chapter 1 the nETWORK and its elements

1.1 a little bit of history

Before the advent of computer networks that were based upon some type of telecommunications system, communication between early computers was performed by human users by carrying instructions between them. This happened until September 1940, when George Stibitz used a teletype machine to send instructions for a problem set from his Model at Dartmouth College in New Hampshire to his Complex Number Calculator in New York and received results back by the same means.

Linking output systems like teletypes to computers was an interest at the Advanced Research Projects Agency (ARPA) when, in 1962, J.C.R. Licklider was hired and developed a working group he called the "Intergalactic Network", a precursor to the ARPANet.

ARPANet was a project within the Advanced Research Projects Agency of the US Department of Defense (also known as DARPA – Defense Advanced Research Projects Agency). (D)ARPA itself was established in 1958 as response to the Soviet Union launching of the first satellite in 1957. This project grew up from the necessity of interconnecting in a reliable manner different networking systems and was based on packet switching. Previously, data communication was based on circuit switching where a dedicated circuit is used for the communication needs of the entities at the end points of the communication channel.

In 1964, researchers at Dartmouth developed the Dartmouth Time Sharing System for distributed users of large computer systems. The same year, at MIT, a research group supported by General Electric and Bell Labs used a DEC computer to route and manage telephone connections.

Throughout the 1960s Leonard Kleinrock, Paul Baran and Donald Davies independently conceptualized and developed network systems which used datagrams or Packet information technology that could be used in a network between computer systems.

In 1965 Thomas Merrill and Lawrence G. Roberts created the first wide area network (WAN).

The first widely used PSTN (Public Switched Telephone Network) switch that used true computer control was the one Western Electric introduced in 1965.

In 1969 the University of California at Los Angeles, SRI (Stanford Research Institute), University of California at Santa Barbara and the University of Utah were connected, setting the foundation of the ARPANet network, using 50 kbit/s circuits.

ARPANet became the technical core of what would become the Internet, and a primary tool in developing the technologies used. ARPANet development was centered around the Request for Comments (RFC) process, still used today for proposing and distributing Internet Protocols and Systems.

RFC 675 - *Specification of Internet Transmission Control Program*, by Vinton Cerf, Yogen Dalal and Carl Sunshine, Network Working Group, December, 1974, contains the first attested use of the term ***internet***, as a shorthand for ***internetworking***; later RFCs repeat this use, so the word started out as an adjective rather than the noun it is today.

As interest in wide spread networking grew and new applications for it were developed, the Internet's technologies spread throughout the rest of the world. The network-agnostic approach in TCP/IP meant that it was easy to use any existing network infrastructure, such as the IPSS X.25 network, to carry Internet traffic. In 1984, University College London replaced its transatlantic satellite links with TCP/IP over IPSS (International Packet Switched Service).

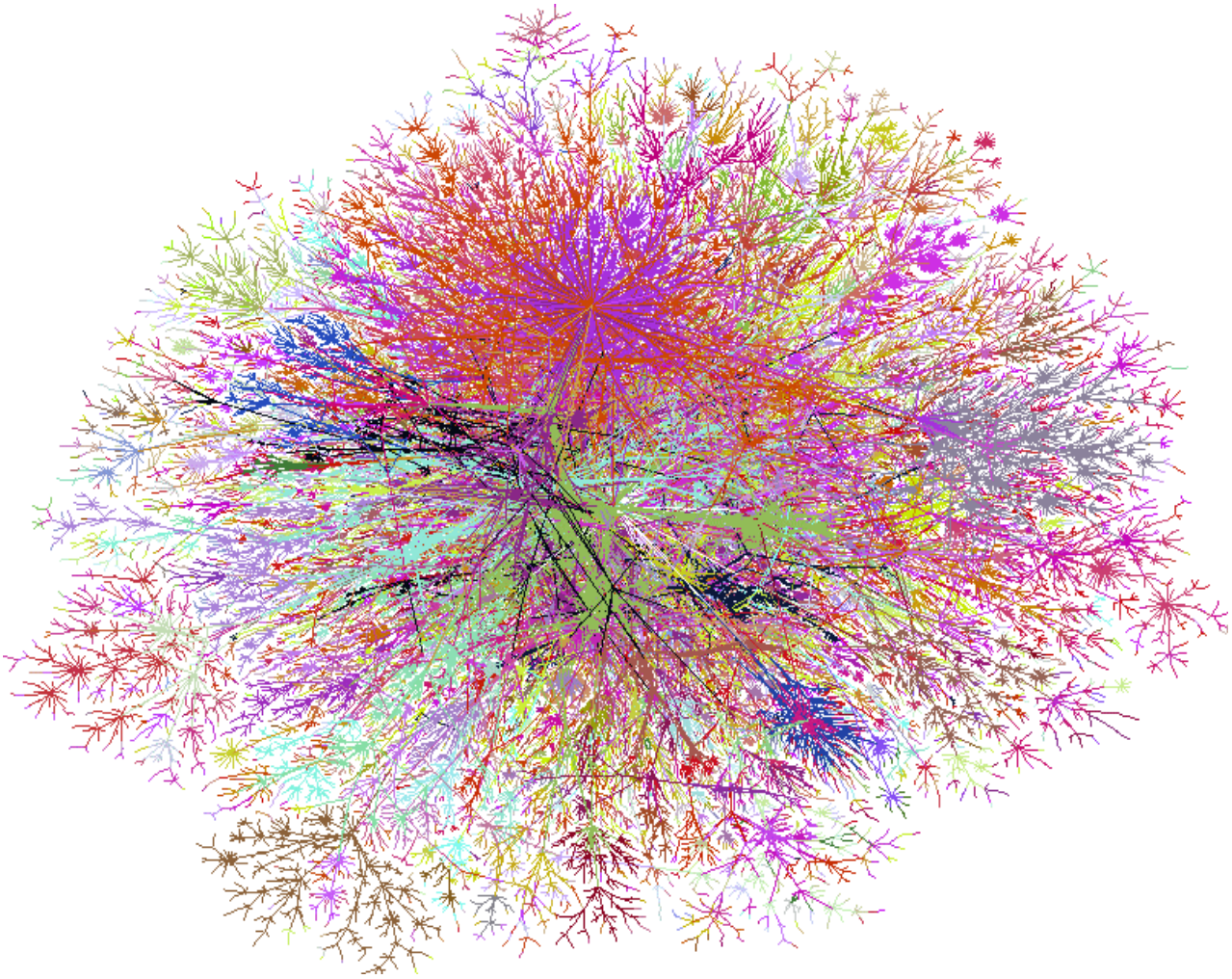
Between 1984 and 1988 CERN began installation and operation of TCP/IP to interconnect its major internal computer systems, workstations, PCs and an accelerator control system. There was considerable resistance in Europe towards more widespread use of TCP/IP and the CERN TCP/IP intranets remained isolated from the Internet until 1989.

The Internet began to penetrate Asia in the late 1980s. Japan, which had built the UUCP (Unix to Unix

Copy Protocol)-based network JUNET in 1984, connected to NSFNet in 1989. It hosted the annual meeting of the Internet Society, INET'92, in Kobe. Singapore developed TECHNET in 1990, and Thailand gained a global Internet connection between Chulalongkorn University and UUNET in 1992.

1.2 the nETWORK in pictures

A global network representation can be viewed courtesy of the Internet Mapping Project. The nodes in this graph represent Autonomous Systems (ASs). Black areas are .mil sites.



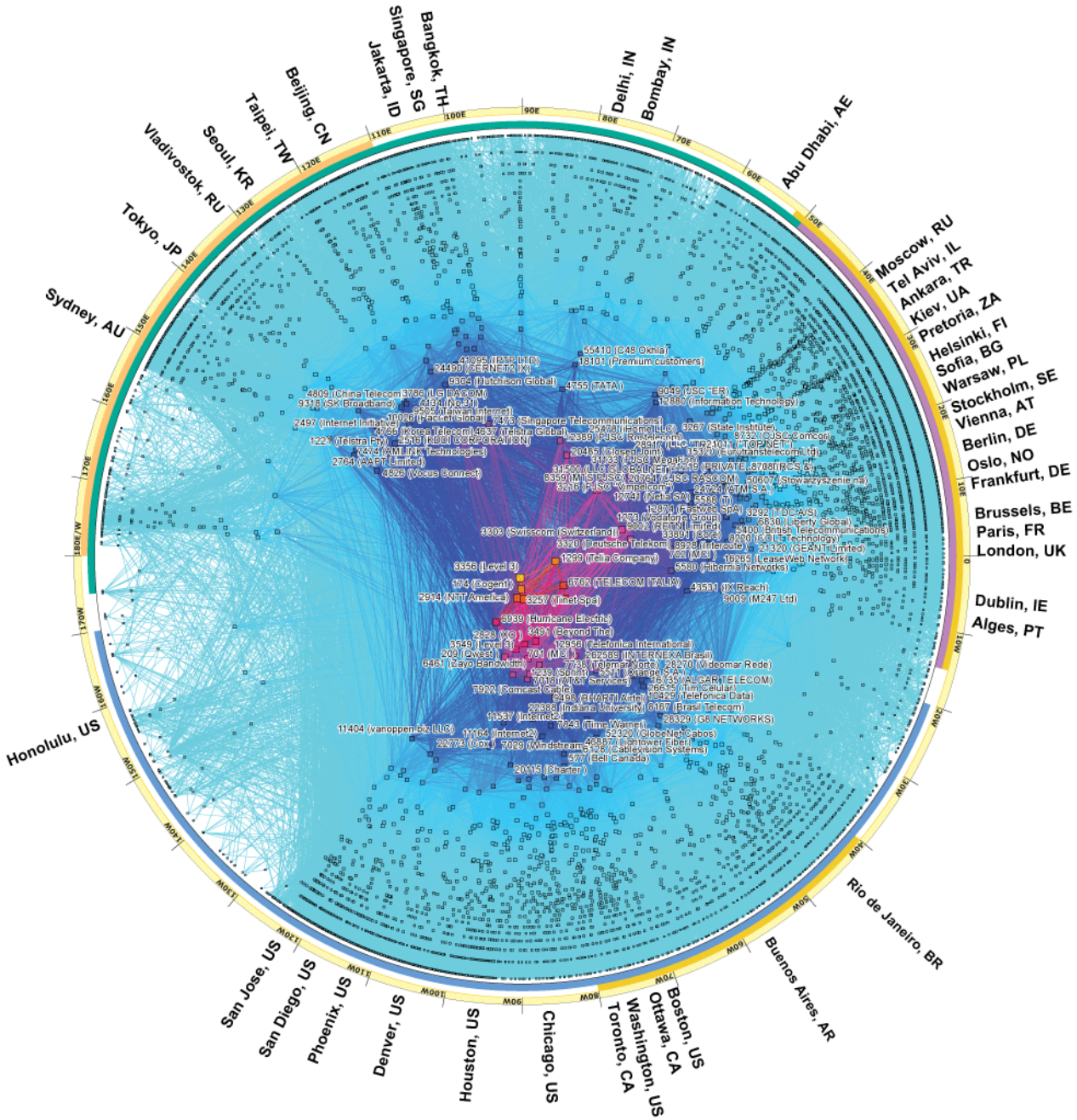
An **Autonomous System** (AS), the unit of routing policy, is an administrative routing domain that can apply its own policy, which is a result of a mutual commercial agreement between autonomous systems. A university, an ISP, or a large company network can own an AS.

When it comes to the geographic distribution of autonomous systems, the pictures below, which represent both the IPv4 and IPv6 topologies, are quite relevant. They are also indicative of o general trend in the evolution of the IP version 6 support.

One must say that the overall penetration level of IPv6 is (based on 21.02.2021 data from Google's IPv6 statistics) at about 32.72 %. Romania has a modest 19.7% IPv6 usage level (as of 21.02.2021). Top five are

India (?!), Belgium, Germany, Malaysia, Greece (?!). The dynamic is, of course, on the IPv6 side. CAIDA (Center for Applied Internet Data Analysis) data shows that IPv6 saw greater relative growth then IPv4, with 23% more ASs and 29% more AS links. In IPv4 the growth was 7% more ASs and 17% links.

Below is presented a snapshot of the IPv4 internet topology as of February 2017, provided by CAIDA. For a historical view of the evolving internet topology, check - http://www.caida.org/research/topology/as_core_network/historical.xml



The IPv4 topology data was obtained by using traceroutes to random destinations, on February 2017 no 2018-2020 data). It contains 50 millions /24 IPv4 networks. The resulting AS topology contains 47, 610 ASs and 148k links. This data was collected from 121 monitors, located in 42 countries from all continents.

This image does not include quite a few private or corporate network, but are still very relevant to the

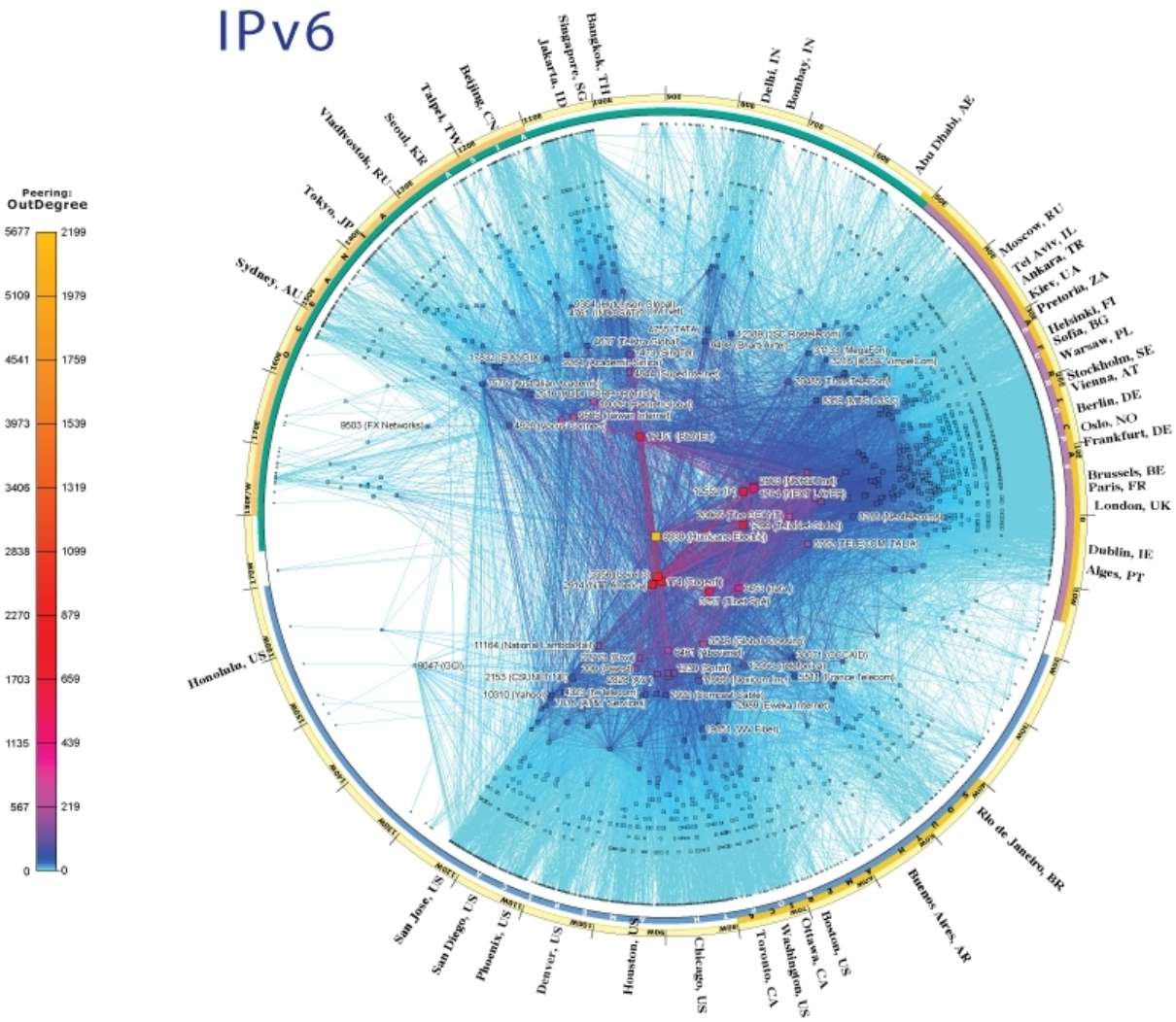
expansion level of the global communication network.

We conclude this internet gallery with the IPv6 internet topology map, from Jan. 2015, picture provided by CAIDA, as well.

In absolute values, the Jan. 2015 figures for IPv6 autonomous systems, as provided by CAIDA, are as follows - 76.1% of globally routeable network prefixes. It shows routes to 4.9 millions IPv6 addresses.

This data was collected from 47 monitors, located in 25 countries from all continents.

More details on the Jan 2015 status at http://www.caida.org/research/topology/as_core_network/



1.3 the infrastructure of the nETWORK

We are tempted to view the network as a set of nodes connected by wires, or, with some imagination effort, the connections can be perceived as wireless. However, beneath this schematic approach belies an entire system of high complexity. And that's because we also need software to make this net work and we also must enforce rules to make sure that the net works properly. Therefore, it makes sense to list the basic

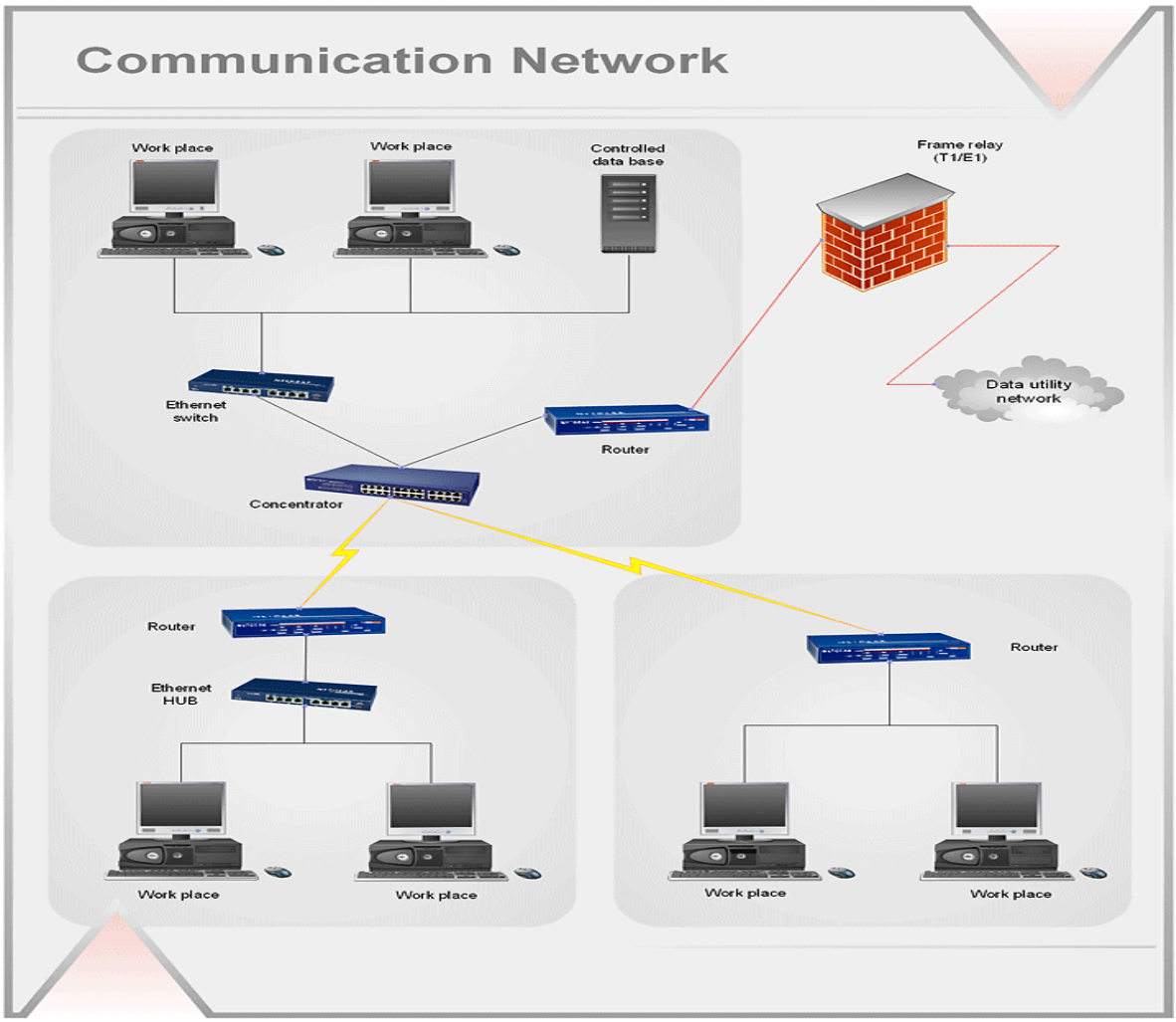
elements of the nETWORK as follows:

- network hardware
- network software
- communication protocols

The structural and functional relationship between these elements represents the **infrastructure** of the network. In a more detailed approach, we can divide the network elements into:

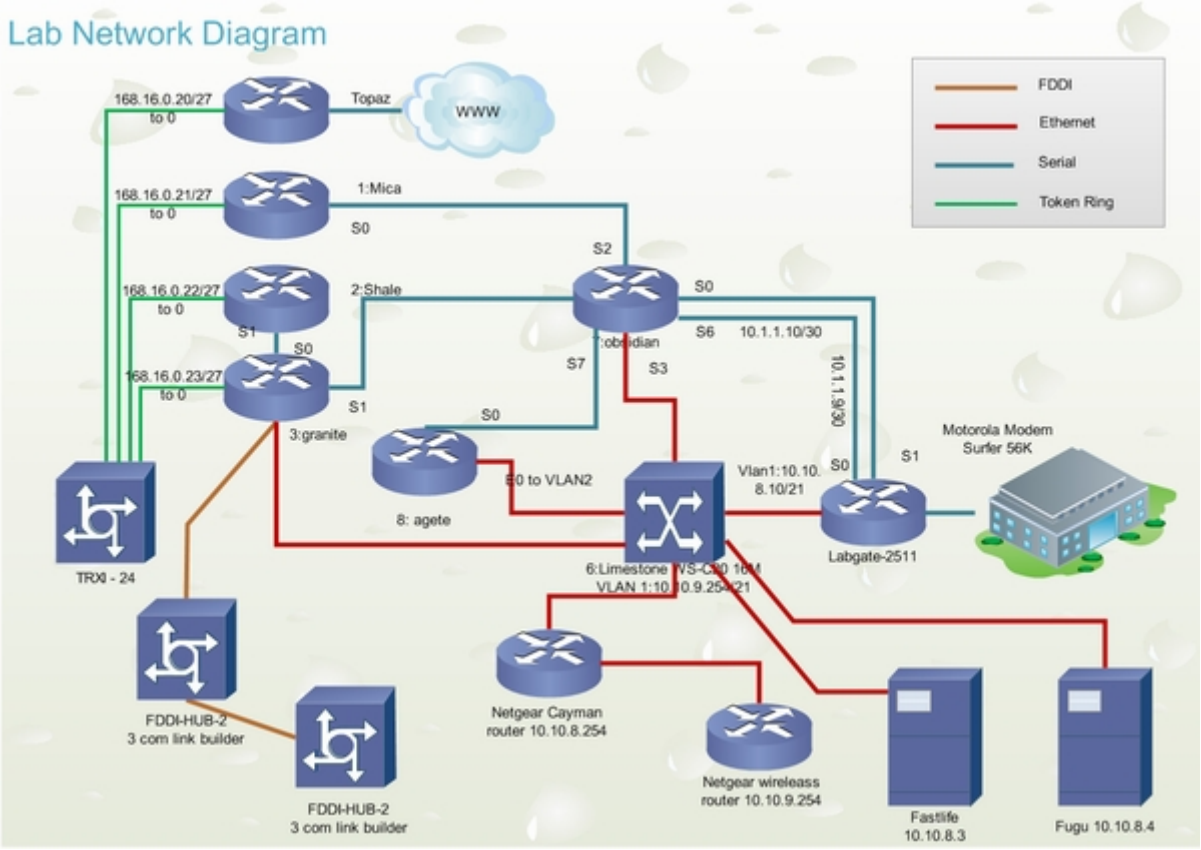
1. **End devices**, or hosts - the sources and destinations of the communication (i.e., devices that generate and transmit messages or receive and interpret messages). These devices act as interface between the end users and the network: Computers, IP phones, mobile phones, PDAs, etc.
2. **Intermediary devices -** devices that give network access to the attached end devices and transport the messages between hosts. These devices accomplish communication functions that ensure the success of the communication process. Examples: hubs, switches, routers, modems, firewalls, etc.
3. **Transmission media -** the physical media that connects the devices, enabling the exchange of messages between them. It may be wired, that is, some copper cable or optical fiber cable, or wireless, that is, some radio link (allows the propagation of electromagnetic waves).
4. **Services -** network-aware software applications (e.g., a web browser) that request network resources (e.g., data) in order to enjoy the end user of the application some provided service.
5. **Processes** – software that runs on network devices in order to support the communication functions - in accordance with the established, also in software, communication rules or protocols - and facilitate the provision of services to the end users. Processes are transparent to the end users.
6. **Messages**: Well-known applications. Includes telephone calls, e-mail, web pages, etc.

1.4 typical representations of a network area



The first image shows a typical office network. The workstations in the bottom left area are connected together through a hub and with the rest of the world through a router. The top left workstations are connected to a switch, then to a concentrator. The concentrator connects through a router to another network, all communications with the public area occurring through a firewall. The symbols used in this diagram are generic.

On the other side, the diagram below, that of a lab, uses the Cisco icons for the representation of network devices and network links. For information regarding the Cisco icons and conventions, check the site



1.5 modeling the network - network devices and their interfaces

When we try to identify ourselves within the nETWORK, we usually run a commands like ipconfig /all (on ms windows platforms). Apart from information related to the computer we work on, the command displays a list of IP related devices – in our example – a wireless adapter, a network card and an ADSL modem, devices which allow us to communicate to a network. Each of these devices exhibit a physical address – six bytes in hexadecimal representation and, for two of them, for which the connection is active, an IP address. These devices, which are, to some extent, part of our workstation, serves as a network interface. This clarifies a first issue, IP addresses are not for computers, as the general belief goes, but for interfaces, like network cards, wireless adapters, firewire ports. Also, interfaces serve as end points for communication links, which may be UTP cables, coaxial cables or radio waves. To make a distinction between a workstation (or another physical network element) and its interfaces, a physical device consisting of a processing unit with one or more interfaces will be called a **network device**.

A **network graph** is a directed multigraph graph NG = (V, A, P_V) whose vertices (nodes) correspond to network devices, the node (vertex) ports are the interfaces of the device and the arcs correspond to physical (they may be radio waves, as well) communication links.

For more details on network graphs, check the notes at:

https://staff.fmi.uvt.ro/~stelian.mihalas/com_net/courses/networks.pdf

```
C:\Documents and Settings\sm>ipconfig /all

Windows IP Configuration

        Host Name . . . . . . . . . . . . : hp8530w
        Primary Dns Suffix  . . . . . . . :
        Node Type . . . . . . . . . . . . : Mixed
        IP Routing Enabled. . . . . . . . : Yes
        WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Wireless Network Connection:

        Media State . . . . . . . . . . . : Media disconnected
        Description . . . . . . . . . . . : Intel(R) Wireless WiFi Link 5300
        Physical Address. . . . . . . . . : 00-16-EA-A2-BC-12

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . :
        Description . . . . . . . . . . . : Intel(R) 82567LM Gigabit Network Con
nection
        Physical Address. . . . . . . . . : 00-23-7D-E7-B9-5A
        Dhcp Enabled. . . . . . . . . . . : No
        IP Address. . . . . . . . . . . . : 192.168.0.2
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.0.1
        DNS Servers . . . . . . . . . . . : 193.231.100.134
                                            193.231.100.130

Ethernet adapter Local Area Connection 3:

        Connection-specific DNS Suffix  . :
        Description . . . . . . . . . . . : SpeedTouch Ethernet Adapter #2
        Physical Address. . . . . . . . . : 00-0E-50-D6-A4-53
        Dhcp Enabled. . . . . . . . . . . : Yes
        Autoconfiguration Enabled . . . . : Yes
        Autoconfiguration IP Address. . . : 169.254.95.117
        Subnet Mask . . . . . . . . . . . : 255.255.0.0
        Default Gateway . . . . . . . . . :

C:\Documents and Settings\sm>
```

1.6 NICs

A **Network Interface Controller (NIC)** or **network card** is a hardware device that handles an interface to a computer network and allows a network-capable device to access that network. The NIC exists on both the Physical and the Data Link layers of the OSI model. At layer 3 (Network) level NICs are identified by their IP (version 4 or 6) addresses.

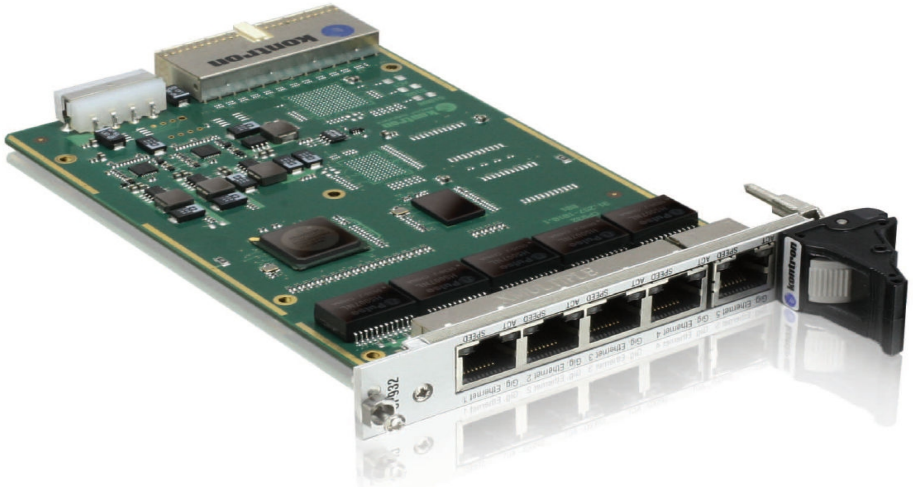
Each NIC can be identified by a unique number, its **MAC (Media Access Control)** address (identifier) which is burned into its **ROM (Read Only Memory)**. This address is globally unique, no two NICs may have the same address (identifier). The first 24 bits of the MAC address represent the so-called **Organizationally Unique Identifier (OUI)** and is manufacturer specific. Mac addresses are usually represented in hexadecimal format, as 6, 2-digit hexadecimal numbers, separated by ":" or "-". A couple of examples:

- 00-0D-60-F1-05-A7 - identifies an IBM (OUI = 00-0D-60) network card
- 00:30:6E:3B:ED:C3 - identifies an HP (OUI = 00:30:6E) network card

The current speed ratings for NICs range from 10Mbps (the original ethernet specification), 100Mbps for Fast ethernet and the new de facto standard of 1000Mbps or Gigabit ethernet. The 10Gbps NICs are into mass production, as well, especially for network backbones.

