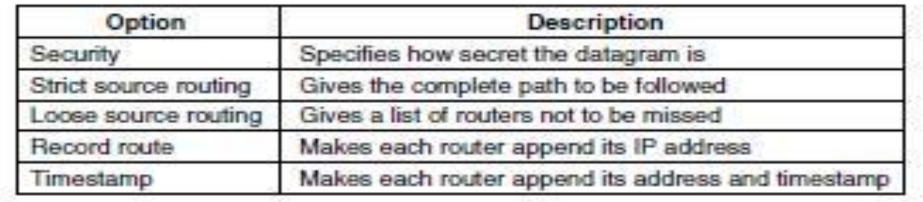
Various combinations of reliability and speed are possible. For digitized voice, fast delivery beats accurate delivery. For file transfer, error-free transmission is more important than fast transmission. The *Type of service* field provided 3 bits to signal priority and 3 bits to signal whether a host cared more about delay, throughput, or reliability. However, no one really knew what to do with these bits at routers, so they were left unused for many years. When differentiated services were designed, IETF threw in the towel and reused this field. Now, the top 6 bits are used to mark the packet with its service class; we described the expedited and assured services earlier in this chapter. The bottom 2 bits are used to carry explicit congestion notification information, such as whether the packet has experienced congestion; we described explicit congestion notification as part of congestion control earlier in this chapter. The *Total length* includes everything in the datagram—both header and data. The maximum length is 65,535 bytes. At present, this upper limit is tolerable, but with future networks, larger datagrams may be needed. The *Identification* field is needed to allow the destination host to determine which packet a newly arrived fragment belongs to. All the fragments of a packet contain the same *Identification* value.

Next comes an unused bit, which is surprising, as available real estate in the IP header is extremely scarce. As an April fool's joke, Bellovin (2003) proposed using this bit to detect malicious traffic. This would greatly simplify security, as packets with the ''evil'' bit set would be known to have been sent by attackers and could just be discarded. Unfortunately, network security is not this simple. Then come two 1-bit fields related to fragmentation. *DF* stands for Don't Fragment. It is an order to the routers not to fragment the packet. Originally, it was intended to support hosts incapable of putting the pieces back together again. Now it is used as part of the process to discover the path MTU, which is the largest packet that can travel along a path without being fragmented. By marking the datagram with the *DF* bit, the sender knows it will either arrive in one piece, or an error message will be returned to the sender. *MF* stands for More Fragments. All fragments except the last one have this bit set. It is needed to know when all fragments of a datagram have arrived. The *Fragment offset* tells where in the current packet this fragment belongs.

All fragments except the last one in a datagram must be a multiple of 8 bytes, the elementary fragment unit. Since 13 bits are provided, there is a maximum of 8192 fragments per datagram, supporting a maximum packet length up to the limit of the *Total length* field. Working together, the *Identification, MF,* and *Fragment offset* fields are used to implement fragmentation as described in Sec. 5.5.5.The *TtL (Time to live)* field is a counter used to limit packet lifetimes. It was originally supposed to count time in seconds, allowing a maximum lifetime of 255 sec. It must be decremented on each hop and is supposed to be decremented multiple times when a packet is queued for a long time in a router. In practice, it just counts hops. When it hits zero, the packet is discarded and a warning packet is sent back to the source host. This feature prevents packets from wandering around forever, something that otherwise might happen if the routing tables ever become corrupted.

When the network layer has assembled a complete packet, it needs to know what to do with it. The *Protocol* field tells it which transport process to give the packet to. TCP is one possibility, but so are UDP and some others. The numbering of protocols is global across the entire Internet. Protocols and other assigned numbers were formerly listed in RFC 1700, but nowadays they are contained in an online database located at www.iana.org. Since the header carries vital information such as addresses, it rates its own checksum for protection, the *Header checksum*. The algorithm is to add up all the 16-bit half words of the header as they arrive, using one's complement arithmetic, and then take the one's complement of the result. For purposes of this algorithm, the *Header checksum* is assumed to be zero upon arrival. Such a checksum is useful for detecting errors while the packet travels through the network. Note that it must be recomputed at each hop because at least one field always changes (the *Time to live* field), but tricks can be used to speed up the computation. The *Source address* and *Destination address* indicate the IP address of the source and destination network interfaces. The *Options* field was designed to provide an escape to allow subsequent versions of the protocol to include information not present in the original design, to permit experimenters to try out new ideas, and to avoid allocating header bits to information that is rarely needed. The options are of variable length. Each begins with a 1-byte code identifying the option. Some options are followed by a 1-byte option length field, and then one or more data bytes. The *Options* field is padded out to a multiple of 4 bytes. Originally, the five options listed in Fig. were defined.

The *Security* option tells how secret the information is. In theory, a military router might use this field to specify not to route packets through certain countries the military considers to be ‘‘bad guys.’’ In practice, all routers ignore it, so its only practical function is to help spies find the good stuff more easily. The *Strict source routing* option gives the complete path from source to destination as a sequence of IP addresses. The datagram is required to follow that exact route. It is most useful for system managers who need to send emergency packets when the routing tables have been corrupted, or for making timing measurements. The *Loose source routing* option requires the packet to traverse the list of routers specified, in the order specified, but it is allowed to pass through other routers on the way. Normally, this option will provide only a few routers, to force a particular path. For example, to force a packet from London to Sydney to go west instead of east, this option might specify routers in New York, Los Angeles, and Honolulu. This option is most useful when political or economic considerations dictate passing through or avoiding certain countries.



Option	Description
Security	Specifies how secret the datagram is
Strict source routing	Gives the complete path to be followed
Loose source routing	Gives a list of routers not to be missed
Record route	Makes each router append its IP address
Timestamp	Makes each router append its address and timestamp

Some of the IP options

The *Record route* option tells each router along the path to append its IP address to the *Options* field. This allows system managers to track down bugs in the routing algorithms (‘‘Why are packets from Houston to Dallas visiting Tokyo first?’’). When the ARPANET was first set up, no packet ever passed through more than nine routers, so 40 bytes of options was plenty. As mentioned above, now it is too small. Finally, the *Timestamp* option is like the *Record route* option, except that in addition to recording its 32-bit IP address, each router also records a 32-bit timestamp. This option, too, is mostly useful for network measurement. Today, IP options have fallen out of favor. Many routers ignore them or do not process them efficiently, shunting them to the side as an uncommon case. That is, they are only partly supported and they are rarely used.

IP Version 6

IP has been in heavy use for decades. It has worked extremely well, as demonstrated by the exponential growth of the Internet. Unfortunately, IP has become a victim of its own popularity: it is close to running out of addresses. Even with CIDR and NAT using addresses more sparingly, the last IPv4 addresses are expected to be assigned by ICANN before the end of 2012. This looming disaster was recognized almost two decades ago, and it sparked a great deal of discussion and controversy within the Internet community about what to do about it. In this section, we will describe both the problem and several proposed solutions. The only long-term solution is to move to larger addresses.

IPv6 (**IP version 6**): is a replacement design that does just that. It uses 128-bit addresses; a shortage of these addresses is not likely any time in the foreseeable future. However, IPv6 has proved very difficult to deploy. It is a different network layer protocol that does not really interwork with IPv4, despite many similarities. Also, companies and users are not really sure why they should want IPv6 in any case. The result is that IPv6 is deployed and used on only a tiny fraction of the Internet (estimates are 1%) despite having been an Internet Standard since 1998. The next several years will be an interesting time, as the few remaining IPv4 addresses are allocated. Will people start to auction off their IPv4 addresses on eBay? Will a black market in them spring up? Who knows? In addition to the address problems, other issues loom in the background. In its early years, the Internet was largely used by universities, high-tech industries, and the U.S. Government (especially the Dept. of Defense). With the explosion of interest in the Internet starting in the mid-1990s, it began to be used by a different group of people, often with different requirements. For one thing, numerous people with smart phones use it to keep in contact with their home bases. For another, with the impending convergence of the computer, communication, and entertainment industries, it may not be that long before every telephone and television set in the world is an Internet node, resulting in a billion machines being used for audio and video on demand. Under these circumstances, it became apparent that IP had to evolve and become more flexible. Seeing these problems on the horizon, in 1990 IETF started work on a new version of IP, one that would never run out of addresses, would solve a variety of other problems, and be more flexible and efficient as well. Its major goals were:

1. Support billions of hosts, even with inefficient addressallocation.
2. Reduce the size of the routing tables.
3. Simplify the protocol, to allow routers to process packets faster.
4. Provide better security (authentication and privacy).
5. Pay more attention to the type of service, particularly for real-time data.
6. Aid multicasting by allowing scopes to be specified.
7. Make it possible for a host to roam without changing its address.
8. Allow the protocol to evolve in the future.
9. Permit the old and new protocols to coexist for years.

The design of IPv6 presented a major opportunity to improve all of the features in IPv4 that fall short of what is now wanted. To develop a protocol that met all these requirements, IETF issued a call for proposals and discussion in RFC 1550. Twenty-one responses were initially received. By December 1992, seven serious proposals were on the table. They ranged from making minor patches to IP, to throwing it out altogether and replacing it with a completely different protocol. One proposal was to run TCP over CLNP, the network layer protocol designed for OSI. With its 160-bit addresses, CLNP would have provided enough address space forever as it could give every molecule of water in the oceans enough addresses (roughly 25) to set up a small network. This choice would also have unified two major network layer protocols. However, many people felt that this would have been an admission that something in the OSI world was actually done right, a statement considered Politically Incorrect in Internet circles. CLNP was patterned closely on IP, so the two are not really that different. In fact, the protocol ultimately chosen differs from

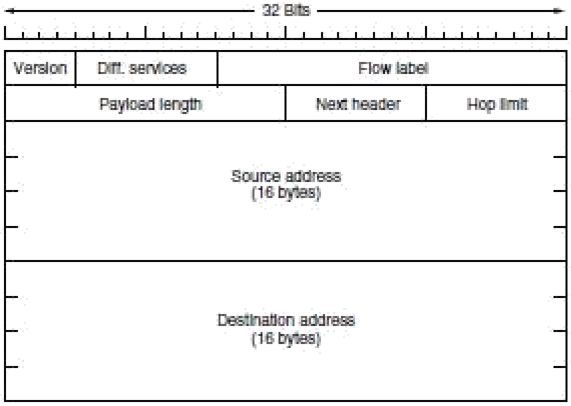
IP far more than CLNP does. Another strike against CLNP was its poor support for service types, something required to transmit multimedia efficiently.

Three of the better proposals were published in *IEEE Network* (Deering, 1993; Francis, 1993; and Katz and Ford, 1993). After much discussion, revision, and jockeying for position, a modified combined version of the Deering and Francis proposals, by now called **SIPP** (**Simple Internet Protocol Plus**) was selected and given the designation **IPv6.** IPv6 meets IETF's goals fairly well. It maintains the good features of IP, discards or deemphasizes the bad ones, and adds new ones where needed. In general, IPv6 is not compatible with IPv4, but it is compatible with the other auxiliary Internet protocols, including TCP, UDP, ICMP, IGMP, OSPF, BGP, and DNS, with small modifications being required to deal with longer addresses. The main features of IPv6 are discussed below. More information about it can be found in RFCs 2460 through 2466.First and foremost, IPv6 has longer addresses than IPv4. They are 128 bits long, which solves the problem that IPv6 set out to solve: providing an effectively unlimited supply of Internet addresses. We will have more to say about addresses shortly. The second major improvement of IPv6 is the simplification of the header. It contains only seven fields (versus 13 in IPv4). This change allows routers to process packets faster and thus improves throughput and delay. We will discuss the header shortly, too. The third major improvement is better support for options. This change was essential with the new header because fields that previously were required are now optional (because they are not used so often). In addition, the way options are represented is different, making it simple for routers to skip over options not intended for them. This feature speeds up packet processing time.

A fourth area in which IPv6 represents a big advance is in security. IETF had its fill of newspaper stories about precocious 12-year-olds using their personal computers to break into banks and military bases all over the Internet. There was a strong feeling that something had to be done to improve security. Authentication and privacy are key features of the new IP. These were later retrofitted to IPv4, however, so in the area of security the differences are not so great any more. Finally, more attention has been paid to quality of service. Various halfhearted efforts to improve QoS have been made in the past, but now, with the growth of multimedia on the Internet, the sense of urgency is greater.

The Main IPv6 Header

The IPv6 header is shown in Fig. 5-56. The *Version* field is always 6 for IPv6 (and 4 for IPv4). During the transition period from IPv4, which has already taken more than a decade, routers will be able to examine this field to tell what kind of packet they have. As an aside, making this test wastes a few instructions in the critical path, given that the data link header usually indicates the network protocol for de-multiplexing, so some routers may skip the check. For example, the Ethernet *Type* field has different values to indicate an IPv4 or an IPv6 payload. The discussions between the ‘‘Do it right’’ and ‘‘Make it fast’’ camps will no doubt be lengthy and vigorous.



The IPv6 fixed header (required).

The ***Differentiated services*** field (originally called *Traffic class*) is used to distinguish the class of service for packets with different real-time delivery requirements. It is used with the Differentiated service architecture for quality of service in the same manner as the field of the same name in the IPv4 packet. Also, the low-order 2 bits are used to signal explicit congestion indications, again in the same way as with IPv4.

The ***Flow label*** field provides a way for a source and destination to mark groups of packets that have the same requirements and should be treated in the same way by the network, forming a pseudo connection. For example, a stream of packets from one process on a certain source host to a process on a specific destination host might have stringent delay requirements and thus need reserved bandwidth. The flow can be set up in advance and given an identifier. When a packet with a nonzero *Flow label* shows up, all the routers can look it up in internal tables to see what kind of special treatment it requires. In effect, flows are an attempt to have it both ways: the flexibility of a datagram network and the guarantees of a virtual-circuit network. Each flow for quality of service purposes is designated by the source address, destination address, and flow number. This design means that up to 220 flows may be active at the same time between a given pair of IP addresses. It also means that even if two flows coming from different hosts but with the same flow label pass through the same router, the router will be able to tell them apart using the source and destination addresses. It is expected that flow labels will be chosen randomly, rather than assigned sequentially starting at 1, so routers are expected to hash them. The *Payload length* field tells how many bytes follow the 40-byte header of Fig. The name was changed from the IPv4 *Total length* field because the meaning was changed slightly: the 40 header bytes are no longer counted as part of the length (as they used to be). This change means the payload can now be 65,535 bytes instead of a mere 65,515 bytes.

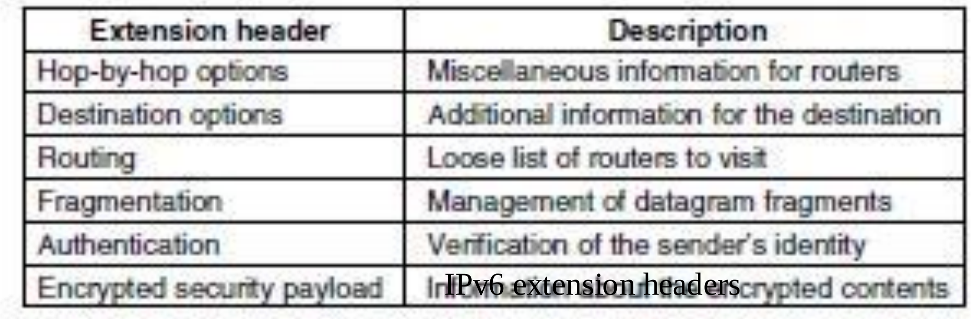
The *Next header* field lets the cat out of the bag. The reason the header could be simplified is that there can be additional (optional) extension headers. This field tells which of the (currently) six extension headers, if any, follow this one.

If this header is the last IP header, the *Next header* field tells which transport protocol handler (e.g., TCP, UDP) to pass the packet to. The *Hop limit* field is used to keep packets from living forever. It is, in practice, the same as the *Time to live* field in IPv4, namely, a field that is decremented on each hop. In theory, in IPv4 it was a time in seconds, but no router used it that way, so the name was changed to reflect the way it is actually used. Next come the *Source address* and *Destination address* fields. Deering's original proposal, SIP, used 8-byte addresses, but during the review process many people felt that with 8-byte addresses IPv6 would run out of addresses within a few decades, whereas with 16-byte addresses it would never run out. Other people argued that 16 bytes was overkill, whereas still others favored using 20-byte addresses to be compatible with the OSI datagram protocol. Still another faction wanted variable-sized addresses. After much debate and more than a few words unprintable in an academic textbook, it was decided that fixed-length 16-byte addresses were the best compromise.

A new notation has been devised for writing 16-byte addresses. They are written as eight groups of four hexadecimal digits with colons between the groups, like this: 8000:0000:0000:0000:0123:4567:89AB: CDEF Since many addresses will have many zeros inside them, three optimizations have been authorized. First, leading zeros within a group can be omitted, so 0123 can be written as 123. Second, one or more groups of 16 zero bits can be replaced by a pair of colons. Thus, the above address now becomes 8000::123:4567:89AB:CDEF Finally, IPv4 addresses can be written as a pair of colons and an old dotted decimal number, for example: ::192.31.20.46 Perhaps it is unnecessary to be so explicit about it, but there are a lot of 16- byte addresses. Specifically, there are a lot of 162128 of them, which is approximately 30010038.If the entire earth, land and water ,were covered with computers,IPV6 would allow 7001023 addresses per square meter. Students of chemistry will notice that this number is larger than Avogadro's number. While it was not the intention to give every molecule on the surface of the earth its own IP address, we are not that far off. In practice, the address space will not be used efficiently, just as the telephone number address space is not (the area code for Manhattan, 212, is nearly full, but that for Wyoming, 307, is nearly empty). In RFC 3194, Durand and Huitema calculated that, using the allocation of telephone numbers as a guide, even in the most pessimistic scenario there will still be well over 1000 IP addresses per square meter of the entire earth's surface (land and water). In any likely scenario, there will be trillions of them per square meter. In short, it seems unlikely that we will run out in the foreseeable future. It is instructive to compare the IPv4 header (Fig.) with the IPv6 header (Fig) to see what has been left out in IPv6. The *IHL* field is gone because the IPv6 header has a fixed length. The *Protocol* field was taken out because the *Next header* field tells what follows the last IP header (e.g., a UDP or TCP segment).All the fields relating to fragmentation were removed because IPv6 takes a different approach to fragmentation. To start with, all IPv6-conformant hosts are expected to dynamically determine the packet size to use. They do this using the path MTU discovery procedure we described in Sec. 5.5.5. In brief, when a host sends an IPv6 packet that is too large, instead of fragmenting it, the router that is unable to forward it drops the packet and sends an error message back to the sending host. This message tells the host to break up all future packets to that destination. Having the host send packets that are the right size in the first place is ultimately much more efficient than having the routers fragment them on the fly. Also, the minimum-size packet that routers must be able to forward has been raised from 576 to 1280 bytes to allow 1024 bytes of data and many headers. Finally, the *Checksum* field is gone because calculating it greatly reduces performance. With the reliable networks now used, combined with the fact that the data link layer and transport layers normally have their own checksums, the value of yet another checksum was deemed not worth the performance price it extracted. Removing all these features has resulted in a lean and mean network layer protocol. Thus, the goal of IPv6—a fast, yet flexible, protocol with plenty of address space—is met by this design.

Extension Headers

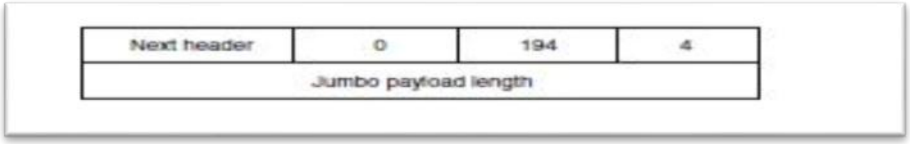
Some of the missing IPv4 fields are occasionally still needed, so IPv6 introduces the concept of (optional) **extension headers**. These headers can be supplied to provide extra information, but encoded in an efficient way. Six kinds of extension headers are defined at present, as listed in Fig. Each one is optional, but if more than one is present they must appear directly after the fixed header, and preferably in the order listed.



Extension header	Description
Hop-by-hop options	Miscellaneous information for routers
Destination options	Additional information for the destination
Routing	Loose list of routers to visit
Fragmentation	Management of datagram fragments
Authentication	Verification of the sender's identity
Encrypted security payload	Information about the encrypted contents

IPv6 extension headers

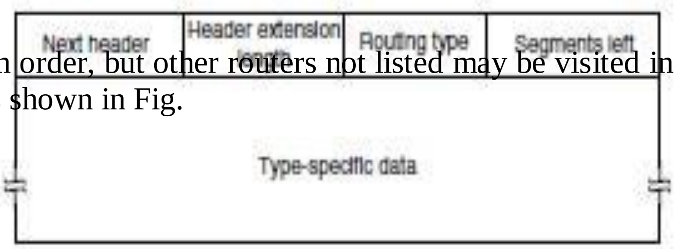
Some of the headers have a fixed format; others contain a variable number of variable-length options. For these, each item is encoded as a (*Type*, *Length*, *Value*) tuple. The *Type* is a 1-byte field telling which option this is. The *Type* values have been chosen so that the first 2 bits tell routers that do not know how to process the option what to do. The choices are: skip the option; discard the packet; discard the packet and send back an ICMP packet; and discard the packet but do not send ICMP packets for multicast addresses (to prevent one bad multicast packet from generating millions of ICMP reports). The *Length* is also a 1-byte field. It tells how long the value is (0 to 255 bytes). The *Value* is any information required, up to 255 bytes. The hop-by-hop header is used for information that all routers along the path must examine. So far, one option has been defined: support of datagrams exceeding 64 KB. The format of this header is shown in. When it is used, the *Payload length* field in the fixed header is set to 0.



Next header	0	194	4
Jumbo payload length			

Figure: The hop-by-hop extension header for large data grams (jumbo grams)

As with all extension headers, this one starts with a byte telling what kind of header comes next. This byte is followed by one telling how long the hop-by-hop header is in bytes, excluding the first 8 bytes, which are mandatory. All extensions begin this way. The next 2 bytes indicate that this option defines the datagram size (code 194) and that the size is a 4-byte number. The last 4 bytes give the size of the datagram. Sizes less than 65,536 bytes are not permitted and will result in the first router discarding the packet and sending back an ICMP error message. Datagrams using this header extension are called **jumbo grams**. The use of jumbo grams is important for supercomputer applications that must transfer gigabytes of data efficiently across the Internet. The destination options header is intended for fields that need only be interpreted at the destination host. In the initial version of IPv6, the only options defined are null options for padding this header out to a multiple of 8 bytes, so initially it will not be used. It was included to make sure that new routing and host software can handle it, in case someone thinks of a destination option some day. The routing header lists one or more routers that must be visited on the way to the destination. It is very similar to the IPv4 loose source routing in that all addresses listed must be visited in order, but other routers not listed may be visited in between. The format of the routing header is shown in Fig.