The actual connector to a NIC may vary from RJ45 (by far, the most widespread) to BNC (coaxial cable)

or no physical connector at all, in the case of a wireless NIC.



A Gigabit network interface card

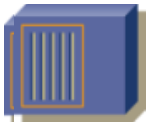
1.7 hubs

A **network hub** or **repeater hub** is a device for connecting multiple twisted pair (TP) or fiber optic Ethernet devices and making them act as a single **network segment**. Hubs work at the Physical layer (layer 1) of the OSI model. This device is a form of multi-port repeater.

The IBM 8242 Ethernet Hub

Hubs do not manage the traffic that comes through them; any packet entering any port is broadcast out on all other ports. Since every packet is being sent out through all other ports, packet collisions may result, affecting the efficiency of the data traffic. Hubs represent a cheap solution for interconnecting computers in small networks.

The Cisco icon for generic hubs is shown below.



1.8 repeaters

A **repeater** is an electronic device that receives a signal and retransmits it at a higher level and/or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances. Basicly, network **repeaters** regenerate incoming electrical, wireless or optical signals.

The Gefen (GTV-ETH-2-COAX) Repeater

Repeaters attempt to preserve signal integrity and extend the distance over which data can safely travel.

Actual network devices that serve as repeaters usually have some other name. **Active hubs**, for example, are repeaters. Active hubs are sometimes also called **multiport repeaters**, but more commonly they are just hubs. Other types of passive hubs are not repeaters.

Repeaters can be found both in wired and wireless networks.

Repeaters are used to lengthen individual network segments to form a larger extended network. That is, in the case of a wired network, repeaters allow a network to be constructed that exceeds the size limit of a single physical segment by allowing additional lengths of cable to be connected. There are some constraints, however. For a repeater to be used, both network segments must be identical-same network protocols for all layers, same media access control method, and the same physical transmission technique.

Repeaters do not have traffic management functions. They do not isolate collision domains or broadcast domains. The Cisco icon for a generic repeater is shown below.

1.9 bridges

Bridges are, in a sense, scaled down switches. Bridges make a simple do/don't decision on which packets to send towards the segments they connect. Filtering is done based on the destination address of the packet. If packet's destination is on the same segment where it originated, it is not forwarded. If it is destined for a station on another LAN, it is connected to a different bridge port and forwarded to that port.

A **network bridge** connects multiple network segments at the Data Link layer (layer 2) of the OSI model.

Using bridges over hubs or switches has several advantages:

- Isolate the collision domains
- Simple bridges are rather inexpensive
- Reduce the size of collision domain by microsegmentation in non-switched networks

- Transparent to protocols above the MAC layer
- Allow the introduction of management/performance information and access control
- LANs interconnected are separate, and physical constraints such as number of stations, repeaters and segment length don't apply
- Help minimize bandwidth usage

On the other side, bridges have several disadvantages:

- Do not scale well to extremely large networks
- Do not limit (isolate) the scope of broadcasts (do not split the broadcast domain)
- Buffering and processing introduces delays
- Bridges are more expensive than repeaters or hubs

1.10 switches

A **network switch** is a network device that connects different **network segments**. Switches operate mostly at the Data Link layer (layer 2) of the OSI model. Switches which operate at Network level or above are called **multilayer switches**.

There are three distinct functions of layer 2 switching:

- address learning
- forward/filter decisions
- loop avoidance

A rack mounted 24-port 3Com switch

Although they do not isolate broadcast domains, switches isolate collision domain, significantly improving data traffic efficiency.

Switches are used to physically connect devices together. Multiple cables can be connected to a switch to enable networked devices to communicate with each other. Switches manage the flow of data across a network by only transmitting a received message to the device for which the message was intended. Each networked device connected to a switch can be identified using a MAC address, allowing the switch to regulate the flow of traffic. This maximizes security and efficiency of the network.

Because of these features, a switch is often considered more "intelligent" than a network hub. Hubs neither provide security, or identification of connected devices. This means that messages have to be

transmitted out of every port of the hub, greatly degrading the efficiency of the network

Mid-to-large sized LANs contain a number of linked managed switches. Small office/home office configurations contain, in general, a single switch or an all-purpose convergence device such as a gateway, which provides access to small office/home broadband services such an ADSL router or a Wi-Fi router.

In switches intended for commercial use, built-in or modular interfaces make it possible to connect different types of networks, including Ethernet, Fibre Channel, RapidIO, ATM, ITU-T G.hn and 802.11.

The icon for Cisco switches is shown below.

1.11 routers

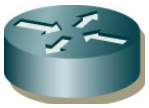
A **router** is a network device used to forward data between **computer networks**. From a functional point of view, a router acts as a Network layer (layer 3) switch. In terms of network topology, routers isolate (split) both the collision domain and the broadcast domain. Routers operate in two different planes:

- the **control plane** - in which the router learns the outgoing interface that is most appropriate for forwarding specific packets to specific destinations
- the **forwarding plane** - responsible for the actual process of sending a packet received on a logical interface to an outbound logical interface.

Routers may provide connectivity within enterprises, between enterprises and the Internet, or between internet service providers' (ISPs) networks. The largest routers (such as the Cisco CRS-1 or Juniper T1600) interconnect the various ISPs, or may be used in large enterprise networks. Smaller routers usually provide connectivity for typical home and office networks. Other networking solutions may be provided by a backbone Wireless Distribution System (WDS), which avoids the costs of introducing networking cables into buildings.

The Cisco 1861 router

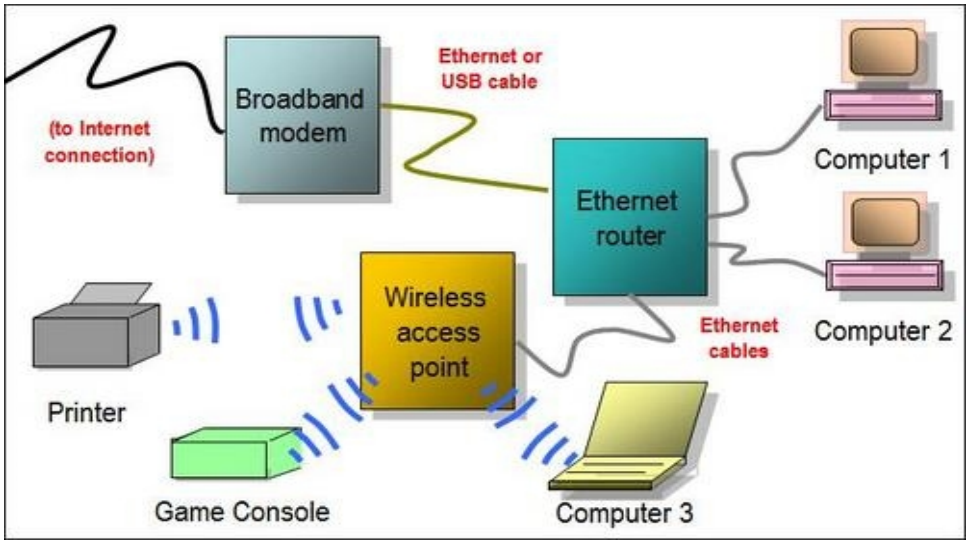
The icon for Cisco generic routers is shown below.



1.12 wireless access points

A **Wireless Access Point** (WAP) is a device that allows wireless communication devices to connect to a wireless network using Wi-Fi, Bluetooth or related standards. The WAP usually connects to a wired network, and can relay data between the wireless devices (such as computers or printers) and wired devices on the network.

An AP is differentiated from a **hotspot**, which is the physical space where the wireless service is provided.



A small office network with one WAP

A typical example of a WAP is the **wireless network interface controller** (WNIC). It is a network card which connects to a radio-based computer network, unlike a regular network interface controller (NIC) which connects to a wire-based network such as token ring or ethernet. A WNIC, just like a NIC, works on the Layer 1 and Layer 2 of the OSI Model. A WNIC is an essential component for wireless desktop computer. This card uses an antenna to communicate through microwaves. A WNIC in a desktop computer usually is connected using the PCI bus. Other connectivity options are USB and PC card. Integrated WNICs are also available, (typically in Mini PCI/PCI Express Mini Card form).

A wireless NIC

1.13 wireless routers

A **Wireless Router** is a device that performs the functions of a router and also includes the functions of a wireless access point. It is used to provide access to the Internet or a private computer network. Depending on the manufacturer and model, it can function in a wired local area network, in a wireless-only LAN, or in a mixed wired and wireless network.

Some of the wireless routers characteristics:

- more often, wireless routers come with either xDSL modem, DOCSIS modem, LTE modem, or fiber optic modem integrated.
- many dual-band wireless routers have data transfer rates exceeding 300 Mbit/s (For the 2.4 GHz band) and 450 Mbit/s (for the 5 GHz band)
- Some wireless routers have one or two USB ports. For wireless routers having one USB port, it is designated for either printer or desktop/mobile external hard disk drive. For wireless routers having two USB ports, one is designated for the printer and the other one is designated for either desktop or mobile external hard disk drive

The picture below shows a typical wireless router, in our case a TP-Link Archer C7. It's a dual band router, with one WAN port, 4 LAN ports and 2 USB ports.

We mention several of the evolving wireless standards of IEEE (the 802.11 set)

- 802.11-1997 – first standard, rates up t0 2 Mbps (2.4GHz frequency)
- 802.11b – 09.1999, speeds up to 11 Mbps
- 802.11g – 06.2003, speeds up to 54 Mbps
- 802.11n – 10.2009, speeds up to 600 Mbps (2.4 and 5 GHz bands) (also known as **Wi-Fi 4**)
- 802.11ac – (**Wi-Fi 5**) – speeds up to 3.4 Gbps, operates mostly in the 5Ghz band – current standard
- 802.11ax – (**Wi-Fi 6**) – speeds 4x previous standards, operates in the 2.4 and 5Ghz bands – next

standard

Current (02.2020) household level wireless routers have data speeds of 1 Gbps.

A typical wireless router for household use

chapter 2 network classification, domains

2.1 classification criteria

We distinguish two main classification criteria for networks:

- coverage area
- the network topology

In terms of span or coverage area, networks can expand from the very basics ones, like home networks, to the global nETWORK. Here is a standard list of network types based on the coverage area criterion:

- personal area networks
- home networks
- local area networks (LANs)
- metropolitan area networks (MANs)
- wide area networks (WANs)
- the global network (the nETWORK)

One may squeeze something in between the lines, like Campus Area Networks, Neighborhood Area Networks or Global Corporate Networks, but they may fit in the LAN or WAN categories, as well.

When it comes to the topological view of the networks, we distinguish three classification criteria:

- physical topologies
- signal topologies
- logical topologies

This course only covers physical topologies. We start our classification process with the types list according to the coverage area criterion.

2.2 personal area networks

This type of network allows devices to communicate over the range of a single person. A common example is a wireless network that connects a computer with its peripherals. Almost every computer has an attached monitor, keyboard, mouse, and printer. Without using wireless, this connection must be done with cables. So many new users have a hard time finding the right cables and plugging them into the right little holes (even though they are usually color coded) that most computer vendors offer the option of sending a technician to the user's home to do it. To help these users, some companies got together to design a short-range wireless network called **Bluetooth** to connect these components without wires. The idea is that if your devices have Bluetooth, then you need no cables. You just put them down, turn them on, and they work together. For many people, this ease of operation is a big plus.

A completely different kind of PAN (Personal Area Network) is formed when an embedded medical device such as a pacemaker, insulin pump, or hearing aid talks to a user-operated remote control. PANs can also be built with other technologies that communicate over short ranges, such as RFID on smartcards and library books.

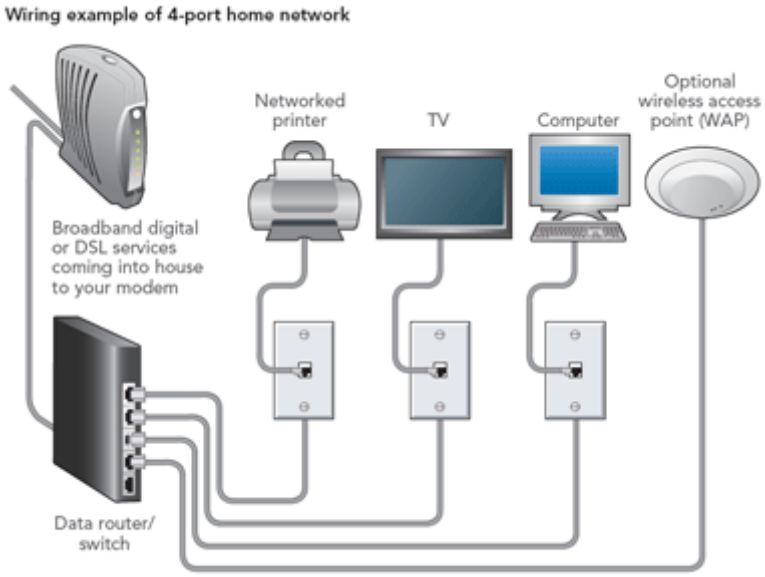
2.3 home networks

Home networking is on the horizon. The fundamental idea is that in the future most homes will be set up for networking. Every device in the home will be capable of communicating with every other device, and all of them will be accessible over the Internet. This is one of those visionary concepts that nobody asked for (like TV remote controls or mobile phones), but once they arrived nobody can imagine how they lived without them.

Many devices are capable of being networked. Some of the more obvious categories (with examples) are as follows:

1. Computers (desktop PC, notebook PC, PDA, tablet, shared peripherals).
2. Entertainment (TV, DVD, VCR, camcorder, camera, stereo, blue ray, MP3 players).
3. Telecommunications (telephone, mobile telephone, intercom, fax).
4. Appliances (microwave, refrigerator, clock, furnace, air conditioner, lights).
5. Telemetry (utility meter, smoke/burglar alarm, thermostat, baby cam).

Home computer networking is already here in a limited way. Many homes already have a device to connect multiple computers to a fast Internet connection. Networked entertainment is not quite here, but as more and more music and movies can be downloaded from the Internet, there will be a demand to connect stereos and televisions to it. Also, people will want to share their own videos with friends and family, so the connection will need to go both ways. Telecommunications gear is already connected to the outside world, but soon it will be digital and go over the Internet.



The average home probably has a dozen clocks (e.g., in appliances), all of which have to be reset twice a year when daylight saving time (summer time) comes and goes. If all the clocks were on the Internet, that resetting could be done automatically. Finally, remote monitoring of the home and its contents is a likely winner. Probably many parents would be willing to spend some money to monitor their sleeping babies on their PDAs when they are eating out, even with a rented teenager in the house. While one can imagine a separate network for each application area, integrating all of them into a single network is probably a better idea.

Home networking has some fundamentally different properties than other network types.

First, the network and devices have to be **easy to install**. The author has installed numerous pieces of hardware and software on various computers over the years, with mixed results. A series of phone calls to the vendor's help desk typically resulted in answers like (1) Read the manual, (2) Reboot the computer, (3) Remove all hardware and software except ours and try again, (4) Download the newest driver from our Web site, and if all else fails, (5) Reformat the hard disk and then reinstall Windows from the CD-ROM. Telling the purchaser of an Internet refrigerator to download and install a new version of the refrigerator's operating system is not going to lead to happy customers. Computer users are accustomed to putting up with products that do not work; the car-, television-, and refrigerator-buying public is far less tolerant. They expect products to work for 100% from the word go.

Second, the network and devices have to be **foolproof in operation**. Air conditioners used to have one knob with four settings: OFF, LOW, MEDIUM, and HIGH. Now they have 30-page manuals. Once they are networked, expect the chapter on security alone to be 30 pages. This will be beyond the comprehension of virtually all the users.

Third, **low price** is essential for success. People will not pay a $50 premium for an Internet thermostat because few people regard monitoring their home temperature from work that important. For $5 extra, it might sell, though.

Fourth, the main application is likely to involve multimedia, so the network needs **sufficient capacity**. There is no market for Internet-connected televisions that show shaky movies at 320 x 240 pixel resolution and 10 frames/sec. Fast Ethernet, the workhorse in most offices, is not good enough for multimedia. Consequently, home networks will need better performance than that of existing office networks and at lower prices before they become mass market items.

Fifth, it must be possible to start out with one or two devices and **expand** the reach of the network **gradually**. This means no format wars. Telling consumers to buy peripherals with IEEE 1394 (FireWire) interfaces and a few years later retracting that and saying USB 2.0 is the interface-of-the-month is going to make consumers skittish. The network interface will have to remain stable for many years; the wiring (if any) will have to remain stable for decades.

Sixth, **security and reliability** will be very important. Losing a few files to an e-mail virus is one thing; having a burglar disarm your security system from his PDA and then plunder your house is something quite different.

An interesting question is whether home networks will be wired or wireless. Most homes already have six networks installed: electricity, telephone, cable television, water, gas, and sewer. Adding a seventh one during construction is not difficult, but retrofitting existing houses is expensive. Cost favors wireless networking, but security favors wired networking. The problem with wireless is that the radio waves they use are quite good at going through fences. Not everyone is overjoyed at the thought of having the neighbors piggybacking on their Internet connection and reading their e-mail on its way to the printer. In the context of a home network, security has to be foolproof, even with inexperienced users. This is easier said than done, even with highly sophisticated users.

In short, home networking offers many opportunities and challenges. Most of them relate to the need to be easy to manage, dependable, and secure, especially in the hands of nontechnical users, while at the same time delivering high performance at low cost.

2.4 local area networks - LANs

Local area networks, generally called LANs, are privately-owned networks within a single building or campus of up to a few kilometers in size. They are widely used to connect personal computers and workstations in company offices and factories to share resources (e.g., printers) and exchange information. LANs are distinguished from other kinds of networks by three characteristics: (1) their size, (2) their transmission technology, and (3) their topology.

LANs are **restricted in size**, which means that the worst-case transmission time is bounded and known in advance. Knowing this bound makes it possible to use certain kinds of designs that would not otherwise be possible. It also simplifies network management.

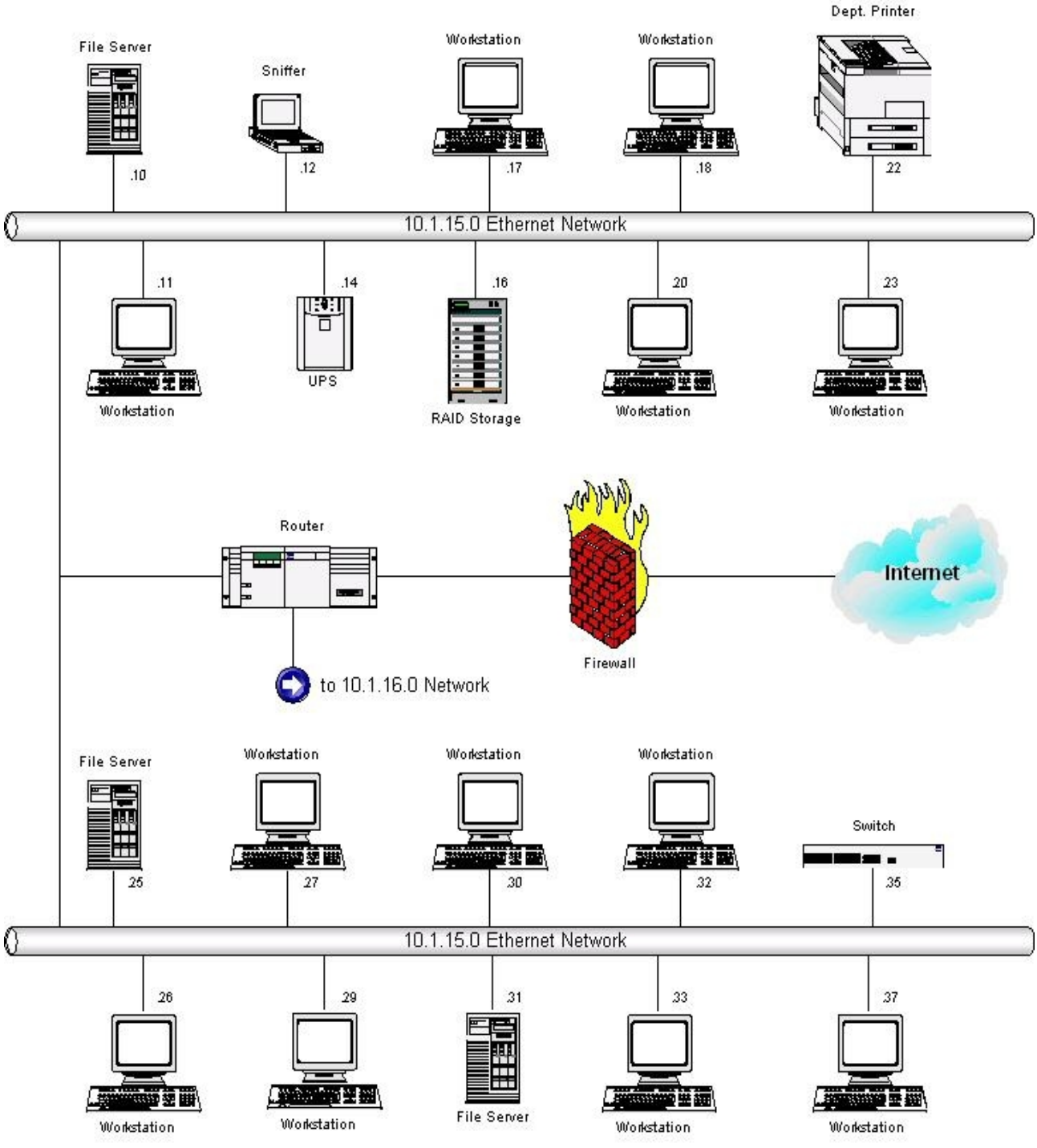
LANs may use a transmission technology consisting of a cable to which all the machines are attached, like the telephone company party lines once used in rural areas. Traditional LANs run at speeds of 100 Mbps to 1000 Mbps, have **low delay** (microseconds or nanoseconds), and make **very few errors**. Newer LANs operate at up to 10 Gbps. In this book, we will adhere to tradition and measure line speeds in megabits/sec (1 Mbps is 1,000,000 bits/sec) and gigabits/sec (1 Gbps is 1,000,000,000 bits/sec).

Various topologies are possible for broadcast LANs. In a **bus** (i.e., a linear cable) network, at any instant at most one machine is the master and is allowed to transmit. All other machines are required to refrain from sending. An arbitration mechanism is needed to resolve conflicts when two or more machines want to transmit simultaneously. The arbitration mechanism may be centralized or distributed. IEEE 802.3, popularly called Ethernet, for example, is a bus-based broadcast network with decentralized control, usually operating at 10 Mbps to 10 Gbps. Computers on an Ethernet can transmit whenever they want to; if two or more packets collide, each computer just waits a random time and tries again later.

A second type of broadcast system is the **ring**. In a ring, each bit propagates around on its own, not waiting for the rest of the packet to which it belongs. Typically, each bit circumnavigates the entire ring in the time it takes to transmit a few bits, often before the complete packet has even been transmitted. As with all other broadcast systems, some rule is needed for arbitrating simultaneous accesses to the ring. Various methods, such as having the machines take turns, are in use. IEEE 802.5 (the IBM token ring), is a ring-based LAN operating at 4 and 16 Mbps. FDDI is another example of a ring network.

Broadcast networks can be further divided into **static** and **dynamic**, depending on how the channel is allocated. A typical static allocation would be to divide time into discrete intervals and use a round-robin algorithm, allowing each machine to broadcast only when its time slot comes up. Static allocation wastes channel capacity when a machine has nothing to say during its allocated slot, so most systems attempt to allocate the channel dynamically (i.e., on demand).

Dynamic allocation methods for a common channel are either **centralized** or **decentralized**. In the centralized channel allocation method, there is a single entity, for example, a bus arbitration unit, which determines who goes next. It might do this by accepting requests and making a decision according to some internal algorithm. In the decentralized channel allocation method, there is no central entity; each machine must decide for itself whether to transmit. You might think that this always leads to chaos, but it does not. Later we will study many algorithms designed to bring order out of the potential chaos.



A typical Local Area Network

2.5 metropolitan area networks - MANs

A metropolitan area network, or MAN, covers a city, large portions of it or a large campus.

A MAN usually interconnects a number of local area networks (LANs) using a high-capacity backbone technology, such as fiber-optical links, and provides up-link services to wide area networks (or WAN) and the Internet. A MAN is optimized for a larger geographical area than a LAN, ranging from several blocks of buildings to entire cities.

The best-known example of a MAN is the cable television network available in many cities. This system grew from earlier community antenna systems used in areas with poor over-the-air television reception. In these early systems, a large antenna was placed on top of a nearby hill and signal was then piped to the subscribers' houses.

At first, these were locally-designed, ad hoc systems. Then companies began jumping into the business, getting contracts from city governments to wire up an entire city. The next step was television programming and even entire channels designed for cable only. Often these channels were highly specialized, such as all news, all sports, all cooking, all gardening, and so on. But from their inception until the late 1990s, they were intended for television reception only.

Starting when the Internet attracted a mass audience, the cable TV network operators began to realize that with some changes to the system, they could provide two-way Internet service in unused parts of the spectrum. At that point, the cable TV system began to morph from a way to distribute television to a metropolitan area network. Cable television is not the only MAN. Recent developments in high-speed wireless Internet access resulted in another MAN, which has been standardized as IEEE 802.16.

2.6 wide area networks - WANs

A wide area network, or WAN, spans a **large geographical area**, often a country or continent. It contains a collection of machines intended for running user (i.e., application) programs. The textbook definition of a WAN is a computer network spanning regions, countries, or even the world. However, in terms of the application of computer networking protocols and concepts, it may be best to view WANs as computer networking technologies used to transmit data over long distances, and between different LANs, MANs and other localized computer networking architectures.

WANs are characterized by high bandwidth, high latency, higher than medium error rates, are flexible in terms of scalability and have heterogeneous administrative structures.

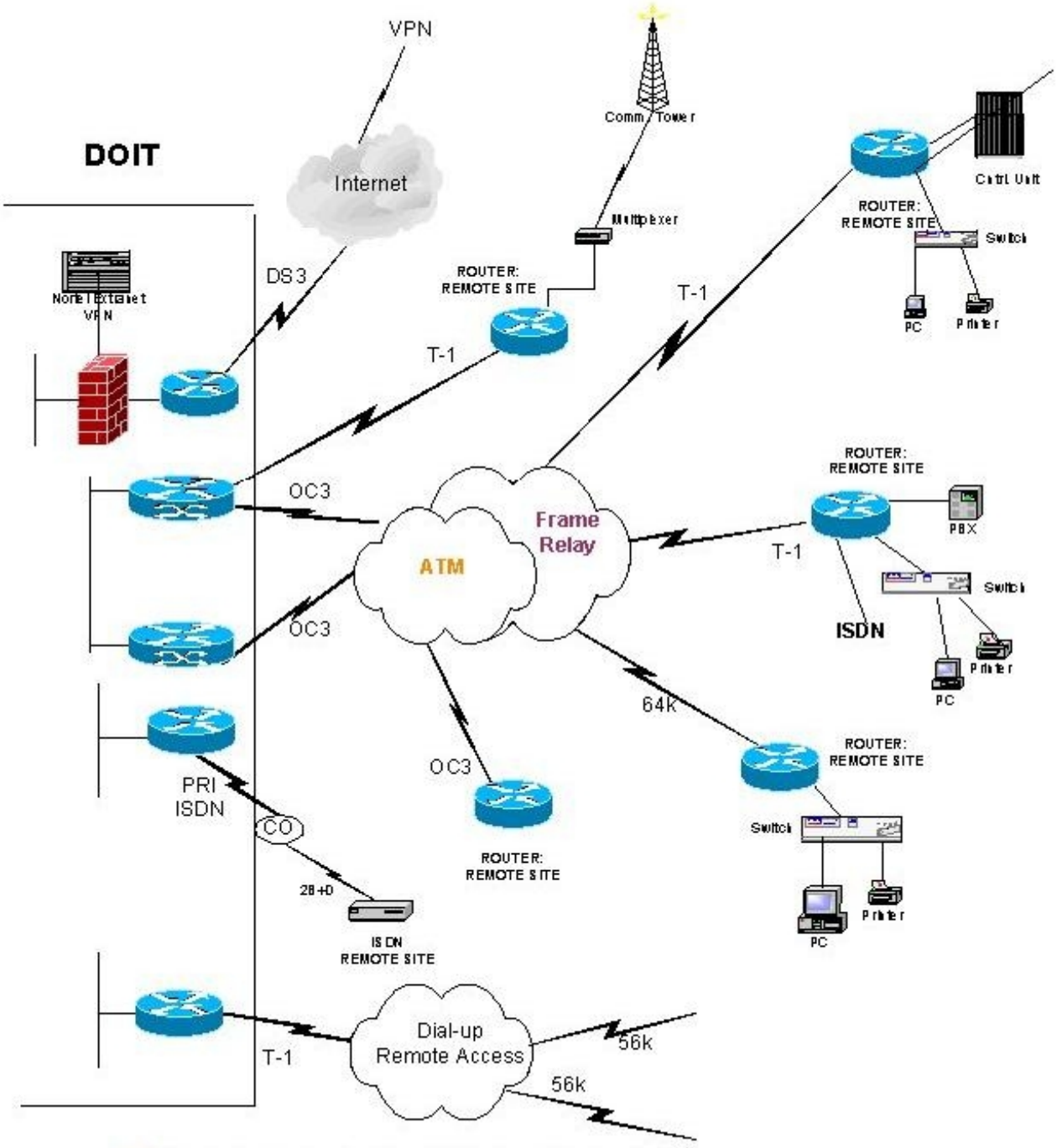
In most wide area networks, the internetwork consists of two distinct components: transmission lines and switching elements. Transmission lines move bits between machines. They can be made of copper wire, optical fiber, or even radio links. Switching elements are specialized computers that connect three or more transmission lines. When data arrive on an incoming line, the switching element must choose an outgoing line on which to forward them. These switching computers have been called by various names in the past; the name router is now most commonly used. Unfortunately, some people pronounce it "rooter" and others have it rhyme with "doubter." Determining the correct pronunciation will be left as an exercise for the reader. (Note: the perceived correct answer may depend on where you live.)

In this model, each host is frequently connected to a LAN on which a router is present, although in some cases a host can be connected directly to a router. The collection of communication lines and routers (but not the hosts) form the internetwork.

A short comment about the term "internetwork" is in order here. Originally, its only meaning was the collection of routers and communication lines that moved packets from the source host to the destination host. However, some years later, it also acquired a second meaning in conjunction with network addressing. Unfortunately, no widely-used alternative exists for its initial meaning, so with some hesitation we will use it in both senses. From the context, it will always be clear which is meant.

In most WANs, the network contains numerous transmission lines, each one connecting a pair of routers.