The extension header for routing

The first 4 bytes of the routing extension header contain four 1-byte integers. The *Next header* and *Header extension length* fields were described above. The *Routing type* field gives the format of the rest of the header. Type 0 says that a reserved 32-bit word follows the first word, followed by some number of IPv6 addresses. Other types may be invented in the future, as needed. Finally, the *Segments left* field keeps track of how many of the addresses in the list have not yet been visited. It is decremented every time one is visited. When it hits 0, the packet is on its own with no more guidance about what route to follow. Usually, at this point it is so close to the destination that the best route is obvious. The fragment header deals with fragmentation similarly to the way IPv4 does. The header holds the datagram identifier, fragment number, and a bit telling whether more fragments will follow.

Open Shortest Path First : Routers connect networks using the Internet Protocol (IP), and OSPF (Open Shortest Path First) is a router protocol used to find the best path for packets as they pass through a set of connected networks. Open Shortest Path First (OSPF) is an Interior Gateway Protocol (IGP) standardized by the Internet Engineering Task Force (IETF) and commonly used in large Enterprise networks. OSPF is a link-state routing protocol providing fast convergence and excellent scalability. Like all link-state protocols, OSPF is very efficient in its use of network bandwidth.

OSPF is a standardized Link-State routing protocol, designed to scale efficiently to support larger networks. OSPF adheres to the following Link State characteristics:

- OSPF employs a hierarchical network design using Areas.
- OSPF will form neighbor relationships with adjacent routers in the same Area.
- Instead of advertising the distance to connected networks, OSPF advertises the status of directly connected links using Link-State Advertisements (LSAs).
- OSPF sends updates (LSAs) when there is a change to one of its links, and will only send the change in the update. LSAs are additionally refreshed every 30 minutes.
- OSPF traffic is multicast either to address 224.0.0.5 (all OSPF routers) or 224.0.0.6 (all Designated Routers).
- OSPF uses the Dijkstra Shortest Path First algorithm to determine the shortest path.
- OSPF is a classless protocol, and thus supports VLSMs. Other characteristics of OSPF include:
- OSPF supports only IP routing.
- OSPF routes have an administrative distance is 110.
- OSPF uses cost as its metric, which is computed based on the bandwidth of the link.

OSPF has no hop-count limit. The OSPF process builds and maintains three separate tables:

- ➢ A neighbor table – contains a list of all neighboring routers.
- ➢ A topology table – contains a list of all possible routes to all known networks within an area.
- ➢ A routing table – contains the best route for each known network.

Link-State Algorithm:

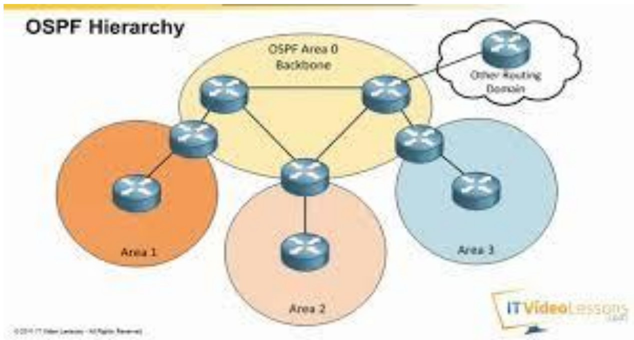
Just like any other Link state routing, OSPF also has the following features:

- ➢ **Advertise about neighborhood**: Instead of sending its entire routing table, a router sends information about its neighborhood only.

- **Flooding**: Each router sends this information to every other router on the internetwork, not just to its neighbors. It does so by a process of flooding. In Flooding, a router sends its information to all its neighbors (through all of its output ports). Every router sends such messages to each of its neighbor, and every router that receives the packet sends copies to its neighbor. Finally, every router has a copy of same information.
- **Active response**: Each outer sends out information about the neighbor when there is a change.

Initialization: When an SPF router is powered up, it initializes its routing-protocol data structures and then waits for indications from lower-layer protocols that its interfaces are functional.

After a router is assured that its interfaces are functioning, it uses the OSPF Hello protocol (sends greeting messages) to acquire neighbors, which are routers with interfaces to a common network. The router sends hello packets to its neighbors and receives their hello packets. These messages are also known as greeting messages. It then prepares an LSP (Link State packet) based on the results of this Hellow protocol.



An example of an internet is shown in Fig. 7.3.1, where R1 is a neighbor of R2 and R4, R2 is a neighbor of R1, R3 and R4, R3 is a neighbor of R2 and R4, R4 is a neighbor of R1, R2 and R3. So each router will send greeting messages to its entire neighbors.

Information from neighbors: A router gets its information about its neighbor by periodically sending them a short greeting packet (this is known as *Hello Message* format). If neighbor responds to this greeting message as expected, it is assumed to be alive and functioning. If it does not, a change is assumed to have occurred and the sending router then alerts the rest of the network in its next LSP, about this neighbor being down. These Greeting messages are small enough that they do not use network resources to a significant amount, unlike the routing table updates in case of a vector-distance algorithm.

Link state packet: The process of router flooding the network with information about its neighborhood is known as *Advertising*. The basis of advertising is a short packet called a *Link state Packet (LSP)*. An LSP usually contains 4 fields: the ID of the advertiser (Identifier of the router which advertises the message), ID of the destination network, The cost, and the ID of the neighbor router. Figure 7.3.2 shows the LSP of a router found after the *Hellow* protocol and Fig. 7.3.3 shows the basic fields of LSP.

Advertiser	Network	Cost	Neighbor
---------------	-------------	---------------	-------------
---------------	--------------	---------------	-------------

Figure 7.3.3 The LSP fields

Link State Database: Every router receives every LSP and then prepares a database, which represents a complete network topology. This Database is known as Link State Database. Figure 7.3.4 shows the database of our sample internetwork. These databases are also known as *topological database.*

Advertiser	Network	Cost	Neighbor
R1	A	4	R4
R1	B	1	R2
R2	B	2	R1
R2	C	5	R3
R3	C	1	R2
R3	D	3	R4
R4	A	1	R1
R4	D	2	R3

Figure 7.3.4 Link State Database

Because every router receives the same LSPs, every router builds the same database. Every router uses it to calculate its routing table. If a router is added or deleted from the system, the whole database must be changed accordingly in all routers.

Shortest Path calculation: After gathering the Link State database, each router applies an algorithm called the Dijkstra algorithm to calculate the shortest distance between any two nodes. The Dijkstra's algorithm calculates the shortest path between two points on a network using a graph made up of nodes and arcs, where nodes are the Routers and the network, while connection between router and network is refer to as arcs.

The algorithm begins to build a tree by identifying its root as shown in Fig. 7.3.5. The router is the root itself. The algorithm then attaches all other nodes that can be reached from that router; this is done with the help of the Link state database.

Figure1
Figure1
Figure2
Figure 7.3.5 Path calculation for router R1

From this shortest path calculation each router makes its routing table, as per our example internet table for router R1 is given in Fig. 7.3.6. All other routers too have a similar routing table made up after this point.

Network	Cost	Next Router
A	4	-----
B	1	-----
C	8	R2
D	7	R4

Figure 7.3.6 Routing table example

Routing Hierarchy in OSPF

Unlike RIP, OSPF can operate within a hierarchy. The largest entity within the hierarchy is the autonomous system (AS), which is a collection of networks under a common administration that share a common routing strategy. OSPF is an intra-AS (interior gateway) routing protocol, although it is capable of receiving routes from and sending routes to other ASs.

An AS can be divided into a number of *areas*, which are groups of contiguous networks and attached hosts. Routers with multiple interfaces can participate in multiple areas. These routers, which are called *Area Border Routers*, maintain separate topological databases for each area.

A topological database is essentially an overall picture of networks in relationship to routers. The topological database contains the collection of LSAs received from all routers in the same area. Because routers within the same area share the same information, they have identical topological databases. We have already seen how these topological databases are made in the previous section.

The term *domain* sometimes is used to describe a portion of the network in which all routers have identical topological databases. Domain is frequently used interchangeably with AS. An area's topology is invisible to entities outside the area. By keeping area topologies separate, OSPF passes less routing traffic than it would if the AS were not partitioned.

Area partitioning creates two different types of OSPF routing, depending on whether the source and the destination are in the same or different areas. Intra-area routing occurs when the source and destination are in the same area; inter-area routing occurs when they are in different areas.

An OSPF backbone is responsible for distributing routing information between areas. It consists of all Area Border Routers, networks not wholly contained in any area, and their attached routers. Figure 7.3.7 shows an example of an internet with several areas. In the Fig. 7.3.7, routers 9, 10, 11, 12 and 13 make up the backbone. If host H1 in Area 3 wants to send a packet to host H2 in Area 1, the packet is sent to Router 4, which then forwards the packet along the backbone to Area Border Router 12, which sends the packet to Router 11, and Router 11 forwards it to Router 10. Router 10 then sends the packet through an intra-area router (Router 3) to be forwarded to Host H2.

The backbone itself is an OSPF area, so all backbone routers use the same procedures and algorithms to maintain routing information within the backbone that any area router would. The backbone topology is invisible to all intra-area routers, as are individual area topologies to the backbone. Areas can be defined in such a way that the backbone is not contiguous. In this case, backbone connectivity must be restored through virtual links. Virtual links are configured between any backbone routers that share a link to a nonbackbone area and function as if they were direct links.

Figure

Different areas in an Autonomous system

OSPF Message Format

In this section we will discuss various message formats used by OSPF, first we will see fixed header, which is common to all messages and then we will look at various variable part, different for different messages used in OSPF.

Fixed Header: All OSPF packets begin with a 24-byte header, as illustrated in Figure 7.3.8. Summary of the functions of different fields are given below:

□ **Version number**—Identifies the OSPF version used.

□ **Type**—Identifies the OSPF packet type as one of the following: o **Hello**—Establishes and maintains neighbor relationships.

o **Database description**—Describes the contents of the topological database. These messages are exchanged when an adjacency is initialized.

o **Link-state request**—Requests pieces of the topological database from neighbor routers. These messages are exchanged after a router discovers (by examining database-description packets) that parts of its topological database are outdated.

QQQ **Link-state update**—Responds to a link-state request packet. These messages also are used for the regular dispersal of LSAs. Several LSAs can be included within a single link-state update packet.

□ **Link-state acknowledgment**—Acknowledges link-state updates packets.

QQQ **Message length**—Specifies the packet length, including the OSPF header, in bytes.

RRR **Source Router IP address**—Identifies the source of the packet.

SSS **Area ID**—Identifies the area to which the packet belongs. All OSPF packets are associated with a single area.

TTT **Checksum**—Checks the entire packet contents for any damage suffered in transit.

UUU **Authentication type**—Contains the authentication type. All OSPF protocol exchanges are authenticated. The authentication type is configurable on per-area basis.

VVV **Authentication**—Contains authentication information.

WWW **Data**—Contains encapsulated upper-layer information.

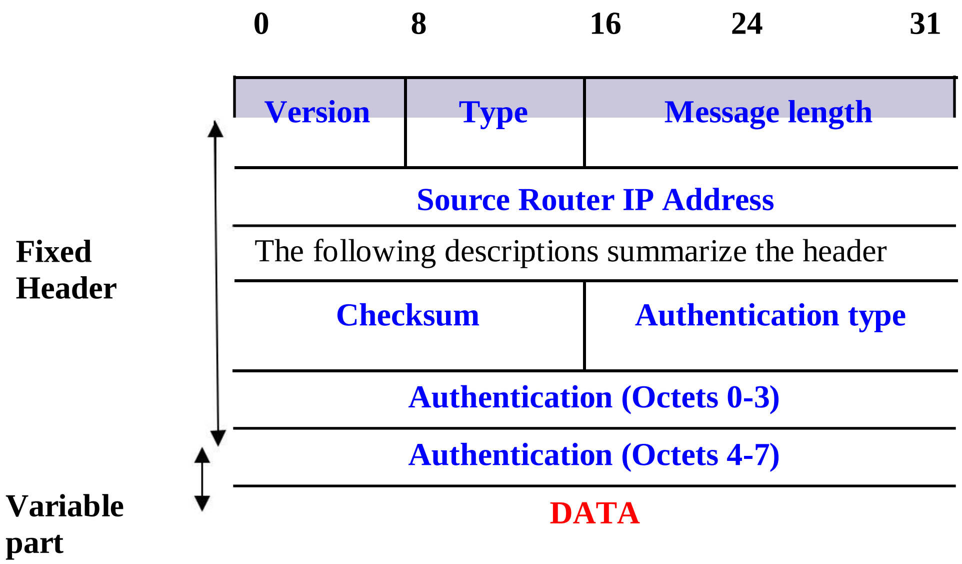


Figure 7.3.8 24-Octet OSPF Message Header

Hellow Message Format: OSPF sends Hellow (greeting) messages on each link periodically to establish and test neighbor reachability. The ormat of this message is shown in Figure 7.3.9. Functions of the header fields are briefly explained below.

p **Fixed Header**: as discussed in previous section and Fig. 7.3.8

q **Network mask**: contains mask for the network over which the message is to be send.

r **Dead Timer**: gives time in seconds after which a non-responding neighbor is considered dead.

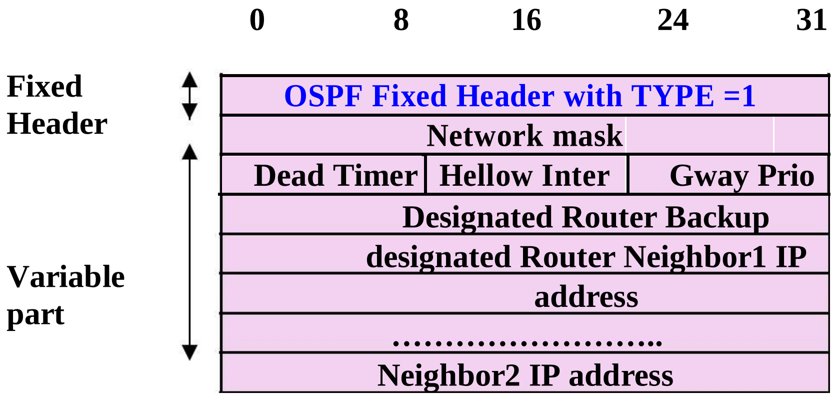
s **Hellow Inter**: means Hellow Interval, it is the normal period, in seconds, between hello messages.

t **Gway Prio**: means gateway priority, it is the interior priority of this router, and is used in selecting the backup designated router.

u **Designated Router**: IP address of the router, which is the designated router for the network as viewed by the sender.

□ **Backup Designated Router**: IP address of the router, which is the Backup designated router for the network as viewed by the sender.

Neighbor IP Address: IP address of all the neighbors from which the sender has recently received Hellow Messages.



0 8 16 24 31

Fixed Header

Variable part

OSPF Fixed Header with TYPE =1

Network mask

Dead Timer | Hellow Inter | Gway Prio

Designated Router Backup

designated Router Neighbor1 IP

address

……………………..

Neighbor2 IP address

Figure 7.3.9 OSPF Hellow Message Format

Database Description message Format: These messages are exchanged by routers to initialize their network topology database. In this exchange one router serves as a master, while other as a slave. The slave acknowledges each database description message with a response. This message is further divided into several messages using **I** and **M** bits. The functions of different fields, as shown in Fig. 7.3.10, are summarized below:

➢ **Fixed Header**: as discussed in previous section and Fig. 7.3.8

➢ **I, M, S bits**: Bit **I** is set to **1** if additional message follows. Bit **S** indicates whether a message was sent by a **master (1)** or a **slave (0).**

➢ **Database Sequence Number**: this is used to sequence the messages so that the receiver can detect if any of the message is missing. Initial message contains a random sequence number **R**; subsequent messages contain sequential integers starting from **R**.

- **Link Type**: describes one link in network topology; it is repeated for each link. Different possible values for Link Type is as follows:

Link Type	Meaning
1	Router Link
2	Network Link
3	Summary Link (IP Network)
4	Summary Link (Link to Border Router)
5	External Link (Link to another site)

1. **Link ID:** gives an identification of the Link, generally an IP address.
2. **Advertising Router**: specifics the router which is advertising this link.

1. **Link sequence Number**: integer to ensure that messages are not mixed or received out of order.
2. **Link Checksum**: Checksum to ensure that the information has not been corrupted.

Link Age: Helps order messages, gives the time (in seconds) since link was established.

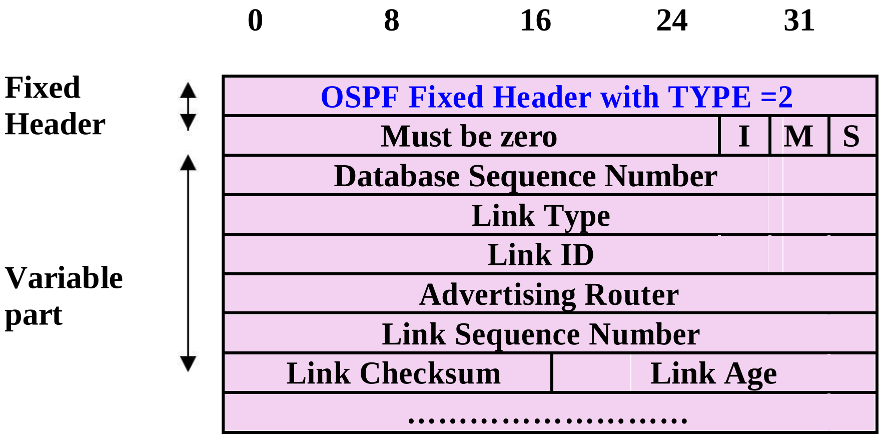


Figure 7.3.10 OSPF Database Description Message Format

Link Status Request Message: After exchanging Database Description message, router may discover that some part of its database is outdated. Link Status message is used to request the neighbor to supply the updated information. The message lists specific links, as shown in Figure 7.3.11. The neighbor responds with the most current information it has about those links. The three fields as shown are repeated for each link, about which status is requested. More than one request message is required if list is long. All the fields have usual meaning as discussed in previous message format.

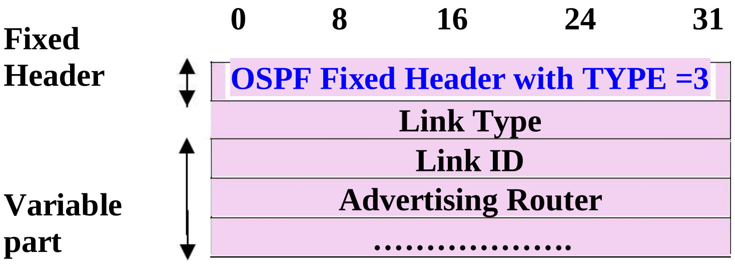

0
8
16
24
31
Fixed
Header
OSPF Fixed Header with TYPE =3
Link Type
Link ID
Advertising Router
……………….
Variable
part

Figure 7.3.11 OSPF Link Status Request Message Format

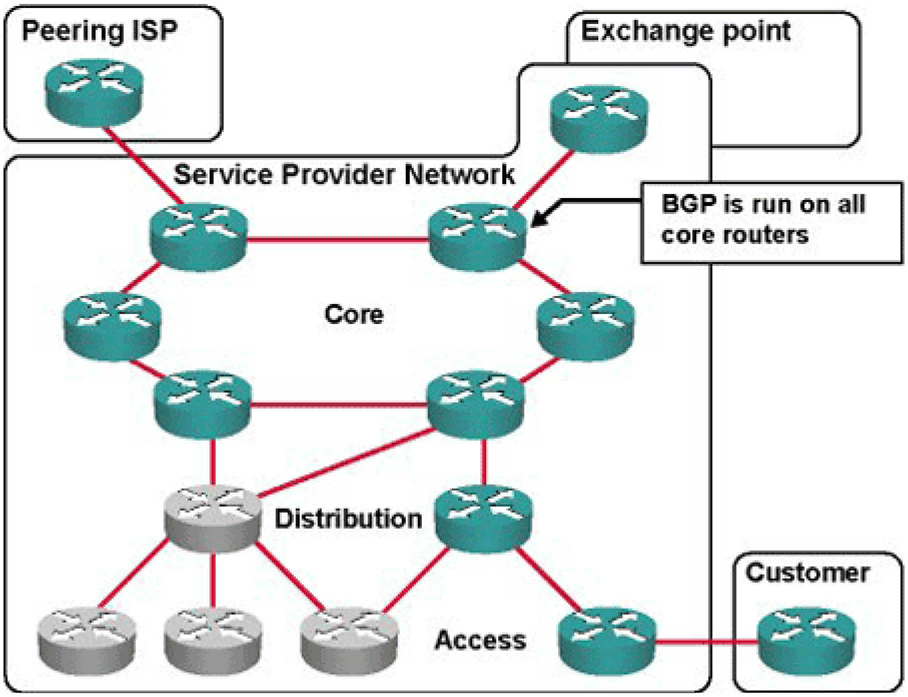
Border Gateway Protocol:

The Border Gateway Protocol (BGP) is the protocol used throughout the Internet to exchange routing information between networks. It is the language spoken by routers on the Internet to determine how packets can be sent from one router to another to reach their final destination. BGP has worked extremely well and continues to the be protocol that makes the Internet work.

The challenge with BGP is that the protocol does not directly include security mechanisms and is based largely on trust between network operators that they will secure their systems correctly and not send incorrect data. Mistakes happen, though, and problems could arise if malicious attackers were to try to affect the routing tables used by BGP.

What is BGP used for?

BGP offers network stability that guarantees routers can quickly adapt to send packets through another reconnection if one internet path goes down. BGP makes routing decisions based on paths, rules or network policies configured by a network administrator. Each BGP router maintains a standard routing tableused to direct packets in transit. This table is used in conjunction with a separate routing table, known as the routing information base (RIB), which is a data table stored on a server on the BGP router. The RIB contains route information both from directly connected external peers, as well as internal peers, and continually updates the routing table as changes occur. BGP is based on TCP/IP and uses client-server topology to communicate routing information, with the client-server initiating a BGP session by sending a request to the server.



BGP makes best-path decisions based on current reachability, hop counts and other path characteristics. In situations where multiple paths are available -- as within a major hosting facility -- BGP can be used to communicate an organization's own preferences in terms of what path traffic should follow in and out of its networks. BGP even has a mechanism for defining arbitrary tags, called communities, which can be used to control route advertisement behavior by mutual agreement among peers.

BGP features:

- The current version of BGP is BGP version 4, based on RFC4271.
- BGP is the path-vector protocol that provides routing information for autonomous systems on the Internet via its AS-Path attribute.
- BGP is a Layer 4 protocol that sits on top of TCP. It is much simpler than OSPF, because it doesn't have to worry about the things TCP will handle.
- Peers that have been manually configured to exchange routing information will form a TCP connection and begin speaking BGP. There is no discovery in BGP.
- Medium-sized businesses usually get into BGP for the purpose of true multi-homing for their entire network.
- An important aspect of BGP is that the AS-Path itself is an anti-loop mechanism. Routers will not import any routes that contain themselves in the AS-Path.