If two routers that do not share a transmission line wish to communicate, they must do this indirectly, via other routers. When a packet is sent from one router to another via one or more intermediate routers, the packet is received at each intermediate router in its entirety, stored there until the required output line is free, and then forwarded. A internetwork organized according to this principle is called a store-and-forward or packet-switched internetwork. Nearly all wide area networks (except those using satellites) have store-and-forward internetworks. When the packets are small and all the same size, they are often called cells.



The Connecticut Dept. of Information Technology WAN

The principle of a packet-switched WAN is so important that it is worth devoting a few more words to it. Generally, when a process on some host has a message to be sent to a process on some other host, the sending host first cuts the message into packets, each one bearing its number in the sequence. These packets are then injected into the network one at a time in quick succession. The packets are transported individually over the network and deposited at the receiving host, where they are reassembled into the original message and delivered to the receiving process.

In this figure, all the packets follow the route ACE, rather than ABDE or ACDE. In some networks all packets from a given message must follow the same route; in others each packet is routed separately. Of course, if ACE is the best route, all packets may be sent along it, even if each packet is individually routed.

Routing decisions are made locally. When a packet arrives at router A, it is up to A to decide if this packet should be sent on the line to B or the line to C. How A makes that decision is called the routing algorithm. Many of them exist. We will study some of them in chapter

Not all WANs are packet switched. A second possibility for a WAN is a satellite system. Each router has an antenna through which it can send and receive. All routers can hear the output from the satellite, and in some cases they can also hear the upward transmissions of their fellow routers to the satellite as well. Sometimes the routers are connected to a substantial point-to-point internetwork, with only some of them having a satellite antenna. Satellite networks are inherently broadcast and are most useful when the broadcast property is important.

2.7 wireless networks

Digital wireless communication is not a new idea. As early as 1901, the Italian physicist Guglielmo Marconi demonstrated a ship-to-shore wireless telegraph, using Morse Code (dots and dashes are binary, after all). Modern digital wireless systems have better performance, but the basic idea is the same.

To a first approximation, wireless networks can be divided into three main categories:

1. System interconnection.

2. Wireless LANs.

3. Wireless WANs.

System interconnection is all about interconnecting the components of a computer using short-range radio. Almost every computer has a monitor, keyboard, mouse, and printer connected to the main unit by cables. So many new users have a hard time plugging all the cables into the right little holes (even though they are usually color coded) that most computer vendors offer the option of sending a technician to the user's home to do it. Consequently, some companies got together to design a short-range wireless network called Bluetooth to connect these components without wires. Bluetooth also allows digital cameras, headsets, scanners, and other devices to connect to a computer by merely being brought within range. No cables, no driver installation, just put them down, turn them on, and they work. For many people, this ease of operation is a big plus.

In the simplest form, system interconnection networks use a master-slave paradigm. The system unit is normally the master, talking to the mouse, keyboard, etc., as slaves. The master tells the slaves what addresses to use, when they can broadcast, how long they can transmit, frequencies they use, and so on.

Many people believe wireless is the wave of the future (e.g., Bi et al., 2001; Leeper, 2001; Varshey and Vetter, 2000) but at least one dissenting voice has been heard. Bob Metcalfe, the inventor of Ethernet, has written: "Mobile wireless computers are like mobile pipeless bathrooms—portapotties. They will be common on vehicles, and at construction sites, and rock concerts. My advice is to wire up your home and stay there" (Metcalfe, 1995). History may record this remark in the same category as IBM's chairman T.J. Watson's 1945 explanation of why IBM was not getting into the computer business: "Four or five computers should be enough for the entire world until the year 2000."

2.8 internetworking

Many networks exist in the world, often with different hardware and software. People connected to one network often want to communicate with people attached to a different one. The fulfillment of this desire requires that different, and frequently incompatible networks, be connected, sometimes by means of machines called gateways to make the connection and provide the necessary translation, both in terms of hardware and software. A collection of interconnected networks is called an internetwork or internet. These terms will be used in a generic sense, in contrast to the worldwide Internet.

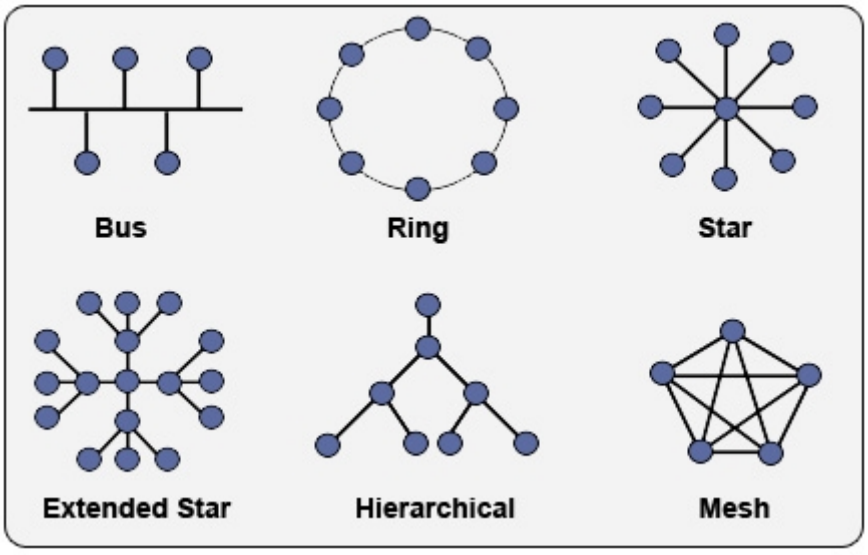
A common form of internet is a collection of LANs connected by a WAN. The only real technical distinction between a internetwork and a WAN in this case is whether hosts are present. If the system within the gray area contains only routers, it is a internetwork; if it contains both routers and hosts, it is a WAN. The real differences relate to ownership and use.

Subnets, networks, and internetworks are often confused. Internetwork makes the most sense in the context of a wide area network, where it refers to the collection of routers and communication lines owned by the network operator. As an analogy, the telephone system consists of telephone switching offices connected to one another by high-speed lines, and to houses and businesses by low-speed lines. These lines and equipment, owned and managed by the telephone company, form the internetwork of the telephone system. The telephones themselves (the hosts in this analogy) are not part of the internetwork. The combination of a internetwork and its hosts forms a network.

An internetwork is formed when distinct networks are interconnected. In our view, connecting a LAN and a WAN or connecting two LANs forms an internetwork, but there is little agreement in the industry over terminology in this area. One rule of thumb is that if different organizations paid to construct different parts of the network and each maintains its part, we have an internetwork rather than a single network. Also, if the underlying technology is different in different parts (e.g., broadcast versus point-to-point), we probably have two networks.

2.9 physical topologies

Due to the complexity of the global network, typical network topological make sense mainly at a local level. A compendium of the most recognizable topological patterns in networks can be seen below.



2.9.1 point-to-point

The simplest topology is a permanent link between two endpoints. Switched point-to-point topologies are the basic model of conventional telephony. The value of a permanent point-to-point network is the value of guaranteed, or nearly so, communications between the two endpoints. The value of an on-demand point-to-point connection is proportional to the number of potential pairs of subscribers, and has been expressed as Metcalfe's Law.

2.9.1.1 permanent (dedicated)

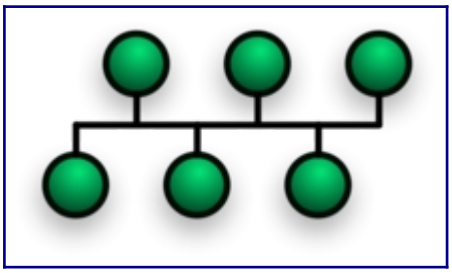
Easiest to understand, of the variations of point-to-point topology, is a point-to-point communication channel that appears, to the user, to be permanently associated with the two endpoints. Children's "tin-can telephone" is one example, with a microphone to a single public address speaker is another. These are examples of *physical dedicated* channels.

Within many switched telecommunications systems, it is possible to establish a permanent circuit. One example might be a telephone in the lobby of a public building, which is programmed to ring only the number of a telephone dispatcher. "Nailing down" a switched connection saves the cost of running a physical circuit between the two points. The resources in such a connection can be released when no longer needed, for example, a television circuit from a parade route back to the studio.

2.9.1.2 switched

Using circuit-switching or packet-switching technologies, a point-to-point circuit can be set up dynamically, and dropped when no longer needed. This is the basic mode of conventional telephony.

2.9.2 bus



Bus network topology

In local area networks where bus topology is used, each machine is connected to a single cable. Each computer or server is connected to the single bus cable through some kind of connector. A terminator is required at each end of the bus cable to prevent the signal from bouncing back and forth on the bus cable. A signal from the source travels in both directions to all machines connected on the bus cable until it finds the MAC address or IP address on the network that is the intended recipient. If the machine address does not match the intended address for the data, the machine ignores the data. Alternatively, if the data does match the machine address, the data is accepted. Since the bus topology consists of only one wire, it is rather inexpensive to implement when compared to other topologies. However, the low cost of implementing the technology is offset by the high cost of managing the network. Additionally, since only one cable is utilized, it can be the single point of failure. If the network cable breaks, the entire network will be down.

2.9.2.1 linear bus

The type of network topology in which all of the nodes of the network are connected to a common transmission medium which has exactly two endpoints (this is the 'bus', which is also commonly referred to as the backbone, or trunk) – all data that is transmitted between nodes in the network is transmitted over this common transmission medium and is able to be received by all nodes in the network virtually simultaneously (disregarding propagation delays).

Note: The two endpoints of the common transmission medium are normally terminated with a device called a terminator that exhibits the characteristic impedance of the transmission medium and which dissipates or absorbs the energy that remains in the signal to prevent the signal from being reflected or propagated back onto the transmission medium in the opposite direction, which would cause interference with and degradation of the signals on the transmission medium.

2.9.2.2 distributed bus

The type of network topology in which all of the nodes of the network are connected to a common transmission medium which has more than two endpoints that are created by adding branches to the main section of the transmission medium – the physical distributed bus topology functions in exactly the same fashion as the physical linear bus topology (i.e., all nodes share a common transmission medium).

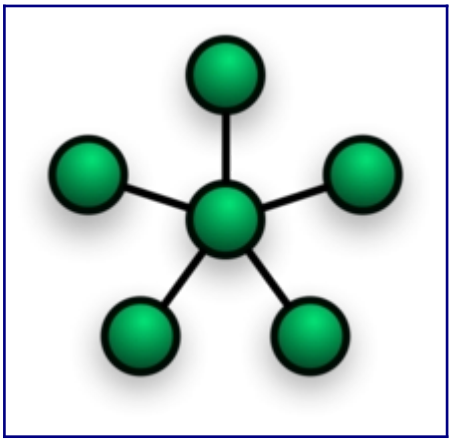
Notes:

1.) All of the endpoints of the common transmission medium are normally terminated with a device called a 'terminator'.

2.) The physical linear bus topology is sometimes considered to be a special case of the physical distributed bus topology – i.e., a distributed bus with no branching segments.

3.) The physical distributed bus topology is sometimes incorrectly referred to as a physical tree topology – however, although the physical distributed bus topology resembles the physical tree topology, it differs from the physical tree topology in that there is no central node to which any other nodes are connected, since this hierarchical functionality is replaced by the common bus.

2.9.3 star



Star network topology

In local area networks with a star topology, each network host is connected to a **central hub**. In contrast to the bus topology, the star topology connects each node to the hub with a point-to-point connection. All traffic that transverses the network passes through the central hub. The hub acts as a signal booster or

repeater. The star topology is considered the easiest topology to design and implement. An advantage of the star topology is the simplicity of adding additional nodes. The primary disadvantage of the star topology is that the hub represents a single point of failure.

Notes

- A point-to-point link (described above) is sometimes categorized as a special instance of the physical star topology – therefore, the simplest type of network that is based upon the physical star topology would consist of one node with a single point-to-point link to a second node, the choice of which node is the 'hub' and which node is the 'spoke' being arbitrary.
- After the special case of the point-to-point link, as in the note above, the next simplest type of network that is based upon the physical star topology would consist of one central node – the 'hub' – with two separate point-to-point links to two peripheral nodes – the 'spokes'.
- Although most networks that are based upon the physical star topology are commonly implemented using a special device such as a **hub** or **switch** as the central node (i.e., the 'hub' of the star), it is also possible to implement a network that is based upon the physical star topology using a computer or even a simple common connection point as the 'hub' or central node – however, since many illustrations of the physical star network topology depict the central node as one of these special devices, some confusion is possible, since this practice may lead to the misconception that a physical star network requires the central node to be one of these special devices, which is not true because a simple network consisting of three computers connected as in the note above also has the topology of the physical star.
- Star networks may also be described as either broadcast multi-access or nonbroadcast multi-access (NBMA), depending on whether the technology of the network either automatically propagates a signal at the hub to all spokes, or only addresses individual spokes with each communication.

2.9.3.1 extended star

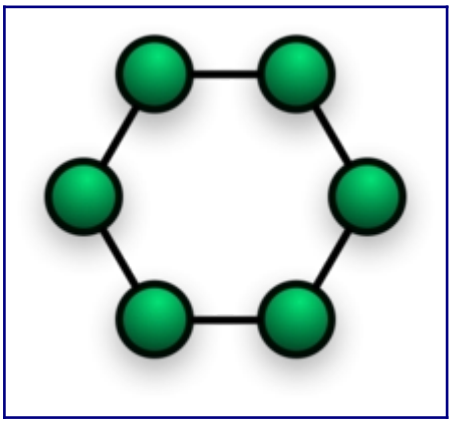
A type of network topology in which a network that is based upon the physical star topology has one or more repeaters between the central node (the 'hub' of the star) and the peripheral or 'spoke' nodes, the repeaters being used to extend the maximum transmission distance of the point-to-point links between the central node and the peripheral nodes beyond that which is supported by the transmitter power of the central node or beyond that which is supported by the standard upon which the physical layer of the physical star network is based.

If the repeaters in a network that is based upon the physical extended star topology are replaced with hubs or switches, then a hybrid network topology is created that is referred to as a physical hierarchical star topology, although some texts make no distinction between the two topologies.

2.9.3.2 distributed star

A type of network topology that is composed of individual networks that are based upon the physical star topology connected together in a linear fashion – i.e., 'daisy-chained' – with no central or top level connection point (e.g., two or more 'stacked' hubs, along with their associated star connected nodes or 'spokes').

2.9.4 ring



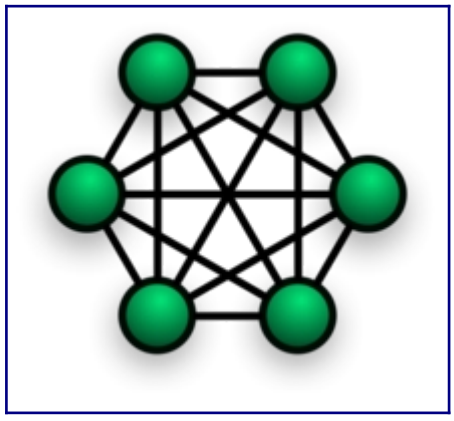
Ring network topology

In local area networks where the ring topology is used, each computer is connected to the network in a **closed loop** or ring. Each machine or computer has a unique address that is used for identification purposes. The signal passes through each machine or computer connected to the ring in one direction. Ring topologies typically utilize a token passing scheme, used to control access to the network. By utilizing this scheme, only one machine can transmit on the network at a time. The machines or computers connected to the ring act as signal boosters or repeaters which strengthen the signals that transverse the network. The primary disadvantage of ring topology is the failure of one machine will cause the entire network to fail.

2.9.5 mesh

The value of fully meshed networks is proportional to the exponent of the number of subscribers, assuming that communicating groups of any two endpoints, up to and including all the endpoints, is approximated by Reed's Law.

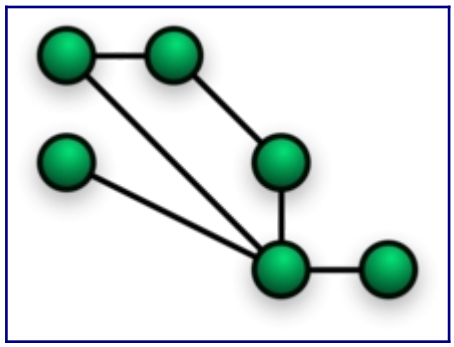
2.9.5.1 fully connected



Fully connected mesh topology

Note: The physical fully connected mesh topology is generally too costly and complex for practical networks, although the topology is used when there are only a small number of nodes to be interconnected.

2.9.5.2 partially connected

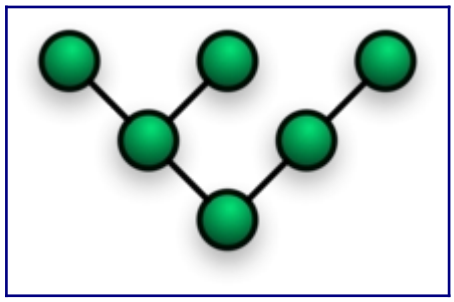


Partially connected mesh topology

The type of network topology in which some of the nodes of the network are connected to more than one other node in the network with a point-to-point link – this makes it possible to take advantage of some of the redundancy that is provided by a physical fully connected mesh topology without the expense and complexity required for a connection between every node in the network.

Note: In most practical networks that are based upon the physical partially connected mesh topology, all of the data that is transmitted between nodes in the network takes the shortest path (or an approximation of the shortest path) between nodes, except in the case of a failure or break in one of the links, in which case the data takes an alternative path to the destination. This requires that the nodes of the network possess some type of logical 'routing' algorithm to determine the correct path to use at any particular time.

2.9.6 tree



Tree network topology

Also known as a **hierarchical network**.

The type of network topology in which a central 'root' node (the top level of the hierarchy) is connected to one or more other nodes that are one level lower in the hierarchy (i.e., the second level) with a point-to-point link between each of the second level nodes and the top level central 'root' node, while each of the second level nodes that are connected to the top level central 'root' node will also have one or more other nodes that are one level lower in the hierarchy (i.e., the third level) connected to it, also with a point-to-point link, the top level central 'root' node being the only node that has no other node above it in the hierarchy (The hierarchy of

the tree is symmetrical.) Each node in the network having a specific fixed number, of nodes connected to it at the next lower level in the hierarchy, the number, being referred to as the 'branching factor' of the hierarchical tree.

1.) A network that is based upon the physical hierarchical topology must have at least three levels in the hierarchy of the tree, since a network with a central 'root' node and only one hierarchical level below it would exhibit the physical topology of a star.

2.) A network that is based upon the physical hierarchical topology and with a branching factor of 1 would be classified as a physical linear topology.

3.) The branching factor, f, is independent of the total number of nodes in the network and, therefore, if the nodes in the network require ports for connection to other nodes the total number of ports per node may be kept low even though the total number of nodes is large – this makes the effect of the cost of adding ports to each node totally dependent upon the branching factor and may therefore be kept as low as required without any effect upon the total number of nodes that are possible.

4.) The total number of point-to-point links in a network that is based upon the physical hierarchical topology will be one less than the total number of nodes in the network.

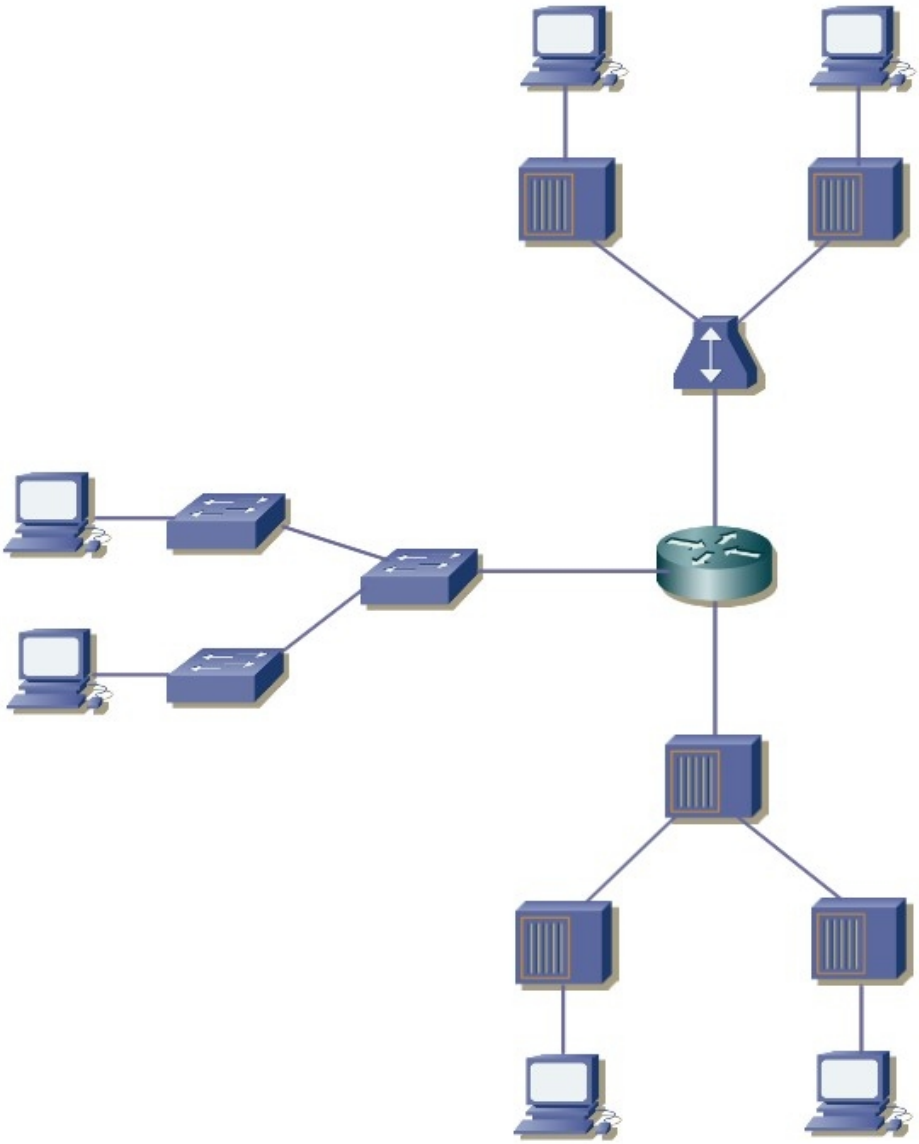
5.) If the nodes in a network that is based upon the physical hierarchical topology are required to perform any processing upon the data that is transmitted between nodes in the network, the nodes that are at higher levels in the hierarchy will be required to perform more processing operations on behalf of other nodes than the nodes that are lower in the hierarchy. Such a type of network topology is very useful and highly recommended.

2.10 collision domains

A **network collision** occurs when more than one device attempts to send a packet on a network segment at the same time.

A **collision domain** is a physical network segment where data packets can collide with one another when being sent on a shared medium, in particular, when using the Ethernet networking protocol.

Remember that every interface of a bridge, switch or router represents a separate collision domain. If we remove all the nodes corresponding to bridges, switches or routers (and the corresponding edges(arcs), as well), we obtain a new graph, called the **collision graph** of the network. Each connected component of the collision graph represents a **collision domain**. The quotient graph (whose nodes correspond to collision domains) will be called the **collision scheme**.



In the network above, after removing bridges, switches and routers, we obtain a network graph with 5 connected components, each component corresponding to a collision domain.

2.11 broadcast domains

A **broadcast domain** associated to an end devices in the network is, essentially, the maximal portion of the network that any broadcast message sent from that device can travel in an unobstructed (unfiltered) way.

Remember that every interface of a router represents a separate broadcast domain. If we remove all the nodes corresponding to routers (and the corresponding edges(arcs), as well), we obtain a new graph, called the **broadcast graph** of the network. Each connected component of the broadcast graph represents a

broadcast domain. The quotient graph (whose nodes correspond to broadcast domains) will be called the **broadcast scheme**.

To identify the broadcast domains of the network in 2.9, we remove only the router (the network device which splits (isolates) broadcast domains. The induced network graph has 3 connected components, each corresponding to a broadcast domain.

chapter 3 communication protocols, the OSI model

3.1 communication protocols

A **communication protocol** is a set of rules that the end points in a telecom link use when they exchange messages.

A protocol is specified in an industry or international standard. All internet related protocols are defined within the frame of ISOC (Internet Society) and its associated bodies, the most important one being IETF (**Internet Engineering Task Force**) via a mechanism called RFC (**Request For Comments**).

Each (potential) protocol is defined by such a document. For a comprehensive list of all the RFCs, check the official site tools.ietf.org/rfc/index (which presents the RFCs in .txt format) or tools.ietf.org/html (which presents the RFCs in .html format, with up to date links embedded in the documents). Pdf format is also available.

Not all RFCs are standards. Each RFC is assigned a designation with regard to its status within the Internet standardization process. This status is one of the following: **Informational**, **Experimental**, **Best Current Practice (BCP)**, **Standards Track**, or **Historic**. Standards-track documents are further divided into **Proposed Standard**, **Draft Standard** and **Internet Standard** documents. The term Historic is applied to deprecated standards-track documents or obsolete RFCs that were published before the standards track was established. Only the IETF, represented by the **Internet Engineering Steer Group** (IESG), can approve standards-track RFCs. Each RFC is static; if the document is changed, it is submitted again and assigned a new RFC number. If an RFC becomes an Internet Standard (STD), it is assigned an STD number but retains its RFC number; however, when an Internet Standard is updated, its number stays the same and it simply refers to a different RFC or set of RFCs. A given Internet Standard, STD *n*, may be RFCs *x* and *y* at a given time, but later the same standard may be updated to be RFC *z* instead. For example, in 2007 RFC 3700 was an Internet Standard - STD 1 - and in May 2008 it was replaced with RFC 5000, so RFC 3700 changed to *Historic*, RFC 5000 became an Internet Standard, and as of May 2008, STD 1 is RFC 5000. When STD 1 is updated again, it will simply refer to a newer RFC that will have completed the standards track, but it will still be STD 1. Best Current Practices work in a similar fashion; BCP *n* refers to a certain RFC or set of RFCs, but which RFC or RFCs may change over time.

The definitive list of Internet Standards is itself an Internet Standard, STD 1: **Internet Official Protocol Standards**. Currently (as of 03.2021) the STD list runs to STD94, while the RFC list has grown up to RFC 9003 (as of March 6, 2021).

The latest standard, STD94, for example, deals with Concise Binary Object Representation (CBOR), is RFC 8949, and became a standard in December 2020.

Besides protocols defined within the RFC frame, communication protocols may be defined as internal specifications, resulting in **proprietary standards** as:

- AppleTalk
- DECnet
- IPX/SPX
- Distributed Systems Architecture (DSA)

3.2 the OSI model

OSI stands for **Open System Interconnection**, an **ISO** (International Standard Organization) standard for worldwide communications that defines a structured framework for implementing protocols in seven

layers. Control is passed from one layer to the next, starting at the application layer at the source node, proceeding to the lower layers, over the links to the next node and back up the hierarchy until the destination node is reached. The structure of the message which is the object of this exchange gets modified along the way, each step down into the layer hierarchy adding a new wrapper around the existing message (usually, consisting of a protocol specific header), while each step up removes the wrapper specific to the layer below.

The seven layers in the OSI model are:

Nr.	Layer	Description	Protocol examples
7	Application	Supports application and end user processes. Provides application services for file transfers, e-mail and other network software services.	DHCP, DNS, FTP, Gopher, HTTP, IMAP4, POP3, SMTP, SNMP, TELNET, TLS (SSL), SOAP
6	Presentation	Translates data from application to network format and vice-verse. May also provide compression and encryption services.	APF, ICA, LPP, NCP, NDR, XDR, X.25 PAD
5	Session	Sets up, manages and terminates connections between communication partners. It handles session and connection coordination.	ASP, NetBIOS, PAP, PPTP, RPC, SMPP, SSH, SDP
4	Transport	Provides data transfer between the end points of the communication partners and is responsible for error recovery and flow control.	DCCP, SCTP, TCP, UDP, WTLS, WTP, XTP
3	Network	Responsible for source to destination delivery of packages, including routing through intermediate nodes. Provides quality of service and error control.	DDP, ICMP, IPSec, IPv4, IPv6, IPX, RIP
2	Data Link	Transfers data between adjacent network nodes and handles errors occurred at the physical level	ARCnet, ARP, ATM, CDP, Ethernet, Frame Relay, HDLC, Token Ring
1	Physical	Translates communication requests from the data link layer into transmissions and receptions of electronic signals at hardware level.	10BASE-T, DSL, Firewire, GSM, ISDN, SONET/SDH, V.92

For a detailed description of these layers, check the site: http://en.wikipedia.org/wiki/OSI_model.

The principles that were applied to arrive at the seven layers of the OSI model can be briefly summarized as follows:.

- A layer should be created where a different abstraction is needed.
- Each layer should perform a well-defined function.
- The function of each layer should be chosen with an eye toward defining internationally standardized protocols.
- The layer boundaries should be chosen to minimize the information flow across the interfaces.
- The number of layers should be large enough that distinct functions need not be thrown together in the same layer out of necessity and small enough that the architecture does not become unwieldy.

3.3 the application layer

The **application** layer contains a variety of protocols that are commonly needed by users. One widely-used application protocol is HTTP (**Hyper Text Transfer Protocol**), which is the basis for the World Wide Web. When a browser wants a Web page, it sends the name of the page it wants to the server using HTTP. The server then sends the page back. Other application protocols are used for file transfer, electronic mail, and network news. Other protocols of interest - DHCP, DNS, FTP, IMAP4 (**Internet Message Access Protocol**), POP3 (**Post Office Protocol**), SMTP (**Simple Mail Transfer Protocol**), SNMP, TELNET.

3.4 the presentation layer

Unlike lower layers, which are mostly concerned with moving bits around, the **presentation** layer is concerned with the syntax and semantics of the information transmitted. In order to make it possible for computers with different data representations to communicate, the data structures to be exchanged can be defined in an abstract way, along with a standard encoding to be used "on the wire."

The presentation layer can be divided into two sublayers:

- the **common application service element** (CASE), which provides services for the application layer and request services from the session layering
- the **special application service element** (SASE), which provides application specific service (protocols) such as:
  - FAM (File Transfer, Acces and Manager)
  - VT (Virtual Terminal)
  - JTM (Job Transfer and Manipulation)
  - RDA (Remote Database Access)
  - DTP (Distributed Transaction Processing)

The presentation layer manages these abstract data structures and allows higher-level data structures (e.g., banking records), to be defined and exchanged. A couple of examples. **Advanced Function Printing** (AFP) is a document format originally defined by IBM to drive its printers and support the typical form printing on laser printers. **External Data Representation Standard** (XDR**)** is a standard for the description and encoding of data. It is useful for transferring data between different computer architectures, using C-like primitives.

3.5 the session layer

The **session** layer allows users on different machines to establish sessions between them. Sessions offer various services, including dialog control (keeping track of whose turn it is to transmit), token management (preventing two parties from attempting the same critical operation at the same time), and synchronization (checkpointing long transmissions to allow them to continue from where they were after a crash).