Autonomous systems:

First a little terminology. In the world of BGP, each routing domain is known as an autonomous system, or AS. What BGP does is help choose a path through the Internet, usually by selecting a route that traverses the least number of autonomous systems: the shortest AS path.

You might need BGP, for example, if your corporate network is connected to two large ISPs. To use BGP you would need an AS number, which you can get from the American Registry of Internet Numbers (ARIN).

Once BGP is enabled, your router will pull a list of Internet routes from your BGP neighbors, who in this case will be your two ISPS. It will then scrutinize them to find the routes with the shortest AS paths. These will be put into the router's routing table. (If you only connect to a single ISP then you don't need BGP. That's because there's only one path to the Internet, so there's no need for a routing protocol to select the best path.)

Generally, but not always, routers will choose the shortest path to an AS. BGP only knows about these paths based on updates it receives.

Route updates:

- ✓ Unlike Routing Information Protocol (RIP), a distance-vector routing protocol which employs the hop count as a routing metric, BGP does not broadcast its entire routing table. At boot, your peer will hand over its entire table. After that, everything relies on updates received. Route updates are stored in a Routing Information Base (RIB).
- ✓ A routing table will only store one route per destination, but the RIB usually contains multiple paths to a destination. It is up to the router to decide which routes will make it into the routing table, and therefore which paths will actually be used. In the event that a route is withdrawn, another route to the same place can be taken from the RIB.
- ✓ The RIB is only used to keep track of routes that could possibly be used. If a route withdrawal is received and it only existed in the RIB, it is silently deleted from the RIB. No update is sent to peers. RIB entries never time out. They continue to exist until it is assumed that the route is no longer valid.

BGP path attributes:

- ✓ In many cases, there will be multiple routes to the same destination. BGP therefore uses path attributes to decide how to route traffic to specific networks.
- ✓ The easiest of these to understand is Shortest AS_Path. What this means is the path which traverses the least number of AS "wins."
- ✓ Another important attribute is Multi_Exit_Disc (Multi-exit discriminator, or MED). This makes it possible to tell a remote AS that if there are multiple exit points on to your network, a specific exit point is preferred.

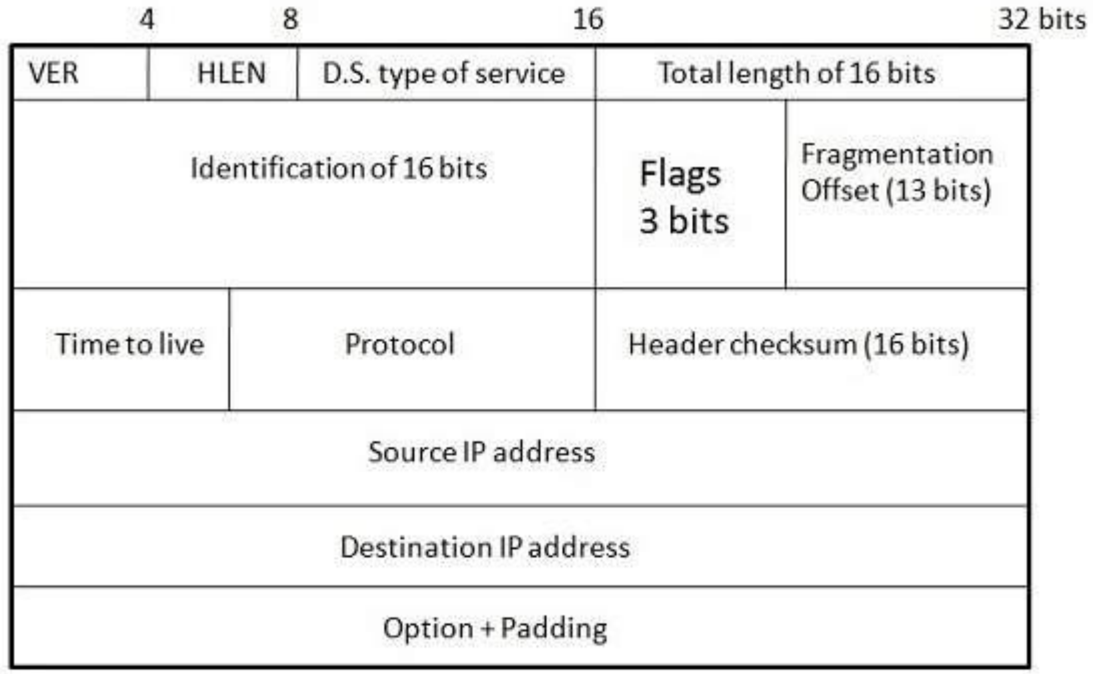
The Origin attribute specifies the origin of a routing update. If BGP has multiple routes, then origin is one of the factors in determining the preferred route.

Internet Protocol:

Internet Protocol is **connectionless** and **unreliable** protocol. It ensures no guarantee of successfully transmission of data.

In order to make it reliable, it must be paired with reliable protocol such as TCP at the transport layer.

Internet protocol transmits the data in form of a datagram as shown in the following diagram:



4	8	16		32 bits
VER	HLEN	D.S. type of service	Total length of 16 bits	
Identification of 16 bits			Flags 3 bits	Fragmentation Offset (13 bits)
Time to live	Protocol		Header checksum (16 bits)	
Source IP address				
Destination IP address				
Option + Padding				

Points to remember:

- The length of datagram is variable.
- The Datagram is divided into two parts: **header** and **data.**

- The length of header is 20 to 60 bytes.
- The header contains information for routing and delivery of the packet.

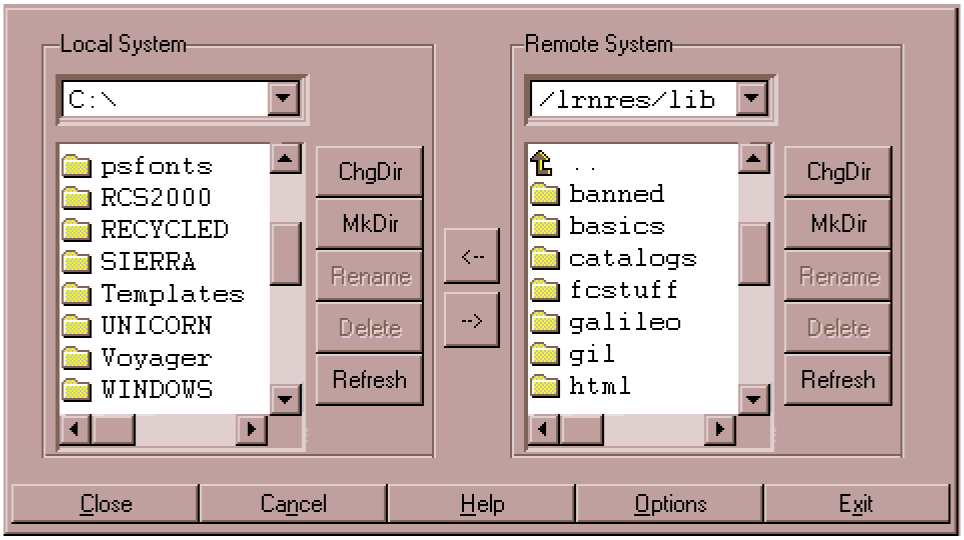
Several protocols are used on the Internet, including Electronic Mail (e-mail), File Transfer Protocol (FTP), HTTP (World Wide Web), News (or Usenet), Gopher and Telnet. Each of these has its own standard and usage.

Electronic Mail

Included in the email protocol are three distinct protocols. SMTP (Simple Mail Transfer Protocol), IMAP (Internet Message Access Protocol) and POP3 (Post Office Protocol 3).
SMTP is a protocol used for sending mail, while IMAP and POP3 are used for receiving. Almost all Internet service providers support all three protocols. However the most popular setup for most providers is to use SMTP for sending mail while using POP3 for receiving.

File Transfer Protocol

File Transfer Protocol, or FTP, is a means of transferring a file from one computer to another. FTP is commonly used for uploading a web page to a web server so that it may be seen on the World Wide Web. A special program, called a client, is usually needed to use FTP.



```
C:\
psfonts
RCS2000
RECYCLED
SIERRA
Templates
UNICORN
Voyager
WINDOWS
```

```
/lrnres/lib
..
banned
basics
catalogs
fcstuff
galileo
gil
html
```

Figure: File Transfer Protocol

HTTP (World Wide Web)

Hyper Text Transfer Protocol, or HTTP, is the protocol used by web server to allow web pages to be shown in a web browser. If you look up into the address bar of your web browser, the place where you type in the address that you want to visit, it has the prefix "http://" in front of the address. Because most web browsers are capable of FTP as well as viewing web pages, the http tells the browser what kind of information to expect.

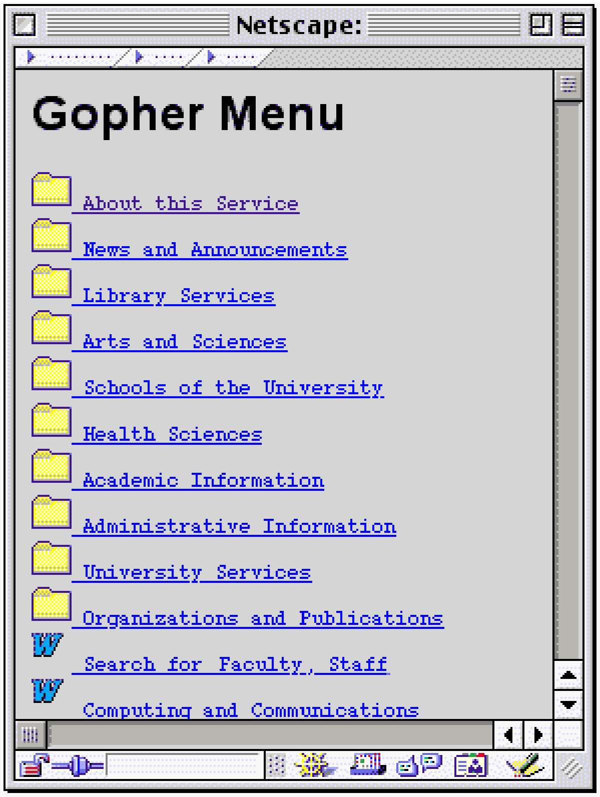
News (or Usenet)

Network News Transfer Protocol (NNTP) is used for serving Usenet posts Usenet is similar to the forums that many web sites have. Usenet has forums that are dedicated to specific companies as well as forums that have a wide range of topics. Usenet is divided into several areas. Some of the forums that are included in Usenet are comp. for discussion of computer-related topics, sci. for discussion of scientific subjects, rec. for discussion of recreational activities (e.g. games and hobbies) and talk. for discussion of contentious issues such as religion and politics.

When most people talk about "the Internet" what they are really referring to is the World Wide Web. The Internet is actually composed of many different components. Some of the components are widely known, such as FTP, while others are not so familiar, such as Gopher and Telnet.

Gopher

Another tool of the Internet is Gopher, a menu-based program that enables you to browse for information without knowing where the material is located. It lets you search a list of resources and then sends the material to you.



Gopher Menu

- About this Service
- News and Announcements
- Library Services
- Arts and Sciences
- Schools of the University
- Health Sciences
- Academic Information
- Administrative Information
- University Services
- Organizations and Publications
- Search for Faculty, Staff
- Computing and Communications

Telnet: Telnet lets you log in to a remote computer just as you would if you were there. So any commands that you would be able to run from the remote computer if you were sitting in front of it, you would be able to run from the computer you logged in from.

Internet control message protocol: ICMP is short for **I***nternet* **C***ontrol* **M***essage* **P***rotocol*. It is an extension to the Internet Protocol (IP) defined by RFC 792. ICMP supports packets containing error, control, and informational messages. It is an error reporting protocol and is used by routers, hosts and network devices to generate error messages when there are problems delivering IP packets.

ICMP communicate control data, information data, and error recovery data across the network. Problems that are less severe than transmission errors result in error conditions that can be reported. For example, suppose some of the physical paths in Internet fail causing the Internet to be partitioned into two sets of networks with no path between the sets. A datagram sent from a host in one set to a host in other cannot be delivered.

The TCP/IP suite includes a protocol called ICMP that IP uses to send error messages when condition such as the one described above arises. The protocol is required for a standard implementation of IP. We will see that the two protocols are co-dependent. IP uses ICMP when it sends an error message, and ICMP uses IP to transport messages.

Following is a brief description of some of the error messages defined by ICMP protocol:

1. **Source Quench** A router or host whose receive communication buffers are nearly full normally triggers this message. A source quench message is sent to the sending host, the receiver is simply requesting the sending host to reduce the rate at which it is transmitting until advised otherwise.
2. **Time Exceeded** A time-exceeded message is sent in two cases. Whenever a router reduces the TIL field in a data gram to zero. The router discards the datagram and sends a time exceeded message. In addition, a time exceeded message is sent by a host if the reassembly timer expires before all fragments from a given datagram arrive,
3. **Route Redirect** A router sends this message to a host that is requesting its routing services. When a host creates a datagram destined for a network, the host sends the datagram to a router, which forwards the datagram to its destination. If a router determines that a host has incorrectly sent a datagram that should be sent to a different router, the router uses route redirect message to cause the host to change its route. In this manner, a route redirect message improves the efficiency of the routing process by informing the req4esting host of a shorter path to the desired destination.
4. **Host Unreachable** Whenever a gateway or a router determines that a datagram cannot be delivered to its final destination (due to link failure or bandwidth congestion), an ICMP host unreachable message is sent to the originating node on the network. Normally, the message includes the reason the host cannot be reached.
5. **Fragmentation and Reassembly** The largest datagram the IP protocol can handle is 64 Kbytes. The maximum datagram size is dictated by the width of the total length field in the IP header. Realistically, most underlying data link technologies cannot accommodate this data size. For example, the maximum size of the data frame supported by Ethernet is 1,514 bytes. Unless something is done about situations like this, IP has to discard data that is delivered to it from upper layer protocols with sizes exceeding the maximum tolerable size by the data link layer. To circumvent this difficulty, IP is built to provide data fragmentation and reassembly.

   Whenever an upper-layer protocol delivers data segments whose sizes exceed the limit allowed by the underlying network, IP breaks the data into smaller pieces that are manageable within the allowed limit. The small data grams are then sent to the target host, which reassembles them for subsequent delivery to an upper-layer protocol.

   Although all data fragments are normally delivered using the same route, in some situations a few of them might traverse alternative routes. Fragments following different routes, however, stand the chance of reaching their-destination out of the order in which they were sent. To allow for recovery from such behaviour, IP employs the fragmentation-offset field in its header. The fragmentation-offset field includes sequencing information that the remote IP host uses to recover the sequence in which the data grams were sent. IP also uses the information in the fragmentation offset field to detect missing fragments~ Data is not passed to the protocol described in the protocol field unless all related fragments are duly received and reordered. This process of fragment recovery and re-sequencing is called data reassembly.

   Fragments belonging to two or more independent large data can be differentiated by IP using identification field. Fragments belonging to the same datagram are uniquely assigned in the identification field. The receiving 110stuses this number to recover the IP fragments to their respective data grams.

   A host that creates a datagram can set a bit in the flag field to specify the fragmentation. Among other bits, the flag field includes a more fragments bit, which is set to I in all fragments belonging to a datagram except for the final fragment. This ensures about the receiving of all fragments of a datagram.

1. **Echo request/Echo reply** - These two ICMP messages are exchanged between ICMP software on any two hosts in a bid to check connectivity between them. The ping command is an example of a diagnostic command commonly used by network users to check for the reach ability of a certain host. Whenever ping is invoked at the command line, ICMP echo request message is sent to the target host. If the target host is operational and connected to the network, it responds with an echo reply message as proof of reach ability. In other words, the reply carries the same data as the request.
2. **Address Mask Request/Reply** A host broadcasts an address mask request when it boots, and routers that receive the request send an address mask reply that contains the correct 2-bit subnet mask being used on the network.

ICMP Message Transport

ICMP uses IP to transport each error message. When a router has an ICMP message to send, it creates datagram and encapsulates the ICMP message in the datagram. It means that the ICMP message placed in the data area of the IP data gram. The· datagram is forwarded as usual with the complete data gram being encapsulated in a frame for transmission.

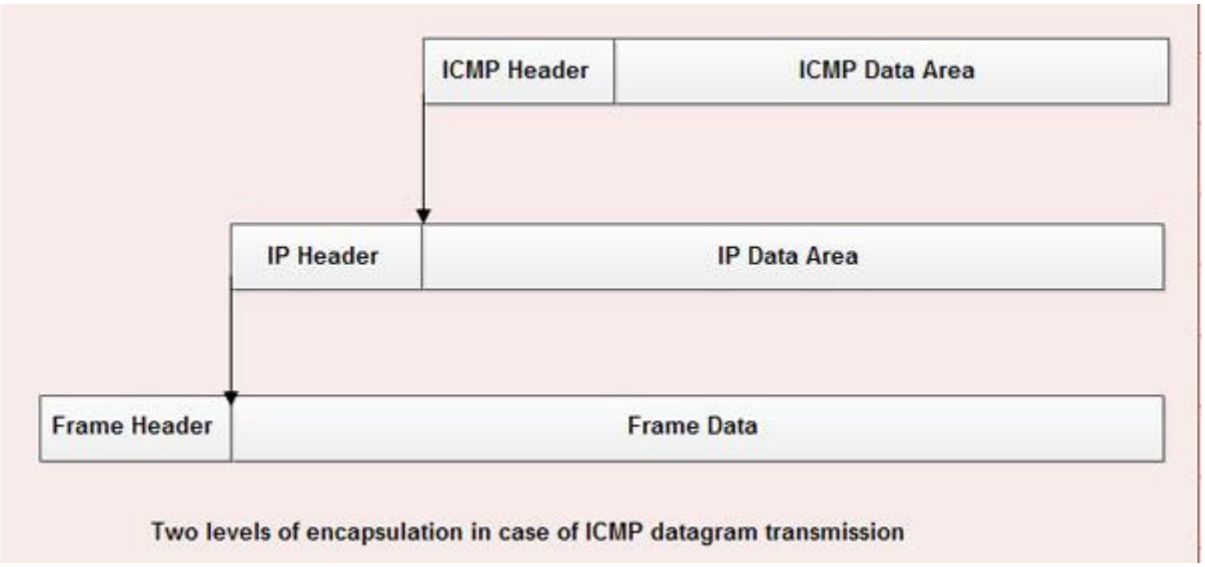
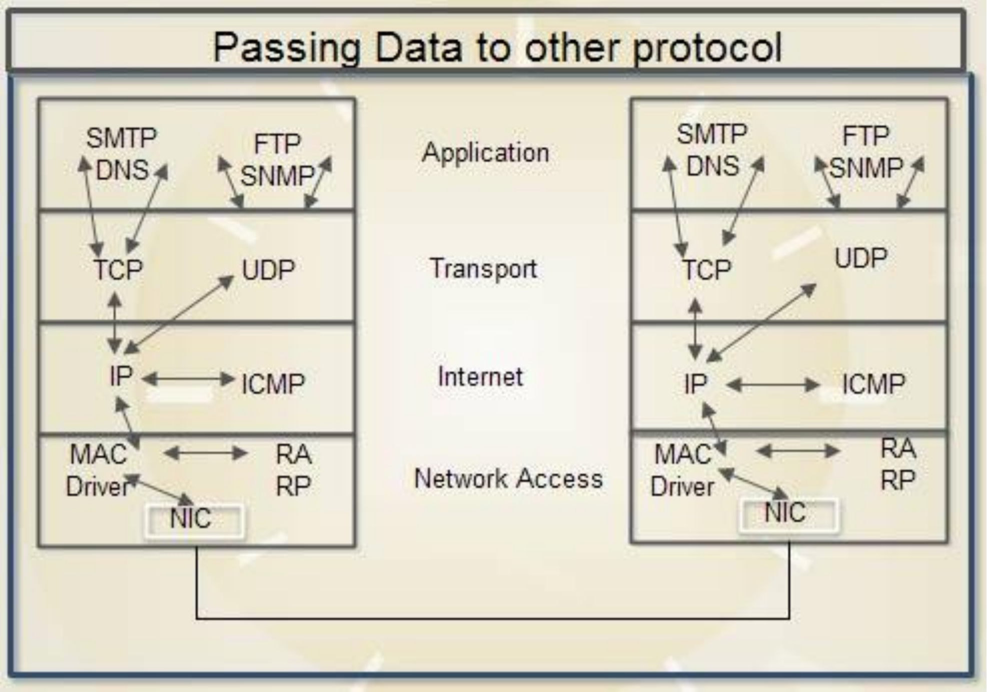


Figure: Passing Data to Other Protocols

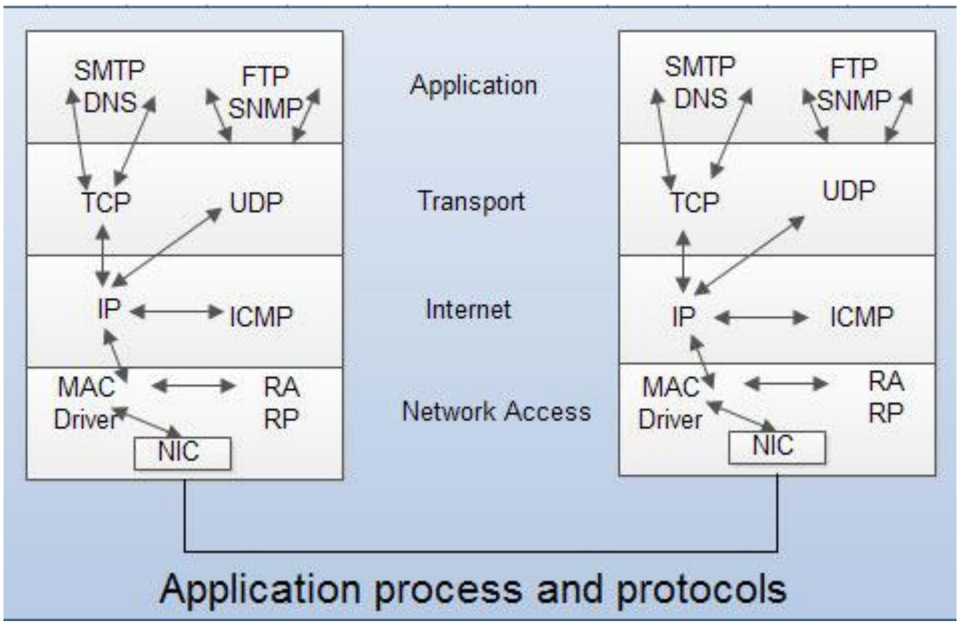


As we know from the previous section that all *TCP/IP* protocols send their data in IP datagrams. Illustrates how data is exchanged across protocol boundaries. A protocol field that is included in IP header assists IP in passing data to other protocols. By TCP/I*P* standards, each protocol that uses IP routing services is assigned a protocol identification number. Setting the protocol field to 6, for example, designates the data as belonging to the TCP protocol, whereas it designates the ICMP protocol. A protocol number of 0 designates the IP protocol, in which case encapsulated data is processed by IP itself.

The Transport or Host-to-Host layer

The host-to-host layer, also known as the Transport layer, corresponds to the transport layer of the OSI model. Protocols defined at this layer accept data from application protocols running at the application layer. Protocols encapsulate it in the protocol header, and deliver the data segment thus formed to the lower IP layer for subsequent routing. Unlike the IP protocol, however, the transport layer is aware of the identity of the final user representative process. As such, the transport layer, in the TCP/I*P* suite, embodies what data communications are all about the delivery of information from an application on one computer to an application on another computer.

At the transport layer, *TCP/IP* defines two transport protocols namely User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). Applications can use one protocol or the other depending on the type of service they desire. To associate data with applications, TCP and UDP include Source and destination port number fields in their headers. These fields are used in much the same way as IP uses the protocol field for, the internal routing of data to applications utilizing their services.



sparingly, the last IPv4 addresses are expected to be assigned by ICANN before the end of 2012. This looming disaster was recognized almost two decades ago, and it sparked a great deal of discussion and controversy within the Internet community about what to do about it. In this section, we will describe both the problem and several proposed solutions. The only long-term solution is to move to larger addresses. **IPv6** (**IP version 6**) is a replacement design that does just that. It uses 128-bit addresses; a shortage of these addresses is not likely any time in the foreseeable future. However, IPv6 has proved very difficult to deploy. It is a different network layer protocol that does not really interwork with IPv4, despite many similarities. Also, companies and users are not really sure why they should want IPv6 in any case. The result is that IPv6 is deployed and used on only a tiny fraction of the Internet (estimates are 1%) despite having been an Internet Standard since 1998. The next several years will be an interesting time, as the few remaining IPv4 addresses are allocated. Will people start to auction off their IPv4 addresses on eBay? Will a black market in them spring up? Who knows? In addition to the address problems, other issues loom in the background. In its early years, the Internet was largely used by universities, high-tech industries, and the U.S. Government (especially the Dept. of Defense). With the explosion of interest in the Internet starting in the mid-1990s, it began to be used by a different group of people, often with different requirements. For one thing, numerous people with smart phones use it to keep in contact with their home bases. For another, with the impending convergence of the computer, communication, and entertainment industries, it may not be that long before every telephone and television set in the world is an Internet node, resulting in a billion machines being used for audio and video on demand. Under these circumstances, it became apparent that IP had to evolve and become more flexible. Seeing these problems on the horizon, in 1990 IETF started work on a new version of IP, one that would never run out of addresses, would solve a variety of other problems, and be more flexible and efficient as well. Its major goals were:

10. Support billions of hosts, even with inefficient addressallocation.
11. Reduce the size of the routing tables.
12. Simplify the protocol, to allow routers to process packets faster.
13. Provide better security (authentication and privacy).
14. Pay more attention to the type of service, particularly for real-time data.

15. Aid multicasting by allowing scopes to be specified.
16. Make it possible for a host to roam without changing its address.
17. Allow the protocol to evolve in the future.
18. Permit the old and new protocols to coexist for years.

The design of IPv6 presented a major opportunity to improve all of the features in IPv4 that fall short of what is now wanted. To develop a protocol that met all these requirements, IETF issued a call for proposals and discussion in RFC 1550. Twenty-one responses were initially received. By December 1992, seven serious proposals were on the table. They ranged from making minor patches to IP, to throwing it out altogether and replacing it with a completely different protocol. One proposal was to run TCP over CLNP, the network layer protocol designed for OSI. With its 160-bit addresses, CLNP would have provided enough address space forever as it could give every molecule of water in the oceans enough addresses (roughly 25) to set up a small network. This choice would also have unified two major network layer protocols. However, many people felt that this would have been an admission that something in the OSI world was actually done right, a statement considered Politically Incorrect in Internet circles. CLNP was patterned closely on IP, so the two are not really that different. In fact, the protocol ultimately chosen differs from
IP far more than CLNP does. Another strike against CLNP was its poor support for service types, something required to transmit multimedia efficiently.

Three of the better proposals were published in *IEEE Network* (Deering, 1993; Francis, 1993; and Katz and Ford, 1993). After much discussion, revision, and jockeying for position, a modified combined version of the Deering and Francis proposals, by now called **SIPP** (**Simple Internet Protocol Plus**) was selected and given the designation **IPv6.** IPv6 meets IETF's goals fairly well. It maintains the good features of IP, discards or deemphasizes the bad ones, and adds new ones where needed. In general, IPv6 is not compatible with IPv4, but it is compatible with the other auxiliary Internet protocols, including TCP, UDP, ICMP, IGMP, OSPF, BGP, and DNS, with small modifications being required to deal with longer addresses. The main features of IPv6 are discussed below. More information about it can be found in RFCs 2460 through 2466.First and foremost, IPv6 has longer addresses than IPv4. They are 128 bits long, which solves the problem that IPv6 set out to solve: providing an effectively unlimited supply of Internet addresses. We will have more to say about addresses shortly. The second major improvement of IPv6 is the simplification of the header. It contains only seven fields (versus 13 in IPv4). This change allows routers to process packets faster and thus improves throughput and delay. We will discuss the header shortly, too. The third major improvement is better support for options. This change was essential with the new header because fields that previously were required are now optional (because they are not used so often). In addition, the way options are represented is different, making it simple for routers to skip over options not intended for them. This feature speeds up packet processing time.

A fourth area in which IPv6 represents a big advance is in security. IETF had its fill of newspaper stories about precocious 12-year-olds using their personal computers to break into banks and military bases all over the Internet. There was a strong feeling that something had to be done to improve security. Authentication and privacy are key features of the new IP. These were later retrofitted to IPv4, however, so in the area of security the differences are not so great any more. Finally, more attention has been paid to quality of service. Various halfhearted efforts to improve QoS have been made in the past, but now, with the growth of multimedia on the Internet, the sense of urgency is greater.

UNIT IV
THE TRANSPORT LAYER

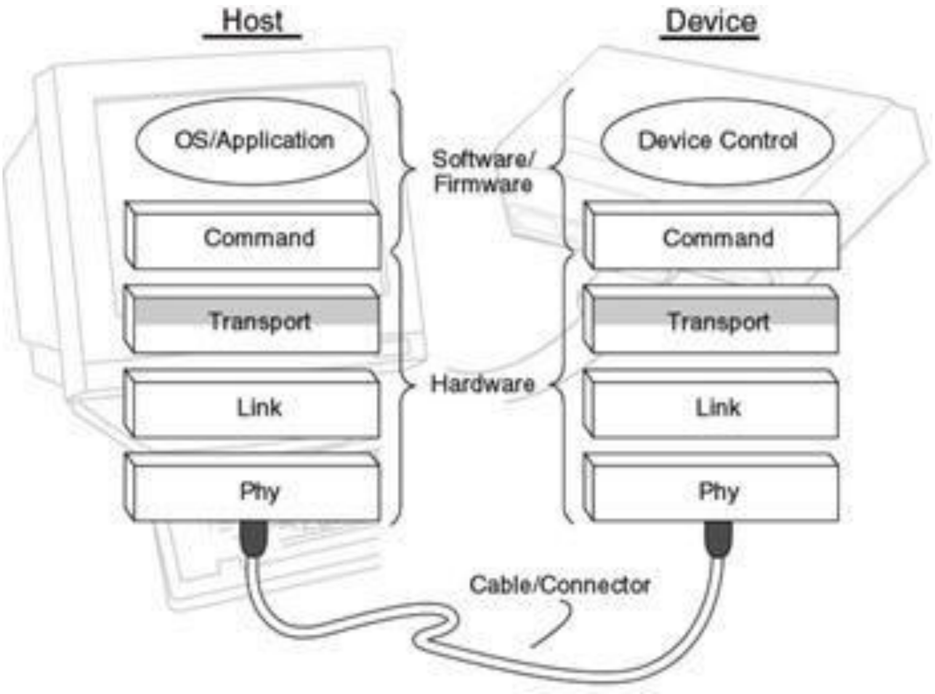
The Transport Service:

Overview of Transport layer:

In computer networking, a **transport layer** provides end-to-end or host-to-host communication services for applications within a layered architecture of network components and protocols. The transport layer provides services such as connection-oriented data stream support, reliability, flow control, and multiplexing.

Transport layer implementations are contained in both the TCP/IP model (RFC 1122), which is the foundation of the Internet, and the Open Systems Interconnection (OSI) model of general networking, however, the definitions of details of the transport layer are different in these models. In the Open Systems Interconnection model the transport layer is most often referred to as **Layer 4** or **L4**.

The best-known transport protocol is the Transmission Control Protocol (TCP). It lent its name to the title of the entire Internet Protocol Suite, TCP/IP. It is used for connection-oriented transmissions, whereas the connectionless User Datagram Protocol (UDP) is used for simpler messaging transmissions. TCP is the more complex protocol, due to its stateful design incorporating reliable transmission and data stream services. Other prominent protocols in this group are the Datagram Congestion Control Protocol (DCCP) and the Stream Control Transmission Protocol (SCTP).



Services

Transport layer services are conveyed to an application via a programming interface to the transport layer protocols. The services may include the following features:

- Connection-oriented communication: It is normally easier for an application to interpret a connection as a data stream rather than having to deal with the underlying connection-less models, such as the datagram model of the User Datagram Protocol (UDP) and of the Internet Protocol (IP).
- Same order delivery: The network layer doesn't generally guarantee that packets of data will arrive in the same order that they were sent, but often this is a desirable feature. This is usually done through the use of segment numbering, with the receiver passing them to the application in order. This can cause head-of-line blocking.
- Reliability: Packets may be lost during transport due to network congestion and errors. By means of an error detection code, such as a checksum, the transport protocol may check that the data is not corrupted, and verify correct receipt by sending an ACK or NACK message to the sender. Automatic repeat request schemes may be used to retransmit lost or corrupted data.
- Flow control: The rate of data transmission between two nodes must sometimes be managed to prevent a fast sender from transmitting more data than can be supported by the receiving data buffer, causing a buffer overrun. This can also be used to improve efficiency by reducing buffer underrun.
- Congestion avoidance: Congestion control can control traffic entry into a telecommunications network, so as to avoid congestive collapse by attempting to avoid oversubscription of any of the processing or link capabilities of the intermediate nodes and networks and taking resource reducing steps, such as reducing the rate of sending packets. For example, automatic repeat requests may keep the network in a congested state; this situation can be avoided by adding congestion avoidance to the flow control, including slow-start. This keeps the bandwidth consumption at a low level in the beginning of the transmission, or after packet retransmission.
- Multiplexing: Ports can provide multiple endpoints on a single node. For example, the name on a postal address is a kind of multiplexing, and distinguishes between different recipients of the same location. Computer applications will each listen for information on their own ports, which enables the use of more than one network service at the same time. It is part of the transport layer in the TCP/IP model, but of the session layer in the OSI model.
- A transport layer protocol provides for **logical communication** between application processes running on different hosts. By "logical" communication, we mean that although the communicating application processes are not physically connected to each other (indeed, they may be on different sides of the planet, connected via numerous routers and a wide range of link types), from the applications' viewpoint, it is as if they were physically connected. Application processes use the logical communication provided by the transport layer to send messages to each other, free for the worry of the details of the

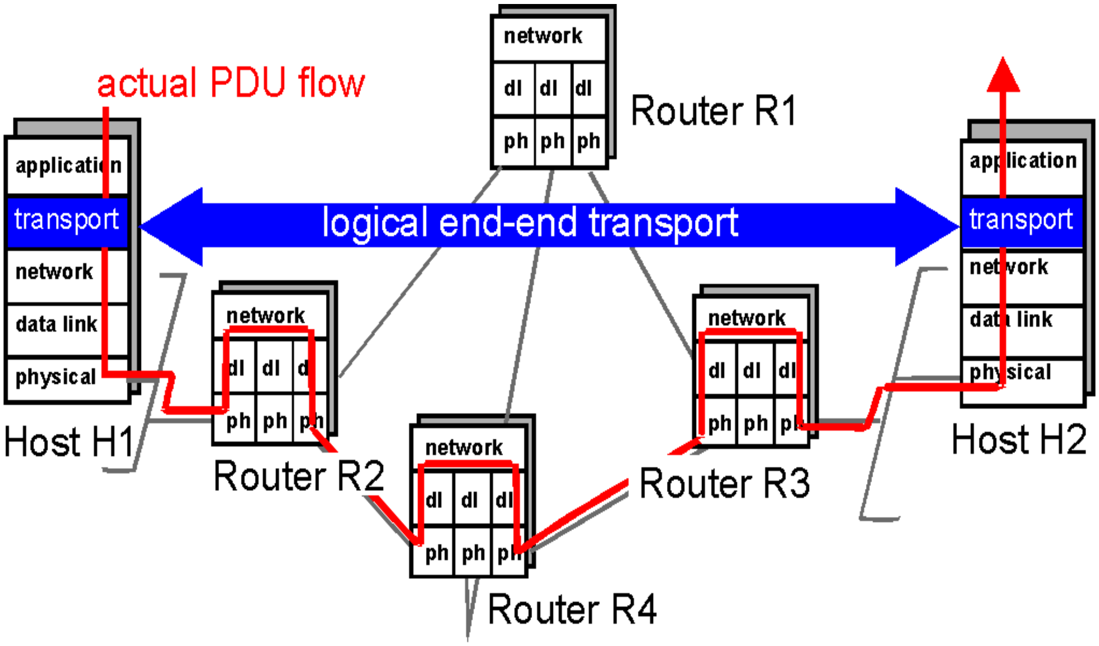
physical infrastructure used to carry these messages. Figure 3.1-1 illustrates the notion of logical communication.

- Transport layer protocols are implemented in the end systems but not in network routers. Network routers only act on the network-layer fields of the layer-3 PDUs; they do not act on the transport-layer fields.

- At the sending side, the transport layer converts the messages it receives from a sending application process into 4-PDUs (that is, transport-layer protocol data units). This is done by (possibly) breaking the application messages into smaller chunks and adding a transport-layer header to each chunk to create 4-PDUs. The transport layer then passes the 4-PDUs to the network layer, where each 4-PDU is encapsulated into a 3-PDU. At the receiving side, the transport layer receives the 4-PDUs from the network layer, removes the transport header from the 4-PDUs, reassembles the messages and passes them to a receiving application process.

- A computer network can make more than one transport layer protocol available to network applications. For example, the Internet has two protocols -- TCP and UDP. Each of these protocols provides a different set of transport layer services to the invoking application.

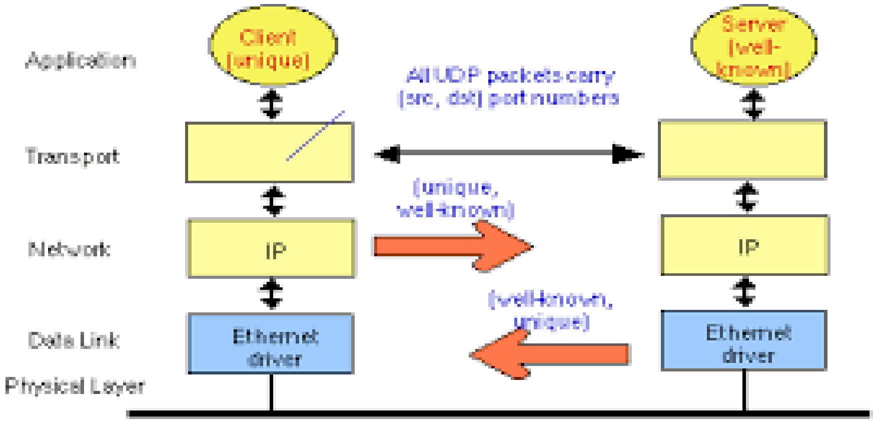
- All transport layer protocols provide an application multiplexing/demultiplexing service. This service will be described in detail in the next section. As discussed in Section 2.1, in addition to multiplexing/demultiplexing service, a transport protocol can possibly provide other services to invoking applications, including reliable data transfer, bandwidth guarantees, and delay guarantees.



UDP:

The **User Datagram Protocol** (**UDP**) is one of the core members of the Internet protocol suite. The protocol was designed by David P. Reed in 1980 and formally defined in RFC 768.

UDP uses a simple connectionless transmission model with a minimum of protocol mechanism. It has no handshaking dialogues, and thus exposes any unreliability of the underlying network protocol to the user's program. There is no guarantee of delivery, ordering, or duplicate protection. UDP provides checksums for data integrity, and port numbers for addressing different functions at the source and destination of the datagram.



With UDP, computer applications can send messages, in this case referred to as datagram, to other hosts on an Internet Protocol (IP) network without prior communications to set up special transmission channels or data paths. UDP is suitable for purposes where error checking and correction is either not necessary or is performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to waiting for delayed packets, which may not be an option in a real-time system.[1] If error correction facilities are needed at the network interface level, an application may use the Transmission Control Protocol (TCP) or Stream Control Transmission Protocol (SCTP)

UDP (User Datagram Protocol) is an alternative communications protocol to Transmission Control Protocol (TCP) used primarily for establishing low-latency and loss tolerating connections between applications on the Internet. Both UDP and TCP run on top of the Internet Protocol (IP) and are sometimes referred to as UDP/IP or TCP/IP. Both protocols send short packets of data, called datagram.

UDP provides two services not provided by the IP layer. It provides port numbers to help distinguish different user requests and, optionally, a checksum capability to verify that the data arrived intact.

TCP has emerged as the dominant protocol used for the bulk of Internet connectivity owing to services for breaking large data sets into individual packets, checking for and resending lost packets and reassembling packets into the correct sequence. But these additional services come at a cost in terms of additional data overhead, and delays called latency.

UDP is an ideal protocol for network applications in which perceived latency is critical such as gaming, voice and video communications, which can suffer some data loss without adversely affecting perceived quality. In some cases, forward error correction techniques are used to improve audio and video quality in spite of some loss.

UDP can also be used in applications that require lossless data transmission when the application is configured to manage the process of retransmitting lost packets and correctly arranging received packets. This approach can help to improve the data transfer rate of large files compared with TCP.

Attributes

A number of UDP's attributes make it especially suited for certain applications.

- It is transaction-oriented, suitable for simple query-response protocols such as the Domain Name System or the Network Time Protocol.
- It provides datagram, suitable for modeling other protocols such as in IP tunneling or Remote Procedure Call and the Network File System.
- It is simple, suitable for bootstrapping or other purposes without a full protocol stack, such as the DHCP and Trivial File Transfer Protocol.
- It is stateless, suitable for very large numbers of clients, such as in streaming media applications for example IPTV
- The lack of retransmission delays makes it suitable for real-time applications such as Voice over IP, online games, and many protocols built on top of the Real Time Streaming Protocol.
- Works well in unidirectional communication, suitable for broadcast information such as in many kinds of service discovery and shared information such as broadcast time or Routing Information Protocol
- UDP provides application multiplexing (via port numbers) and integrity verification (via checksum) of the header and payload.[4] If transmission reliability is desired, it must be implemented in the user's application.

- The UDP header consists of 4 fields, each of which is 2 bytes (16 bits).[1] The use of the fields "Checksum" and "Source port" is optional in IPv4 (pink background in table). In IPv6 only the source port is optional (see below).
- Source port number

- This field identifies the sender's port when meaningful and should be assumed to be the port to reply to if needed. If not used, then it should be zero. If the source host is the client, the port number is likely to be an ephemeral port number. If the source host is the server, the port number is likely to be a well-known port number.[2]

- Destination port number

- This field identifies the receiver's port and is required. Similar to source port number, if the client is the destination host then the port number will likely be an ephemeral port number and if the destination host is the server then the port number will likely be a well-known port number.[2]

- Length

∉ A field that specifies the length in bytes of the UDP header and UDP data. The minimum length is 8 bytes because that is the length of the header. The field size sets a theoretical limit of 65,535 bytes (8 byte header + 65,527 bytes of data) for a UDP datagram. The practical limit for the data length which is imposed by the underlying IPv4 protocol is 65,507 bytes (65,535 − 8 byte UDP header − 20 byte IP header).[2]

- In IPv6 Jumbo grams it is possible to have UDP packets of size greater than 65,535 bytes.[5] RFC 2675 specifies that the length field is set to zero if the length of the UDP header plus UDP data is greater than 65,535.

- Checksum

- The checksum field is used for error-checking of the header and data. If no checksum is generated by the transmitter, the field uses the value all-zeros.[6] This field is not optional for IPv6.[7]

Reliable byte stream (TCP):

A **reliable byte stream** is a common service paradigm in computer networking; it refers to a byte stream in which the bytes which emerge from the communication channel at the recipient are exactly the same, and in exactly the same order, as they were when the sender inserted them into the channel.

The classic example of a reliable byte stream communication protocol is the Transmission Control Protocol, one of the major building blocks of the Internet.

A reliable byte stream is not the only reliable service paradigm which computer network communication protocols provide, however; other protocols (e.g. SCTP) provide a reliable message stream, i.e. the data is divided up into distinct units, which are provided to the consumer of the data as discrete objects.

Connection-oriented (TCP):

- Flow control: keep sender from overrunning receiver
- Congestion control: keep sender from overrunning network

Characteristics of TCP Reliable Delivery:

TCP provides a **reliable, byte-stream, full-duplex inter-process communications service** to application programs/processes. The service is **connection-oriented** and uses the concept of **port numbers** to identify processes.

Reliable

All data will be delivered correctly to the destination process, without errors, even though the underlying packet delivery service (IP) is unreliable -- see later.

Connection-oriented

Two process which desire to communicate using TCP must first request a **connection**. A connection is closed when communication is no longer desired.

Byte-stream

An application which uses the TCP service is unaware of the fact that data is broken into **segments** for transmission over the network.

Full-duplex

Once a TCP connection is established, application data can flow in both directions simultaneously -- note, however, that many application protocols do not take advantage of this.

Port Numbers

Port numbers identify processes/connections in TCP.

Edge Systems and Reliable Transport

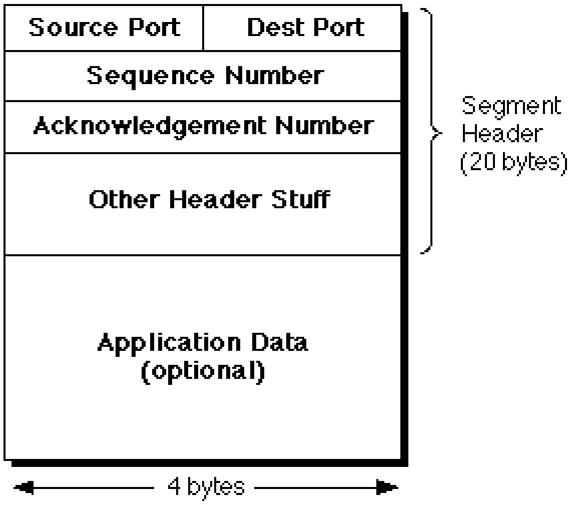
1. An **edge system** is any computer (host, printer, even a toaster...) which is "connected to" the Internet -- that is, it has access to the Internet's packet delivery system, but doesn't itself form part of that delivery system.