An example of a Session Layer protocol is the **Session Layer Protocol**, also known as X.225 or ISO 8327. In case of a connection loss this protocol may try to recover the connection. If a connection is not used for a long period, the Session Layer Protocol may close it and re-open it. It provides for either full duplex or half-duplex operation and provides synchronization points in the stream of exchanged messages.

Another protocol of interest is SMPP (**Short Message Peer to Peer** protocol) which is used for short

messages (sms) deliveries in the cell phone industry.

3.6 the transport layer

The basic function of the **transport** layer is to accept data from above, split it up into smaller units (called **segments**) if needed, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end. Furthermore, all this must be done efficiently and in a way that isolates the upper layers from the inevitable changes in the hardware technology.

The transport layer also determines what type of service to provide to the session layer, and, ultimately, to the users of the network. The most popular type of transport connection is an error-free point-to-point channel that delivers messages or bytes in the order in which they were sent. However, other possible kinds of transport service are the transporting of isolated messages, with no guarantee about the order of delivery, and the broadcasting of messages to multiple destinations. The type of service is determined when the connection is established. (As an aside, an error-free channel is impossible to achieve; what people really mean by this term is that the error rate is low enough to ignore in practice.)

The transport layer is a true end-to-end layer, all the way from the source to the destination. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine, using the message headers and control messages. In the lower layers, the protocols are between each machine and its immediate neighbors, and not between the ultimate source and destination machines, which may be separated by many routers.

3.7 the network layer

The network layer controls the operation of the internetwork. A key design issue is determining how **packets** are routed from source to destination. Routes can be based on static tables that are "wired into" the network and rarely changed. They can also be determined at the start of each conversation, for example, a terminal session (e.g., a login to a remote machine). Finally, they can be highly dynamic, being determined anew for each packet, to reflect the current network load.

If too many packets are present in the internetwork at the same time, they will get in one another's way, forming bottlenecks. The control of such congestion also belongs to the network layer. More generally, the quality of service provided (delay, transit time, jitter, etc.) is also a network layer issue.

When a packet has to travel from one network to another to get to its destination, many problems can arise. The addressing used by the second network may be different from the first one. The second one may not accept the packet at all because it is too large. The protocols may differ, and so on. It is up to the network layer to overcome all these problems to allow heterogeneous networks to be interconnected.

In broadcast networks, the routing problem is simple, so the network layer is often thin or even nonexistent.

3.8 the data link layer

The main task of the data link layer is to transform a raw transmission facility into a line that appears free of undetected transmission errors to the network layer. It accomplishes this task by having the sender break up the input data into data **frames** (typically a few hundred or a few thousand bytes) and transmit the frames sequentially. If the service is reliable, the receiver confirms correct receipt of each frame by sending back an acknowledgment frame.

Another issue that arises in the data link layer (and most of the higher layers as well) is how to keep a fast transmitter from drowning a slow receiver in data. Some traffic regulation mechanism is often needed to let the transmitter know how much buffer space the receiver has at the moment. Frequently, this flow regulation and the error handling are integrated.

Broadcast networks have an additional issue in the data link layer: how to control access to the shared channel. A sublayer of the data link layer, the medium access control (**MAC**) sublayer, deals with this problem.

3.9 the physical layer

The physical layer is concerned with transmitting raw **bits** over a communication channel. The design issues have to do with making sure that when one side sends a 1 bit, it is received by the other side as a 1 bit, not as a 0 bit.

Typical questions here are how many volts should be used to represent a 1 and how many for a 0, how many nanoseconds a bit lasts, whether transmission may proceed simultaneously in both directions, how the initial connection is established and how it is torn down when both sides are finished, and how many pins the network connector has and what each pin is used for. The design issues here largely deal with mechanical,electrical, timing interfaces and the physical transmission medium.

3.10 protocol data units (PDUs)

The term **protocol data unit** (**PDU**) has the following meanings:

1. Information that is delivered as a unit among peer entities of a network and that may contain control information, address information, or data.
2. In a layered system, a unit of data which is specified in a protocol of a given layer and which consists of protocol-control information and user data of that layer. For example: iSCSI PDU

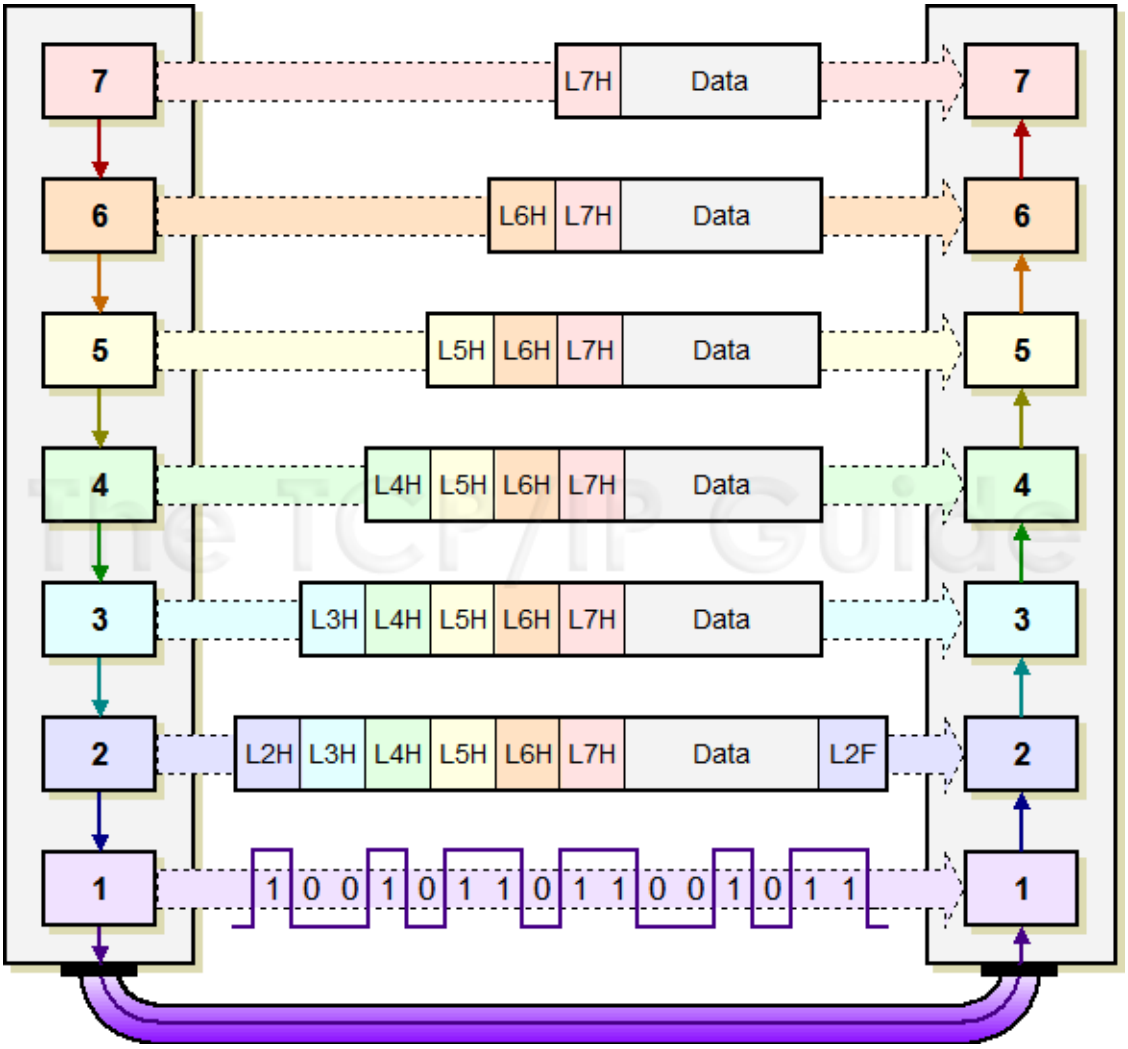
PDUs are relevant in relation to each of the first 4 layers of the OSI model as follows:

1. The Layer 1 (Physical Layer) PDU is the **bit**
2. The Layer 2 (Data Link Layer) PDU is the **frame**
3. The Layer 3 (Network Layer) PDU is the **packet**
4. The Layer 4 (Transport Layer) PDU is the **segment** (e.g. TCP segment)

(Layer 5 and above are referred to as **data**.)

Given a context pertaining to a specific layer, PDU is sometimes used as a synonym for its representation at that layer.

3.11 data encapsulation



The OSI model data encapsulation

The communication between layers higher than layer one is **logical**; the only hardware connection is at the physical layer. Thus, in order for a protocol to communicate, it must pass down its PDU to the next lower layer for transmission. We've also already seen that using the OSI terminology, lower layers are said to provide **services** to the layers immediately above them. One of the services each layer provides is this function: to handle and manage data received from the layer above.

At any particular layer N, a PDU is a complete message that implements the protocol at that layer. However, when this "layer N PDU" is passed down to layer N-1, it becomes the **data** that the layer N-1 protocol is supposed to **service**. Thus, the layer N protocol data unit (PDU) is called the layer N-1 **service data unit** (SDU). The job of layer N-1 is to transport this SDU, which it does in turn by placing the layer N SDU into its own PDU format, preceding the SDU with its own headers and appending footers as necessary. This process is called **data encapsulation**, because the entire contents of the higher-layer message are encapsulated as the data payload of the message at the lower layer.

What does layer N-1 do with its PDU? It of course passes it down to the next lower layer, where it is treated as a layer N-2 SDU. Layer N-2 creates a layer N-2 PDU containing the layer N-1 SDU and layer N-2's headers and footers. And the so the process continues, all the way down to the physical layer. In the theoretical model, we end up with a message at layer 1 consisting of application-layer data encapsulated with headers and/or footers from each of layers 7 through 2 in turn, as shown in the image above.

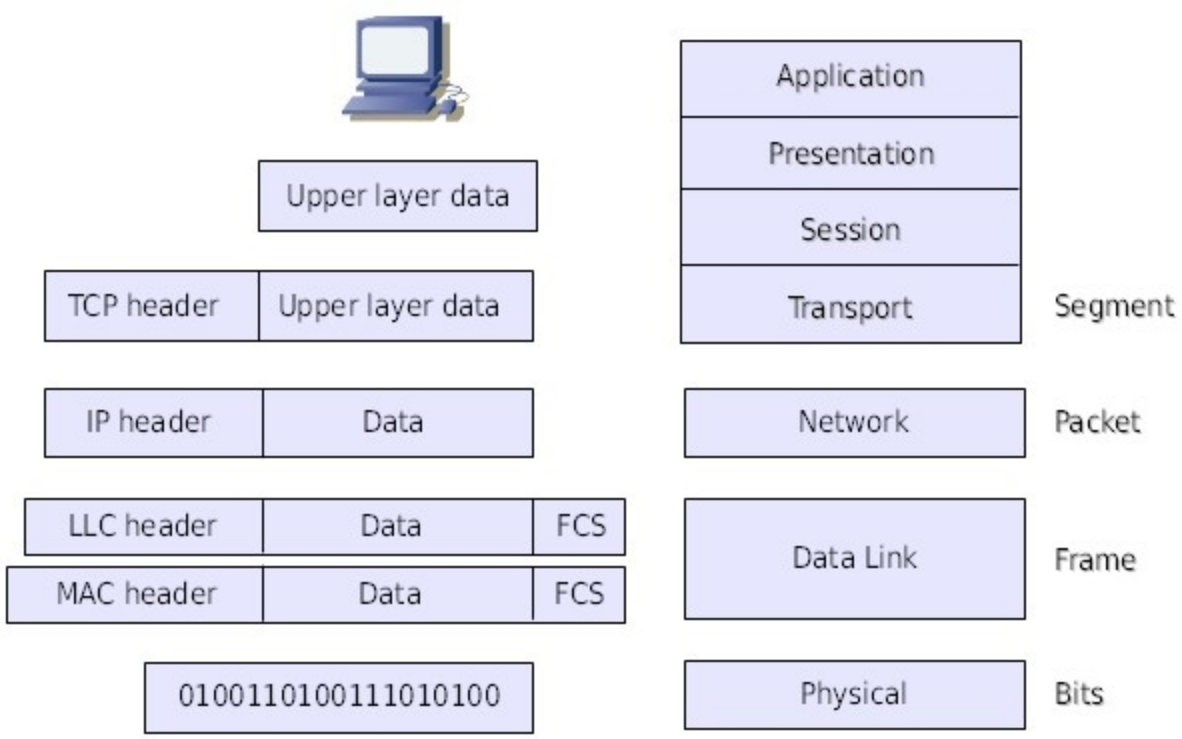
3.12 data, segments, packets, frames and bits

To communicate and exchange information, each layer uses its own specific PDUs. These hold the control information attached to the data at each layer of the model. They are usually attached to the header in front of the data field but can also be in the trailer, or end, of it.

Each PDU is attached to the data by encapsulating it at each layer of the OSI model, and each has a specific name depending on the information provided in each header. This PDU information is read only by the peer layer on the receiving device. After it's read, it's stripped off, and the data is then handed to the next layer up.

The figure below shows the PDUs and how they attach control information to each layer. This figure demonstrates how the upper-layer user data is converted for transmission on the network.

The data stream is handed down to the Transport layer, which sets up a virtual circuit to the receiving device by sending over a synch packet. Next, the data stream is broken up into smaller pieces, and a Transport layer header (a PDU) is created and attached to the header of the data field; now the piece of data is called a **segment**. Each segment is sequenced so the data stream can be put back together on the receiving side exactly as it was transmitted.



PDU types in the encapsulation process

Each segment is then handed to the Network layer for network addressing and routing through the internetwork. Logical addressing (for example, IP) is used to get each segment to the correct network. The Network layer protocol adds a control header to the segment handed down from the Transport layer, and what we have now is called a **packet** or **datagram**. Remember that the Transport and Network layers work together to rebuild a data stream on a receiving host, but it's not part of their work to place their PDUs on a local network segment—which is the only way to get the information to a router or host.

It's the Data Link layer that's responsible for taking packets from the Network layer and placing them on

the network medium (cable or wireless). The Data Link layer encapsulates each packet in a **frame**, and the frame's header carries the hardware address of the source and destination hosts. If the destination device is on a remote network, then the frame is sent to a router to be routed through an internetwork. Once it gets to the destination network, a new frame is used to get the packet to the destination host. (In the picture above, **FCS** stands for Frame Check Sequence, consists of 4 bytes and is a CRC (Cyclic Redundancy Check) which verifies the integrity of the frame).

To put this frame on the network, it must first be put into a digital signal. Since a frame is really a logical group of 1s and 0s, the Physical layer is responsible for encoding these digits into **bits**, a digital signal, which is read by devices on the same local network.

The receiving devices will synchronize on the digital signal and extract (decode) the 1s and 0s from the digital signal. At this point the devices build the frames, run a CRC, and then check their answer against the answer in the frame's FCS field. If it matches, the packet is pulled from the frame, and what's left of the frame is discarded.

This process is called **de-encapsulation**. The packet is handed to the Network layer, where the address is checked. If the address matches, the segment is pulled from the packet, and what's left of the packet is discarded. The segment is processed at the Transport layer, which rebuilds the data stream and acknowledges to the transmitting station that it received each piece. It then happily hands the data stream to the upper-layer application.

At a transmitting device, the data encapsulation method works like this:

1. User information is converted to **data** for transmission on the network.

2. Data is converted to **segments** and a reliable connection is set up between the transmitting and receiving hosts.

3. Segments are converted to **packets** or **datagrams**, and a logical address is placed in the header so each packet can be routed through an internetwork.

4. Packets or datagrams are converted to **frames** for transmission on the local network. Hardware (Ethernet) addresses are used to uniquely identify hosts on a local network segment.

5. Frames are converted to **bits**, and a digital encoding and clocking scheme is used.

3.13 the DoD (TCP/IP) model

While the OSI model is the current and the widely accepted model for structural classification of the communication protocols, we have to mention the model which preceded it, as well.

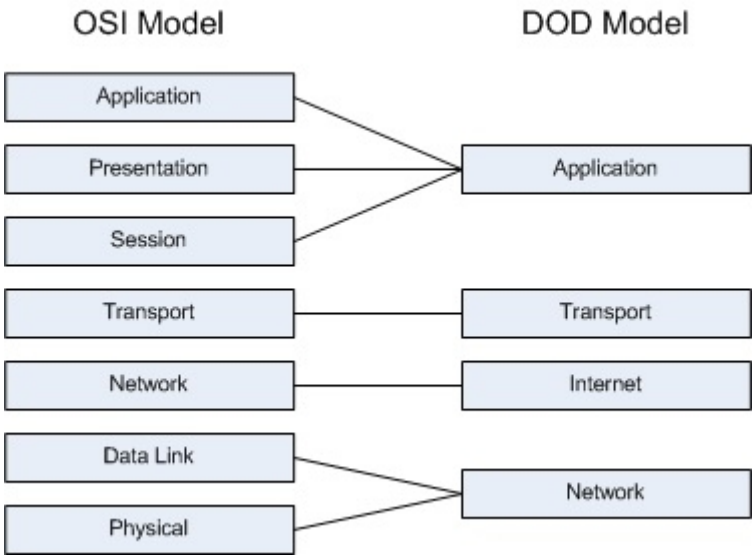
This model is the **Department of Defense** model (DoD), also known as the TCP/IP model. It was developed in the 1970's for DARPA's Internetwork Project, project that later developed into what we call now the Internet.

The core Internet protocols fit very well in this model as well, newer and more specialized communication protocols are better suited for the OSI model.

The DoD model consists of four layers only:

- the Process (**Application)** layer, corresponding to the top three layers in the OSI modeling
- the **Host-to-Host** layer
- the **Internet** layer, corresponding to the Network layer in OSI
- the **Network Access** layer, corresponding to the Data Link + Physical layers in the OSI modeling

This correspondence can be viewed in the picture below.


OSI Model
DOD Model
Application
Presentation
Session
Transport
Network
Data Link
Physical
Application
Transport
Internet
Network

chapter 4 sockets

4.1 sockets - basics

Although the socket concept covers a variety of cases, like Unix sockets (end-point(s) in local interprocess communication) or end-point of a bi-directional communication link in the Berkeley sockets, we will limit ourselves, within the scope of this course, to the most used variety, namely internet sockets. An (internet) **socket** is a logical entity which describes the end point(s) of a communication link between two IP entities (entities which implement the Internet Protocol). Sockets are identified by the IP address and the port number.

IP addresses come in two flavors, as defined in by the two versions of the Internet Protocol currently in use, version 4 and version 6.

Version 4 IP addresses are 32 bits long and are represented as a sequence of 4 8-bit integers, dot separated, ranging from 0 to 255, like 193.231.200.67.

Version 6 IP addresses are 128 bits long and are represented by a sequence of eight 16-bit integers, separated by columns (:). An example - 2008:0CB8:85A4:0000:0000:8F2C:0371:7714.

Port numbers range from 0 to 65535 (2^16 – 1) and are split into 3 categories:

1. **well known ports** - ranging from 0 to 1023 – these ports are under the control of IANA (Internet Assigned Number Authority), a selective list is shown in the table below:

Port number	UDP protocol	TCP protocol	Other
1		TCPMUX	
5	Remote Job Entry (RJE)		
7	Echo		
15	NETSTAT		
20		FTP - data	
21		FTP – control	
22	Secure Shell		
23	Telnet		
25	Simple Mail Transfer Protocol (SMTP)		
41	Graphics		
42	ARPA Host Name Server Protocol		WINS
43		WHOIS	
53	Domain Name System (DNS)		
57		Mail Transfer Protocol (MTP)	
67	BOOTP		
68	BOOTP		
69	TFTP		
79		Finger	

80		HTTP	
107		Remote Telnet	
109		Post Office Protocol 2 (POP2)	
110		POP3	
115		Simple FTP (SFTP)	
118	SQL services		
123	Network Time Protocol (NTP)		
137	NetBIOS Name Service		
138	NetBIOS Datagram Service		
139	NetBIOS Session Service		
143	Internet Message Access Protocol (IMAP)		
156	SQL service		
161	Simple Network Management Protocol (SNMP)		
162	SNMP Trap		
179		Border Gateway Protocol (BGP)	
194		Internet Relay Chat (IRC)	
213	IPX		

2. **registered ports** - ranging from 1024 to 49151 – registered by ICANN, as a convenience to the community, should be accessible to ordinary users. A list of some of these ports is shown below:

Port number	UDP protocol	TCP protocol	Other
1080		SOCKS proxy	
1085	WebObjects		
1098	RMI activation		
1099	RMI registry		
1414		IBM WebSphere MQ	
1521		Oracle DB default listener	
2030	Oracle services for Microsoft Transaction Server		
2049	Network File System		
2082		CPanel default	
3306	MySQL DB system		
3690	Subversion version control system		
3724	World of Warcraft online gaming		
4664		Google Desktop Search	
5050		Yahoo Messenger	
5190		ICQ and AOL IM	
5432	PostgreSQL DB system		
5500		VNC remote desktop protocol	

5800		VNC over HTTP	
6000/6001		X11	
6881-6887	BitTorrent		
6891-6900	Windows Live Messenger – File transfer		
6901	Windows Live Messenger – Voice		
8080		Apache Tomcat	
8086/8087	Kaspersky AV Control Center		
8501	Duke Nukem 3D		
9043		WebSphere Application Server	
14567	Battlefield 1942		
24444	NetBeans IDE		
27010/27015	Half-Life, Counter-Strike		
28910	Nintendo Wi-Fi Connection		
33434	traceroute		

3. **dynamic (private) ports**, ranging from 49152 to 65535

4.2 posix sockets

The socket APIs (Application Programming Interface) are rooted into the POSIX socket API. POSIX stands for Portable Operating System Interface – a common name for a set of IEEE standards used to define APIs. This family of standards dates back to 1988 and is identified as IEE 1003 or ISO/IEC 9945.

The socket communication is, in general, asymmetric. One of the two communicating entities plays the role of a **server**. The server listens for incoming requests at a certain port. This port number is public, and together with the IP address identifies the server. The actual communication is initiated by the **client,** who sends a connection request to the server. If the connection request Is accepted, the server creates (in general) another socket, which is dedicated to the communication with that particular client. The closure of this communication link can be initiated by either the client or by the server.

To create a client socket, two calls are necessary. The first one creates a file descriptor (fd) which is basically a number which identifies an I/O channel (not different from the file descriptor resulted from a fopen() call which opens a file).

The prototype of this call is the following:

```
int socket(int family, int type, int protocol);
```

The **family** parameter specifies the address family of the socket and may take one of the following values, the list itself depending on the implementation platform:

- `AF_APPLETALK`

- `AF_INET` – most used, indicates an IP version 4 address
- `AF_INET6` – indicates an IP version 6 address
- `AF_IPX`
- `AF_KEY`
- `AF_LOCAL`
- `AF_NETBIOS`
- `AF_ROUTE`
- `AF_TELEPHONY`
- `AF_UNSPEC`

The **type** parameter specifies the socket stream type and may take the following values:

- `SOCK_DGRM` – the transport level protocol is UDP
- `SOCK_STREAM` – the transport level protocol is TCP
- `SOCK_RAW` – used to generate/receive packets of a type that the kernel doesn't explicitly support

The value of the **protocol** parameter is set to 0, except for raw sockets.

The function `socket()` returns an integer. In case of success, it is the identifier of a file descriptor (fd), and if the call fails, it is an error code.

The second call connects the client to the server. Here is the signature of the `connect()` call.

```
int connect(int sock_fd, struct sockaddr * server_addr, int addr_len);
```

The `sockaddr` structure varies depending on the protocol selected. For reference purposes, let's display it, together with another associated structure – `sockaddr_in`, both used in the context of IPv4.

```
struct sockaddr {
        ushort  sa_family;
        char    sa_data[14];
};
struct sockaddr_in {
        short   sin_family;
        u_short sin_port;
        struct  in_addr sin_addr;
        char    sin_zero[8];
};
```

Both structures have the same size and have compatible content, therefore casting is allowed.

The `sock_fd` parameter identifies the local socket and the `server_addr` parameter contains information about the server we try to connect to. `addr_len` is just the size (in bytes) of this structure.

The `connect()` call returns 0, in case of success or a negative error indicator, in case of failure.

To create a server socket, four calls are necessary. Here are the prototypes of these calls:

```
int socket(int family, int type, int protocol);
int bind(int sock_fd, struct sockaddr * my_addr, int addr_len);
int listen(int sock_fd, int backlog);
int accept(int sock_fd, struct sockaddr * client_addr, int * addr_len);
```

The `bind()` function merely associates the socket to a specified port, while the `listen()` function sets the server in listening mode.

A few remarks. Why not binding the client socket to a particular port, as well? Well, nobody stops us from invoking the bind() function on a client socket, but this is not exactly relevant. While the server port has to be known, because the client must know both the IP address (or the URL, if that is the case) and the port of the server, it is not important to know the port of the client. The assignment of a port to a client socket is done by the operating system, and this solution is quite satisfactory.

The `accept()` call causes the process to block until a client connects to the server. Thus, it wakes up the process when a connection from a client has been successfully established. It returns a new file descriptor, and all communication on this connection should be done using the new file descriptor. In case of an error, the return value is an error code. The first parameter is the fd of the local listening socket, and the second one is a reference pointer to the address of the client at the other end of the connection. The third parameter is a pointer to the size of this structure.

chapter 5 the application layer

5.1 purpose

This is the layer where all the applications are found. The layers below the application layer are there to provide reliable transport, but they do not do real work for users. In this chapter we will study some network applications from user's point of view.

However, even in the application layer there is a need for support protocols, to allow the applications to function. We will look at some of these protocols and the applications they support with a special emphasis on HTTP, DHCP – Dynamic Host Configuration Protocol and DNS – the Domain Name System.

The different protocols and applications covered in this chapter include the following:

- HTTP
- Telnet
- FTP
- TFTP
- DHCP/BootP
- NFS
- SMTP
- SNMP
- DNS

5.2 HTTP

5.2.1 what is http?

HTTP stands for **H**yper**T**ext **T**ransfer **P**rotocol while **hypertext** means text containing links to another text. HTTP was created by by Tim Berners-Lee in 1990 at CERN as a mean to store scientific data. It quickly evolved into the preferred communication protocol over the internet.

The first official version – HTTP 1.0 – dates from 05/95 and is the object of RFC 1945 (www.apps.ietf.org/rfc/rfc1945.html). It is authored by Tim Berners-Lee, Roy Fielding and Henrik Nielsen.

The second version, namely HTTP 1.1, was the object of several RFCs, of which we mention RFC 2068 (01/97), RFC 2616 (06/99), RFC 2617 (06/99) and RFC 2774 (02/00). The current specification was set in 2014 by the RFC 7230 family.

For a complete specification of the different HTTP versions, check the official HTTP site – www.w3.org/Protocols . As a site for understanding how HTTP works, we recommend www.jmarshall.com/easy/http.

The next version of the HTTP specification was HTTP 2.0 (or HTTP/2). It has been released as RFC 7540 in May 2015 - https://tools.ietf.org/html/rfc7540

The working group for HTTP/2 addressed the following issues and goals:

- Negotiation mechanism that allows clients and servers to elect to use HTTP 1.1, 2.0, or potentially other non-HTTP protocols.
- Maintain high-level similarity with HTTP 1.1 (for example with methods, status codes, header fields, and URIs, and most header fields)
- Decrease latency to improve page load speed in web browsers by considering:
  - Data compression of HTTP headers
  - Server push technologies
  - Fixing the head-of-line blocking problem in HTTP 1
  - Loading page elements in parallel over a single TCP connection
- Support common existing use cases of HTTP, such as desktop web browsers, mobile web browsers, web APIs, web servers at various scales, proxy servers, reverse proxy servers, firewalls, and content delivery networks

HTTP/2 is currently (03.2021) used by over 50% and is supported by all major browsers.

The proposed successor of HTTP/2 is HTTP/3. As of March 2021 the HTTP/3 specification is an Internet draft and already has several implementations. Currently (03.2021) 7.32% of desktop web browsers support HTTP/3 and 9.3% of the top 10 million web sites support HTTP/3.

While HTTP/1.1 and HTTP/2 use TCP as their transport layer protocol, HTTP/3 uses QUIC, a transport layer protocol developed initially by Google where user space congestion control is used over UDP.

5.2.2 the structure of http transactions

HTTP follows the client – server model. The client sends a request message to the server. The server answers with a response message. These messages may have different contents, but they also have some common structural elements, as follows:

1. an initial line
2. zero or more header lines
3. a blank line (CR/LF)
4. an optional message body

```
<initial line>
      Header1: value1
...
      Headern: valuen

<optional data block>
```

5.2.3 the initial request line

Contains 3 elements, separated by spaces:

- a **command** (method) name (like GET, POST, HEAD, ...)
- a file specification (**path**) (the part of the URL after the host name)

- the HTTP **version** (usually, HTTP/1.0).

Here is an example of an initial request line:

```
GET /path/to/the/file/index.html HTTP/1.0
```

5.2.4 http commands (methods)

As of HTTP 1.1, there are 8 HTTP commands (methods) that are widely supported. Here is their list:

1. GET
2. HEAD
3. POST
4. CONNECT
5. DELETE
6. OPTIONS
7. PUT
8. TRACE

Three other commands are listed, as well, in the HTTP 1.1 specification, but lack of support makes them obsolete. These commands are:

- LINK
- UNLINK
- PATCH

The HEAD command is identical to the GET command in all respects but one. The only difference is that the response must not have a body. All the information requested is returned in the header section of the response.

5.2.5 the GET and POST methods

The GET method means retrieve whatever information (in the form of an entity) is identified by the Request-URI. If the Request-URI refers to a data-producing process, it is the produced data which shall be returned as the entity in the response and not the source text of the process, unless that text happens to be the output of the process.

The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. POST is designed to allow a uniform method to cover the following functions:

- Annotation of existing resources;
- Posting a message to a bulletin board, newsgroup, mailing list, or similar group of articles;
- Providing a block of data, such as the result of submitting a form, to a data-handling process;
- Extending a database through an append operation.

The actual function performed by the POST method is determined by the server and is usually dependent on the Request-URI. The posted entity is subordinate to that URI in the same way that a file is subordinate to a directory containing it, a news article is subordinate to a newsgroup to which it is posted, or a record is subordinate to a database.

The action performed by the POST method might not result in a resource that can be identified by a URI. In this case, either 200 (OK) or 204 (No Content) is the appropriate response status, depending on whether or not the response includes an entity that describes the result.

5.2.6 differences between GET and POST

1. The method GET is intended for getting (retrieving) data, while POST may involve anything, like storing or updating data, or ordering a product, or sending E-mail
2. When used for form data submission, GET attaches this data to the URL of the request, after the "?" character, as a sequence of "name=value" pairs, separated by the character "&" or ";" On the other side, form data submitted by POST may be encoded either as above (using `application/x-www-form-urlencoded` content type), or in the message body, (encoded as `multipart/form-data`).
3. A POST request requires an extra transmission to retrieve the message body, while a GET request allows data sent via the URL to be processed immediately.