2. A **transport service** provides communications between application processes running on edge systems. As we have already seen, application processes communicate with each another using **application protocols** such as HTTP and SMTP. The interface between an application process and the transport service is normally provided using the **socket** mechanism.

Most application protocols require **reliable data transfer**, which in the Internet is provided by the **TCP** transport service/protocol. Note: some applications **do not** require reliability, so the unreliable **UDP** transport service/protocol is also provided as an alternative

TCP Segments

TCP slices (dices?) the incoming byte-stream data into **segments** for transmission across the Internet. A segment is a highly-structured data package consisting of an administrative **header** and some **application data**.



Source and Destination Port Numbers

We have already seen that TCP server processes wait for connections at a pre-agreed port number. At connection establishment time, TCP first allocates a **client port number** -- a port number by which the client, or initiating, process can be identified. Each segment contains both port numbers.

Segment and Acknowledgment Numbers

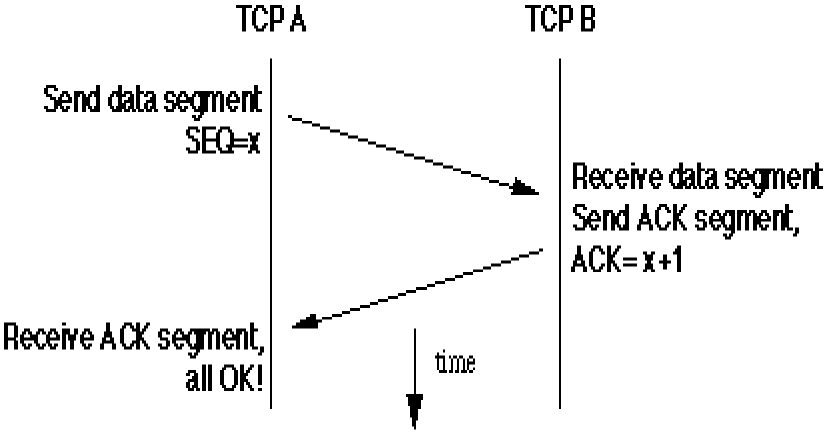
Every transmitted segment is identified with a 32-bit **Sequence number**[2], so that it can be explicitly acknowledged by the receipient. The Acknowledgment Number identifies the last segment recived by the originator of this segment.

Application Data

Optional because some segments convey only **control information** -- for example, an ACK segment has a valid acknowledgment number field, but no data. The data field can be any size up to the currently configured **MSS** for the whole segment.

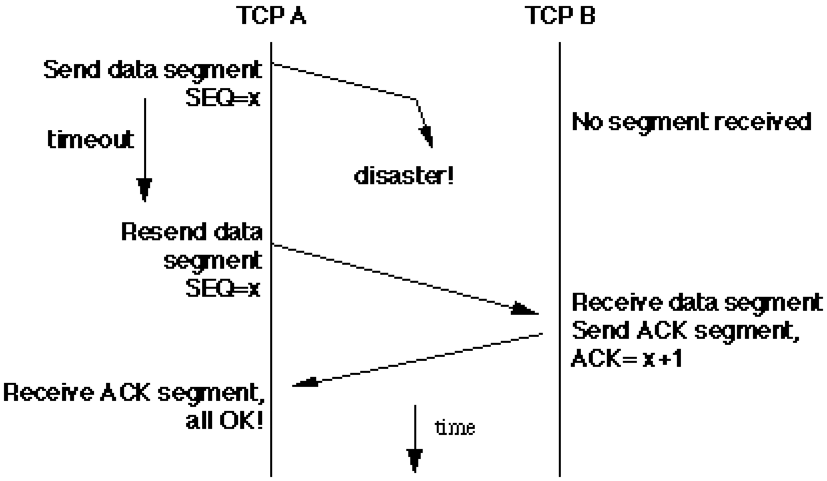
TCP Operation

When a segment is received correct and intact at its destination, an **acknowledgment** (ACK) segment is returned to the sending TCP. This ACK contains the sequence number of the last byte correctly received, incremented by 1[3]. ACKs are cumulative -- a single ACK can be sent for several segments if, for example, they all arrive within a short period of time.



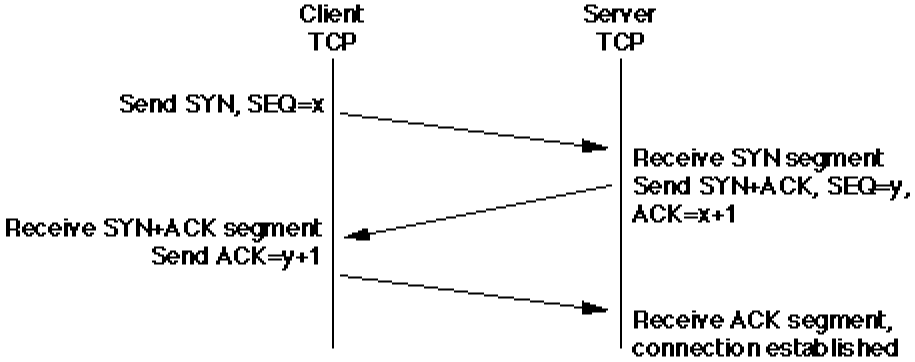
The network service can fail to deliver a segment. If the sending TCP waits for **too long**[4] for an acknowledgment, it times out and resends the segment, on the assumption that the datagram has been lost.

In addition, the network can potentially deliver duplicated segments, and can deliver segments out of order. TCP buffers or discards out of order or duplicated segments appropriately, using the byte count for identification.



TCP Connections:

An application process requests TCP to establish, or open, a (reliable) connection to a server process running on a specified edge-system, and awaiting connections at a known port number. After allocating an unused client-side port number[5], TCP initiates an exchange of connection establishment "control segments":



- This exchange of segments is called a **3-way handshake** (for obvious reasons), and is necessary because any one of the three segments can be lost, etc. The **ACK** and **SYN** segment names refer to "control bits" in the TCP header: for example, if the **ACK** bit is set, then this is an **ACK** segment.
- Each TCP chooses an **random initial sequence number** (the **x** and **y** in this example). This is crucial to the protocol's operation if there's a small chance that "old" segments (from a closed connection) might be interpreted as valid within the current connection.
- A connection is **closed** by another 3-way handshake of control segments. It's possible for a connection to be **half open** if one end requests close, and the other doesn't respond with an appropriate segment.

Optional: TCP Flow Control, Congestion Control and Slow Start

TCP attempts to make the best possible use of the underlying network, by sending data at the highest possible rate that won't cause segment loss. There are two aspects to this:

Flow Control

The two TCPs involved in a connection each maintain a **receive window** for the connection, related to the size of their **receive buffers**. For TCP "**A**", this is the maximum number of bytes that TCP "**B**" should send to it before "blocking" and waiting for an ACK. All TCP segments contain a **window** field, which is used to inform the other TCP of the sender's receive window size -- this is called "advertising a window size". At any time, for example, TCP **B** can have multiple segments "**in-flight**" -- that is, sent but not yet ACK'd -- up to TCP **A**'s advertised window.

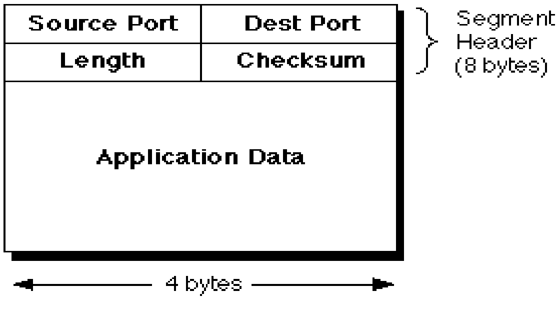
Congestion Avoidance and Control

When a connection is initially established, the TCPs know nothing at all about the speed, or capacity, of the networks which link them. The built-in "**slow start**" algorithm controls the rate at which segments are initially sent, as TCP tentatively discovers reasonable numbers for the connection's **Round Trip Time (RTT)** and its variability. TCP also slowly increases the number of segments "in-flight", since this increases the utilisation of the network.

Every TCP in the entire Internet is attempting to make full use of the available network, by increasing the number of "in-flight" segments it has outstanding. Ultimately there will come a point where the sum of the traffic, in some region of the network exceeds one or more router's buffer space, at which time segments will be dropped. When TCP "times out", and has to resend a dropped segment, it takes this as an indication that it (and all the other TCPs) have pushed the network just a little too hard. TCP immediately reduces its **congestion window** to a low value, and slowly, slowly allows it to increase again as ACKs are received.

User Datagram Protocol:

The **User Datagram Protocol (UDP)** provides an alternative, connectionless, transport service to TCP for applications where reliable stream service is not needed. UDP datagrams can be droppped, duplicated or delivered out of order, exactly as for IP.



The UDP transport service adds to IP the ability to deliver a datagram to a specified destination process using a port abstraction, in an analogous way to that used by TCP.

UDP segments (also commonly called **datagrams**, see later) have a minimal (8-byte) header. Data transfer with UDP has no initial connection overhead, and (obviously) no waiting for ACK segments as in TCP. Some typical UDP-based services include DNS, streaming multimedia and "Voice over IP" applications.

Connection management:

TCP Connection Management:
Recall:

TCP sender, receiver establish “connection” before exchanging data segments - to initialize TCP variables.

Client:

connection initiaton Socket clientSocket = new Socket("hostname","port number");

Server:

Contacted by client Socket connectionSocket = welcome Socket. accept ();

Flow control:

- Based on window mechanism
- Aims at sharing the bandwidth fairly between the users
- Timeout and retransmission
- measurement of the round trip time (RTT)

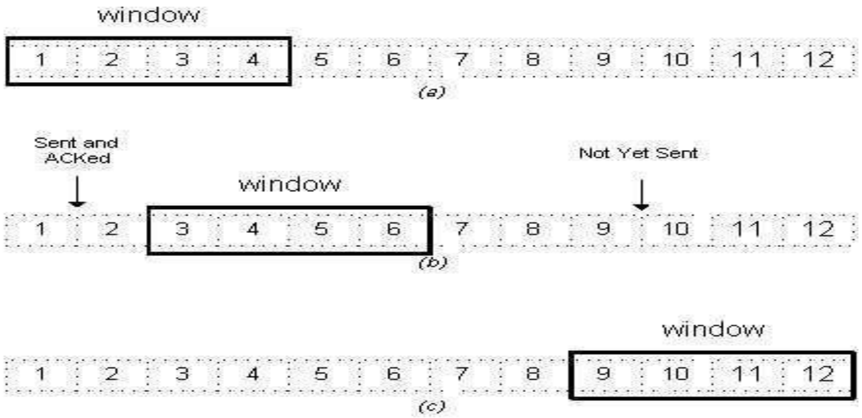
∉ Estimating variance of RTT (Jacobson’s algorithm)

- Exponential backoff of RTO
- Slow start

- Dynamic window control
- Fast retransmit
- Fast recovery
- Selective acknowledgement, SACK (an optional addition)

One of TCP's primary functions is to properly match the transmission rate of the sender to that of the receiver and the network. It is important for the transmission to be at a high enough rate to ensure good performance, but also to protect against overwhelming the network or receiving host.

TCP's 16-bit window field is used by the receiver to tell the sender how many bytes of data the receiver is willing to accept. Since the window field is limited to a maximum of 16 bits, this provides for a maximum window size of 65,535 bytes.
The window size advertised by the receiver tells the sender how much data, starting from the current position in the TCP data byte stream can be sent without waiting for further acknowledgements. As data is sent by the sender and then acknowledged by the receiver, the window slides forward to cover more data in the byte stream. This concept is known as a "sliding window".



The window boundary is eligible to be sent by the sender. Those bytes in the stream prior to the window have already been sent and acknowledged. Bytes ahead of the window have not been sent and must wait for the window to "slide" forward before they can be transmitted by the sender. A receiver can adjust the window size each time it sends acknowledgements to the sender. The maximum transmission rate is ultimately bound by the receiver's ability to accept and process data. However, this technique implies an implicit trust arrangement between the TCP sender and receiver. It has been shown that aggressive or **unfriendly** TCP software implementations can take advantage of this trust relationship to unfairly increase the transmission rate or even to intentionally cause network overload situations.

As we will see shortly, the sender and also the network can play a part in determining the transmission rate of data flow as well. It is important to consider the limitation on the window size of 65,535 bytes. Consider a typical internetwork that may have link speeds of up to 1 Gb/s or more. On a 1 Gb/s network 125,000,000 bytes can be transmitted in one second. TCP stations are communicating on this link, at best 65,535/125,000,000 or only about .0005 of the bandwidth will be used in each direction each second.

Recognizing the need for larger windows on high-speed networks, the Internet Engineering Task Force released a standard for a “window scale option” defined in RFC 1323. This standard effectively allows the window to increase from 16 to 32 bits or over 4 billion bytes of data in the window.Flow control is a function for **the control of the data flow** within an OSI layer or between adjacent layers. In other words it **limits the amount of data transmitted** by the sending transport entity to a level, or rate, that the receiver can manage.

Flow control is a good example of a protocol function that must be implemented in several layers of the OSI architecture model. At the transport level flow control will allow the transport protocol entity in a host to **restrict the flow of data over a logical connection** from the transport protocol entity in another host. However, one of the services of the network level is to **prevent congestion**. Thus the network level also uses flow control to restrict the flow of network protocol data units (NPDUs).

The flow control mechanisms used in the transport layer vary for the different classes of service. Since the different classes of service are determined by the quality of service of the underlying data network which transports the transport protocol data units (TPDUs), it is these which influence the type of flow control used.

Thus flow control becomes a much more **complex issue** at the transport layer than at lower levels like the data link level.

Two reasons for this are:

- Flow control must interact with transport users, transport entities, and the network service.
- Long and variable transmission delays between transport entities.

Flow control causes **Queuing amongst transport users, entities, and the network service**. We take a look at the four possible queues that form and what control policies are at work **here**.

The transport entity is responsible for generating one or more **transport protocol data units (TPDUs)** for passing onto the network layer. The network layer delivers the TPDUs to the receiving transport entity which then takes out the data and passes it on to the destination user. There are two reasons why the receiving transport entity would want to **control the flow of TPDUs**:

- The receiving user cannot keep up with the flow of data
- The receiving transport entity itself cannot keep up with the flow of TPDUs

When we say that a user or transport entity cannot keep up with the data flow, we mean that the **receiving buffers** are filling too quickly and will **overflow and lose data** unless the rate of incoming data is slowed.

Four possible ways to cope with the problem are:

- **Let it be and do nothing**
- **Refuse any more TPDUs from the network service**
- **Use a fixed sliding-window protocol**
- **Use a credit scheme**

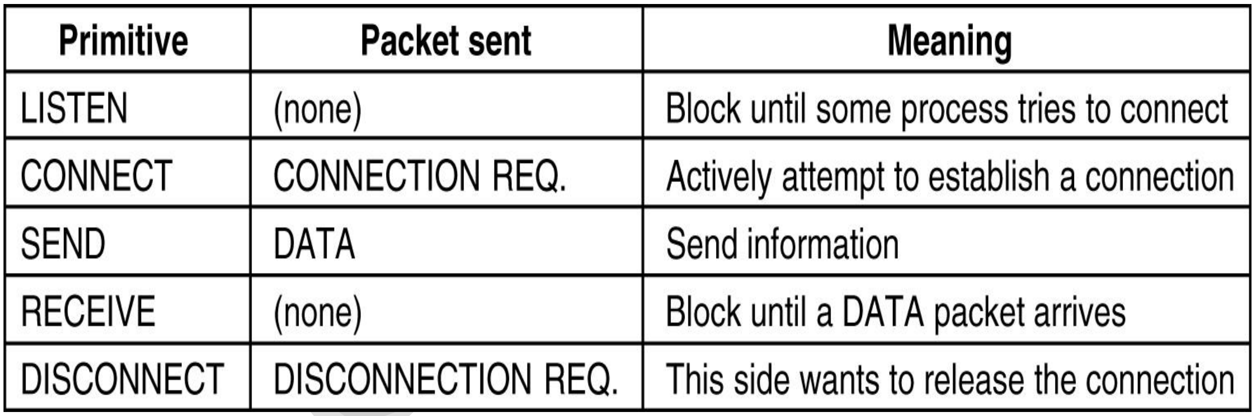
There are different issues to be considered with transport flow control over different levels of network service. The more unreliable the network service provided the more complex flow control mechanism that may be needed to be used by the Transport Layer. The credit scheme works well with the different network services although specific issues need to be addressed as with a **Reliable Non-sequencing Network Service** and an **Unreliable Network Service**.

The **credit scheme** seems most suited for flow control in the **transport layer** with all types of network service. It gives the receiver the **best control over data flow** and helps provide a smooth traffic flow. **Sequence numbering of credit allocations** handles the arrival of ACK/CREDIT TPDUs out of order, and a **window timer** will ensure **deadlock** does not occur in a network environment where TPDUs can be lost.

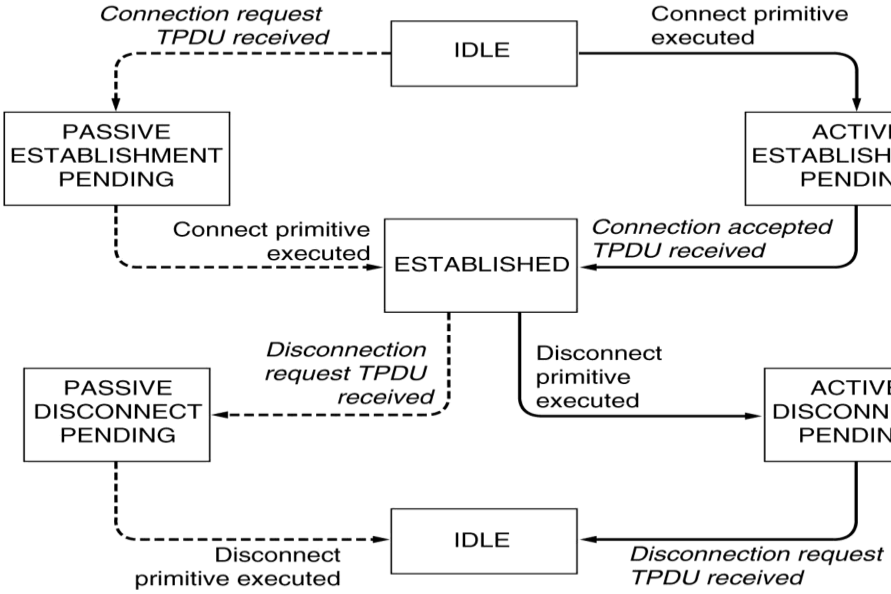
The Transport Services:

a) **Services Provided to the Upper Layers**
b) **Transport Service Primitives**
c) **Berkeley Sockets**

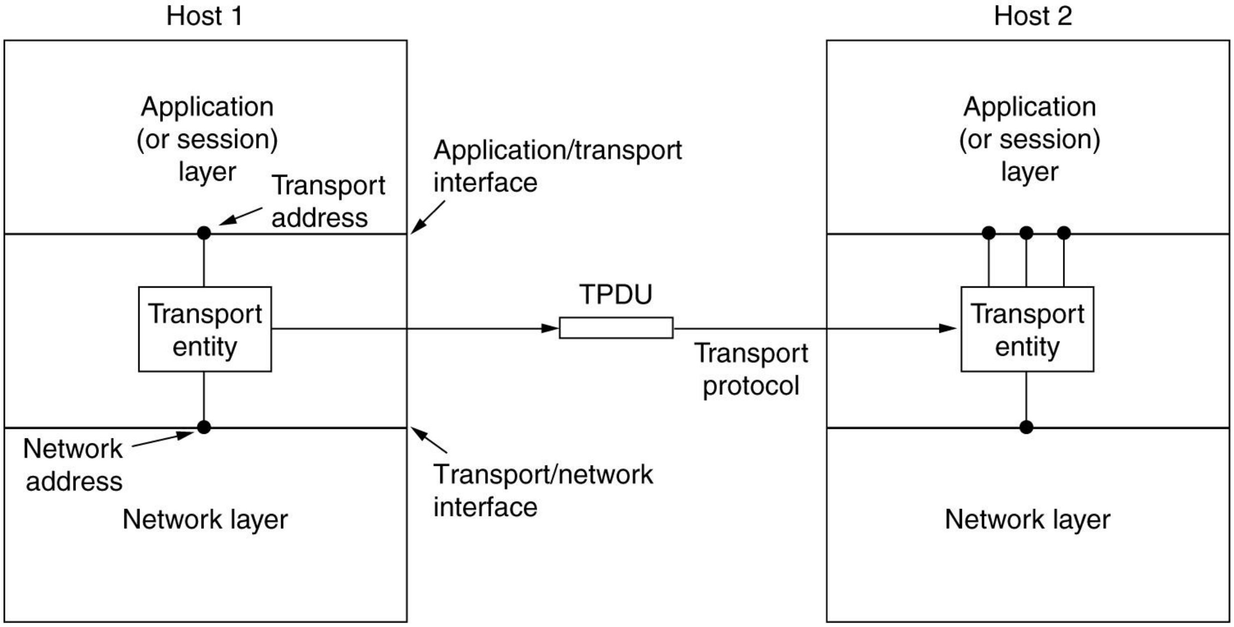
Transport Service Primitives



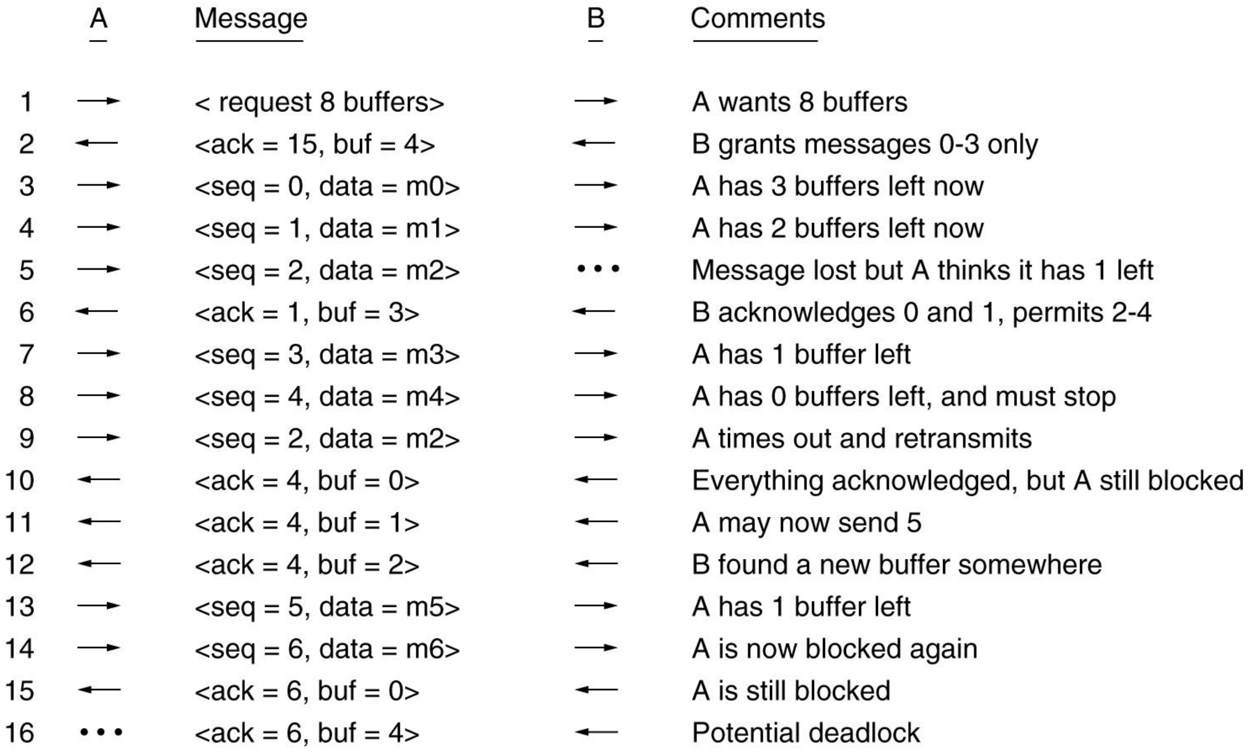
Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection


Connection request TPDU received
IDLE
Connect primitive executed
PASSIVE ESTABLISHMENT PENDING
ACTIVE ESTABLISHMENT PENDING
Connect primitive executed
ESTABLISHED
Connection accepted TPDU received
Disconnection request TPDU received
Disconnect primitive executed
PASSIVE DISCONNECT PENDING
ACTIVE DISCONNECT PENDING
IDLE
Disconnect primitive executed
Disconnection request TPDU received

Services Provided to the Upper Layers



Flow Control and Buffering:



	A	Message	B	Comments
1	→	< request 8 buffers>	→	A wants 8 buffers
2	←	<ack = 15, buf = 4>	←	B grants messages 0-3 only
3	→	<seq = 0, data = m0>	→	A has 3 buffers left now
4	→	<seq = 1, data = m1>	→	A has 2 buffers left now
5	→	<seq = 2, data = m2>	• • •	Message lost but A thinks it has 1 left
6	←	<ack = 1, buf = 3>	←	B acknowledges 0 and 1, permits 2-4
7	→	<seq = 3, data = m3>	→	A has 1 buffer left
8	→	<seq = 4, data = m4>	→	A has 0 buffers left, and must stop
9	→	<seq = 2, data = m2>	→	A times out and retransmits
10	←	<ack = 4, buf = 0>	←	Everything acknowledged, but A still blocked
11	←	<ack = 4, buf = 1>	←	A may now send 5
12	←	<ack = 4, buf = 2>	←	B found a new buffer somewhere
13	→	<seq = 5, data = m5>	→	A has 1 buffer left
14	→	<seq = 6, data = m6>	→	A is now blocked again
15	←	<ack = 6, buf = 0>	←	A is still blocked
16	• • •	<ack = 6, buf = 4>	←	Potential deadlock

Retransmission:

TCP is relegated to rely mostly upon implicit signals it learns from the network and remote host. TCP must make an educated guess as to the state of the network and trust the information from the remote host in order to control the rate of data flow. This may seem like an awfully tricky problem, but in most cases TCP handles it in a seemingly simple and straightforward way.

A sender's implicit knowledge of network conditions may be achieved through the use of a **timer**. For each TCP segment sent the sender expects to receive an acknowledgement within some period of time otherwise an error in the form of a timer expiring signals that that something is wrong.

Somewhere in the end-to-end path of a TCP connection a segment can be lost along the way. Often this is due to congestion in network routers where excess packets must be dropped. TCP not only must correct for this situation, but it can also **learn** something about network conditions from it.

Whenever TCP transmits a segment the sender starts a timer which keeps track of how long it takes for an acknowledgment for that segment to return. This timer is known as the **retransmission timer**. If an acknowledgement is returned before the timer expires,

which by default is often initialized to 1.5 seconds, the timer is reset with no consequence. If however an acknowledgement for the segment does not return within the timeout period, the sender would retransmit the segment and double the retransmission timer value for each consecutive timeout up to a maximum of about 64 seconds. If there are serious network problems, segments may take a few minutes to be successfully transmitted before the sender eventually times out and generates an error to the sending application.

Fundamental to the timeout and retransmission strategy of TCP is the measurement of the **round-trip time** between two communicating TCP hosts. The round-trip time may vary during the TCP connection as network traffic patterns fluctuate and as routes become available or unavailable.

A TCP option negotiated in the TCP connection establishment phase sets the number of bits by which the window is right-shifted in order to increase the value of the window. TCP keeps track of when data is sent and at what time acknowledgements covering those sent bytes are returned. TCP uses this information to calculate an estimate of round trip time. As packets are sent and acknowledged, TCP adjusts its round-trip time estimate and uses this information to come up with a reasonable timeout value for packets sent. If acknowledgements return quickly, the round-trip time is short and the retransmission timer is thus set to a lower value. This allows TCP to quickly retransmit data when network response time is good, alleviating the need for a long delay between the occasional lost segment. The converse is also true. TCP does not retransmit data too quickly during times when network response time is long.

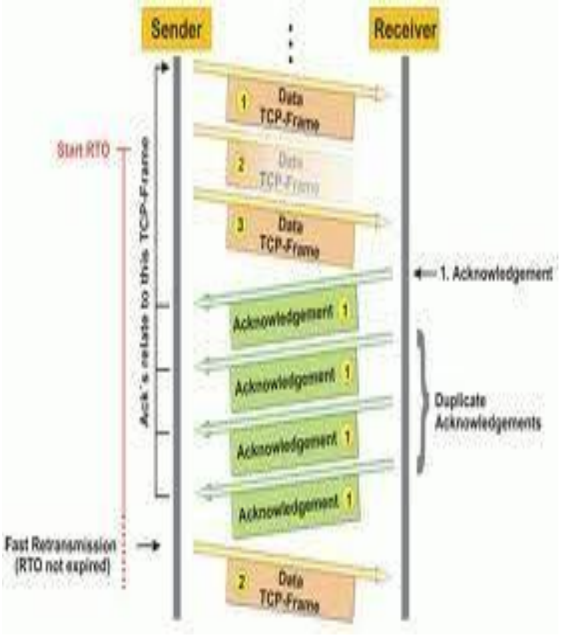
If a TCP data segment is lost in the network, a receiver will never even know it was once sent. However, the sender is waiting for an acknowledgement for that segment to return. In one case, if an acknowledgement doesn't return, the sender's retransmission timer expires which causes a retransmission of the segment. If however the sender had sent at least one additional segment after the one that was lost and that later segment is received correctly, the receiver does not send an acknowledgement for the later, out of order segment.

The receiver cannot acknowledgement out of order data; it must acknowledge the last contiguous byte it has received in the byte stream prior to the lost segment. In this case, the receiver will send an acknowledgement indicating the last contiguous byte it has received. If that last contiguous byte was already acknowledged, we call this a duplicate ACK. The reception of duplicate ACKs can implicitly tell the sender that a segment may have been lost or delayed. The sender knows this because the receiver only generates a duplicate ACK when it receives other, out of order segments. In fact, the Fast Retransmit algorithm described later uses duplicate ACKs as a way of speeding up the retransmission process.

A TCP sender uses a timer to recognize lost segments. If an acknowledgement is not received for a particular segment within a specified time (a function of the estimated Round-trip delay time), the sender will assume the segment was lost in the network, and will retransmit the segment.

Duplicate acknowledgement is the basis for the fast retransmit mechanism which works as follows: after receiving a packet (e.g. with sequence number 1), the receiver sends an acknowledgement by adding 1 to the sequence number (i.e., acknowledgement number 2) which means that the receiver receives the packet number 1 and it expects packet number 2 from the sender. Let's assume that three subsequent packets have been lost. In the meantime the receiver receives packet numbers 5 and 6. After receiving packet number 5, the receiver sends an acknowledgement, but still only for sequence number 2. When the receiver receives packet number 6, it sends yet another acknowledgement value of 2. Because the sender receives more than one acknowledgement with the same sequence number (2 in this example) this is called duplicate acknowledgement.

The fast retransmit enhancement works as follows: if a TCP sender receives a specified number of acknowledgements which is usually set to three duplicate acknowledgements with the same acknowledge number (that is, a total of four acknowledgements with the same acknowledgement number), the sender can be reasonably confident that the segment with the next higher sequence number was dropped, and will not arrive out of order. The sender will then retransmit the packet that was presumed dropped before waiting for its timeout.



TCP Congestion control:

The standard fare in TCP implementations today can be found in RFC 2581 [2]. This reference document specifies four standard congestion control algorithms that are now in common use. Each of the algorithms noted within that document was actually designed long before the standard was published [9], [11]. Their usefulness has passed the test of time.

The four algorithms, Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery are described below.

Slow Start

Slow Start, a requirement for TCP software implementations is a mechanism used by the sender to control the transmission rate, otherwise known as sender-based flow control. This is accomplished through the return rate of acknowledgements from the receiver. Inother words, the rate of acknowledgements returned by the receiver determine the rate at which the sender can transmit data.

When a TCP connection first begins, the Slow Start algorithm initializes a **congestion window** to one segment, which is the maximum segment size (MSS) initialized by the receiver during the connection establishment phase. When acknowledgements are returned by the receiver, the congestion window increases by one segment for each acknowledgement returned. Thus, the sender can transmit the minimum of the congestion window and the advertised window of the receiver, which is simply called the **transmission window**.

Slow Start is actually not very slow when the network is not congested and network response time is good. For example, the first successful transmission and acknowledgement of a TCP segment increases the window to two segments. After successful transmission of these two segments and acknowledgements completes, the window is increased to four segments. Then eight segments, then sixteen segments and so on, doubling from there on out up to the maximum window size advertised by the receiver or until congestion finally does occur.

At some point the congestion window may become too large for the network or network conditions may change such that packets may be dropped. Packets lost will trigger a timeout at the sender. When this happens, the sender goes into congestion avoidance mode as described in the next section.

Congestion avoidance:

During the initial data transfer phase of a TCP connection the Slow Start algorithm is used. However, there may be a point during Slow Start that the network is forced to drop one or more packets due to overload or congestion. If this happens, Congestion Avoidance is used to slow the transmission rate. However, Slow Start is used in conjunction with Congestion Avoidance as the means to get the data transfer going again so it doesn't slow down and stay slow.

In the Congestion Avoidance algorithm a retransmission timer expiring or the reception of duplicate ACKs can implicitly signal the sender that a network congestion situation is occurring. The sender immediately sets its transmission window to one half of the current window size (the minimum of the congestion window and the receiver's advertised window size), but to at least two segments. If congestion was indicated by a timeout, the congestion window is reset to one segment, which automatically puts the sender into Slow Start mode. If congestion was indicated by duplicate ACKs, the Fast Retransmit and Fast Recovery algorithms are invoked (see below).

As data is received during Congestion Avoidance, the congestion window is increased. However, Slow Start is only used up to the halfway point where congestion originally occurred. This halfway point was recorded earlier as the new transmission window. After this halfway point, the congestion window is increased by one segment for all segments in the transmission window that are acknowledged. This mechanism will force the sender to more slowly grow its transmission rate, as it will approach the point where congestion had previously been detected.

DEC bit:

DECbit is a technique implemented in routers to avoid congestion. Its utility is to predict possible congestion and prevent it. This protocol works with TCP.

When a router wants to signal congestion to the sender, it adds a bit in the header of packets sent. When a packet arrives at the router, the router calculates the average queue length for the last (busy + idle) period plus the current busy period. (The router is busy when it is transmitting packets, and idle otherwise). When the average queue length exceeds 1, then the router sets the congestion indication bit in the packet header of arriving packets.

When the destination replies, the corresponding ACK includes a bit of congestion. The sender receives the ACK and calculates how many packets it received with the congestion indication bit set to one. If less than half of the packets in the last window had the congestion indication bit set, then the window is increased linearly. Otherwise, the window is decreased exponentially.

This technique provides distinct advantages:

- Dynamically manages the window to avoid congestion and increasing freight if it detects congestion.
- Try to balance bandwidth with respect to the delay.

Note that this technique does not allow for effective use of the line, because it fails to take advantage of the available bandwidth. Besides, the fact that the tail has increased in size from one cycle to another does not always mean there is congestion.

DECbit (Destination Experiencing Congestion Bit) Developed for the Digital Network Architecture Basic idea One bit allocated in packet header Any router experiencing congestion sets bit Destination returns bit to source Source adjusts rate based on bits Note that responsibility is shared Routers identify congestion Hosts act to avoid congestion Key Questions:

- When should router signal congestion?
- How should end hosts react?

Congestion avoidance RED:

Random early detection (**RED**), also known as **random early discard** or **random early drop** is a queueing discipline for a network scheduler suited for congestion avoidance. In the conventional tail drop algorithm, a router or other network component buffers as many packets as it can, and simply drops the ones it cannot buffer. If buffers are constantly full, the network is congested. Tail drop distributes buffer space unfairly among traffic flows. Tail drop can also lead to TCP global synchronization as all TCP connections "hold back" simultaneously, and then step forward simultaneously. Networks become under-utilized and flooded by turns. RED addresses these issues.

Operation

RED monitors the average queue size and drops (or marks when used in conjunction with ECN) packets based on statistical probabilities. If the buffer is almost empty, all incoming packets are accepted. As the queue grows, the probability for dropping an incoming packet grows too. When the buffer is full, the probability has reached 1 and all incoming packets are dropped.

RED is more fair than tail drop, in the sense that it does not possess a bias against bursty traffic that uses only a small portion of the bandwidth. The more a host transmits, the more likely it is that its packets are dropped as the probability of a host's packet being dropped is proportional to the amount of data it has in a queue. Early detection helps avoid TCP global synchronization.

- RED provides congestion avoidance by controlling the queue size at the gateway.
- RED notifies the source before the congestion actually happens rather than wait till it actually occurs.
- RED provides a mechanism for the gateway to provide some feedback to the source on congestion status.

Advantages of RED gateways:

- Congestion Avoidance
  - If the RED gateway drops packets when avgQ reached maxQ, the avgQ will never exceed maxQ.
- Appropriate time scales
  - Source will not be notified of transient congestion.
- No Global Synchronization.
  - All connection wont back off at same time.
- Simple
- High link utilization
- Fair

QoS:

Quality of service (**QoS**) is the overall performance of a telephony or computer network, particularly the performance seen by the users of the network.To quantitatively measure quality of service, several related aspects of the network service are often considered, such as error rates, bandwidth, throughput, transmission delay, availability, jitter, etc.

Quality of service is particularly important for the transport of traffic with special requirements. In particular, much technology has been developed to allow computer networks to become as useful as telephone networks for audio conversations, as well as supporting new applications with even stricter service demands.

In the field of telephony, quality of service was defined by the ITU in 1994. Quality of service comprises requirements on all the aspects of a connection, such as service response time, loss, signal-to-noise ratio, crosstalk, echo, interrupts, frequency response, loudness levels, and so on. A subset of telephony QoS is grade of service (GoS) requirements, which comprises aspects of a connection relating to capacity and coverage of a network, for example guaranteed maximum blocking probability and outage probability.

In the field of computer networking and other packet-switched telecommunication networks, the traffic engineering term refers to resource reservation control mechanisms rather than the achieved service quality. Quality of service is the ability to provide different priority to different applications, users, or data flows, or to guarantee a certain level of performance to a data flow.

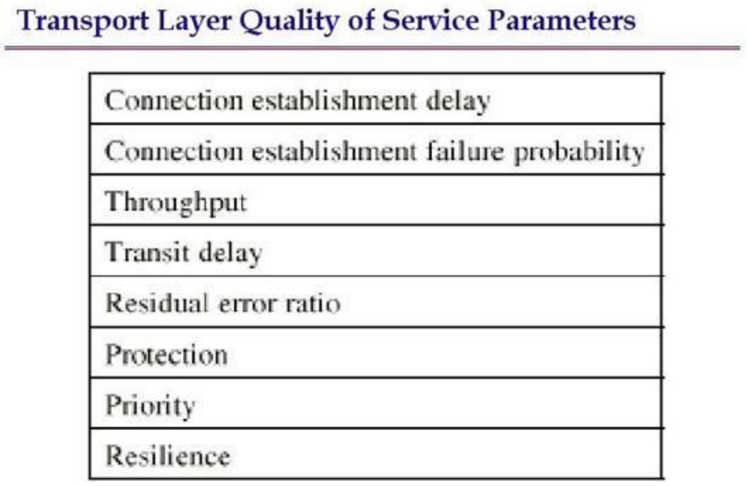
For example, a required bit rate, delay, jitter, packet dropping probability and/or bit error rate may be guaranteed. Quality of service guarantees are important if the network capacity is insufficient, especially for real-time streaming multimedia applications such as voice over IP, online games and IP-TV, since these often require fixed bit rate and are delay sensitive, and in networks where the capacity is a limited resource, for example in cellular data communication.

A network or protocol that supports QoS may agree on a traffic contract with the application software and reserve capacity in the network nodes, for example during a session establishment phase. During the session it may monitor the achieved level of performance, for example the data rate and delay, and dynamically control scheduling priorities in the network nodes. It may release the reserved capacity during a tear down phase.

A best-effort network or service does not support quality of service. An alternative to complex QoS control mechanisms is to provide high quality communication over a best-effort network by over-provisioning the capacity so that it is sufficient for the expected peak traffic load. The resulting absence of network congestion eliminates the need for QoS mechanisms.

QoS is sometimes used as a quality measure, with many alternative definitions, rather than referring to the ability to reserve resources. Quality of service sometimes refers to the level of quality of service, i.e. the guaranteed service quality.[3] High QoS is often confused with a high level of performance or achieved service quality, for example high bit rate, low latency and low bit error probability.

An alternative and disputable definition of QoS, used especially in application layer services such as telephony and streaming video, is requirements on a metric that reflects or predicts the subjectively experienced quality. In this context, QoS is the acceptable cumulative effect on subscriber satisfaction of all imperfections affecting the service. Other terms with similar meaning are the quality of experience (QoE) subjective business concept, the required “user perceived performance”,[4] the required “degree of satisfaction of the user” or the targeted “number of happy customers”. Examples of measures and measurement methods are mean opinion score (MOS), perceptual speech quality measure (PSQM) and perceptual evaluation of video quality (PEVQ). See also Subjective video quality.



Transport Layer Quality of Service Parameters

Connection establishment delay
Connection establishment failure probability
Throughput
Transit delay
Residual error ratio
Protection
Priority
Resilience

Qualities of traffic

In packet-switched networks, quality of service is affected by various factors, which can be divided into “human” and “technical” factors. Human factors include: stability of service, availability of service, delays, user information. Technical factors include: reliability, scalability, effectiveness, maintainability, grade of service, etc.

Many things can happen to packets as they travel from origin to destination, resulting in the following problems as seen from the point of view of the sender and receiver:

Low throughput

Due to varying load from disparate users sharing the same network resources, the bit rate (the maximum throughput) that can be provided to a certain data stream may be too low for realtime multimedia services if all data streams get the same scheduling priority.

Dropped packets

The routers might fail to deliver (drop) some packets if their data loads are corrupted, or the packets arrive when the router buffers are already full. The receiving application may ask for this information to be retransmitted, possibly causing severe delays in the overall transmission.

Errors

Sometimes packets are corrupted due to bit errors caused by noise and interference, especially in wireless communications and long copper wires. The receiver has to detect this and, just as if the packet was dropped, may ask for this information to be retransmitted.

Latency

It might take a long time for each packet to reach its destination, because it gets held up in long queues, or it takes a less direct route to avoid congestion. This is different from throughput, as the delay can build up over time, even if the throughput is almost normal. In some cases, excessive latency can render an application such as VoIP or online gaming unusable.

Jitter

Packets from the source will reach the destination with different delays. A packet's delay varies with its position in the queues of the routers along the path between source and destination and this position can vary unpredictably. This variation in delay is known as jitter and can seriously affect the quality of streaming audio and/or video.

Out-of-order delivery

When a collection of related packets is routed through a network, different packets may take different routes, each resulting in a different delay. The result is that the packets arrive in a different order than they were sent. This problem requires special additional protocols responsible for rearranging out-of-order packets to an isochronous state once they reach their destination. This is especially important for video and VoIP streams where quality is dramatically affected by both latency and lack of sequence.

Applications

A defined quality of service may be desired or required for certain types of network traffic, for example:

- Streaming media specifically
    - Internet protocol television (IPTV)
    - Audio over Ethernet
    - Audio over IP
- IP telephony also known as Voice over IP (VoIP)
- Videoconferencing
- Tele-presence
- Storage applications such as iSCSI and FCoE
- Circuit Emulation Service
- Safety-critical applications such as remote surgery where availability issues can be hazardous
- Network operations support systems either for the network itself, or for customers' business critical needs
- Online games where real-time lag can be a factor
- Industrial control systems protocols such as Ethernet/IP which are used for real-time control of machinery

These types of service are called inelastic, meaning that they require a certain minimum level of bandwidth and a certain maximum latency to function. By contrast, elastic applications can take advantage of however much or little bandwidth is available. Bulk file transfer applications that rely on TCP are generally elastic.

Application Layer

At the top of the TCP/IP protocol architecture is the Application Layer . This layer includes all processes that use the Transport Layer protocols to deliver data. There are many applications protocols. Most provide user services, and new services are always being added to this layer.

The most widely known and implemented applications protocols are:

Telnet

The Network Terminal Protocol, which provides remote login over the network.

FTP

The File Transfer Protocol, which is used for interactive file transfer.

SMTP

The Simple Mail Transfer Protocol, which delivers electronic mail.

HTTP

The Hypertext Transfer Protocol, which delivers Web pages over the network.

While HTTP, FTP, SMTP, and telnet are the most widely implemented TCP/IP applications, you will work with many others as both a user and a system administrator. Some other commonly used TCP/IP applications are:

Domain Name Service (DNS)

Also called name service , this application maps IP addresses to the names assigned to network devices. DNS is discussed in detail in this book.

Open Shortest Path First (OSPF)

Routing is central to the way TCP/IP works. OSPF is used by network devices to exchange routing information. Routing is also a major topic of this book.