5.2.7 the initial response (status) line

Contains 3 elements, separated by spaces (although the reason phrase may contain spaces, as well):

- the HTTP **version** of the response
- a response **status code** (a number)
- a response status **reason phrase** (a human readable response status)

Here is an example of an initial response line:

```
HTTP/1.0 404 Not Found
```

5.2.8 the status code

A three-digit integer, where the first digit identifies the general category of response:

- **1xx** indicates an informational message only
- **2xx** indicates success of some kind
- **3xx** redirects the client to another URL
- **4xx** indicates an error on the client's part
- **5xx** indicates an error on the server's part

The most common status codes are:

- **200 OK** - the request succeeded, and the resulting resource (e.g. file or script output) is returned in the message body.

- **404 Not Found** - the requested resource doesn't exist.
- **301 Moved Permanently**
- **302 Moved Temporarily**
- **303 See Other** (HTTP 1.1 only) - the resource has moved to another URL (given by the **Location:** response header), and should be automatically retrieved by the client. This is often used by a CGI script to redirect the browser to an existing file.
- **500 Server Error** - an unexpected server error. The most common cause is a server-side script that has bad syntax, fails, or otherwise can't run correctly.

5.2.9 header lines

A header line consists of two parts, header **name** and header **value**, separated a semicolon. The HTTP 1.0 version specifies 16 headers, none of them mandatory, while the HTTP 1.1 version specifies 46 of them, out of which, one (Host) is mandatory. Although the header names are not case sensitive, header values are.

A couple of examples of header lines:

```
User-agent: Mozilla/3.0Gold
Last-Modified: Fri, 31 Dec 1999 23:59:59 GMT
```

Header lines which begin with spaces or tabs are parts of the previous header line.

5.2.10 the message body

An HTTP message may have a body of data sent after the header lines. The most common use of the message body is in a response, that is, where the requested resource is returned to the client, or perhaps explanatory text if there's an error. In a request, this is where user-entered data or uploaded files are sent to the server.

If an HTTP message includes a body, the header lines of the message are used to describe the body. In particular,

- the **Content-Type:** header gives the MIME-type of the data in the body, such as **text/html** or **image/jpg**.
- the **Content-Length:** header gives the number of bytes in the body.

5.2.11 mime types/subtypes

MIME stands for Multipurpose Internet Mail Extensions. Each extension consists of a type and a subtype. RFC 1521 (www.apps.ietf.org/rfc/rfc1521.html) defines 7 types and several subtypes, although the list of admissible subtypes is much longer.

Here is the list of the seven types, together with the subtypes defined in this particular RFC.

1. **text**, with subtype plain
2. **multipart**, with subtypes mixed, alternative, digest, parallel
3. **message**, with subtypes rfc822, partial, external-body

4. **application**, with subtypes octet-stream, postscript
5. **image**, with subtypes jpeg, gif
6. **audio**, with subtype basic
7. **video**, with subtype mpeg

5.2.12 an example of an http transaction

To retrieve the file at the URL http://web.info.uvt.ro/path/file.html

first open a socket to the host **web.info.uvt.ro**, port 80 (use the default port of 80 because none is specified in the URL). Then, send something like the following through the socket:

```
GET /path/file.html HTTP/1.0
From: someuser@yahoo.com
User-Agent: HTTPTool/1.0
[blank line here]
```

The server should respond with something like the following, sent back through the same socket:

```
HTTP/1.0 200 OK
Date: Fri, 31 Dec 1999 23:59:59 GMT
Content-Type: text/html
Content-Length: 1354

<html>
<body>
<h1>Happy birthday!</h1>
(more file contents)
.
.
</body>
</html>
```

After sending the response, the server closes the socket.

5.2.13 HTTP/2 protocol overview

HTTP/2 provides an **optimized transport** for HTTP semantics. HTTP/2 supports all of the core features of HTTP/1.1 but aims to be more efficient in several ways.

The basic protocol unit in HTTP/2 is a **frame**. Each frame type serves a different purpose. For example, HEADERS and DATA frames form the basis of HTTP requests and responses; other frame types like SETTINGS, WINDOW_UPDATE, and PUSH_PROMISE are used in support of other HTTP/2 features.

Multiplexing of requests is achieved by having each HTTP request/response exchange associated with its own stream. Streams are largely independent of each other, so a blocked or stalled request or response does not prevent progress on other streams.

Flow control and **prioritization** ensure that it is possible to efficiently use multiplexed streams. Flow control helps to ensure that only data that can be used by a receiver is transmitted. Prioritization ensures that limited resources can be directed to the most important streams first.

HTTP/2 adds a new interaction mode whereby a server can **push responses** to a client. Server push allows a server to speculatively send data to a client that the server anticipates the client will need, trading off some network usage against a potential latency gain. The server does this by synthesizing a request, which it sends as a PUSH_PROMISE frame. The server is then able to send a response to the synthetic request on a separate stream.

Because HTTP header fields used in a connection can contain large amounts of redundant data, frames that contain them are **compressed**. This has especially advantageous impact upon request sizes in the common case, allowing many requests to be compressed into one packet.

An interesting review of the novelties in HTTP/2 can be found at:
https://developers.google.com/web/fundamentals/performance/http2

5.3 telnet

Telnet is the chameleon of protocols—its specialty is terminal emulation. It allows a user on a remote client machine, called the **Telnet client**, to access the resources of another machine, the **Telnet server**. Telnet achieves this by pulling a fast one on the Telnet server and making the client machine appear as though it were a terminal directly attached to the local network. This projection is actually a software image - a virtual terminal that can interact with the chosen remote host.

These emulated terminals are of the text-mode type and can execute refined procedures like displaying menus that give users the opportunity to choose options from them and access the applications on the duped server. Users begin a Telnet session by running the Telnet client software and then logging into the Telnet server.

Telnet was developed in 1969 beginning with RFC 15, extended in RFC 854, and standardized as Internet Engineering Task Force (IETF) Internet Standard STD 8, one of the first Internet standards.

The well-known port for telnet is **port 23**.

5.4 file transfer protocol - FTP

File Transfer Protocol (FTP) is the protocol that actually lets us transfer files, and it can accomplish this between any two machines using it. But FTP isn't just a protocol; it's also a program.

Operating as a protocol, FTP is used by applications. As a program, it's employed by users to perform file tasks by hand. FTP also allows for access to both directories and files and can accomplish certain types of directory operations, such as relocating into different ones. FTP teams up with Telnet to transparently log you into the FTP server and then provides for the transfer of files.

Accessing a host through FTP is only the first step, though. Users must then be subjected to an authentication login that's probably secured with passwords and usernames implemented by system administrators to restrict access. But you can get around this somewhat by adopting the username "anonymous" - though what you'll gain access to will be limited.

Even when employed by users manually as a program, FTP's functions are limited to listing and manipulating directories, typing file contents, and copying files between hosts. It can't execute remote files as

programs.

Most common web browsers provide support for some of the FTP functionality, especially for file retrieval from FTP servers. FTP URL syntax is described in RFC 1738, taking the form

```
ftp://[<user>[:<password>]@]<host>[:<port>]/<url-path>
```

The current specification for FTP is RFC 959 (October 1985), amended with several proposed standards, like RFCs 2228 and 2428.

There are 2 well-known ports associated with FTP – **port 20** for data exchange and **port 21** for control commands.

It is worth mentioning that in Jan 2021 Chrome disabled the support for FTP, step followed by other browsers, like Firefox.

5.5 trivial file transfer protocol - TFTP

Trivial File Transfer Protocol (TFTP) is the stripped-down, stock version of FTP, but it's the protocol of choice if you know exactly what you want and where to find it, plus it's so easy to use and it's fast too! It doesn't give you the abundance of functions that FTP does, though. TFTP has no directory-browsing abilities; it can do nothing but send and receive files. This compact little protocol also skimps in the data department, sending much smaller blocks of data than FTP, and there's no authentication as with FTP, so it's insecure. Few sites support it because of the inherent security risks and is mostly used for local data exchange, in conjunction with DHCP and the BootP protocols.

The TFTP PDU header consists, in general of 16 bits for the OpCode followed (in general) by 16 bits for either an error code or a block number. Depending on the operation type (RRQ/WRQ), the OpCode field may be followed by a string (file specification), as well. For details, check RFC 1350 or an useful reference site - http://www.networksorcery.com/enp/protocol/tftp.htm

TFTP is implemented on top of UDP, while FTP is implemented mainly on top of TCP.

The original specification of TFP dates back to 1980, as an IEN (Internet Experiment Note) and was later formalized in RFC 1350.

The current versions of the TFTP specification are the object of RFCs 2347, 2348, 2349, 3617.

The well-known port for TFTP is **port 69**. Data transfer must be initiated at this port, but subsequent data transfer may occur at another port, as negotiated by the client and tse server.

5.5.1 protocol description

1. The initiating host A sends an RRQ (read request) or WRQ (write request) packet to host S at the well-known port number 69, containing the filename and transfer mode.
2. S replies with an ACK (acknowledgment) packet to WRQ and directly with a DATA packet to RRQ. Packet is sent from a newly allocated ephemeralport, all future packets to host S should be to this port.
3. The source host sends numbered DATA packets to the destination host, all but the last containing a full-sized block of data (512 bytes). The destination host replies with numbered ACK packets for all DATA packets.
4. The final DATA packet must contain less than a full-sized block of data to signal that it is the last. If the size of the transferred file is an exact multiple of the block-size, the source sends a final DATA packet containing 0 bytes of data.
5. Receiver responds to each DATA with associated numbered ACK. Sender responds to the first

received ACK of a block with DATA of the next block.

6. If an ACK is not eventually received, a retransmit timer resends DATA packet.

5.6 dynamic host configuration - DHCP and bootstrap - BootP protocol

Dynamic Host Configuration Protocol (DHCP) gives IP addresses to hosts. It allows easier administration and works well in small-to-even-very-large network environments. All types of hardware can be used as a DHCP server, including a Cisco router.

DHCP differs from BootP in that BootP gives an IP address to a host, but the host's hardware address must be entered manually in a BootP table. You can think of DHCP as a dynamic BootP. But remember that BootP is also used to send an operating system that a host can boot from. DHCP can't do that.

But there is a lot of information a DHCP server can provide to a host when the host is requesting an IP address from the DHCP server. Here's a list of the information a DHCP server can provide:

- IP address
- Subnet mask
- Domain name
- Default gateway (routers)
- DNS
- WINS (Windows Internet Name Service) information

A DHCP server can give us even more information than this, but the items in that list are the most common.

There are several well-known port associated to DHCP - **port 546** for DHCPv6 clients, **port 547** for DHCPv6 servers, **port 647** for the DHCP Failover Protocol. BootP servers listen at the well-known **port 67** while BootP clients use the well-known **port 68**. DHCP itself uses these 2 well-known ports, as well.

5.6.1 history

RFC 1531 initially defined DHCP as a standard-track protocol in October 1993, succeeding the Bootstrap Protocol (BOOTP). The next update, RFC 2131 released in 1997 is the current DHCP definition for Internet Protocol version 4 (IPv4) networks. The extensions of DHCP for IPv6 (DHCPv6) were published as RFC 3315.

5.6.2 technical overview

Dynamic Host Configuration Protocol automates network-parameter assignment to network devices from one or more fault-tolerant DHCP servers. Even in small networks, DHCP is useful because it can make it easy to add new machines to the network.

When a DHCP-configured client (a computer or any other network-aware device) connects to a network, the DHCP client sends a broadcast query requesting necessary information from a DHCP server. The DHCP server manages a pool of IP addresses and information about client configuration parameters such as default gateway, domain name, the DNS servers, other servers such as time servers, and so forth. On receiving a valid request, the server assigns the computer an IP address, a lease (length of time the allocation is valid), and other IP configuration parameters, such as the subnet mask and the default gateway. The query is

typically initiated immediately after booting, and must complete before the client can initiate IP-based communication with other hosts.

Depending on implementation, the DHCP server may have three methods of allocating IP-addresses:

- *dynamic allocation*: A network administrator assigns a range of IP addresses to DHCP, and each client computer on the LAN has its IP software configured to request an IP address from the DHCP server during network initialization. The request-and-grant process uses a lease concept with a controllable time period, allowing the DHCP server to reclaim (and then reallocate) IP addresses that are not renewed (dynamic re-use of IP addresses).
- *automatic allocation*: The DHCP server permanently assigns a free IP address to a requesting client from the range defined by the administrator. This is like dynamic allocation, but the DHCP server keeps a table of past IP address assignments, so that it can preferentially assign to a client the same IP address that the client previously had.
- *static allocation*: The DHCP server allocates an IP address based on a table with MAC address/IP address pairs, which are manually filled in (perhaps by a network administrator). Only requesting clients with a MAC address listed in this table will be allocated an IP address. This feature (which is not supported by all devices) is variously called *Static DHCP Assignment* (by DD-WRT), *fixed-address* (by the dhcpd documentation), *DHCP reservation* or *Static DHCP* (by Cisco/Linksys), and *IP reservation* or *MAC/IP binding* (by various other router manufacturers).

5.6.3 technical details

DHCP uses the same two ports assigned by IANA for BOOTP: 67/UDP for sending data to the server, and 68/UDP for data to the client.

DHCP operations fall into four basic phases: IP discovery, IP lease offer, IP request, and IP lease acknowledgment.

When a DHCP client and server are on the same subnet, they will communicate via UDP broadcasts. When the client and server are on different subnets, IP discovery and IP request messages are sent via UDP broadcasts, but IP lease offer and IP lease acknowledgment messages are sent via unicast.

5.6.4 DHCP discovery

The client broadcasts messages on the physical subnet to discover available DHCP servers. Network administrators can configure a local router to forward DHCP packets to a DHCP server from a different subnet. This client-implementation creates a User Datagram Protocol (UDP) packet with the broadcast destination of 255.255.255.255 or the specific subnet broadcast address.

A DHCP client can also request its last-known IP address (in the example below, 192.168.1.100). If the client remains connected to a network for which this IP is valid, the server might grant the request. Otherwise, it depends whether the server is set up as authoritative or not. An authoritative server will deny the request, making the client ask for a new IP immediately. A non-authoritative server simply ignores the request, leading to an implementation-dependent timeout for the client to give up on the request and ask for a new IP address.

5.6.5 DHCP offer

When a DHCP server receives an IP lease request from a client, it reserves an IP address for the client and extends an IP lease offer by sending a DHCPOFFER message to the client. This message contains the client's MAC address, the IP address that the server is offering, the subnet mask, the lease duration, and the IP address of the DHCP server making the offer.

The server determines the configuration based on the client's hardware address as specified in the CHADDR (Client Hardware Address) field. Here the server, 192.168.1.1, specifies the IP address in the YIADDR (Your IP Address) field.

5.6.6 DHCP request

A client can receive DHCP offers from multiple servers, but it will accept only one DHCP offer and broadcast a DHCP request message. Based on the Transaction ID field in the request, servers are informed whose offer the client has accepted. When other DHCP servers receive this message, they withdraw any offers that they might have made to the client and return the offered address to the pool of available addresses.

5.6.7 DHCP acknowledgment

When the DHCP server receives the DHCPREQUEST message from the client, the configuration processes enters its final phase. The acknowledgment phase involves sending a DHCPACK packet to the client. This packet includes the lease duration and any other configuration information that the client might have requested. At this point, the IP configuration process is completed.

The protocol expects the DHCP client to configure its network interface with the negotiated parameters.

After the client obtains an IP address, the client may use the Address Resolution Protocol (ARP) to prevent IP conflicts caused by overlapping address pools of DHCP servers.

5.6.8 DHCP information

A DHCP client may request more information than the server sent with the original DHCPOFFER. The client may also request repeat data for a particular application. For example, browsers use *DHCP Inform* to obtain web proxy settings via WPAD. Such queries do not cause the DHCP server to refresh the IP expiry time in its database.

5.6.9 DHCP releasing

The client sends a request to the DHCP server to release the DHCP information and the client deactivates its IP address. As client devices usually do not know when users may unplug them from the network, the protocol does not mandate the sending of *DHCP Release*.

5.6.10 client configuration parameters

A DHCP server can provide optional configuration parameters to the client. RFC 2132 describes the available DHCP options defined by Internet Assigned Numbers Authority (IANA) - DHCP and BOOTP PARAMETERS.

A DHCP client can select, manipulate and overwrite parameters provided by a DHCP server.

5.6.11 options

An option exists to identify the vendor and functionality of a DHCP client. The information is a variable-length string of characters or octets which has a meaning specified by the vendor of the DHCP client. One method that a DHCP client can utilize to communicate to the server that it is using a certain type of hardware or firmware is to set a value in its DHCP requests called the Vendor Class Identifier (VCI) (Option 60). This method allows a DHCP server to differentiate between the two kinds of client machines and process the requests from the two types of modems appropriately. Some types of set-top boxes also set the VCI (Option 60) to inform the DHCP server about the hardware type and functionality of the device. The value that this option is set to give the DHCP server a hint about any required extra information that this client needs in a DHCP response.

5.6.12 DHCP relaying

In small networks DHCP typically uses broadcasts. However, in some circumstances, unicast addresses will be used, for example: when networks have a single DHCP server that provides IP addresses for multiple subnets. When a router for such a subnet receives a DHCP broadcast, it converts it to unicast (with a destination MAC/IP address of the configured DHCP server, source MAC/IP of the router itself). The GIADDR field of this modified request is populated with the IP address of the router interface on which it received the original DHCP request. The DHCP server uses the GIADDR field to identify the subnet of the originating device in order to select an IP address from the correct pool. The DHCP server then sends the DHCP OFFER back to the router via unicast. The router then converts the DHCP OFFER back to a broadcast, sent out on the interface of the original device.

5.7 network file system - NFS

Network File System (NFS) is a protocol specializing in file sharing. It allows two different types of file systems to interoperate. It works like this: Suppose the NFS server software is running on an NT server, and the NFS client software is running on a Unix host. NFS allows for a portion of the RAM on the NT server to transparently store Unix files, which can, in turn, be used by Unix users. Even though the NT file system and Unix file system are unlike - they have different case sensitivity, filename lengths, security, and so on - both Unix users and NT users can access that same file with their normal file systems, in their normal way.

NFS usually operates on ports like 111 and 2049, but there is no particular well-known port associated to NFS.

5.7.1 the original NFS version

The implementation details are defined in RFC 1094. Sun used version 1 only for in-house experimental purposes. When the development team added substantial changes to NFS version 1 and released it outside of Sun, they decided to release the new version as v2, so that version interoperation and RPC version fallback could be tested.

5.7.2 NFSv2

Version 2 of the protocol (defined in RFC 1094, March 1989) originally operated entirely over UDP. Its designers meant to keep the protocol stateless, with locking (for example) implemented outside of the core protocol.

5.7.3 NFSv3

Version 3 (RFC 1813, June 1995) added:

- support for 64-bit file sizes and offsets, to handle files larger than 2 gigabytes (GB);
- support for asynchronous writes on the server, to improve write performance;
- additional file attributes in many replies, to avoid the need to re-fetch them;
- a READDIRPLUS operation, to get file handles and attributes along with file names when scanning a directory;
- assorted other improvements.

At the time of introduction of Version 3, vendor support for TCP as a transport-layer protocol began increasing. While several vendors had already added support for NFS Version 2 with TCP as a transport, Sun Microsystems added support for TCP as a transport for NFS at the same time it added support for Version 3. Using TCP as a transport made using NFS over a WAN more feasible.

5.7.4 NFSv4

Version 4 (RFC 3010, December 2000; revised in RFC 3530, April 2003), influenced by AFS and CIFS, includes performance improvements, mandates strong security, and introduces a stateful protocol. Version 4 became the first version developed with the Internet Engineering Task Force (IETF) after Sun Microsystems handed over the development of the NFS protocols.

NFS version 4 minor version 1 (NFSv4.1) has been approved by the IESG and received an RFC number since Jan 2010. The NFSv4.1 specification aims to provide protocol support to take advantage of clustered server deployments including the ability to provide scalable parallel access to files distributed among multiple servers (pNFS extension). NFS version 4.2 specification was published in 11.2016 as RFC 7862.

5.8 simple mail transfer protocol - SMTP

Simple Mail Transfer Protocol (SMTP), answering our ubiquitous call to e-mail, uses a spooled, or queued, method of mail delivery. Once a message has been sent to a destination, the message is spooled to a device - usually a disk. The server software at the destination posts a vigil, regularly checking this queue for messages. When it detects them, it proceeds to deliver them to their destination.

SMTP is used to send mail; POP3 and IMAP are used to receive mail.

SMTP was initially defined in 1982 in RFC 821, which became STD 10, and has its last update in RFC 5321, which includes the extended SMTP (ESMTP) additions.

The well-known port for SMTP is **port 25**.

5.9 simple network management protocol - SNMP

Simple Network Management Protocol (SNMP) collects and manipulates data about devices on the IP networks. It gathers data by polling the devices from a management station at fixed or random intervals, requiring them to disclose certain information. When all is well, SNMP receives something called a *baseline* - a report delimiting the operational traits of a healthy network. This protocol can also stand as a watchdog over the network, quickly notifying managers of any sudden turn of events. These network watchdogs are called *agents,* and when aberrations occur, agents send an alert called a **trap** to the management station.

An SNMP-managed network consists of three key components:

- Managed device
- Agent - software which runs on managed devices
- Network management system (NMS) - software which runs on the manager

A ***managed device*** is a network node that implements an SNMP interface that allows unidirectional (read-only) or bidirectional access to node-specific information. Managed devices exchange node-specific information with the NMSs. Sometimes called network elements, the managed devices can be any type of device, including, but not limited to, routers, access servers, switches, bridges, hubs, IP telephones, IP video cameras, computer hosts, and printers.

An ***agent*** is a network-management software module that resides on a managed device. An agent has local knowledge of management information and translates that information to or from an SNMP specific form.

A ***network management system*** (NMS) executes applications that monitor and control managed devices. NMSs provide the bulk of the processing and memory resources required for network management. One or more NMSs may exist on any managed network.

There are 2 well-known ports associated to SNMP - **port 161** for general SNMP messages and **port 162** for trap management.

There are 3 versions of the SNMP protocol – SNPMv1 through SNMPv3. The first RFCs for v1 (RFC 1065–RFC 1067) appeared in 1988. SNMPv2 (RFC 1441–RFC 1452), revises version 1 and includes improvements in the areas of performance, security, confidentiality, and manager-to-manager communications. SNMPv3 primarily added security and remote configuration enhancements to SNMP and is defined in RFCs 3411-3418, also known as **STD0062**.

5.10 the domain name system - DNS

Although programs theoretically could refer to hosts, mailboxes, and other resources by their network (e.g., IP) addresses, these addresses are hard for people to remember. Also, sending e-mail to **tana@128.111.24.41** means that if Tana's ISP or organization moves the mail server to a different machine with a different IP address, her e-mail address has to change. Consequently, ASCII names were introduced to decouple machine names from machine addresses. In this way, Tana's address might be something like **tana@art.ucsb.edu**. Nevertheless, the network itself understands only numerical addresses, so some mechanism is required to convert the ASCII strings to network addresses. In the following sections we will study how this mapping is accomplished in the Internet.

Way back in the ARPANET, there was simply a file, **hosts.txt**, that listed all the hosts and their IP addresses. Every night, all the hosts would fetch it from the site at which it was maintained. For a network of a few hundred large timesharing machines, this approach worked reasonably well.

However, when thousands of minicomputers and PCs were connected to the net, everyone realized that this approach could not continue to work forever. For one thing, the size of the file would become too large. However, even more important, host name conflicts would occur constantly unless names were centrally managed, something unthinkable in a huge international network due to the load and latency. To solve these

problems, **DNS** (the **Domain Name System**) was invented.

The essence of DNS is the invention of a hierarchical, domain-based naming scheme and a distributed database system for implementing this naming scheme. It is primarily used for mapping host names and e-mail destinations to IP addresses but can also be used for other purposes. DNS is defined in RFCs 1034 and 1035.

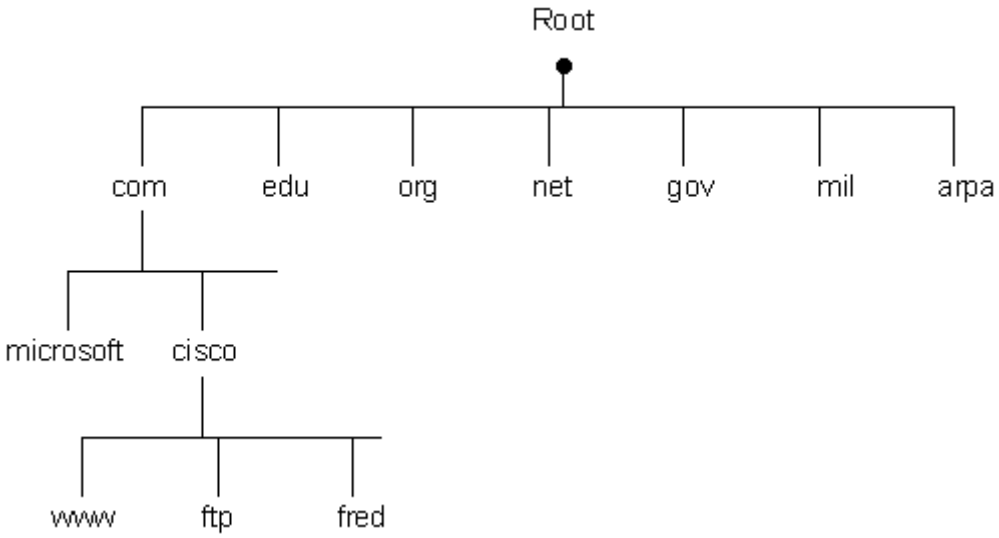
Very briefly, the way DNS is used is as follows. To map a name onto an IP address, an application program calls a library procedure called the **resolver**, passing it the name as a parameter. We already saw an example of a resolver, **gethostbyname**. The resolver sends a UDP packet to a local DNS server, which then looks up the name and returns the IP address to the resolver, which then returns it to the caller. Armed with the IP address, the program can then establish a TCP connection with the destination or send it UDP packets.

5.10.1 the DNS name space

Managing a large and constantly changing set of names is a nontrivial problem. In the postal system, name management is done by requiring letters to specify (implicitly or explicitly) the country, state or province, city, and street address of the addressee. By using this kind of hierarchical addressing, there is no confusion between the Marvin Anderson on Main St. in White Plains, N.Y. and the Marvin Anderson on Main St. in Austin, Texas. DNS works the same way. Conceptually, the Internet is divided into over 1500 top-level domains (1503, as of 03.2021), where each domain covers many hosts. Each domain is partitioned into subdomains, and these are further partitioned, and so on. All these domains can be represented by a tree, as shown in the figure below. The leaves of the tree represent domains that have no subdomains (but do contain machines, of course). A leaf domain may contain a single host, or it may represent a company and contain thousands of hosts.

The top-level domains come in two flavors: generic and countries. The original generic domains were *com (commercial), edu* (educational institutions), *gov* (the U.S. Federal Government), *int* (certain international organizations), *mil* (the U.S. armed forces), *net* (network providers), and *org* (nonprofit organizations). The country domains include one entry for every country, as defined in ISO 3166.

In November 2000, ICANN approved four new, general-purpose, top-level domains, namely, *biz* (businesses), *info* (information), *name* (people's names), and *pro* (professions, such as doctors and lawyers). In addition, three more specialized top-level domains were introduced at the request of certain industries. These are *aero* (aerospace industry), *coop* (co-operatives), and *museum* (museums). Many other top-level domains have been added ever since.



The original top level domains and some subdomains

As an aside, as the Internet becomes more commercial, it also becomes more contentious. Take *pro*, for example. It was intended for certified professionals. But who is a professional? And certified by whom? Doctors and lawyers clearly are professionals. But what about freelance photographers, piano teachers, magicians, plumbers, barbers, exterminators, tattoo artists, mercenaries, and prostitutes? Are these occupations professional and thus eligible for *pro* domains? And if so, who certifies the individual practitioners?

In general, getting a second-level domain, such as *name-of-company*.com, is easy. It merely requires going to a registrar for the corresponding top-level domain (*com* in this case) to check if the desired name is available and not somebody else's trademark. If there are no problems, the requester pays a small annual fee and gets the name. By now, virtually every common (English) word has been taken in the *com* domain. Try household articles, animals, plants, body parts, etc. Nearly all are taken.

Each domain is named by the path upward from it to the (unnamed) root. The components are separated by periods (pronounced "dot"). Thus, the engineering department at Sun Microsystems might be **eng.sun.com**., rather than a UNIX-style name such as */com/sun/eng.* Notice that this hierarchical naming means that **eng.sun.com***.* does not conflict with a potential use of *eng* in **eng.yale.edu**., which might be used by the Yale English department.

Domain names can be either absolute or relative. An absolute domain name always ends with a period (e.g., *eng.sun.com.*), whereas a relative one does not. Relative names have to be interpreted in some context to uniquely determine their true meaning. In both cases, a named domain refers to a specific node in the tree and all the nodes under it.

Domain names are **case insensitive**, so *edu*, *Edu*, and *EDU* mean the same thing. Component names can be up to 63 characters long, and full path names must not exceed 255 characters.

In principle, domains can be inserted into the tree in two different ways. For example, **cs.yale.edu** could equally well be listed under the *us* country domain as **cs.yale.ct.us**. In practice, however, most organizations in the United States are under a generic domain, and most outside the United States are under the domain of their country. There is no rule against registering under two top-level domains, but few organizations except multinationals do it (e.g., *sony.com* and *sony.nl*).

Each domain controls how it allocates the domains under it. For example, Japan has domains *ac.jp* and *co.jp* that mirror *edu* and *com*. The Netherlands does not make this distinction and puts all organizations directly under *nl*. Thus, all three of the following are university computer science departments:

1. **cs.yale.edu** (Yale University, in the United States)
2. **cs.vu.nl** (Vrije Universiteit, in The Netherlands)
3. **cs.keio.ac.jp** (Keio University, in Japan)

To create a new domain, permission is required of the domain in which it will be included. For example, if a VLSI group is started at Yale and wants to be known as **vlsi.cs.yale.edu**, it has to get permission from whoever manages **cs.yale.edu**. Similarly, if a new university is chartered, say, the University of Northern South Dakota, it must ask the manager of the *edu* domain to assign it **unsd.edu**. In this way, name conflicts are avoided and each domain can keep track of all its subdomains. Once a new domain has been created and registered, it can create subdomains, such as **cs.unsd.edu**, without getting permission from anybody higher up the tree.

Naming follows organizational boundaries, not physical networks. For example, if the computer science and electrical engineering departments are located in the same building and share the same LAN, they can nevertheless have distinct domains. Similarly, even if computer science is split over Babbage Hall and Turing Hall, the hosts in both buildings will normally belong to the same domain.

5.10.2 resource records

Every domain, whether it is a single host or a top-level domain, can have a set of **resource records**

associated with it. For a single host, the most common resource record is just its IP address, but many other kinds of resource records also exist. When a resolver gives a domain name to DNS, what it gets back are the resource records associated with that name. Thus, the primary function of DNS is to **map domain names onto resource records**.

A resource record is a five-tuple. Although they are encoded in binary for efficiency, in most expositions, resource records are presented as ASCII text, one line per resource record. The format we will use is as follows:

Domain_name Time_to_live Class Type Value

1. The **Domain_name** tells the domain to which this record applies. Normally, many records exist for each domain and each copy of the database holds information about multiple domains. This field is thus the primary search key used to satisfy queries. The order of the records in the database is not significant.
2. The **Time_to_live** field gives an indication of how stable the record is. Information that is highly stable is assigned a large value, such as 86400 (the number of seconds in 1 day). Information that is highly volatile is assigned a small value, such as 60 (1 minute). We will come back to this point later when we have discussed caching.
3. The third field of every resource record is the **Class**. For Internet information, it is always **IN**. For non-Internet information, other codes can be used, but in practice, these are rarely seen.
4. The Type field tells what kind of record this is. The most important types are listed below:

- The most important record type is the **A** (Address) record. It holds a 32-bit IP address for some host. Every Internet host must have at least one IP address so that other machines can communicate with it. Some hosts have two or more network connections, in which case they will have one type *A* resource record per network connection (and thus per IP address). DNS can be configured to cycle through these, returning the first record on the first request, the second record on the second request, and so on.
- The next most important record type is the **MX** record. It specifies the name of the host prepared to accept e-mail for the specified domain. It is used because not every machine is prepared to accept e-mail. If someone wants to send e-mail to, for example, *bill@microsoft.com*, the sending host needs to find a mail server at *microsoft.com* that is willing to accept e-mail. The *MX* record can provide this information.
- The **NS** records specify name servers. For example, every DNS database normally has an *NS* record for each of the top-level domains, so, for example, e-mail can be sent to distant parts of the naming tree. We will come back to this point later.
- **CNAME** records allow aliases to be created. For example, a person familiar with Internet naming in general and wanting to send a message to someone whose login name is *paul* in the computer science department at M.I.T. might guess that *paul@cs.mit.edu* will work. Actually, this address will not work, because the domain for M.I.T.'s computer science department is *lcs.mit.edu*. However, as a service to people who do not know this, M.I.T. could create a *CNAME* entry to point people and programs in the right direction. An entry like this one might do the job:
- Like *CNAME*, **PTR** points to another name. However, unlike *CNAME*, which is really just a macro definition, *PTR* is a regular DNS datatype whose interpretation depends on the context in which it is found. In practice, it is nearly always used to associate a name with an IP address to allow lookups of the IP address and return the name of the corresponding machine. These are called **reverse lookups**.
- **HINFO** records allow people to find out what kind of machine and operating system a domain corresponds to.
- Finally, **TXT** records allow domains to identify themselves in arbitrary ways. Both of these record types are for user convenience. Neither is required, so programs cannot count on getting them (and probably cannot deal with them if they do get them).

5. Finally, we have the **Value** field. This field can be a number, a domain name, or an ASCII string. The semantics depend on the record type.

The process of resolving a domain name into an address starts by making an inquiry into the root database to locate the domain's DNS servers. This enables an inquiry into these servers to find the address associated with the Hostname. The actual mechanism doesn't really matter for our purposes here either; it is more important to gain an understanding of the concept for now.

Taking a look at the records that are held in these DNS servers can reveal the concept of the resolving process clearly enough. The manner in which these records are held can vary from one DNS server to another, depending on the particular DNS software it is running, but the basic nature of the records is the same. To illustrate the function of the most common records, I'll provide a sample DNS record file. If you were to create an entry for a domain in a DNS server, you would need to create these records. Note, however, that many web based interfaces to domain record files automatically generate some of these records. If you are going to use such an interface, it may not provide mechanisms for you to create those records. That simply makes your life a little easier. Here is the sample domain file:

```
$ORIGIN mydomain.com.

      @ IN SOA dns1.mydomain.com. hostmaster.mydomain.com.
      ( 20030915001 ; serial 172800 ; refresh 3600 ; retry
      1728000 ; expire 172800 ; minimum )

      IN NS dns1.mydomain.com.
      IN NS dns2.mydomain.com.
      IN A 192.168.251.10
      IN MX 5 mail.mydomain.com

      localhost IN A 127.0.0.1
      mail IN A 192.168.251.5
      imapmail IN CNAME imap.mydomain2.com.

      offsite.mydomain.com. IN A 192.168.200.33

      www.mydomain.com. IN A 192.168.251.10
      www.mydomain.com. IN MX 10 webmail.mydomain2.com.

      $ORIGIN subdomain.mydomain.com.
      www IN A 192.168.251.100
```

The first few lines pertain to the domain itself, and may begin with a $ORIGIN record. The $ORIGIN simply provides a name to be appended to each name in the file that does not end with a period. For example, "mail", above, doesn't end with a period and so will be taken as "mail.mydomain.com"; "offsite.mydomain.com." does end in a period and so is taken as is (forget the period and it will be "mail.mydomain.com.mydomain.com" which is probably not what you wanted!).

Next we have a "Start Of Authority", SOA, record. There are several interesting items in this record, only a couple of which we're concerned with here. First there's the name of the machine on which this file was created, followed by the name of the responsible person in "dotted email address" form (replace the first dot with an @ sign). These are meant for documentation and don't affect the way DNS works. Next, enclosed in parentheses, are some parameters and their values. "Serial" is a serial number that is incremented each time a change is made to the file, and indicates the version level of this file to the DNS system. The rest of these parameters do some pretty interesting things, but are beyond our scope here. For now, you can use the above values - they're fairly standard.

Next we have some NS records. These state which Name Servers there are for this domain. Note that the "name" on each of these records (the first entry) is null, but since it does not end in a period it has the "mydomain.com" suffix added to it. Alternatively, we could specify "mydomain.com." (with the period). Skip the next couple of records - I'll come back to them.

Now we have an "A" address record for the name "localhost". "A" records specify the IP address for a host. This record specifies the loopback address 127.0.0.1, which is what you would expect for "localhost"! This is followed by an address record specifying that there is a machine called mail.mydomain.com at 192.168.251.5, which, for the sake of this example, is an address on our local network. To show that the address can be anywhere on the internet, the "A" record for "offsite" points to a machine in a different network. Assuming that 192.168.251 is our class C network address, "www" then points to host address 10 on our network. This is (presumably) the address to which all http transactions will be sent.

Going back to those skipped records, note that the null named address record points to host address 10. The effect of this is that if somebody uses "http://mydomain.com" omitting the "www" it will still go to the same machine.

The other record we skipped is the MX, or Mail exchanger (email server), record. This one has a null name and so pertains to mail sent to mydomain.com, as in someone@mydomain.com. There is a number right after the MX. This is the priority of this record. The lower the number, the higher the priority, so that if there are two or more records that would apply to a given name, the lowest number would be tried first. Usually, in smaller networks, there's only the one mail server anyway. A little further down in our file there's an MX record for the host "www". This one will send any mail to someone@www.mydomain.com to the machine webmail.mydomain2.com. As you can see, the email server can be on this or any other network. In these examples the mail server machines are specified by name but could be specified by number. Personally, I prefer to use the name so that if I move or renumber a server I only have to change the "A" record that specifies its address and all other references will resolve to the new number.

In the middle of our file we have a "CNAME" record. This is used to define an alias. CNAME actually stands for "Canonical Name" (yawn) but I like to think it stands for "See Name" since it defines an alias! Our CNAME defines the name "imapmail" within mydomain.com and points it at a host called "imap" in the domain "mydomain2.com". Again, the target machine could be anywhere on the Internet.

At the bottom of our sample file I have shown an example of using $ORIGIN to create a subdomain. Here there is only one host defined in the subdomain. Its full name is www.subdomain.mydomain.com and it has an address that is still within our class C network. Once again, however, the target address could be anywhere on the Internet.

"Subdomain" in this sense is an organization of names etc, as opposed to a subnet in IP addressing, which has physical implications.

To summarize the purpose of these records:

- To define a domain, use an SOA record with one or more NS records.
- To define a host (a machine or a name) use an A record.
- To send mail somewhere use an MX record.
- To define an alias use a CNAME record.
- Use $ORIGIN to define the suffix to be appended to names being defined or referenced, and to define subnets.

Remember that DNS is used to resolve a name to an address, and the address can be anywhere on the Internet.

5.11 the forgotten layers – presentation

The **Presentation layer** gets its name from its purpose: It presents data to the Application layer and is responsible for data translation and code formatting. This layer is essentially a translator and provides coding and conversion functions. A successful data-transfer technique is to adapt the data into a standard format before transmission. Computers are configured to receive this generically formatted data and then convert the data back into its native format for actual reading (for example, EBCDIC to ASCII). By providing translation services, the Presentation layer ensures that data transferred from the Application layer of one system can be read by the Application layer of another one.

The main services provided by this layer are:

- data conversion
- character code translation
- compression
- encryption and decryption
- serialization

Some Presentation layer standards are involved in multimedia operations too. The following serve to direct graphic and visual image presentation:

- **PICT** A picture format used by Macintosh programs for transferring QuickDraw graphics.
- **TIFF** Tagged Image File Format; a standard graphics format for high-resolution, bitmapped images.
- **JPEG** Photo standards brought to us by the Joint Photographic Experts Group.

Other standards guide movies and sound:

- **MIDI** Musical Instrument Digital Interface (sometimes called Musical Instrument Device Interface), used for digitized music.
- **MPEG** Increasingly popular Moving Picture Experts Group standard for the compression and coding of motion video for CDs. It provides digital storage and bit rates up to 1.5Mbps.
- **QuickTime** For use with Macintosh programs; manages audio and video applications.
- **RTF** Rich Text Format, a file format that lets you exchange text files between different word processors, even in different operating systems.

5.12 the forgotten layers – session

The **Session layer** is responsible for setting up, managing, and then tearing down sessions between Presentation layer entities. This layer also provides dialogue control between devices, or nodes. It coordinates communication between systems, and serves to organize their communication by offering three different modes: **simplex** (one-way), **half duplex** (both directions, but only one at a time), and **full duplex**. To sum up, the Session layer basically keeps different applications' data separate from other applications' data.

The following are some examples of Session layer protocols and interfaces (according to Cisco):

- **Structured Query Language (SQL) -** developed by IBM to provide users with a simpler way to define their information requirements on both local and remote systems.
- **Remote Procedure Call (RPC) -** a broad client/server redirection tool used for disparate service environments. Its procedures are created on clients and performed on servers.
- **AppleTalk Session Protocol (ASP) -** another client/server mechanism, which both establishes and

maintains sessions between AppleTalk client and server machines.

- **Digital Network Architecture Session Control Protocol (DNA SCP) -** a DECnet Session layer protocol.
- **PAP** – Printer Access Protocols

chapter 6 the transport layer, TCP and UDP

6.1 functions

The **Transport Layer** is responsible for delivering data to the appropriate application process on the host computers. This involves statistical multiplexing of data from different application processes, i.e. forming data packets, and adding source and destination port numbers in the header of each Transport Layer data packet. Together with the source and destination IP address, the port numbers constitutes a network socket, i.e., an identification address of the process-to-process communication. In the OSI model, this function is supported by the Session Layer.

Some Transport Layer protocols, for example TCP, but not UDP, support virtual circuits, i.e., provide connection oriented communication over an underlying packet oriented datagram network. A byte-stream is delivered while hiding the packet mode communication for the application processes. This involves connection establishment, dividing of the data stream into packets called segments, segment numbering and reordering of out-of order data.

Finally, some Transport Layer protocols, for example TCP, but not UDP, provide end-to-end reliable communication, i.e. error recovery by means of error detecting code and automatic repeat request (ARQ) protocol. The ARQ protocol also provides flow control, which may be combined with congestion avoidance.

UDP is a very simple protocol, and does not provide virtual circuits, nor reliable communication, delegating these functions to the application program. UDP packets are called datagrams, rather than segments.

TCP is used for many protocols, including HTTP web browsing and email transfer. UDP may be used for multicasting and broadcasting, since retransmissions are not possible to a large amount of hosts. UDP typically gives higher throughput and shorter latency, and is therefore often used for real-time multimedia communication where packet loss occasionally can be accepted, for example IP-TV and IP-telephony, and for online computer games.

In many non-IP-based networks, for example X.25, Frame Relay and ATM, the connection oriented communication is implemented at network layer or data link layer rather than the Transport Layer. In X.25, in telephone network modems and in wireless communication systems, reliable node-to-node communication is implemented at lower protocol layers.

6.2 services

There is a long list of services that can be optionally provided by the Transport Layer. None of them are compulsory, because not all applications require all available services.

- Connection-oriented: This is normally easier to deal with than connection-less models, so where the Network layer only provides a connection-less service, often a connection-oriented service is built on top of that in the Transport Layer.
- 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, so the Transport Layer provides it. The simplest way of doing this is to give each packet a number, and allow the receiver to reorder the packets.
- Reliable data: Packets may be lost in routers, switches, bridges and hosts due to network congestion, when the packet queues are filled and the network nodes have to delete packets. Packets may be lost or corrupted in Ethernet due to interference and noise, since Ethernet does not retransmit corrupted packets. Packets may be delivered in the wrong order by an underlying network. Some Transport Layer protocols, for example TCP, can fix this. By means of an error detection code, for

example a checksum, the transport protocol may check that the data is not corrupted, and verify that by sending an ACK message to the sender. Automatic repeat request schemes may be used to retransmit lost or corrupted data. By introducing segment numbering in the Transport Layer packet headers, the packets can be sorted in order. Of course, error free is impossible, but it is possible to substantially reduce the numbers of undetected errors.

- Flow control: The amount of memory on a computer is limited, and without flow control a larger computer might flood a computer with so much information that it can't hold it all before dealing with it. Nowadays, this is not a big issue, as memory is cheap while bandwidth is comparatively expensive, but in earlier times it was more important. Flow control allows the receiver to respond before it is overwhelmed. Sometimes this is already provided by the network, but where it is not, the Transport Layer may add it on.
- Congestion avoidance: Network congestion occurs when a queue buffer of a network node is full and starts to drop packets. Automatic repeat request 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.
- Byte orientation: Rather than dealing with things on a packet-by-packet basis, the Transport Layer may add the ability to view communication just as a stream of bytes. This is nicer to deal with than random packet sizes, however, it rarely matches the communication model which will normally be a sequence of messages of user defined sizes.
- Ports: (Part of the Transport Layer in the TCP/IP model, but of the Session Layer in the OSI model) Ports are essentially ways to address multiple entities in the same location. For example, the first line of a postal address is a kind of port, and distinguishes between different occupants of the same house. Computer applications will each listen for information on their own ports, which is why you can use more than one network-based application at the same time.

6.3 transport service primitives

To allow users to access the transport service, the transport layer must provide some operations to application programs, that is, a transport service interface. Each transport service has its own interface. In this section, we will first examine a simple (hypothetical) transport service and its interface to see the bare essentials. In the following section we will look at a real example.

The transport service is similar to the network service, but there are also some important differences. The main difference is that the network service is intended to model the service offered by real networks, warts and all. Real networks can lose packets, so the network service is generally unreliable.

The (connection-oriented) transport service, in contrast, is reliable. Of course, real networks are not error-free, but that is precisely the purpose of the transport layer—to provide a reliable service on top of an unreliable network.

As an example, consider two processes connected by pipes in UNIX. They assume the connection between them is perfect. They do not want to know about acknowledgments, lost packets, congestion, or anything like that. What they want is a 100 percent reliable connection. Process *A* puts data into one end of the pipe, and process *B* takes it out of the other. This is what the connection-oriented transport service is all about - hiding the imperfections of the network service so that user processes can just assume the existence of an error-free bit stream.

As an aside, the transport layer can also provide unreliable (datagram) service. However, there is relatively little to say about that, so we will mainly concentrate on the connection-oriented transport service in this chapter. Nevertheless, there are some applications, such as client-server computing and streaming multimedia, which benefit from connectionless transport, so we will say a little bit about it later on.

A second difference between the network service and transport service is whom the services are intended for. The network service is used only by the transport entities. Few users write their own transport entities, and thus few users or programs ever see the bare network service. In contrast, many programs (and thus programmers) see the transport primitives. Consequently, the transport service must be convenient and

easy to use.

To get an idea of what a transport service might be like, consider the five primitives listed below. This transport interface is truly bare bones, but it gives the essential flavor of what a connection-oriented transport interface has to do. It allows application programs to establish, use, and then release connections, which is sufficient for many applications.

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

The primitives for a simple transport service

To see how these primitives might be used, consider an application with a server and a number of remote clients. To start with, the server executes a LISTEN primitive, typically by calling a library procedure that makes a system call to block the server until a client turns up. When a client wants to talk to the server, it executes a CONNECT primitive. The transport entity carries out this primitive by blocking the caller and sending a packet to the server. Encapsulated in the payload of this packet is a transport layer message for the server's transport entity.

A quick note on terminology is now in order. For lack of a better term, we will reluctantly use the somewhat ungainly acronym TPDU (**Transport Protocol Data Unit**) for messages sent from transport entity to transport entity. Thus, TPDUs (exchanged by the transport layer) are contained in packets (exchanged by the network layer). In turn, packets are contained in frames (exchanged by the data link layer). When a frame arrives, the data link layer processes the frame header and passes the contents of the frame payload field up to the network entity. The network entity processes the packet header and passes the contents of the packet payload up to the transport entity.

Getting back to our client-server example, the client's CONNECT call causes a CONNECTION REQUEST TPDU to be sent to the server. When it arrives, the transport entity checks to see that the server is blocked on a LISTEN (i.e., is interested in handling requests). It then unblocks the server and sends a CONNECTION ACCEPTED TPDU back to the client. When this TPDU arrives, the client is unblocked and the connection is established.

Data can now be exchanged using the SEND and RECEIVE primitives. In the simplest form, either party can do a (blocking) RECEIVE to wait for the other party to do a SEND. When the TPDU arrives, the receiver is unblocked. It can then process the TPDU and send a reply. As long as both sides can keep track of whose turn it is to send, this scheme works fine.

Note that at the transport layer, even a simple unidirectional data exchange is more complicated than at the network layer. Every data packet sent will also be acknowledged (eventually). The packets bearing control TPDUs are also acknowledged, implicitly or explicitly. These acknowledgments are managed by the transport entities, using the network layer protocol, and are not visible to the transport users. Similarly, the transport entities will need to worry about timers and retransmissions. None of this machinery is visible to the transport users. To the transport users, a connection is a reliable bit pipe: one user stuffs bits in and they magically appear at the other end. This ability to hide complexity is the reason that layered protocols are such a powerful tool.

When a connection is no longer needed, it must be released to free up table space within the two transport entities. Disconnection has two variants: asymmetric and symmetric. In the asymmetric variant, either transport user can issue a DISCONNECT primitive, which results in a DISCONNECT TPDU being sent

to the remote transport entity. Upon arrival, the connection is released.

In the symmetric variant, each direction is closed separately, independently of the other one. When one side does a DISCONNECT, that means it has no more data to send but it is still willing to accept data from its partner. In this model, a connection is released when both sides have done a DISCONNECT.

6.4 classes of transport protocols in the OSI model

The OSI model defines five classes of transport protocols ranging from Transport Protocol Class 0 through Transport Protocol Class 4 (TP0, TP1, TP2, TP3 & TP4). The protocols increase in complexity from 0-4. TP0-3 work only with connection-oriented communications, in which a session connection must be established before any data is sent; TP4 also works with both connection-oriented and connectionless communications.

Transport Protocol Class 0 (TP0) performs **segmentation** (fragmentation) and **reassembly** functions. TP0 discerns the size of the smallest maximum protocol data unit (PDU) supported by any of the underlying networks, and segments the packets accordingly. The packet segments are reassembled at the receiver.

Transport Protocol Class 1 (TP1) performs segmentation (fragmentation) and reassembly, plus **error recovery**. TP1 sequences protocol data units (PDUs) and will retransmit PDUs or reinitiate the connection if an excessive number of PDUs are unacknowledged.

Transport Protocol Class 2 (TP2) performs segmentation and reassembly, as well as **multiplexing** and **demultiplexing** of data streams over a single virtual circuit. **Multiplexing** is a method by which multiple analog message signals or digital data streams are combined into one signal over a shared medium. The multiplexed signal is transmitted over a communication channel, which may be a physical transmission medium. The multiplexing divides the capacity of the low-level communication channel into several higher-level logical channels, one for each message signal or data stream to be transferred. A reverse process, known as demultiplexing, can extract the original channels on the receiver side.

Transport Protocol Class 3 (TP3) offers error recovery, segmentation and reassembly, and multiplexing and demultiplexing of data streams over a single virtual circuit. TP3 also sequences PDUs and retransmits them or reinitiates the connection if an excessive number are unacknowledged.

Transport Protocol Class 4 (TP4) offers error recovery, performs segmentation and reassembly, and supplies multiplexing and demultiplexing of data streams over a single virtual circuit. TP4 sequences PDUs and retransmits them or reinitiates the connection if an excessive number are unacknowledged. TP4 provides reliable transport service and functions with either connection-oriented or connectionless network service. TP4 is the most commonly used of all the OSI transport protocols, which is similar to the Transmission Control Protocol (TCP) in the TCP/IP suite.

Both TP4 and TCP are built to provide a reliable connection oriented end-to-end transport service on top of an unreliable network service. The network service may loose packets, store them, deliver them in the wrong order or even duplicate packets. Both protocols have to be able to deal with the most severe problems e.g. a subnetwork stores valid packets and sends them at a later date. TP4 and TCP have a connect, transfer and a disconnect phase. The principles of doing this are also quite similar.

One difference between TP4 and TCP to be mentioned is that TP4 uses ten different TPDU (Transport Protocol Data Unit) types whereas TCP knows only one. This makes TCP simpler but every TCP header has to have all possible fields and therefore the TCP header is at least 20 bytes long whereas the TP4 header takes at least 5 bytes. Another difference is the way both protocols react in case of a call collision. TP4 opens two bidirectional connections between the TSAPs (Transport Service Access Points) whereas TCP opens just one connection. TP4 uses a different flow control mechanism for its messages, it also provides means for quality of service measurement.

6.5 comparison of OSI transport protocols

The OSI model defines five classes of connection-mode transport protocols designated class 0 (TP0) to class 4 (TP4). Class 0 contains no error recovery, and was designed for use on network layers that provide error-free connections. Class 4 is closest to TCP, although TCP contains functions, such as the graceful close, which OSI assigns to the Session Layer. All OSI connection-mode protocol classes provide expedited data and preservation of record boundaries. Detailed characteristics of the classes are shown in the following table:

Feature Name	***TP0***	***TP1***	***TP2***	***TP3***	***TP4***
Connection oriented network	Yes	Yes	Yes	Yes	Yes
Connectionless network	No	No	No	No	Yes
Concatenation and separation	No	Yes	Yes	Yes	Yes
Segmentation and reassembly	Yes	Yes	Yes	Yes	Yes
Error Recovery	No	Yes	No	Yes	Yes
Reinitiate connection (excessive unacknowledgements)	No	Yes	No	Yes	No
multiplexing and demultiplexing over a single virtual circuit	No	No	Yes	Yes	Yes
Explicit flow control	No	No	Yes	Yes	Yes
Retransmission on timeout	No	No	No	No	Yes
Reliable Transport Service	No	Yes	No	Yes	Yes

6.6 the other protocols in the TCP/IP model

Besides the well known TCP and UDP protocols, which will be detailed within this chapter, it is worth mentioning a few other protocols operating at this layer.

- **DCCP - Datagram Congestion Control Protocol** – a message oriented protocol, implementing reliable connection setup, teardown, ECN (Explicit Congestion Notification), congestion control, and feature negotiation. DCCP was published as RFC 4340, a proposed standard, by the IETF in March, 2006. RFC 4336 provides an introduction. Linux had an implementation of DCCP first released in Linux kernel version 2.6.14 (released October 28, 2005).
- **SCTP - Stream Control Transmission Protocol** – ensures reliable, in-sequence transport of messages with congestion control. The protocol was defined by the IETF Signaling Transport (SIGTRAN) working group in 2000, and is maintained by the IETF Transport Area (TSVWG) working group. RFC 4960 defines the protocol. RFC 3286 provides an introduction.

6.7 comparison of TCP/IP transport protocols

	UDP	TCP	DCCP	SCTP
Packet header size	8 Bytes	20-60 Bytes	12 or 16 Bytes	12 Bytes + Variable Chunk Header
Transport Layer packet entity	Datagram	Segment	Datagram	Datagram
Port numbering	Yes	Yes	Yes	Yes
Error detection	Optional	Yes	Yes	Yes
Reliability: Error recovery by automatic repeat request (ARQ)	No	Yes	No	Yes
Virtual circuits: Sequence numbering and reordering	No	Yes	Yes	Optional
Flow control	No	Yes	No	Yes
Congestion avoidance: Variable congestion window, slow start, time outs	No	Yes	Yes	Yes
Multiple streams	No	No	No	Yes
ECN support	No	Yes	Yes	Yes
NAT friendly	Yes	Yes	Yes	No

6.8 the UDP protocol

The **User Datagram Protocol** offers only a minimal transport service - non-guaranteed datagram delivery - and gives applications direct access to the datagram service of the IP layer. UDP is used by applications that do not require the level of service of TCP or that wish to use communications services (e.g., multicast or broadcast delivery) not available from TCP.

UDP is almost a null protocol; the only services it provides over IP are checksumming of data and multiplexing by port number. Therefore, an application program running over UDP must deal directly with end-to-end communication problems that a connection-oriented protocol would have handled -- e.g., retransmission for reliable delivery, packetization and reassembly, flow control, congestion avoidance, etc., when these are required. The fairly complex coupling between IP and TCP will be mirrored in the coupling between UDP and many applications using UDP.

UDP was defined in RFC 768, Aug. 1980. RFC 4113 deals with MIB (Management Information Base) for UDP, while RFC 8085 specifies UDP Usage Guidelines.

6.9 the UDP PDU

The data part of the PDU comes from the upper layer. The header part is detailed below.

00-03	04-07	08-11	12-15	16-19	20-23	24-27	28-31
Source port				Destination port			

Length	Checksum
Data	

- **Source port** - 16 bits
- **Destination port** - 16 bits
- **Length** – 16 bits - The length in bytes of the UDP header and the encapsulated data. The minimum value for this field is 8.
- **Checksum** – 16 bits - Computed as the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded as needed with zero bytes at the end to make a multiple of two bytes. If the checksum is cleared to zero, then checksuming is disabled. If the computed checksum is zero, then this field must be set to 0xFFFF.
- **Data** – variable length

6.10 the TCP protocol

The beginnings of TCP date back to 1974, when Vint Cerf and Bob Kahn published an article called "*A Protocol for Packet Network Interconnection",* in which they described an internetworking protocol based on packet-switching among the nodes of the network. At its core, it had a **Transmission Control Program** that incorporated both connection-oriented links and datagram services between hosts. The monolithic Transmission Control Program was later split into two components consisting of the *Transmission Control Protocol* at the connection-oriented layer and the *Internet Protocol* at the internetworking (datagram) layer. The model became later known informally as *TCP/IP*, although formally it was henceforth called the *Internet Protocol Suite*.

The first formal specification of TCP was in RFC 675, Dec. 1974. It was followed by RFC 793, from Sept. 1981, which ended up as STD 7.

Other TCP related RFCs are 1122, 1323, 1379, 1948, 2018.

A later RFC (5681) deals with the Flow Control capability of TCP.

Newer RFCs, like RFC 6824 and 7323 deal with TCP Extensions for Multipath Operation with Multiple Addresses and TCP Extensions for High Performance, respectively.

A dedicated RFC (7414) summarizes the documents related to TCP, it is informational only.

TCP provides a communication service at an intermediate level between an application program and the Internet Protocol (IP). That is, when an application program desires to send a large chunk of data across the Internet using IP, instead of breaking the data into IP-sized pieces and issuing a series of IP requests, the software can issue a single request to TCP and let TCP handle the IP details.

IP works by exchanging pieces of information called packets. A packet is a sequence of bytes and consists of a **header** followed by a **body**. The header describes the packet's destination and, optionally, the routers to use for forwarding until it arrives at its final destination. The body contains the data which IP is transmitting.

Due to network congestion, traffic load balancing, or other unpredictable network behavior, IP packets can be lost or delivered out of order. TCP detects these problems, requests retransmission of lost packets, rearranges out-of-order packets, and even helps minimize network congestion to reduce the occurrence of the other problems. Once the TCP receiver has finally reassembled a perfect copy of the data originally transmitted, it passes that datagram to the application program. Thus, TCP abstracts the application's communication from the underlying networking details.

6.11 the TCP PDU

The data part of the PDU comes from the upper layer. The header part is detailed below.

00-03	04-05	06-08	10-15	16-19	20-23	24-27	28-31
Source port				Destination port			
Sequence number							
Acknowledgment number							
Data offset	reserved	ECN	Control bits	Window			
Checksum				Urgent pointer			
Options and padding							
Data							

- **Source port** - 16 bits
- **Destination port** - 16 bits
- **Sequence number** - 32 bits - The sequence number of the first data byte in this segment. If the SYN bit is set, the sequence number is the initial sequence number and the first data byte is initial sequence number + 1.
- **Acknowledgment number** - 32 bits - If the ACK bit is set, this field contains the value of the next sequence number the sender of the segment is expecting to receive. Once a connection is established this is always sent.
- **Data offset** – 4 bits - The number of 32-bit words in the TCP header. This indicates where the data begins. The length of the TCP header is always a multiple of 32 bits.
- **Reserved** – 3 bits – must be cleared to zero.
- **ECN** – Explicit Congestion Notification – 3 bits – NCE - Added in RFC 3168
    1. N – NS - Nonce sum - intended to protect against accidental or malicious concealment of marked packets from the TCP sender
    2. C – CWR - Congestion Window Reduced, used by the data sending part to inform the receiving part that the congestion window has been reduced
    3. E – ECE - ECN echo
- **Control bits** – 6 bits – UAPRSF
    1. U – URG - Urgent pointer valid flag
    2. A – ACK - Acknowledgment number valid flag
    3. P – PSH - Push flag
    4. R – RST - Reset connection flag
    5. S – SYN - Synchronize sequence numbers flag
    6. F – FIN - End of data flag
- **Window** - 16 bits, unsigned - The number of data bytes beginning with the one indicated in the acknowledgment field which the sender of this segment is willing to accept. Windows are used to control the amount of outstanding, unacknowledged data segments.
- **Checksum** - 16 bits - This is computed as the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the TCP header, and the data, padded as

needed with zero bytes at the end to make a multiple of two bytes.