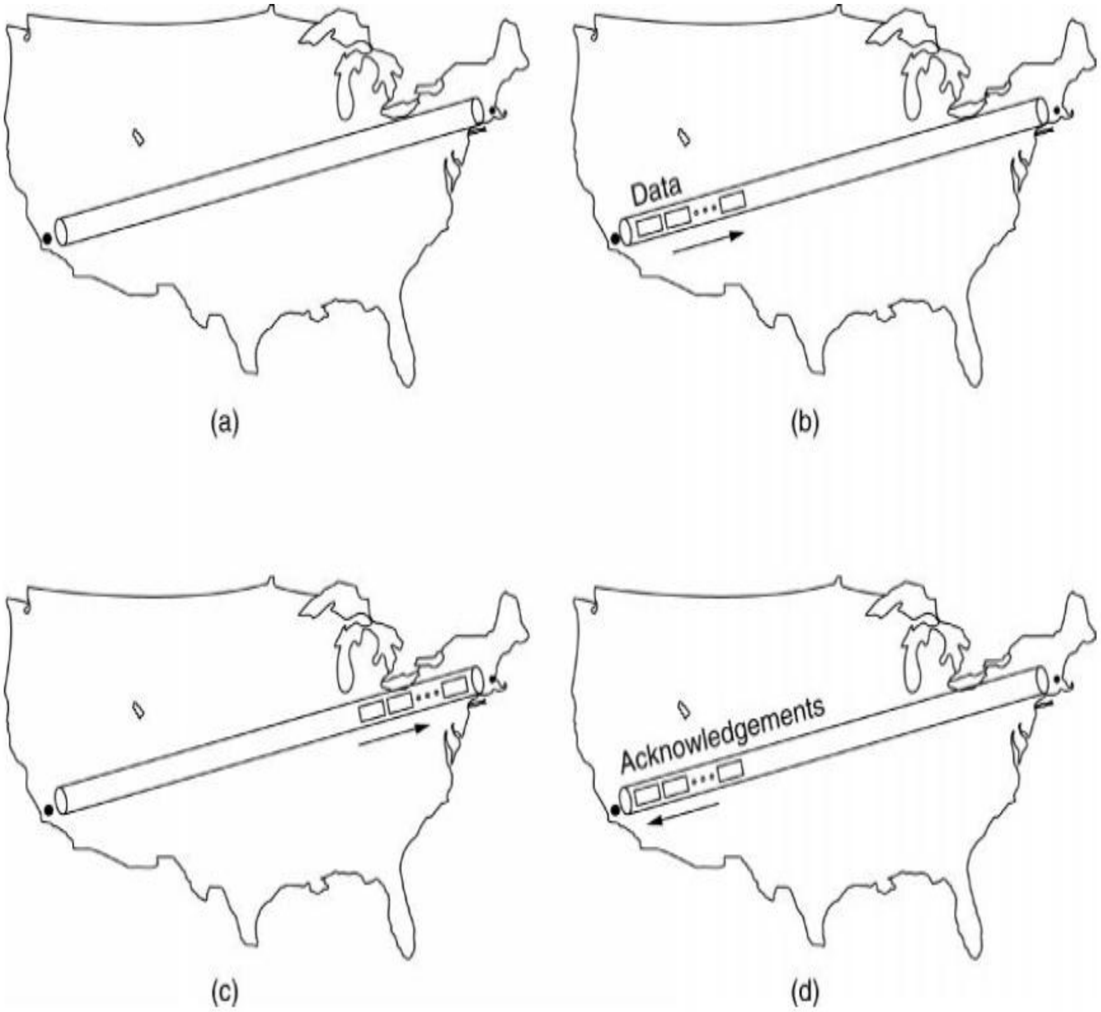
Network File system (NFS)

This protocol allows files to be shared by various hosts on the network.Some protocols, such as telnet and FTP, can only be used if the user has some knowledge of the network.

Other protocols, like OSPF, run without the user even knowing that they exist. As system administrator, you are aware of all these applications and all the protocols in the other TCP/IP layers.

Performance problems in computer networks

- Overloads Example 1: TPDU containing the bad parameter when broadcast may clog the n/w results in broadcast storm due to error message
- synchronous overload due to power failure-DHCP contacted for booting
- Apart from this problems due to insufficient memory TPDUs lost
- Not setting the timeout correctly the TPDUs lost
- Gigabit n/w pose new problems
- The next figure explains this here the transmission line used only for .5msec greatly reducing the efficiency



- The useful quantity is the Bandwidth-Delay product
- The product is the capacity of the pipe from sender to receiver and back to sender in bits
- In the above example it is 40 million bits but the actual utilisation is only 1.25 percent of the pipe capacity
- therefore for good performance the receiver window must be at least as large as the Bandwidth-Delay product
- Another performance problem could be jitter to avoid a small standard deviation is used

The basic loop for improving network performance.

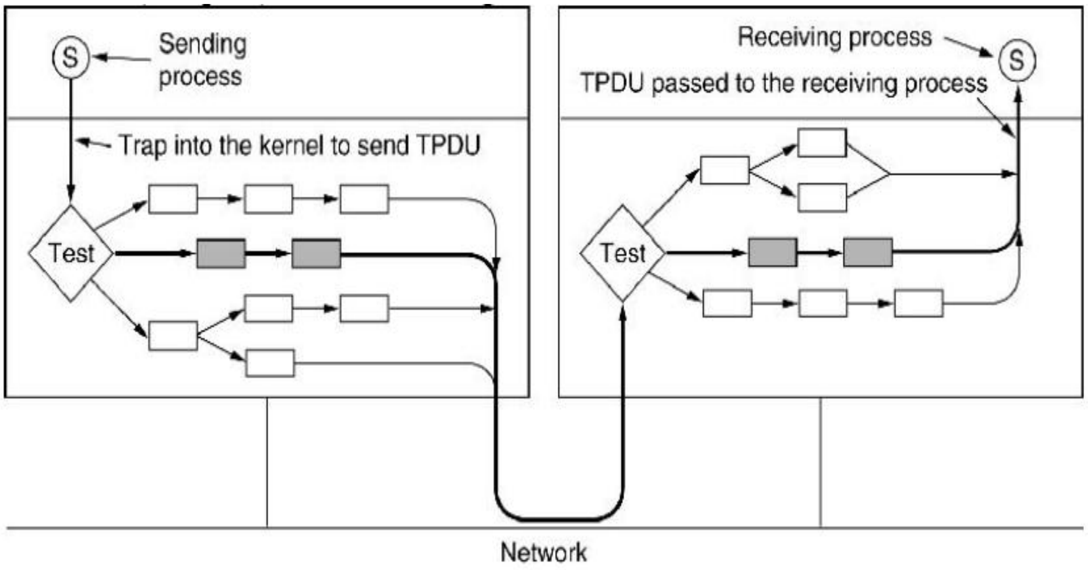
- Measure relevant network parameters, performance.
- Try to understand what is going on.
- Change one parameter Precautions taken while measuring
- Sample size should be large enough
- Samples should be representative
- To be careful while using coarse grained clock
- Nothing unexpected going on while tests are conducted
- Caching problem
- Understanding the measurements
- Extrapolation of the result

System Design for Better Performance
Rules:

- CPU speed is more important than network speed.
- Reduce packet count to reduce software overhead.
- Minimize context switches.
- Minimize copying.
- You can buy more bandwidth but not lower delay.
- Avoiding congestion is better than recovering from it.
- Avoid timeouts.

Fast TPDU Processing

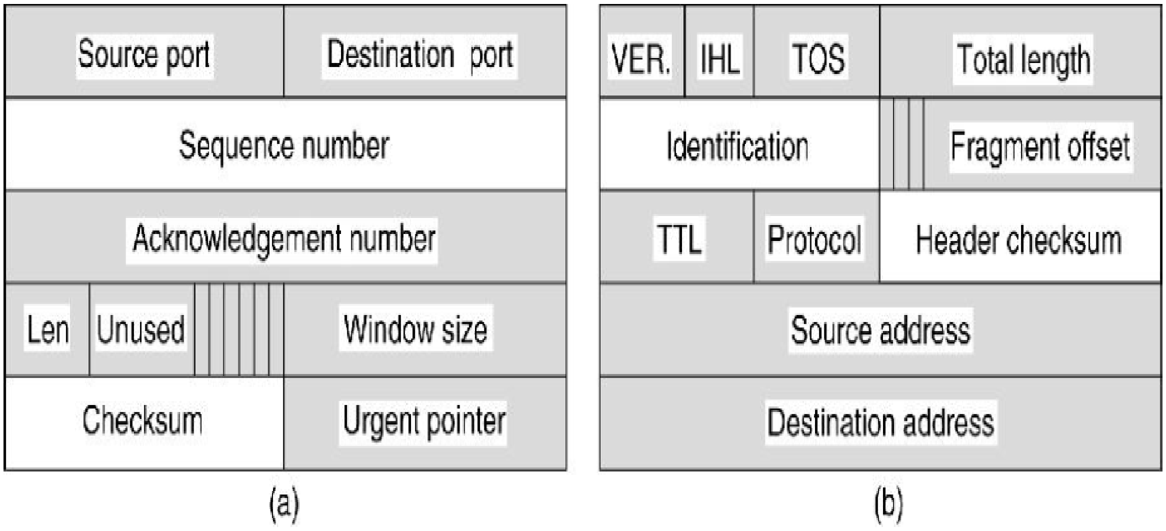
- TPDU processing overhead has two components
- one –overhead per TPDU
- other – overhead per byte
- Example take the sending side
- first the sending side traps to kernel to SEND
- if it is a normal case then the state is ESTABLISHED and typically this path is taken (fast path) shown in the figure below


S
Sending process
Trap into the kernel to send TPDU
Test
Receiving process
S
TPDU passed to the receiving process
Test
Network

The fast path from sender to receiver is shown with a heavy line.
The processing steps on this path are shaded.

Another example

- In the TCP header the fields that are same between consecutive TPDUs on a one way flow are shaded
- All sending TCP entity has to copy from the prototype header into the output buffer
- It handovers the header and data to the special IP procedure for sending a regular max TPDU
- IP then copies its prototype header and makes the packet ready the above figure



(a) TCP header. (b) IP header. In both cases, the shaded fields are taken from the prototype without change.

Fast path processing at receiver side

- step 1: locating the connection record for the incoming TPDU
- The TPDU checked to see if it is normal case
- If all checks are met then a fast procedure is called
- Many TCP implementations use Header Prediction
- The other two areas where major performance gain are possible are

Buffer management

Timer Management

- The timer management done by the timing wheel
- There are some problems and the possible solution posed by the Gigabit protocols
- Problems

Sequence Numbers

Communication Speeds

Go back n protocol and its poor performance

gigabit lines are bandwidth limited

Results of new application

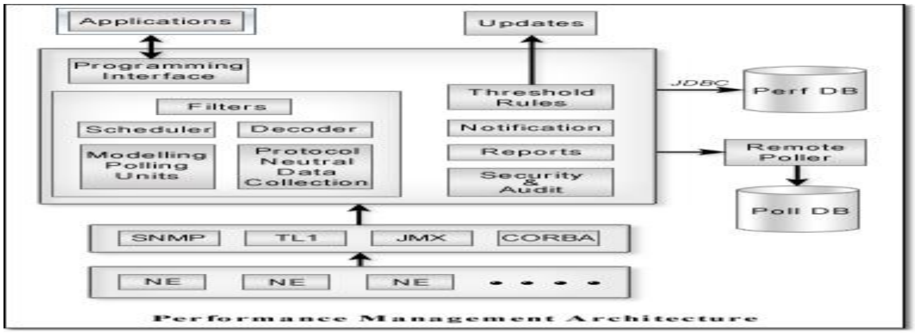
Network Performance Management

The main task of Network management systems is to monitor and control the network infrastructure. As computer networks increase in size, heterogeneity and complexity, effective management of such networks becomes more important and much difficult. In the present business world, each and every hour counts. An hour of network downtime means an hour of lost business and hence lost opportunities. Pro-actively managing the Network's health and performance is indispensable for any mission critical business and is something which the administrators of large networks have to put in place.

The main challenges in this area include

- Identifying the data that needs to be collected
- Interpreting the collected data
- Disseminating the data and
- Presenting the data which helps in network Performance management.

Performance Management Architecture



Performance Management Architecture

Web NMS Performance module is made up of the following components:

Data Collection service (Protocol neutral Data collection)

- This provides multi protocol Data collection support. Data can be collected from TL1 devices, CORBA devices etc. Default data collection takes place with respect to SNMP devices.
- User can plug in his own Protocol provider to facilitate data collection for that protocol.
- Data collection process can be Customized to suit user requirements.
- Observers can be set to monitor data collection process and get notified .

Data Collection Objects (Modeling Polling units)

Data collection process has been well-studied and modeled using objects that define

- what data to collect (PollingObjects and PolledData) and
- where to store the collected data

Scheduler

This component takes care of scheduling

- **Periodic data collection**
  Data Collection takes place at specified time intervals. Based on the time interval given, the Scheduler schedules the Data Collection process.
- **Periodic Report generation**
  Based on the settings you specify i.e. which report should be generated when (day and time) and how often, the time of report generation will be scheduled and at the appropriate day and hour, the report will be produced.
- **Tables clean up**
  You can specify the periodicity as to how often you want to delete the tables which hold collected data. If table clean up is not done then the number of tables will increase and soon database will be full.

Filter

The filter otherwise called as Poll Filter allows manipulation of PolledData objects before they are added to the database. The manipulations will be some kind of addition, modification or deletion of PolledData objects .

Decoder

The data collected for the device can be converted into any other format and stored in database. This process of conversion is called Decoding and is taken care of by Data decoder. Normally the collected data is of type Long or String. One of the decoding practice can be that the collected string data which is in the format 3,88,7,9 (denoting memory usage) might be parsed and separated as individual long values like 3 88 7 and 9.

Threshold Rules

The collected data needs some kind of monitoring which is done by applying Threshold rules on the collected data. These rules are nothing but Threshold objects each having a value, type, severity etc associated with it. These threshold objects will be associated with PolledData objects that define what data to collect and thus monitor constantly whenever data gets collected. Any violation in Threshold rule will result in notifications sent to the administrator.

Notification

Notification refers to messages sent to the network users for intimating some aspect of network happening. By default Web NMS Performance module supports notifications via three means:

- **Threshold Notification**
  Whenever collected data exceeds Threshold value, a Threshold event is generated and sent to Fault module which handles it.
- **Collected data Observers (Poll observers)**
  You can set Observers to get informed when data is collected. You might like to do so when you want to do something with data before it gets stored in database.
- **Poll Unit Observers**
  You may wish to receive notifications whenever any change is made in existing Data collection configuration. Poll units are objects which hold definition of what data to collect and from where.

Reporting

The collected data can be grouped into meaningful sets and represented in formatted manner called Reports. You can create your own reports and add it to existing set of reports. Reports can be scheduled to be generated periodically. Many types of reports are provided including " On demand " reports.

Security and Audit

Authorization privileges are available using which the administrator can create user accounts and associate Performance related operations permitted for him. User Based Views can also be created and coupled with permissions on what the user can do on Performance objects.

Programming Interface

This refers to a rich set of API methods which help you in customizing, extending and configuring Performance module to suit your requirements. Javadocs for API methods are also available along with extensive Help documents.

Configuring and Customizing Performance module

You may have your own set of requirements for which you would like to customize and configure Web NMS Performance module. Following gives you the list which you can configure and customize:

1. Data collection
2. Data storage
3. Reports
4. Threshold generation
5. Graphs
6. Distributed Polling
7. Performance Client
8. Managed Object inputs to Performance module

You can do the above listed using **API methods** and using **Configuration files**.

Configuration files

- Configuration files are available in XML format
- You can modify configuration files before Server startup and see the changes
- These are stored under *<Web NMS Home>/conf* directory
- These are updated when settings are changed via Client User Interface

API methods

- These are used when you want to configure Performance objects at runtime
- You are required to get the handle of the API to use it's methods
- ***PollAPI*** is the most importantly used interface to configure Data collection parameters
- PollAPI can be accessed through RMI. When RMI is enabled by running the RMI registry, it will be published with the RMI handle / PollAPI on the server

The following code snippet is used for obtaining the handle for PerformanceAPI.

UNIT V
INTRODUCTION TO APPLICATION LAYER

Introduction:

The application layer implements an interactive request/response protocol. The client reads commands from standard input, converts them into server requests, sends them to the server, and prints the response returned by the server. The server fields client requests and sends back appropriate responses.

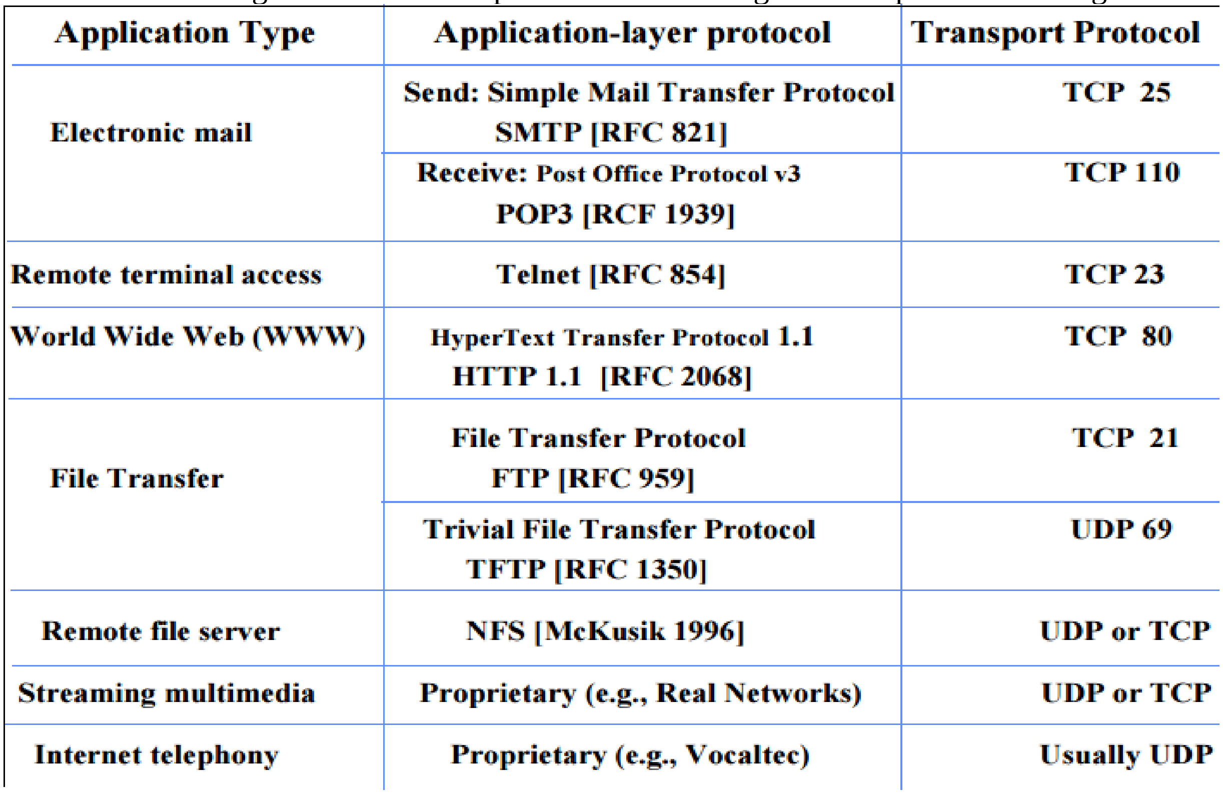
An application layer protocol defines how application processes (clients and servers), running on different end systems, pass messages to each other. In particular, an application layer protocol defines:

The types of messages, e.g., request messages and response messages.

The syntax of the various message types, i.e., the fields in the message and how the fields are delineated.

The semantics of the fields, i.e., the meaning of the information that the field is supposed to contain;

Rules for determining when and how a process sends messages and responds to messages.



Application Type	Application-layer protocol	Transport Protocol
Electronic mail	Send: Simple Mail Transfer Protocol SMTP [RFC 821]	TCP 25
	Receive: Post Office Protocol v3 POP3 [RCF 1939]	TCP 110
Remote terminal access	Telnet [RFC 854]	TCP 23
World Wide Web (WWW)	HyperText Transfer Protocol 1.1 HTTP 1.1 [RFC 2068]	TCP 80
File Transfer	File Transfer Protocol FTP [RFC 959]	TCP 21
	Trivial File Transfer Protocol TFTP [RFC 1350]	UDP 69
Remote file server	NFS [McKusik 1996]	UDP or TCP
Streaming multimedia	Proprietary (e.g., Real Networks)	UDP or TCP
Internet telephony	Proprietary (e.g., Vocaltec)	Usually UDP

.

SMTP (Simple Mail Transfer Protocol):

- ✓ One of the most popular network service is electronic mail (e-mail).
- ✓ The TCP/IP protocol that supports electronic mail on the Internet is called Simple Mail Transfer Protocol (SMTP).
- ✓ SMTP transfers messages from senders' mail servers to the recipients' mail servers using TCP connections.

- ✓ Users based on e-mail addresses.
- ✓ SMTP provides services for mail exchange between users on the same or different computers.

Following the client/server model:

- ▪ SMTP has two sides: a client side which executes on a sender's mail server, and server side which executes on recipient's mail server.
- ▪ Both the client and server sides of SMTP run on every mail server.
- ▪ When a mail server sends mail (to other mail servers), it acts as an SMTP client.
- ▪ When a mail server receives mail (from other mail servers) it acts as an SMTP server.

TELNET (Terminal Network):

- ❖ TELNET is client-server application that allows a user to log onto remote machine and lets the user to access any application program on a remote computer.
- ❖ TELNET uses the NVT (Network Virtual Terminal) system to encode characters on the local system.
- ❖ On the server (remote) machine, NVT decodes the characters to a form acceptable to the remote machine.
- ❖ TELNET is a protocol that provides a general, bi-directional, eight-bit byte oriented communications facility.
- ❖ Many application protocols are built upon the TELNET protocol

FTP (File Transfer Protocol):

- ➢ FTP is the standard mechanism provided by TCP/IP for copying a file from one host to another.
- ➢ FTP differs form other client-server applications because it establishes 2 connections between hosts.
- ➢ FTP is built on a client-server architecture and uses separate control and data connections between the client and the server.
- ➢ One connection is used for data transfer, the other for control information (commands and responses).
- ➢ It transfer data reliably and efficiently.

Multipurpose Internet Mail Extensions (MIME):

- ➢ It is an extension of SMTP that allows the transfer of multimedia messages.
- ➢ If binary data is included in a message MIME headers are used to inform the receiving mail agent:
- ➢ Content-Transfer-Encoding: Header alerts the receiving user agent that the message body has been ASCII encoded and the type of encoding used.
- ➢ Content-Type: Header informs the receiving mail agent about the type of data included in the message.