The sending and receiving TCP entities exchange data in the form of **segments**. A TCP segment consists of a fixed 20-byte header (plus an optional part) followed by zero or more data bytes. The TCP software decides how big segments should be. It can accumulate data from several writes into one segment or can split data from one write over multiple segments. Two limits restrict the segment size. First, each segment, including the TCP header, must fit in the 65,515-byte IP payload. Second, each network has a **maximum transfer unit**, or MTU, and each segment must fit in the MTU. In practice, the MTU is generally 1500 bytes (the Ethernet payload size) and thus defines the upper bound on segment size.

chapter 7 the network layer – the IP protocol

7.1 functions

The Network Layer is responsible for end-to-end (source to destination) packet delivery including routing through intermediate hosts, whereas the Data Link Layer is responsible for node-to-node (hop-to-hop) frame delivery on the same link.

The Network Layer provides the functional and procedural means of transferring variable length data sequences from a source to a destination host via one or more networks while maintaining the quality of service and error control functions.

Functions of the Network Layer include:

- **Connection model**: connection-oriented and connectionless communication

  For example, snail mail is connectionless, in that a letter can travel from a sender to a recipient without the recipient having to do anything. On the other hand, the telephone system is connection-oriented, because the other party is required to pick up the phone before communication can be established. The OSI Network Layer protocol can be either connection-oriented, or connectionless. In contrast, the TCP/IP Internet Layer supports only the connectionless Internet Protocol (IP); but connection-oriented protocols exist higher at other layers of that model.

- **Host addressing**

  Every host in the network needs to have a unique address which determines where it is. This address will normally be assigned from a hierarchical system, so you can be "Fred Murphy" to people in your house, "Fred Murphy, Main Street 1" to Dubliners, or "Fred Murphy, Main Street 1, Dublin" to people in Ireland, or "Fred Murphy, Main Street 1, Dublin, Ireland" to people anywhere in the world. On the Internet, addresses are known as Internet Protocol (IP) addresses.

- **Message forwarding**

  Since many networks are partitioned into subnetworks and connect to other networks for wide-area communications, networks use specialized hosts, called gateways or routers to forward packets between networks. This is also of interest to mobile applications, where a user may move from one location to another, and it must be arranged that his messages follow him. Version 4 of the Internet Protocol (IPv4) was not designed with this feature in mind, although mobility extensions exist. IPv6 has a better designed solution.

7.2 the IP protocol

The **Internet Protocol** (**IP**) is a protocol used for communicating data across a packet-switched internetwork using the Internet Protocol Suite, also referred to as TCP/IP.

IP is the primary protocol in the Internet Layer of the Internet Protocol Suite and has the task of delivering distinguished protocol datagrams (packets) from the source host to the destination host solely based on their addresses. For this purpose the Internet Protocol defines addressing methods and structures for datagram encapsulation. The first major version of addressing structure, now referred to as Internet Protocol Version 4 (IPv4) is still the dominant protocol of the Internet, although the successor, Internet Protocol Version 6 (IPv6) is being deployed actively worldwide, although the average penetration level is a little above 5% as of March 2016.

Because of the abstraction provided by encapsulation, IP can be used over a heterogeneous network, i.e., a network connecting computers may consist of a combination of Ethernet, ATM, FDDI, Wi-Fi, token ring, or others. Each link layer implementation may have its own method of addressing (or possibly the complete lack of it), with a corresponding need to resolve IP addresses to data link addresses. This address resolution is handled by the Address Resolution Protocol (ARP) for IPv4 and Neighbor Discovery Protocol (NDP) for IPv6.

7.2.1 reliability

The design principles of the Internet protocols assume that the network infrastructure is inherently unreliable at any single network element or transmission medium and that it is dynamic in terms of availability of links and nodes. No central monitoring or performance measurement facility exists that tracks or maintains the state of the network. For the benefit of reducing network complexity, the intelligence in the network is purposely mostly located in the end nodes of each data transmission, cf. end-to-end principle. Routers in the transmission path simply forward packets to next known local gateway matching the routing prefix for the destination address.

As a consequence of this design, the Internet Protocol only provides best effort delivery and its service can also be characterized as **unreliable**. In network architectural language it is a **connection-less** protocol, in contrast to so-called connection-oriented modes of transmission. The lack of reliability allows any of the following fault events to occur:

- data corruption
- lost data packets
- duplicate arrival
- out-of-order packet delivery; meaning, if packet 'A' is sent before packet 'B', packet 'B' may arrive before packet 'A'. Since routing is dynamic and there is no memory in the network about the path of prior packets, it is possible that the first packet sent takes a longer path to its destination.

The only assistance that the Internet Protocol provides in Version 4 (IPv4) is to ensure that the IP packet header is error-free through computation of a checksum at the routing nodes. This has the side-effect of discarding packets with bad headers on the spot. In this case no notification is required to be sent to either end node, although a facility exists in the Internet Control Message Protocol (ICMP) to do so.

IPv6, on the other hand, has abandoned the use of IP header checksums for the benefit of rapid forwarding through routing elements in the network.

The resolution or correction of any of these reliability issues is the responsibility of an upper layer protocol. For example, to ensure in-order delivery the upper layer may have to cache data until it can be passed to the application.

In addition to issues of reliability, this dynamic nature and the diversity of the Internet and its components provide no guarantee that any particular path is actually capable of, or suitable for performing the data transmission requested, even if the path is available and reliable. One of the technical constraints is the size of data packets allowed on a given link. An application must assure that it uses proper transmission characteristics. Some of this responsibility lies also in the upper layer protocols between application and IP. Facilities exist to examine the maximum transmission unit (MTU) size of the local link, as well as for the entire projected path to the destination when using IPv6. The IPv4 internetworking layer has the capability to automatically fragment the original datagram into smaller units for transmission. In this case, IP does provide re-ordering of fragments delivered out-of-order.

Transmission Control Protocol (TCP) is an example of a protocol that will adjust its segment size to be smaller than the MTU. User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) disregard MTU size thereby forcing IP to fragment oversized datagrams.

7.2.2 IP addressing and routing

Perhaps the most complex aspects of IP are IP addressing and routing. Addressing refers to how end hosts become assigned IP addresses and how subnetworks of IP host addresses are divided and grouped together. IP routing is performed by all hosts, but most importantly by internetwork routers, which typically use either interior gateway protocols (IGPs) or external gateway protocols (EGPs) to help make IP datagram forwarding decisions across IP connected networks

7.3 version history

In May 1974, the Institute of Electrical and Electronic Engineers (IEEE) published a paper entitled "A Protocol for Packet Network Interconnection". The paper's authors, Vint Cerf and Bob Kahn, described an internetworking protocol for sharing resources using packet-switching among the nodes. A central control component of this model was the "Transmission Control Program" (TCP) that incorporated both connection-oriented links and datagram services between hosts. The monolithic Transmission Control Program was later divided into a modular architecture consisting of the Transmission Control Protocol at the connection-oriented layer and the Internet Protocol at the internetworking (datagram) layer. The model became known informally as TCP/IP, although formally it was henceforth referenced as the Internet Protocol Suite.

The Internet Protocol is one of the determining elements that define the Internet. The dominant internetworking protocol (Internet Layer) in use today is IPv4; with number 4 assigned as the formal protocol version number carried in every IP datagram. IPv4 is described in RFC 791 (1981).

The successor to IPv4 is IPv6. Its most prominent modification from Version 4 is the addressing system. IPv4 uses 32-bit addresses (c. 4 billion, or 4.3×10^{9}, addresses) while IPv6 uses 128-bit addresses (c. 340 undecillion, or 3.4×10^{38} addresses). Although adoption of IPv6 has been slow, as of June 2008, all United States government systems have demonstrated basic infrastructure support for IPv6 (if only at the backbone level).

Version numbers 0 through 3 were development versions of IPv4 used between 1977 and 1979. Version number 5 was used by the Internet Stream Protocol (IST), an experimental stream protocol. Version numbers 6 through 9 were proposed for various protocol models designed to replace IPv4: SIPP (Simple Internet Protocol Plus, known now as IPv6), TP/IX (RFC 1475), PIP (RFC 1621) and TUBA (TCP and UDP with Bigger Addresses, RFC 1347). Version number 6 was eventually chosen as the official assignment for the successor Internet protocol, subsequently standardized as IPv6.

7.4 IP version 4

Internet Protocol version 4 (**IPv4**) is the fourth revision in the development of the Internet Ptotocol (IP) and it is the first version of the protocol to be widely deployed. Together with version 6 (IPv6), it is at the core of standards-based internetworking methods of the Internet. IPv4 is still by far the most widely deployed Internet (Network) layer protocol, as IPv6 is still in its infancy of deployment.

IPv4 is described in IETF publication RFC 791 (September 1981), replacing an earlier definition (RFC 760, January 1980).

IPv4 is a connectionless protocol for use on packet-switched Link layer networks (e.g., Ethernet). It operates on a best effort delivery model, in that it does not guarantee delivery, nor does it assure proper sequencing, or avoid duplicate delivery. These aspects, including data integrity, are addressed by an upper layer transport protocol (e.g., Transmission Control Protocol).

7.5 addressing

IPv4 uses 32-bit (four-byte) addresses, which limits the address space to 4,294,967,296 (2^{32}) possible unique addresses. However, some are reserved for special purposes such as private networks (~18 million addresses) or multicast addresses (~270 million addresses). This reduces the number of addresses that can potentially be allocated for routing on the public Internet. As addresses are being incrementally delegated to end users, an IPv4 address shortage has been developing, however network addressing architecture redesign via classful network design, Classless Inter-Domain Routing, and network address translation (NAT) has significantly delayed the inevitable exhaustion.

This limitation has stimulated the development of IPv6, which is currently in the early stages of deployment, and is the only long-term solution.

7.5.1 address representations

IPv4 addresses are usually written in dot-decimal notation, which consists of the four octets of the address expressed in decimal and separated by periods. This is the base format used in the conversion in the following table:

Notation	*Value*	*Conversion from dot-decimal*
Dot-decimal notation	`192.0.2.235`	N/A
Dotted Hexadecimal	`0xC0.0x00.0x02.0xEB`	Each octet is converted to hexadecimal
Dotted Octal	`0300.0000.0002.0353`	Each octet is converted into octal
Hexadecimal	`0xC00002EB`	Concatenation from the dotted hexadecimal
Decimal	`3221226219`	The 32-bit number expressed in decimal
Octal	`030000001353`	The 32-bit number expressed in octal

Most of these formats should work in all browsers. Additionally, in dotted format, each octet can be of any of the different bases. For example, `192.0x00.0002.235` is a valid (though unconventional) equivalent to the above addresses.

A final form is not really a notation since it is rarely written in an ASCII string notation. That form is a binary form of the hexadecimal notation in binary. This difference is merely the representational difference between the string "0xCF8E83EB" and the 32-bit integer value 0xCF8E83EB. This form is used for assigning the source and destination fields in a software program.

7.5.2 allocation

Originally, an IP address was divided into two parts, the network identifier represented in the most significant (highest order) octet of the address and the host identifier using the rest of the address. The latter was therefore also called the **rest field**. This enabled the creation of a maximum of 256 networks. Quickly this was found to be inadequate.

To overcome this limit, the high order octet of the addresses was redefined to create a set of *classes* of networks, in a system which later became known as classful networking. The system defined five classes, Class A, B, C, D, and E. The Classes A, B, and C had different bit lengths for the new network identification.

The rest of an address was used as previously to identify a host within a network, which meant that each network class had a different capacity to address hosts. Class D was for allocated for multicast addressing and Class E was reserved for future applications.

Around 1985, **subnets** were introduced to allow classful network to be subdivided

Around 1987, Variable Length Subnet Mask (VLSM) was introduced. VLSM is used to implement subnets of different sizes.

Around 1993, Classless Inter-Domain Routing was introduced. **CIDR** is used to implement supernetting. Supernetting allow route aggregation. CIDR introduced prefix notation which is also known as CIDR notation. Prefix/CIDR notation is now used in the three cases of classless IP addressing: subnetting, VLSM/subnets of different sizes, CIDR/supernetting.

The original system of IP adresses classes was replaced with (CIDR), and the class-based scheme was dubbed **classful**, by contrast. CIDR's primary advantage is to allow repartitioning of any address space so that smaller or larger blocks of addresses may be allocated to users.

The hierarchical structure created by CIDR and overseen by the Internet Assigned Numbers Authority (IANA) and its Regional Internet Registries (RIRs), manages the assignment of Internet addresses worldwide. Each RIR maintains a publicly-searchable WHOIS database that provides information about IP address assignments; information from these databases plays a central role in numerous tools that attempt to locate IP addresses geographically.

7.5.3 special-use addresses

Reserved address blocks

CIDR address block	*Description*	*Reference*
0.0.0.0/8	Current network (only valid as source address)	RFC 1700
10.0.0.0/8	Private network	RFC 1918
127.0.0.0/8	Loopback	RFC 5735
169.254.0.0/16	Link-Local	RFC 3927
172.16.0.0/12	Private network	RFC 1918
192.0.0.0/24	Reserved (IANA)	RFC 5735
192.0.2.0/24	TEST-NET-1, Documentation and example code	RFC 5735
192.88.99.0/24	IPv6 to IPv4 relay	RFC 3068
192.168.0.0/16	Private network	RFC 1918
198.18.0.0/15	Network benchmark tests	RFC 2544
198.51.100.0/24	TEST-NET-2, Documentation and examples	RFC 5737
203.0.113.0/24	TEST-NET-3, Documentation and examples	RFC 5737
224.0.0.0/4	Multicasts (former Class D network)	RFC 3171
240.0.0.0/4	Reserved (former Class E network)	RFC 1700
255.255.255.255	Broadcast	RFC 919

7.5.4 (virtual) private networks

Of the approximately four billion addresses allowed in IPv4, three ranges of address are reserved for private networking use. These ranges are not routable outside of private networks and private machines cannot directly communicate with public networks. They can, however, do so through network address translation.

The following are the three ranges reserved for private networks (RFC 1918):

Name	*Address range*	*Number of addresses*	*Classful description*	*Largest CIDR block*
24-bit block	10.0.0.0–10.255.255.255	16,777,216	Single Class A	10.0.0.0/8
20-bit block	172.16.0.0–172.31.255.255	1,048,576	Contiguous range of 16 Class B blocks	172.16.0.0/12
16-bit block	192.168.0.0–192.168.255.255	65,536	Contiguous range of 256 Class C blocks	192.168.0.0/16

Packets addressed with private addresses are deliberately ignored by all public routers. Therefore, it is not possible to communicate between two private networks (e.g., two branch offices) via the public Internet without special facilities. This is accomplished with virtual private networks (VPNs).

VPNs establish tunneling connections across the public network such that the endpoints of the tunnel function as routers for private network packets. These routers encapsulate or package the privately addressed packets with headers in the routable public network so that they can be delivered to the opposing router, at the other end of the tunnel, via the public network and stripped of their public addressing headers and delivered locally to the destination.

Optionally, the encapsulated packet can be encrypted to secure the data while it travels over the public network.

7.5.5 link-local addressing

RFC 5735 defines an address block, 169.254.0.0/16, for the special use in link-local addressing. These addresses are only valid on the link, such as a local network segment or point-to-point connection, that a host is connected to. These addresses are not routable and like private addresses cannot be the source or destination of packets traversing the Internet. Link-local addresses are primarily used for address autoconfiguration (Zeroconf) when a host cannot obtain an IP address from a DHCP server or other internal configuration methods.

When the address block was reserved, no standards existed for mechanisms of address autoconfiguration. Filling the void, Microsoft created an implementation called Automatic Private IP Addressing (APIPA). Due to Microsoft's market power, APIPA has been deployed on millions of machines and has, thus, become a de facto standard in the industry. Many years later, the IETF defined a formal standard for this functionality, RFC 3927, entitled **Dynamic Configuration of IPv4 Link-Local Addresses**.

7.5.6 localhost

The address range 127.0.0.0–127.255.255.255 (127.0.0.0/8 in CIDR notation) is reserved for localhost communication. Addresses within this range should never appear outside a host computer and packets sent to this address are returned as incoming packets on the same virtual network device (known as loopback).

7.5.7 addresses ending in 0 or 255

It is a common misunderstanding that addresses ending with an octet of 0 or 255 can never be assigned to hosts. This is only true of networks with subnet masks of at least 24 bits — Class C networks in the old

classful addressing scheme, or in CIDR, networks with masks of */24* to */32* (or 255.255.255.0–255.255.255.255).

In classful addressing (now obsolete with the advent of CIDR), there are only three possible subnet masks: Class A, 255.0.0.0 or /8; Class B, 255.255.0.0 or /16; and Class C, 255.255.255.0 or /24. For example, in the subnet 192.168.5.0/255.255.255.0 (or 192.168.5.0/24) the identifier 192.168.5.0 refers to the entire subnet, so it cannot also refer to an individual device in that subnet.

A broadcast address is an address that allows information to be sent to all machines on a given subnet, rather than a specific machine. Generally, the broadcast address is found by obtaining the bit complement of the subnet mask and performing a bitwise OR operation with the network identifier. In other words, the broadcast address is the last address in the range belonging to the subnet. In our example, the broadcast address would be 192.168.5.255, so to avoid confusion this address also cannot be assigned to a host. On a Class A, B, or C subnet, the broadcast address always ends in 255.

However, this does not mean that every addresses ending in 255 cannot be used as a host address. For example, in the case of a Class B subnet 192.168.0.0/255.255.0.0 (or 192.168.0.0/16), equivalent to the address range 192.168.0.0–192.168.255.255, the broadcast address is 192.168.255.255. However, one can assign 192.168.1.255, 192.168.2.255, etc. (though this can cause confusion). Also, 192.168.0.0 is the network identifier and so cannot be assigned, but 192.168.1.0, 192.168.2.0, etc. can be assigned (though this can also cause confusion).

With the advent of CIDR, broadcast addresses do not necessarily end with 255.

In general, the first and last addresses in a subnet are used as the network identifier and broadcast address, respectively. All other addresses in the subnet can be assigned to hosts on that subnet.

7.5.8 address resolution

Hosts on the Internet are usually known not by IP addresses, but by names (e.g., www.wikipedia.org, www.whitehouse.gov, www.freebsd.org, www.berkeley.edu). The routing of IP packets across the Internet is not directed by such names, but by the numeric IP addresses assigned to such domain names. This requires translating (or resolving) domain names to addresses.

The Domain Name System (DNS) provides such a system for converting names to addresses and addresses to names. Much like CIDR addressing, the DNS naming is also hierarchical and allows for subdelegation of name spaces to other DNS servers.

The domain name system is often described in analogy to the telephone system directory information systems in which subscriber names are translated to telephone numbers.

7.5.9 address space exhaustion

Since the 1980s it has been apparent that the number of available IPv4 addresses is being exhausted at a rate that was not initially anticipated in the design of the network. This was the driving factor for the introduction of classful networks, for the creation of CIDR addressing, and finally for the redesign of the Internet Protocol, based on a larger address format (IPv6).

Today, there are several driving forces for the acceleration of IPv4 address exhaustion:

- Mobile devices — laptop computers, PDAs, mobile phones
- Always-on devices — ADSL modems, cable modems
- Rapidly growing number of Internet users

The accepted and standardized solution is the migration to IPv6. The address size jumps dramatically from 32 bits to 128 bits, providing a vastly increased address space that allows improved route aggregation across the Internet and offers large subnet allocations of a minimum of 2^{64} host addresses to end-users. Migration to IPv6 is in progress but is expected to take considerable time.

Methods to mitigate the IPv4 address exhaustion are:

- Network address translation (NAT)
- Use of private networks
- Dynamic Host Configuration Protocol (DHCP)
- Name-based virtual hosting
- Tighter control by Regional Internet Registries on the allocation of addresses to Local Internet Registries
- Network renumbering to reclaim large blocks of address space allocated in the early days of the Internet

As of April 2008, predictions of exhaustion date of the unallocated IANA pool seem to converge to between February 2010 and May 2011.

7.5.10 network address translation

The rapid pace of allocation of the IPv4 addresses and the resulting shortage of address space since the early 1990s led to several methods of more efficient use. One method was the introduction of network address translation (NAT). NAT devices masquerade an entire, private network 'behind' a single public IP address, permitting the use of private addresses within the private network. Most mass-market consumer Internet access providers rely on this technique.

7.6 IPv4 PDU

The data part of the PDU comes from the upper layer. The header part is detailed below.

00-03	04-07	08-15	16-18	19-31
Version	IHL	Differentiated sevices	Total length	
Identification			flags	Fragment offset
TTL		Protocol	header checksum	
Source IP address				
Destination IP address				
Options and padding				
Data				

- **Version** - 4 bits - possible values:
  1. 4 - IP version 4
  2. 5 - ST - ST datagram mode
  3. 6 - IP version 6
  4. 7 - TP/IX - the next internet
  5. 8 - PIP - the P internet protocol
- **IHL** – Internet Header Length - 4 bits - Specifies the length of the IP packet header in 32 bit words
- **Differentiated services** - 8 bits - Specifies the parameters for the type of service requested; defined in RFCs 2474 and 2597

- **Total length** - 16 bits - contains the length of the datagram
- **Identification** – 16 bits - Used to identify the fragments of one datagram from those of another. The originating protocol module of an internet datagram sets the identification field to a value that must be unique for that source-destination pair and protocol for the time the datagram will be active in the internet system
- **Flags** - 3 bits - values:
  1. R - reserved - must be cleared to zero
  2. DF - do not fragment - controls the fragmentation of the datagram
  3. MF - more fragments
- **Fragment offset - 13 bits** - Specifies the offset (in eight bytes blocks) of a particular fragment relative to the beginning of the original unfragmented IP datagram
- **Protocol** - 8 bits - Specifies the next encapsulated protocol
- **Header checksum** - 16 bits - A 16 bit one's complement checksum of the IP header and IP options
- **Source IP address** - 32 bits
- **Destination IP address** - 32 bits
- **Options** - variable length
- **Padding** - variable length - Used as a filler to guarantee that the data starts on a 32 bit boundary

7.7 IP version 6

Internet Protocol version 6 (**IPv6**) is the next-generation Internet Protocol version designated as the successor to IPv4, the first implementation used in the Internet that is still in dominant use currently. It is an Internet Layer protocol for packet-switched internetworks. The main driving force for the redesign of Internet Protocol is the foreseeable IPv4 address exhaustion. IPv6 was defined in December 1998 by the Internet Engineering Task Force (IETF) with the publication of an Internet standard specification, RFC 2460.

IPv6 has a vastly larger address space than IPv4. This results from the use of a 128-bit address, whereas IPv4 uses only 32 bits. The new address space thus supports 2^{128} (about 3.4×10^{38}) addresses. This expansion provides flexibility in allocating addresses and routing traffic and eliminates the primary need for network address translation (NAT), which gained widespread deployment as an effort to alleviate IPv4 address exhaustion.

IPv6 also implements new features that simplify aspects of address assignment (stateless address autoconfiguration) and network renumbering (prefix and router announcements) when changing Internet connectivity providers. The IPv6 subnet size has been standardized by fixing the size of the host identifier portion of an address to 64 bits to facilitate an automatic mechanism for forming the host identifier from Link Layer media addressing information (MAC address).

Network security is integrated into the design of the IPv6 architecture. Internet Protocol Security (IPsec) was originally developed for IPv6, but found widespread optional deployment first in IPv4 (into which it was back-engineered). The IPv6 specifications mandate IPsec implementation as a fundamental interoperability requirement.

As of March 2016, the IPV6 penetration level is slightly above 5%, although it has been implemented on all major operating systems in use in commercial, business, and home consumer environments.

7.7.1 motivation and origins

The first publicly used version of the Internet Protocol, Version 4 (IPv4), provides an addressing capability of about 4 billion addresses (2^{32}). This was deemed sufficient in the early design stages of the Internet when the explosive growth and worldwide proliferation of networks was not anticipated.

During the first decade of operation of the TCP/IP-based Internet, by the late 1980s, it became apparent that methods had to be developed to conserve address space. In the early 1990s, even after the introduction of classless network redesign, it became clear that this would not suffice to prevent IPv4 address exhaustion and that further changes to the Internet infrastructure were needed. By the beginning of 1992, several proposed systems were being circulated, and by the end of 1992, the IETF announced a call for white papers (RFC 1550) and the creation of the "IP Next Generation" (IPng) area of working groups.

The Internet Engineering Task Force adopted IPng on July 25, 1994, with the formation of several IPng working groups. By 1996, a series of RFCs were released defining Internet Protocol Version 6 (IPv6), starting with RFC 2460.

The technical discussion, development and introduction of IPv6 was not without controversy and the design has been criticized for lack of interoperability with IPv4 and other aspects.

Incidentally, the IPng architects could not use version number 5 as a successor to IPv4, because it had been assigned to an experimental flow-oriented streaming protocol (Internet Stream Protocol), similar to IPv4, intended to support video and audio.

It is widely expected that IPv4 will be supported alongside IPv6 for the foreseeable future. IPv4 - only nodes are not able to communicate directly with IPv6 nodes, and will need assistance from an intermediary.

7.8 features and differences from IPv4

In most regards, IPv6 is a conservative extension of IPv4. Most transport- and application-layer protocols need little or no change to operate over IPv6; exceptions are application protocols that embed internet-layer addresses, such as FTP or NTPv3.

IPv6 specifies a new packet format, designed to minimize packet-header processing. Since the headers of IPv4 packets and IPv6 packets are significantly different, the two protocols are not interoperable.

7.8.1 larger address space

The most important feature of IPv6 is a much larger address space than that of IPv4: addresses in IPv6 are 128 bits long, compared to 32-bit addresses in IPv4.

The very large IPv6 address space supports a total of 2^{128} (about 3.4×10^{38}) addresses—or approximately 4.6×10^{28} (roughly 2^{95}) addresses for each of the roughly 7.4 billion (7.4×10^{9}) people alive in 2016. In another perspective, there is the same number of IP addresses per person as the number of atoms in a metric ton of carbon.

While these numbers are impressive, it was not the intent of the designers of the IPv6 address space to assure geographical saturation with usable addresses. Rather, the longer addresses allow a better, systematic, hierarchical allocation of addresses and efficient route aggregation. With IPv4, complex Classless Inter-Domain Routing (CIDR) techniques were developed to make the best use of the small address space. Renumbering an existing network for a new connectivity provider with different routing prefixes is a major effort with IPv4, as discussed in RFC 2071 and RFC 2072. With IPv6, however, changing the prefix announced by a few routers can in principle renumber an entire network since the host identifiers (the least-significant 64 bits of an address) can be independently self-configured by a host.

The size of a subnet in IPv6 is 2^{64} addresses (64-bit subnet mask), the square of the size of the entire IPv4 Internet. Thus, actual address space utilization rates will likely be small in IPv6, but network management and routing will be more efficient because of the inherent design decisions of large subnet space and hierarchical route aggregation.

7.8.2 stateless address autoconfiguration

IPv6 hosts can configure themselves automatically when connected to a routed IPv6 network using ICMPv6 (**Internet Control Message Protocol**) router discovery messages. When first connected to a network, a host sends a link-local multicast **router solicitation** request for its configuration parameters; if configured suitably, routers respond to such a request with a *router advertisement* packet that contains network-layer configuration parameters.

If IPv6 stateless address autoconfiguration is unsuitable for an application, a network may use stateful configuration with the DHCP for IPv6 (DHCPv6) or hosts may be configured statically.

Routers present a special case of requirements for address configuration, as they often are sources for autoconfiguration information, such as router and prefix advertisements. Stateless configuration for routers can be achieved with a special router renumbering protocol.

7.8.3 multicast

Multicast, the ability to send a single packet to multiple destinations, is part of the base specification in IPv6. This is unlike IPv4, where it is optional (although usually implemented).

IPv6 **does not implement broadcast**, which is the ability to send a packet to all hosts on the attached link. The same effect can be achieved by sending a packet to the link-local ***all hosts*** **multicast group**. It therefore lacks the notion of a broadcast address—the highest address in a subnet (the broadcast address for that subnet in IPv4) is considered a normal address in IPv6.

Most environments, however, do not currently have their network infrastructures configured to route multicast packets; multicasting on a single subnet will work, but global multicasting might not.

IPv6 multicast shares common features and protocols with IPv4 multicast, but also provides changes and improvements. When even the smallest IPv6 global routing prefix is assigned to an organization, the organization is also assigned the use of 4.2 billion globally routable source-specific IPv6 multicast groups to assign for inner-domain or cross-domain multicast applications [RFC 3306]. In IPv4 it was very difficult for an organization to get even one globally routable cross-domain multicast group assignment and implementation of cross-domain solutions was very arcane [RFC 2908]. IPv6 also supports new multicast solutions, including Embedded Rendezvous Point [RFC 3956] which simplifies the deployment of cross domain solutions.

7.8.4 mandatory network layer security

Internet Protocol Security (IPsec), the protocol for IP encryption and authentication, forms an integral part of the base protocol suite in IPv6. IPsec support is mandatory in IPv6; this is unlike IPv4, where it is optional (but usually implemented). IPsec, however, is not widely used at present except for securing traffic between IPv6 Border Gateway Protocol routers.

7.8.5 simplified processing by routers

A number of simplifications have been made to the packet header, and the process of packet forwarding has been simplified, in order to make packet processing by routers simpler and hence more efficient. Specifically:

- The packet header in IPv6 is simpler than that used in IPv4, with many rarely used fields moved to separate options; in effect, although the addresses in IPv6 are four times larger, the (option-less) IPv6 header is only twice the size of the (option-less) IPv4 header.
- IPv6 routers do not perform fragmentation. IPv6 hosts are required to either perform PMTU discovery, perform end-to-end fragmentation, or to send packets smaller than the IPv6 minimum MTU size of 1280 octets.
- The IPv6 header is not protected by a checksum; integrity protection is assumed to be assured by both a link layer checksum and a higher layer (TCP, UDP, etc.) checksum. In effect, IPv6 routers do not need to recompute a checksum when header fields (such as the TTL or Hop Count) change. This improvement may have been made less necessary by the development of routers that perform checksum computation at link speed using dedicated hardware, but it is still relevant for software based routers.
- The Time-to-Live field of IPv4 has been renamed to *Hop Limit*, reflecting the fact that routers are no longer expected to compute the time a packet has spent in a queue.

7.8.6 mobility

Unlike mobile IPv4, Mobile IPv6 (MIPv6) avoids triangular routing and is therefore as efficient as normal IPv6. IPv6 routers may also support Network Mobility (NEMO) [RFC 3963] which allows entire subnets to move to a new router connection point without renumbering. However, since neither MIPv6 nor MIPv4 or NEMO are widely deployed today, this advantage is mostly theoretical.

7.8.7 options extensibility

IPv4 has a fixed size (40 octets) of option parameters. In IPv6, options are implemented as additional extension headers after the IPv6 header, which limits their size only by the size of an entire packet. The extension header mechanism allows IPv6 to be easily 'extended' to support future services for QoS, security, mobility, etc. without a redesign of the basic protocol.

7.8.8 jumbograms

IPv4 limits packets to 65535 (2^{16} - 1) octets of payload. IPv6 has optional support for packets over this limit, referred to as jumbograms, which can be as large as 4294967295 (2^{32} - 1) octets. The use of jumbograms may improve performance over high-MTU links. The use of jumbograms is indicated by the Jumbo Payload Option header.