POP is also called as POP3 protocol:

This is a protocol used by a mail server in conjunction with SMTP to receive and holds mail for hosts.POP3 mail server receives e-mails and filters them into the appropriate user folders. When a user connects to the mail server to retrieve his mail, the messages are downloaded from mail server to the user's hard disk.

HTTP (Hypertext Transfer Protocol):

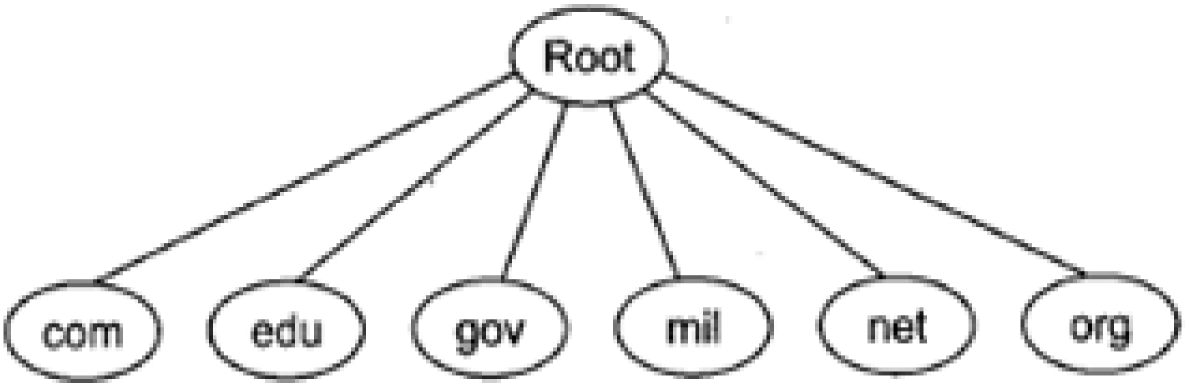
This is a protocol used mainly to access data on the World Wide Web (www).The Hypertext Transfer Protocol (HTTP) the Web's main application-layer protocol although current browsers can access other types of servers A repository of information spread all over the world and linked together. The HTIP protocol transfer data in the form of plain text, hyper text, audio, video and so on.HTTP utilizes TCP connections to send client requests and server replies.

Domain Name System (DNS):

To identify an entity, TCP/IP protocol uses the IP address which uniquely identifies the connection of a host to the Internet.DNS is a hierarchical system, based on a distributed database, that uses a hierarchy of Name Servers to resolve Internet host names into the corresponding IP addresses required for packet routing by issuing a DNS query to a name server.However, people refer to use names instead of address. Therefore, we need a system that can map a name to an address and conversely an address to name.

- ➢ In TCP/IP, this is the domain name system.
- ➢ DNS in the Internet: DNS is protocol that can be used in different platforms.
- ➢ Domain name space is divided into three categories.

Generic Domain: The generic domain defines registered hosts according, to their generic behaviour. Each node in the tree defines a domain which is an index to the domain name space database.

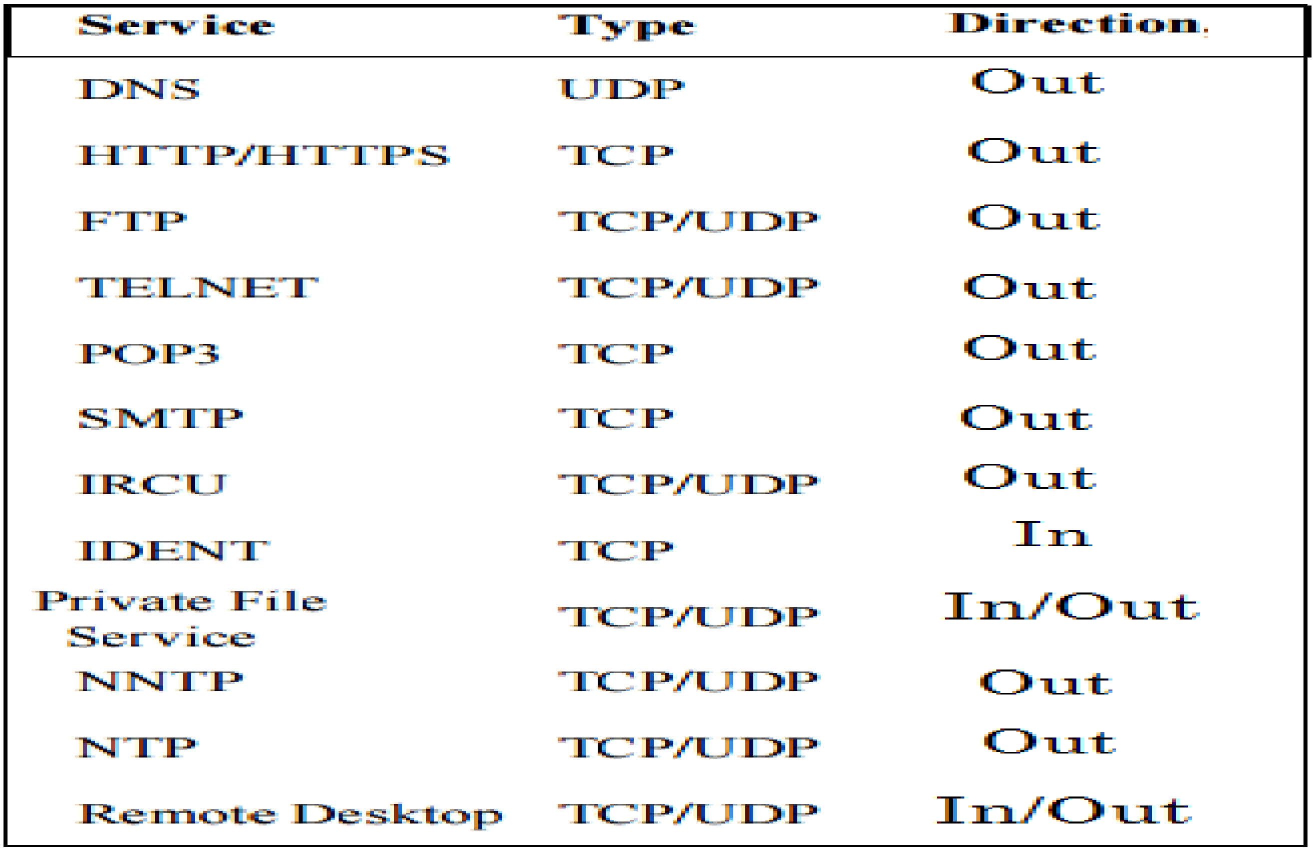


Generic domain

Country Domain: The country domain section follows the same format as the generic domain but uses 2 characters country abbreviations (e.g., US for United States) in place of 3 characters.

Inverse Domain: The inverse domain is used to map an address to a name.

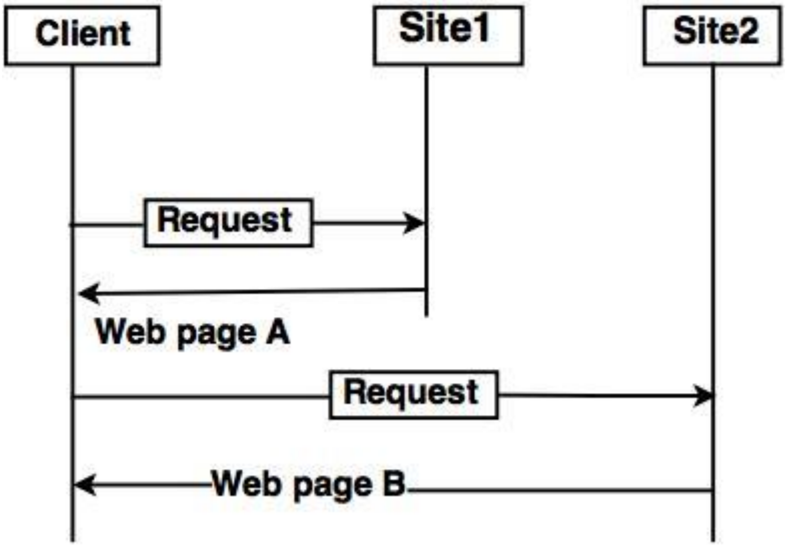
Overview of Services



Service	Type	Direction.
DNS	UDP	Out
HTTP/HTTPS	TCP	Out
FTP	TCP/UDP	Out
TELNET	TCP/UDP	Out
POP3	TCP	Out
SMTP	TCP	Out
IRCU	TCP/UDP	Out
IDENT	TCP	In
Private File Service	TCP/UDP	In/Out
NNTP	TCP/UDP	Out
NTP	TCP/UDP	Out
Remote Desktop	TCP/UDP	In/Out

World Wide Web: WWW is a set of programs, standards and protocols that allow the text, images, animations, sounds and videos to be stored, accessed and linked together in form of web sites. The WWW project was developed at CERN, the European Center for Nuclear Research in 1989.It has a unique combination of flexibility, portability, and user-friendly features that distinguishes it from the other services provided by the Internet.

- The World Wide Web (WWW) is a collection of documents and other web resources which are identified by URLs, interlinked by hypertext links, and can be accessed and searched by browsers via the Internet.
- World Wide Web is also called the Web and it was invented by Tim Berners-Lee in 1989.
- Website is a collection of web pages belonging to a particular organization.
- The pages can be retrieved and viewed by using browser.



Architecture of WWW

Let us go through the scenario shown in above fig.

- The client wants to see some information that belongs to site 1.
- It sends a request through its browser to the server at site 2.
- The server at site 1 finds the document and sends it to the client.

www architecture:WWW is basically a distributed client-server service. It this, a client can access the services from a server using a browser.

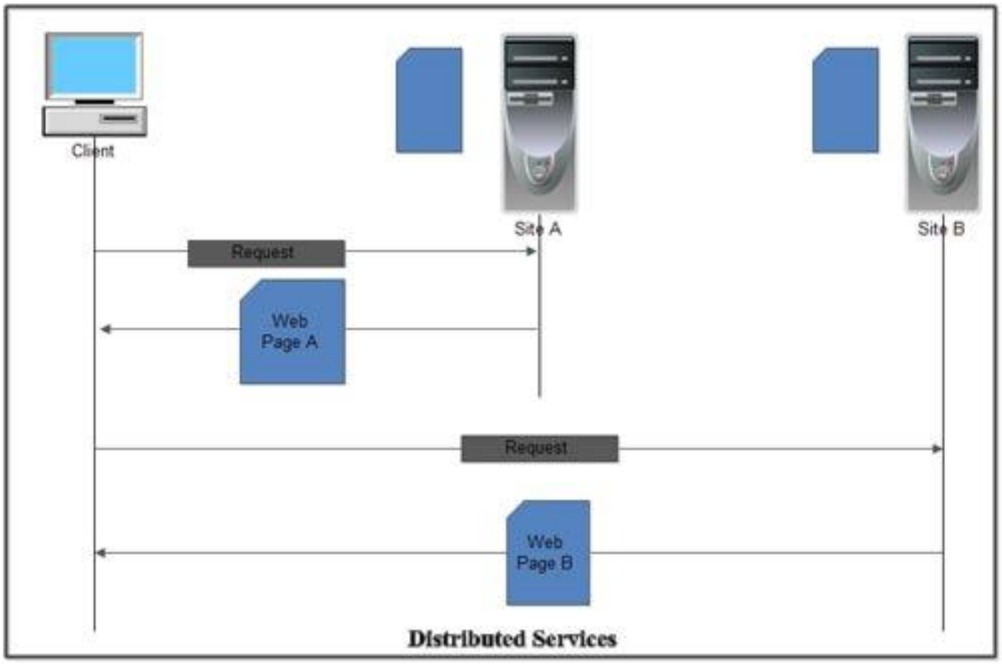
• These services are usually distributed over many locations called sites or websites.

• From the user's point of view web consists of a vast worldwide collection of documents called web pages. These web pages reside on different sites or machines all over the world.

• Each web page can contain link to other pages any where in the world. By clicking on such link user can access another web page.

• This kind of link can be in form of string of text or picture, sound, movie clip etc.

• Such a text or image that enables the user to link to another web page is called hyperlink.



- The string of text that points to another web page is called hypertext. The difference between the normal text and hypertext is that, when you take the mouse pointer over it, it changes into a hand shaped cursor. Such a text is sometime, underlined and blue is colour.
- Hypermedia is enhanced form of a hyperlink which not only links to the other pages or other sections within the same page but can also link with various medium like sound, animation, movie clip etc, Hypermedia is grouping of different media like sound, graphics, animations and text in a single file.
- These hyperlinks are created with the help of specialized language called Hypertext Mark up Language (HTML).
- In order to access these web pages on different sites, each of these pages has a specific address called Uniform Resource Locator (URL).
- Web pages are viewed with a program called a browser.

Client (Browser):

- Web browser is a program, which is used to communicate with web server on the Internet.
- Each browser consists of three parts: a controller, client protocol and interpreter.
- The controller receives input from input device and use the programs to access the documents.
- After accessing the document, the controller uses one of the interpreters to display the document on the screen.

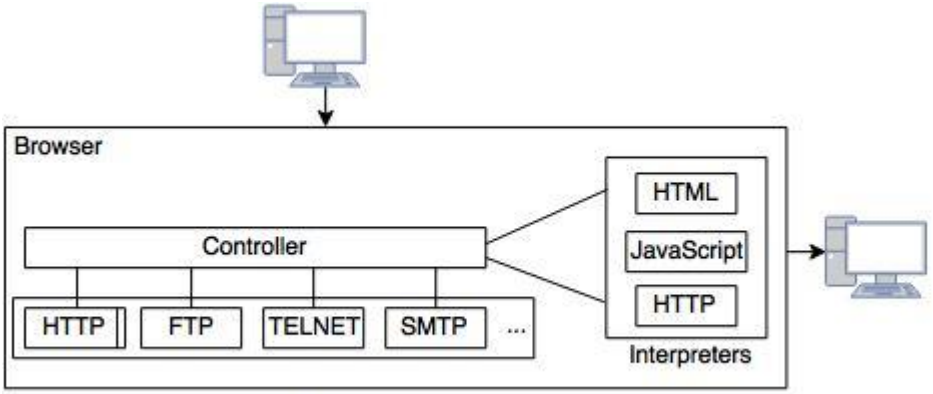


Fig: Cllient (Browser)

Server:

- A computer which is available for the network resources and provides service to the other computer on request is known as server.
- The web pages are stored at the server.
- Server accepts a TCP connection from a client browser.
- It gets the name of the file required.
- Server gets the stored file. Returns the file to the client and releases the top connection.

Uniform Resource Locater (URL)

- The URL is a standard for specifying any kind of information on the Internet.
- The URL consists of four parts: protocol, host computer, port and path.
- The protocol is the client or server program which is used to retrieve the document or file. The protocol can be ftp or http.
- The host is the name of computer on which the information is located.
- The URL can optionally contain the port number and it is separated from the host name by a colon.
- Path is the pathname of the file where the file is stored.

SNMP (Simple Network Management Protocol):

A large part of being a system administrator is collecting accurate information about your servers and infrastructure. There are a number of tools and options for gathering and processing this type of information. Many of them are built upon a technology called **SNMP**.

SNMP stands for simple network management protocol. It is a way that servers can share information about their current state, and also a channel through which an administer can modify pre-defined values. While the protocol itself is very simple, the structure of programs that implement SNMP can be very complex.

In this guide, we will introduce you to the basics of the SNMP protocol. We will go over its uses, the way that the protocol is typically used in a network, the differences in its protocol versions, and more.

Basic Concepts

SNMP is a protocol that is implemented on the application layer of the networking stack (click here to learn about networking layers). The protocol was created as a way of gathering information from very different systems in a consistent manner. Although it can be used in connection to a diverse array of systems, the method of querying information and the paths to the relevant information are standardized.

There are multiple versions of the SNMP protocol, and many networked hardware devices implement some form of SNMP access. The most widely used version is SNMPv1, but it is in many ways insecure. Its popularity largely stems from its ubiquity and long time in the wild. Unless you have a strong reason not to, we recommend you use SNMPv3, which provides more advanced security features.

In general, a network being profiled by SNMP will mainly consist of devices containing SNMP **agents**. An agent is a program that can gather information about a piece of hardware, organize it into predefined entries, and respond to queries using the SNMP protocol.

The component of this model that queries agents for information is called an SNMP **manager**. These machines generally have data about all of the SNMP-enabled devices in their network and can issue requests to gather information and set certain properties.

SNMP Managers

An SNMP manager is a computer that is configured to poll SNMP agent for information. The management component, when only discussing its core functionality, is actually a lot less complex than the client configuration, because the management component simply requests data.

The manager can be any machine that can send query requests to SNMP agents with the correct credentials. Sometimes, this is implemented as part of a monitoring suite, while other times this is an administrator using some simple utilities to craft a quick request.

Almost all of the commands defined in the SNMP protocol (we will go over these in detail later) are designed to be *sent* by a manager component. These include GetRequest, GetNextRequest, GetBulkRequest, SetRequest, InformRequest, and Response. In addition to these, a manager is also designed to *respond to* Trap, and Response messages.

SNMP Agents

SNMP agents do the bulk of the work. They are responsible for gathering information about the local system and storing them in a format that can be queried.updating a database called the "management information base", or **MIB**.

The MIB is a hierarchical, pre-defined structure that stores information that can be queried or set. This is available to well-formed SNMP requests originating from a host that has authenticated with the correct credentials (an SNMP manager).

The agent computer configures which managers should have access to its information. It can also act as an intermediary to report information on devices it can connect to that are not configured for SNMP traffic. This provides a lot of flexibility in getting your components online and SNMP accessible.

SNMP agents respond to most of the commands defined by the protocol. These include GetRequest, GetNextRequest, GetBulkRequest, SetRequest and InformRequest. In addition, an agent is designed to send Trap messages.

Understanding the Management Information Base

The most difficult part of the SNMP system to understand is probably the **MIB**, or management information base. The MIB is a database that follows a standard that the manager and agents adhere to. It is a hierarchical structure that, in many areas, is globally standardized, but also flexible enough to allow vendor-specific additions.

The MIB structure is best understood as a top-down hierarchical tree. Each branch that forks off is labeled with both an identifying number (starting with 1) and an identifying string that are unique for that level of the hierarchy. You can use the strings and numbers interchangeably.

To refer to a specific node of the tree, you must trace the path from the unnamed root of the tree to the node in question. The lineage of its parent IDs (numbers or strings) are strung together, starting with the most general, to form an address. Each junction in the hierarchy is represented by a dot in this notation, so that the address ends up being a series of ID strings or numbers separated by dots. This entire address is known as an object identifier, or **OID**.

Hardware vendors that embed SNMP agents in their devices sometimes implement custom branches with their own fields and data points. However, there are standard MIB branches that are well defined and can be used by any device.

The standard branches we will be discussing will all be under the same parent branch structure. This branch defines information that adheres to the MIB-2 specification, which is a revised standard for compliant devices.

The base path to this branch is:

1.3.6.1.2.1

This can also be represented in strings like:

iso.org.dod.internet.mgmt.mib-2

The section 1.3.6.1 or iso.org.dod.internet is the OID that defines internet resources. The 2 or mgmtthat follows in our base path is for a management subcategory. The 1 or mib-2 under that defines the MIB-2 specification.

This is a great resource for familiarizing yourself with the MIB tree. This particular page represents the connecting nodes at the junction we have been talking about. You can check what is further up and down the tree by checking out the "superior" and "subsidiary" references respectively.

Another similar tool is a SNMP Object Navigator provided by Cisco. This can be used to drill down into the hierarchy to find information you need. A similar tree is provided by SolarWinds.

Basically, if we want to query our devices for information, most of the paths will begin with 1.3.6.1.2.1. You can browse the tree interfaces to learn what kind of information is available to query and set.

SNMP Protocol Commands

One of the reasons that SNMP has seen such heavy adoption is the simplicity of the commands available. There are very few operations to implement or remember, but they are flexible enough to address the utility requirements of the protocol.

The following PDUs, or protocol data units, describe the exact messaging types that are allowed by the protocol:

- **Get**: A Get message is sent by a manager to an agent to request the value of a specific OID. This request is answered with a Response message that is sent back to the manager with the data.
- **GetNext**: A GetNext message allows a manager to request the next sequential object in the MIB. This is a way that you can traverse the structure of the MIB without worrying about what OIDs to query.
- **Set**: A Set message is sent by a manager to an agent in order to change the value held by a variable on the agent. This can be used to control configuration information or otherwise modify the state of remote hosts. This is the only write operation defined by the protocol.

- **GetBulk**: This manager to agent request functions as if multiple GetNext requests were made. The reply back to the manager will contain as much data as possible (within the constraints set by the request) as the packet allows.
- **Response**: This message, sent by an agent, is used to send any requested information back to the manager. It serves as both a transport for the data requested, as well as an acknowledgement of receipt of the request. If the requested data cannot be returned, the response contains error fields that can be set with further information. A response message must be returned for any of the above requests, as well as Inform messages.
- **Trap**: A trap message is generally sent by an agent to a manager. Traps are asynchronous notifications in that they are unsolicited by the manager receiving them. They are mainly used by agents to inform managers of events that are happening on their managed devices.
- **Inform**: To confirm the receipt of a trap, a manager sends an Inform message back to the agent. If the agent does not receive this message, it may continue to resend the trap message.

With these seven data unit types, SNMP is capable of querying for and sending information about your networked devices.

Protocol Versions

The SNMP protocol has gone through many changes since it was first introduced. The initial spec was formulated with RFC 1065, 1066, and 1067 in 1988. By the simple fact that it has been around so long, this version is still widely supported. However, there are many security issues with the protocol, including authenticating in plain text, so its use is highly discouraged, especially when used on unprotected networks.

Work on version 2 of the protocol was initiated in 1993 and offers some substantial improvements on the earlier standard. Included in this version was a new "party-based" security model meant to address the security issues inherent with the prior revision. However, the new model was not very popular because it was difficult to understand and implement.

Because of this, a few "spin-offs" of version 2 were created, each of which kept the bulk of the version 2 improvements, but swapped out the security model. In SNMPv2c, community-based authentication, the same model used in v1, was reintroduced. This was the most popular version of the v2 protocol. Another implementation, called SNMPv2u, uses user-based security, although this was never very popular. This allowed for per-user authentication settings.

In 1998, the third (and current) version of the SNMP protocol entered as a spec proposal. From a user's perspective, the most relevant change was the adoption of a user-based security system. It allows you to set a user's authentication requirements as one of these models:

- **NoAuthNoPriv**: Users connecting with this level have no authentication in place and no privacy of the messages they send and receive.
- **AuthNoPriv**: Connections using this model must authenticate, but messages are sent without any encryption.

- **AuthPriv**: Authentication is required and messages are encrypted.
  In addition to authentication, an access control mechanism was implemented to provide granular control over which branches a user can access. Version 3 also has the ability to leverage the security provided by the transport protocols, such as SSH or TLS