7.8.9 addressing

The increased length of network addresses emphasizes a most important change when moving from

IPv4 to IPv6. IPv6 addresses are 128 bits long, whereas IPv4 addresses are 32 bits; where the IPv4 address space contains roughly 4.3×10^9 (4.3 billion) addresses, IPv6 has enough room for 3.4×10^{38} (340 trillion trillion trillion) unique addresses.

IPv6 addresses are normally written with hexadecimal digits and colon separators like `2001:db8:85a3::8a2e:370:7334`, as opposed to the dot-decimal notation of the 32 bit IPv4 addresses. IPv6 addresses are typically composed of two logical parts: a 64-bit (sub-)network prefix, and a 64-bit host part.

IPv6 addresses are classified into three types: unicast addresses which uniquely identify network interfaces, anycast addresses which identify a group of interfaces—mostly at different locations—for which traffic flows to the nearest one, and multicast addresses which are used to deliver one packet to many interfaces. Broadcast addresses are not used in IPv6. Each IPv6 address also has a 'scope', which specifies in which part of the network it is valid and unique. Some addresses have node scope or link scope; most addresses have global scope (i.e. they are unique globally).

Some IPv6 addresses are used for special purposes, like the loopback address. Also, some address ranges are considered special, like link-local addresses (for use in the local network only) and solicited-node multicast addresses (used in the Neighbor Discovery Protocol).

7.8.10 IPv6 in the domain name system

IPv6 addresses are represented in the Domain Name System by *AAAA* resource records (so-called *quad-A* records) for forward lookups. Reverse lookup takes place under `ip6.arpa` (previously `ip6.int`), where name space is allocated by the ASCII representation of nibble units (digits) of the hexadecimal IP address. This scheme, which is an adaptation of the IPv4 method under *in-addr.arpa*, is defined in RFC 3596.

Consider a host named *derrick* with Unique Local Address `fdda:5cc1:23:4::1f`. To represent this address in the DNS two records are needed:

```
derrick.example.com.    IN    AAAA    fdda:5cc1:23:4::1f
```

in the forward mapping zone. For the reverse mapping zone the (rather awkward) record

```
f.1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.4.0.0.0.3.2.0.0.1.c.c.5.a.d.d.f    IN
PTR   derrick.example.com.
```

takes care of the mapping of the IPv6 address back to the name *derrick*.

7.9 IPv6 PDU

The data part of the PDU comes from the upper layer. The header part is detailed below.

00-03	04-11	12-15	16-23	24-31
Version	Traffic class	Flow label		
Payload length			Next header	Hop limit
Source address - 128 bits				
Destination address - 128 bits				
Data				

- **Version** - 4 bits - IPv6 version number
- **Traffic class** - 8 bits - Internet traffic priority delivery value
- **Flow label** - 20 bits - Specifies special router handling from source to destination(s) for a sequence of packets
- **Payload length** - 16 bits unsigned - Specifies the length of the data in the packet. When cleared to zero, the option is a hop-by-hop Jumbo payload
- **Next header** – 8 bits -Specifies the next encapsulated protocol. The values are compatible with those specified for the IPv4 protocol field
- **Hop limit** – 8 bits unsigned - For each router that forwards the packet, the hop limit is decremented by 1. When the hop limit field reaches zero, the packet is discarded. This replaces the TTL field in the IPv4 header that was originally intended to be used as a time based hop limit
- **Source address -** 128 bits
- **Destination address -** 128 bits

7.10 whatever happened to IP version 5?

Depending on who responds to this question, there are two (not that much) different answers.

The first one, goes as follows - version 5 was in fact intentionally skipped to avoid confusion, or at least to rectify it. The problem with version 5 relates to an experimental TCP/IP protocol called the **Internet Stream Protocol, Version 2**, originally defined in RFC 1190. This protocol was originally seen by some as being a peer of IP at the Internet Layer in the TCP/IP architecture, and in its standard, these packets were assigned IP version 5 to differentiate them from “normal” IP packets (version 4). This protocol apparently never went anywhere, but to be absolutely sure that there would be no confusion, version 5 of the Internet Protocol was skipped over in favor of version 6.

The other story of the version 5 demise is that this used to designate a UNIX based, experimental, non-IP, streaming protocol, called ST2, described in RFC 1819. Also, the fact that the IP address size jumped from 2^5 bits to 2^7 bits (skipping the 2^6 bits size) made the designation of the emerging standard as IP version 6 a more natural choice.

chapter 8 IP addresses, internet control protocols

8.1 network addressing

The **network address** (which can also be called the network number) uniquely identifies each network. Every machine on the same network shares that network address as part of its IP address. In the IP address 172.16.30.56, for example, 172.16 is the network address.

The **node address** is assigned to, and uniquely identifies, each machine on a network. This part of the address must be unique because it identifies a particular machine - an individual - as opposed to a network, which is a group. This number can also be referred to as a *host address*. In the sample IP address 172.16.30.56, the 30.56 is the node address.

The designers of the Internet decided to create classes of networks based on network size. For the small number of networks possessing a very large number of nodes, they created the rank **Class A network**. At the other extreme is the **Class C network**, which is reserved for the numerous networks with a small number of nodes. The class distinction for networks between very large and very small is predictably called the **Class B network**.

Subdividing an IP address into a network and node address is determined by the class designation of one's network.

To ensure efficient routing, Internet designers defined a mandate for the leading-bits section of the address for each different network class. For example, since a router knows that a Class A network address always starts with a 0, the router might be able to speed a packet on its way after reading only the first bit of its address. This is where the address schemes define the difference between a Class A, a Class B, and a Class C address. In the next sections, We'll discuss the differences between these three classes, followed by a discussion of the Class D and Class E addresses (class A, B and C are the only ranges that are used to address hosts in our networks).

8.1.1 network address range: class A

The designers of the IP address scheme said that the first bit of the first byte in a Class A network address must always be off, or 0. This means a Class A address must be between 0 and 127, inclusive.

Consider the following network address:

0xxxxxxx

If we turn the other 7 bits all off and then turn them all on, we'll find the Class A range of network addresses:

00000000 = 0

01111111 = 127

So, a Class A network is defined in the first octet between 0 and 127, and it can't be less or more. (yes, I know 0 and 127 are not valid in a class A network - we'll talk about illegal addresses in a minute.)

8.1.2 network address range: class B

In a Class B network, the RFCs state that the first bit of the first byte must always be turned on, but the second bit must always be turned off. If you turn the other 6 bits all off and then all on, you will find the range

for a Class B network:

10000000 = 128

10111111 = 191

As you can see, a Class B network is defined when the first byte is configured from 128 to 191.

8.1.3 network address range: class C

For Class C networks, the RFCs define the first 2 bits of the first octet as always turned on, but the third bit can never be on. Following the same process as the previous classes, convert from binary to decimal to find the range. Here's the range for a Class C network:

11000000 = 192

11011111 = 223

So, if you see an IP address that starts at 192 and goes to 223, you'll know it is a Class C IP address.

8.1.4 network address ranges: classes D and E

The addresses between 224 and 255 are reserved for Class D and E networks. Class D (224–239 or **1110** starting addresses) is used for multicast addresses and Class E (240–255 or **1111** starting addresses) for scientific purposes, but I'm not going into these types of addresses in this book (and you don't need to know them).

8.1.5 network addresses: special purpose

Some IP addresses are reserved for special purposes, so network administrators can't ever assign these addresses to nodes. The table below lists the members of this exclusive little club and the reasons why they're included in it.

ADDRESS	FUNCTION
Network address of all 0s	Designates **this** network or segment
Network address of all 1s	Designates **all** networks
Network 127.0.0.1	Reserved for **loopback** tests. Designates the local node and allows that node to send a packet to itself without generating network traffic.
Node address of all 0s	Interpreted to mean **network address** or any host on specified network.
Node address of all 1s	Interpreted to mean **all nodes** on the specified network.
Entire IP address set to all 0s – 0.0.0.0	Used by Cisco routers to designate the default route. Could also mean **any network**.
Entire IP address set to all 1s – 255.255.255.255	**Broadcast** to all nodes on the current network; sometimes called an "all 1s broadcast" or limited broadcast.

8.2 class A addresses

In a Class A network address, the first byte is assigned to the network address, and the three remaining bytes are used for the node addresses. The Class A format is:

network.node.node.node

For example, in the IP address 49.22.102.70, the 49 is the network address, and 22.102.70 is the node address. Every machine on this particular network would have the distinctive network address of 49.

Class A network addresses are one byte long, with the first bit of that byte reserved and the 7 remaining bits available for manipulation (addressing). As a result, the maximum number of Class A networks that can be created is 128. Why? Because each of the 7 bit positions can be either a 0 or a 1, thus 2^7 or 128.

To complicate matters further, the network address of all 0s (0000 0000) is reserved to designate the default route (see the table in the previous section). Additionally, the address 127, which is reserved for diagnostics, can't be used either, which means that you can really only use the numbers 1 to 126 to designate Class A network addresses. This means the actual number of usable Class A network addresses is 128 minus 2, or **126**.

Each Class A address has three bytes (24-bit positions) for the node address of a machine. This means there are 2^24 - or 16,777,216 - unique combinations and, therefore, precisely that many possible unique node addresses for each Class A network. Because node addresses with the two patterns of all 0s and all 1s are reserved, the actual maximum usable number of nodes for a Class A network is 2^24 - 2, which equals 16,777,214. Either way, that's a huge amount of hosts on a network segment!

8.2.1 class A valid host IDs

Here's an example of how to figure out the valid host IDs in a Class A network address:

- All host bits off is the network address: 10.0.0.0.
- All host bits on is the broadcast address: 10.255.255.255.

The valid hosts are the numbers in between the network address and the broadcast address: 10.0.0.1 through 10.255.255.254. Notice that 0s and 255s can be valid host IDs. All you need to remember when trying to find valid host addresses is that the host bits can't all be turned off or all be on at the same time.

8.3 class B addresses

In a Class B network address, the first two bytes are assigned to the network address and the remaining two bytes are used for node addresses. The format is:

network.**network**.node.node

For example, in the IP address 172.16.30.56, the network address is 172.16, and the node address is 30.56. With a network address being two bytes (8 bits each), there would be 2^16 unique combinations.

But the Internet designers decided that all Class B network addresses should start with the binary digit 1, then 0. This leaves 14 bit positions to manipulate, therefore 16,384 (that is, 2^14) unique Class B network addresses.

A Class B address uses two bytes for node addresses. This is 2^16 minus the two reserved patterns (all 0s

and all 1s), for a total of 65,534 possible node addresses for each Class B network.

8.3.1 class B valid host IDs

Here's an example of how to find the valid hosts in a Class B network:

- All host bits turned off is the network address: 172.16.0.0.
- All host bits turned on is the broadcast address: 172.16.255.255.

The valid hosts would be the numbers in between the network address and the broadcast address: 172.16.0.1 through 172.16.255.254.

8.4 class C addresses

The first three bytes of a Class C network address are dedicated to the network portion of the address, with only one measly byte remaining for the node address. The format is

network.**network**.**network**.node

Using the example IP address 192.168.100.102, the network address is 192.168.100, and the node address is 102.

In a Class C network address, the first three bit positions are always the binary 110. The calculation is: 3 bytes, or 24 bits, minus 3 reserved positions, leaves 21 positions. Hence, there are 2^21, or 2,097,152 possible Class C networks.

Each unique Class C network has one byte to use for node addresses. This leads to 2^8 or 256, minus the two reserved patterns of all 0s and all 1s, for a total of 254 node addresses for each Class C network.

8.4.1 class C valid host IDs

Here's an example of how to find a valid host ID in a Class C network:

- All host bits turned off is the network ID: 192.168.100.0.
- All host bits turned on is the broadcast address: 192.168.100.255.

The valid hosts would be the numbers in between the network address and the broadcast address: 192.168.100.1 through 192.168.100.254.

8.5 private IP addresses

The people who created the IP addressing scheme also created what we call private IP addresses.

These addresses can be used on a private network, but they're not routable through the Internet.

This is designed for the purpose of creating a measure of well-needed security, but it also conveniently saves valuable IP address space.

If every host on every network had to have real routable IP addresses, we would have run out of IP addresses to hand out years ago. But by using private IP addresses, ISPs, corporations, and home users

only need a relatively tiny group of bona fide IP addresses to connect their networks to the Internet. This is economical because they can use private IP addresses on their inside networks and get along just fine.

To accomplish this task, the ISP and the corporation—the end user, no matter who they are - need to use something called a *Network Address Translation (NAT)*, which basically takes a private IP address and converts it for use on the Internet. Many people can use the same real IP address to transmit out onto the Internet. Doing things this way saves megatons of address space - good for us all! I'll provide an introduction to what NAT is in the section "Introduction to Network Address Translation (NAT)."

The reserved private addresses are listed in the table below.

ADDRESS CLASS	RESERVED ADDRESS SPACE
Class A	10.0.0.0 through 10.255.255.255
Class B	172.16.0.0 through 172.31.255.255
Class C	192.168.0.0 through 192.168.255.255

8.6 broadcast addresses

Even though I've referred to broadcast addresses throughout Chapters 1 and 2, I really haven't gone into their different terms and uses. Here are the four different types I'd like to define:

1. **Layer 2 broadcasts -** These are sent to all nodes on a LAN.
2. **Broadcasts (layer 3) -** These are sent to all nodes on the network.
3. **Unicast -** These are sent to a single destination host.
4. **Multicast -** These are packets sent from a single source, and transmitted to many devices on different networks.

First, understand that layer 2 broadcasts are also known as **hardware broadcasts** - they only go out on a LAN, and they usually don't go past the LAN boundary (router) unless they become a unicast packet (discussed in a minute). The typical hardware address is 6 bytes (48 bits) and looks something like 0c.43.a4.f3.12.c2. The broadcast would be all 1s in binary and all Fs in hexadecimal, as in FF.FF.FF.FF.FF.FF.

Then there's the plain old broadcast addresses at layer 3. Broadcast messages are meant to reach all hosts on a broadcast domain. These are the network broadcasts that have all host bits on. Here's an example that you're already familiar with: The network address of 172.16.0.0 255.255.0.0 would have a broadcast address of 172.16.255.255 - all host bits on. Broadcasts can also be "all networks and all hosts," as indicated by 255.255.255.255. A good example of a broadcast message is an **Address Resolution Protocol** (ARP) request. When a host has a packet, it knows the logical address (IP) of the destination. To get the packet to the destination, the host needs to forward the packet to a default gateway if the destination resides on a different IP network. If the destination is on the local network, the source will forward the packet directly to the destination. Because the source doesn't have the MAC address it needs to forward the frame to, it sends out a broadcast, something that every device in the local broadcast domain will listen to. This broadcast says, in essence, "If you are the owner of IP address 192.168.2.3, please forward your MAC address to me," with the source giving the appropriate information.

A **unicast** is different because it's a broadcast that becomes directed to an actual destination IP address - in other words, it's directed to a specific host, and a DHCP client request is a good example of how a unicast works. Here's an example: Your host on a LAN sends out an FF.FF.FF.FF.FF.FF layer 2 broadcast and 255.255.255.255 layer 3 destination broadcast looking for a DHCP server on the LAN. The router will see that this is a broadcast meant for the DHCP server because it has a destination port number of 67 (BootP server), and will forward the request to the IP address of the DHCP server on another LAN. So, basically, if

your DHCP server IP address is 172.16.10.1, your host just sends out a 255.255.255.255 DHCP client broadcast request, and the router changes that broadcast to the specific destination address of 172.16.10.1.

Be careful here - a host address can really look like a broadcast address! For example, a Unicast is a broadcast that's forwarded to a specific host. Is the IP address 172.16.128.255/18 a valid host or a broadcast address? Sure looks like a broadcast address!

First, /18 is 255.255.192.0. This makes the valid networks 64.0 and 128.0, where the broadcast address for the 172.16.128.0 subnet is 172.16.191.255, so the valid hosts are 128.1 through 191.254. 172.16.128.255 is a valid host and is a unicast! Don't worry - I'll show you how to subnet this in the next chapter. For now, understand that just because an IP address has a 0 or 255 in the address doesn't always make it a broadcast.

Multicast is a different beast entirely. At first glance, it appears to be a hybrid of unicast and broadcast communication, but that isn't quite the case. Multicast does allow point-to-multipoint communication, which is similar to broadcasts, but it happens in a different manner. The crux of *multicast* is that it enables multiple recipients to receive messages without flooding the messages to all hosts on a broadcast domain.

Multicast works by sending messages or data to IP *multicast group* addresses. Routers then forward copies of the packet out every interface that has hosts *subscribed* to that group address. This is where multicast differs from broadcast messages - with multicast communication, copies of packets, in theory, are sent only to subscribed hosts.

There are several different groups that users or applications can subscribe to. The range of multicast addresses starts with 224.0.0.0, and goes through 239.255.255.255. As you can see, this range of addresses falls within IP Class D address space based on classful IP assignment.

8.7 NAT - network address translation

IP addresses are scarce. An ISP might have a /16 (formerly class B) address, giving it 65,534 host numbers. If it has more customers than that, it has a problem. For home customers with dial-up connections, one way around the problem is to dynamically assign an IP address to a computer when it calls up and logs in and take the IP address back when the session ends. In this way, a single /16 address can handle up to 65,534 active users, which is probably good enough for an ISP with several hundred thousand customers. When the session is terminated, the IP address is reassigned to another caller. While this strategy works well for an ISP with a moderate number of home users, it fails for ISPs that primarily serve business customers.

The problem is that business customers expect to be on-line continuously during business hours. Both small businesses, such as three-person travel agencies, and large corporations have multiple computers connected by a LAN. Some computers are employee PCs; others may be Web servers. Generally, there is a router on the LAN that is connected to the ISP by a leased line to provide continuous connectivity. This arrangement means that each computer must have its own IP address all day long. In effect, the total number of computers owned by all its business customers combined cannot exceed the number of IP addresses the ISP has. For a /16 address, this limits the total number of computers to 65,534. For an ISP with tens of thousands of business customers, this limit will quickly be exceeded.

To make matters worse, more and more home users are subscribing to ADSL or Internet over cable. Two of the features of these services are (1) the user gets a permanent IP address and (2) there is no connect charge (just a monthly flat rate charge), so many ADSL and cable users just stay logged in permanently. This development just adds to the shortage of IP addresses. Assigning IP addresses on-the-fly as is done with dial-up users is of no use because the number of IP addresses in use at any one instant may be many times the number the ISP owns.

Also , many ADSL and cable users have two or more computers at home, often one for each family member, and they all want to be on-line all the time using the single IP address their ISP has given them. The solution here is to connect all the PCs via a LAN and put a router on it. From the ISP's point of view, the family is now the same as a small business with a handful of computers. Welcome to Jones, Inc.

The problem of running out of IP addresses is not a theoretical problem that might occur at some point in

the distant future. It is happening right here and right now. The long-term solution is for the whole Internet to migrate to IPv6, which has 128-bit addresses. This transition is slowly occurring, but it will be years before the process is complete. As a consequence, some people felt that a quick fix was needed for the short term. This quick fix came in the form of **NAT** (Network Address Translation), which is described in RFC 3022 and which we will summarize below.

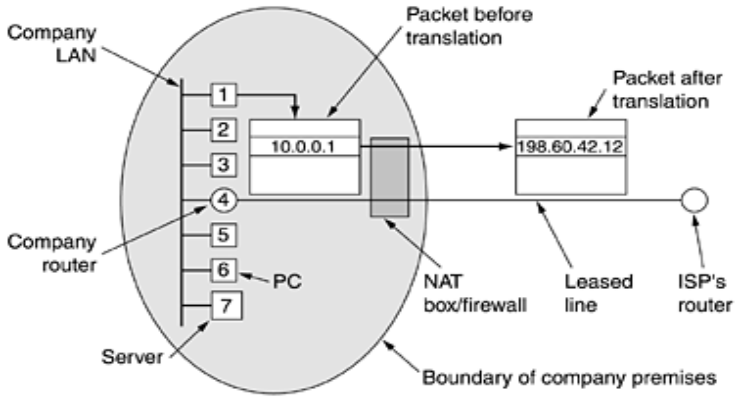
The basic idea behind NAT is to assign each company a single IP address (or at most, a small number of them) for Internet traffic. *Within* the company, every computer gets a unique IP address, which is used for routing intramural traffic. However, when a packet exits the company and goes to the ISP, an address translation takes place. To make this scheme possible, three ranges of IP addresses have been declared as private. Companies may use them internally as they wish. The only rule is that no packets containing these addresses may appear on the Internet itself. The three reserved ranges are:

10.0.0.0	– 10.255.255.255/8	(16,777,216 hosts)
172.16.0.0	– 172.31.255.255/12	(1,048,576 hosts)
192.168.0.0	– 192.168.255.255/16	(65,536 hosts)

The first range provides for 16,777,216 addresses (except for 0 and -1, as usual) and is the usual choice of most companies, even if they do not need so many addresses.

The operation of NAT is shown below. Within the company premises, every machine has a unique address of the form 10.x.y.z. However, when a packet leaves the company premises, it passes through a NAT box that converts the internal IP source address, 10.0.0.1 in the figure, to the company's true IP address, 198.60.42.12 in this example. The NAT box is often combined in a single device with a firewall, which provides security by carefully controlling possible to integrate the NAT box into the company's router.

Placement and operation of a NAT box.



So far we have glossed over one tiny little detail: when the reply comes back (e.g., from a Web server), it is naturally addressed to 198.60.42.12, so how does the NAT box know which address to replace it with? Herein lies the problem with NAT. If there were a spare field in the IP header, that field could be used to keep track of who the real sender was, but only 1 bit is still unused. In principle, a new option could be created to hold the true source address, but doing so would require changing the IP code on all the machines on the entire Internet to handle the new option. This is not a promising alternative for a quick fix.

What actually happened is as follows. The NAT designers observed that most IP packets carry either TCP or UDP payloads. When we studied TCP and UDP, we saw that both of these have headers containing a source port and a destination port. Below we will just discuss TCP ports, but exactly the same story holds for UDP ports. The ports are 16-bit integers that indicate where the TCP connection begins and ends. These ports provide the field needed to make NAT work.

When a process wants to establish a TCP connection with a remote process, it attaches itself to an

unused TCP port on its own machine. This is called the source port and tells the TCP code where to send incoming packets belonging to this connection. The process also supplies a destination port to tell who to give the packets to on the remote side. Ports 0–1023 are reserved for well-known services. For example, port 80 is the port used by Web servers, so remote clients can locate them. Each outgoing TCP message contains both a source port and a destination port. Together, these ports serve to identify the processes using the connection on both ends.

An analogy may make the use of ports clearer. Imagine a company with a single main telephone number. When people call the main number, they reach an operator who asks which extension they want and then puts them through to that extension. Ports are an extra 16-bits of addressing that identify which process gets which incoming packet.

Using the *Source port* field, we can solve our mapping problem. Whenever an outgoing packet enters the NAT box, the 10.*x.y.z* source address is replaced by the company's true IP address. In addition, the TCP *Source port* field is replaced by an index into the NAT box's 65,536-entry translation table. This table entry contains the original IP address and the original source port. Finally, both the IP and TCP header checksums are recomputed and inserted into the packet. It is necessary to replace the *Source port* because connections from machines 10.0.0.1 and 10.0.0.2 may both happen to use port 5000, for example, so the *Source port* alone is not enough to identify the sending process.

When a packet arrives at the NAT box from the ISP, the *Source port* in the TCP header is extracted and used as an index into the NAT box's mapping table. From the entry located, the internal IP address and original TCP *Source port* are extracted and inserted into the packet. Then both the IP and TCP checksums are recomputed and inserted into the packet. The packet is then passed to the company router for normal delivery using the 10.*x.y.z* address.

NAT can also be used to alleviate the IP shortage for ADSL and cable users. When the ISP assigns each user an address, it uses 10.*x.y.z* addresses. When packets from user machines exit the ISP and enter the main Internet, they pass through a NAT box that translates them to the ISP's true Internet address. On the way back, packets undergo the reverse mapping. In this respect, to the rest of the Internet, the ISP and its home ADSL/cable users just looks like a big company.

Although this scheme sort of solves the problem, many people in the IP community regard it as an abomination-on-the-face-of-the-earth. Briefly summarized, here are some of the objections. First, NAT violates the architectural model of IP, which states that every IP address uniquely identifies a single machine worldwide. The whole software structure of the Internet is built on this fact. With NAT, thousands of machines may (and do) use address 10.0.0.1.

Second, NAT changes the Internet from a connectionless network to a kind of connection-oriented network. The problem is that the NAT box must maintain information (the mapping) for each connection passing through it. Having the network maintain connection state is a property of connection-oriented networks, not connectionless ones. If the NAT box crashes and its mapping table is lost, all its TCP connections are destroyed. In the absence of NAT, router crashes have no effect on TCP. The sending process just times out within a few seconds and retransmits all unacknowledged packets. With NAT, the Internet becomes as vulnerable as a circuit-switched network.

Third, NAT violates the most fundamental rule of protocol layering: layer *k* may not make any assumptions about what layer *k* + 1 has put into the payload field. This basic principle is there to keep the layers independent. If TCP is later upgraded to TCP-2, with a different header layout (e.g., 32-bit ports), NAT will fail. The whole idea of layered protocols is to ensure that changes in one layer do not require changes in other layers. NAT destroys this independence.

Fourth, processes on the Internet are not required to use TCP or UDP. If a user on machine *A* decides to use some new transport protocol to talk to a user on machine *B* (for example, for a multimedia application), introduction of a NAT box will cause the application to fail because the NAT box will not be able to locate the TCP *Source port* correctly.

Fifth, some applications insert IP addresses in the body of the text. The receiver then extracts these addresses and uses them. Since NAT knows nothing about these addresses, it cannot replace them, so any attempt to use them on the remote side will fail. FTP, the standard **File Transfer Protocol** works this way and can fail in the presence of NAT unless special precautions are taken. Similarly, the H.323 Internet telephony protocol has this property and can fail in the presence of NAT. It may be possible to patch NAT to work with H.323, but patching the code every time a new application comes along is not a good idea.

Sixth, since the TCP *Source port* field is 16 bits, at most 65,536 machines can be mapped onto an IP address. Actually, the number is slightly less because the first 4096 ports are reserved for special uses. However, if multiple IP addresses are available, each one can handle up to 61,440 machines. These and other problems with NAT are discussed in RFC 2993. In general, the opponents of NAT say that by fixing the problem of insufficient IP addresses with a temporary and ugly hack, the pressure to implement the real solution, that is, the transition to IPv6, is reduced, and this is a bad thing.

8.8 the Internet Control Message Protocol - ICMP

The operation of the Internet is monitored closely by the routers. When something unexpected occurs, the event is reported by the **ICMP** (**Internet Control Message Protocol**), which is also used to test the Internet. About a dozen types of ICMP messages are defined. The most important ones are listed in the figure below. Each ICMP message type is encapsulated in an IP packet.

The principal ICMP message types.

Message type	Description
Destination unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet
Redirect	Teach a router about geography
Echo	Ask a machine if it is alive
Echo reply	Yes, I am alive
Timestamp request	Same as Echo request, but with timestamp
Timestamp reply	Same as Echo reply, but with timestamp

The DESTINATION UNREACHABLE message is used when the subnet or a router cannot locate the destination a packet with the *DF* bit cannot be delivered because a "small-packet" network stands in the way.

The TIME EXCEEDED message is sent when a packet is dropped because its counter has reached zero. This event is a symptom that packets are looping, that there is enormous congestion, or that the timer values are being set too low.

The PARAMETER PROBLEM message indicates that an illegal value has been detected in a header field. This problem indicates a bug in the sending host's IP software or in the software of a router transited.

The SOURCE QUENCH message was formerly used to throttle hosts that were sending too many packets. When a host received this message, it was expected to slow down. It is rarely used any more because when congestion occurs, these packets tend to add more fuel to the fire. Congestion control in the Internet is now done largely in the transport layer.

The REDIRECT message is used when a router notices that a packet seems to be routed wrong. It is used by the router to tell the sending host about the probable error.

The ECHO and ECHO REPLY messages are used to see if a given destination is reachable and alive. Upon receiving the ECHO message, the destination is expected to send an ECHO REPLY message back. The TIMESTAMP REQUEST and TIMESTAMP REPLY messages are similar, except that the arrival time of the message and the departure time of the reply are recorded in the reply. This facility is used to measure network performance.

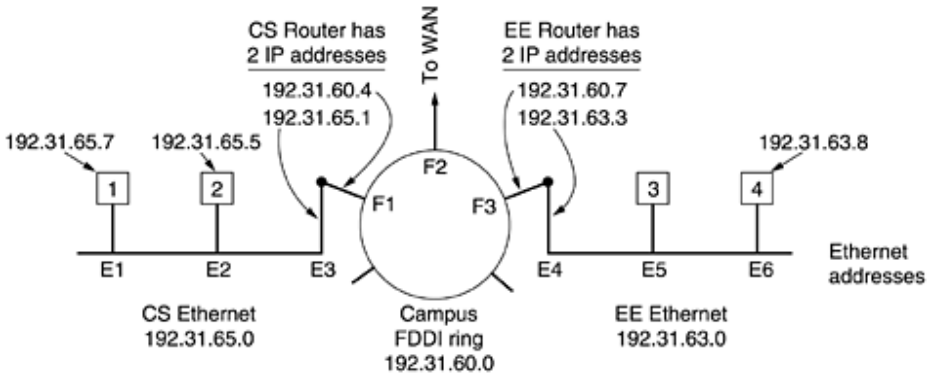
In addition to these messages, others are defined. See www.iana.org/assignments/icmp-parameters.

8.9 ARP - the Address Resolution Protocol

Although every machine on the Internet has one (or more) IP addresses, these cannot actually be used for sending packets because the data link layer hardware does not understand Internet addresses. Nowadays, most hosts at companies and universities are attached to a LAN by an interface board that only understands LAN addresses. For example, every Ethernet board ever manufactured comes equipped with a 48-bit Ethernet address. Manufacturers of Ethernet boards request a block of addresses from a central authority to ensure that no two boards have the same address (to avoid conflicts should the two boards ever appear on the same LAN). The boards send and receive frames based on 48-bit Ethernet addresses. They know nothing at all about 32-bit IP addresses.

The question now arises: How do IP addresses get mapped onto data link layer addresses, such as Ethernet? To explain how this works, let us use the example below, in which a small university with several class C (now called /24) networks is illustrated. Here we have two Ethernets, one in the Computer Science Dept., with IP address 192.31.65.0 and one in Electrical Engineering, with IP address 192.31.63.0. These are connected by a campus backbone ring (e.g., FDDI) with IP address 192.31.60.0. Each machine on an Ethernet has a unique Ethernet address, labeled *E1* through *E6*, and each machine on the FDDI (**Fiber Distributed Data Interface**) ring has an FDDI address, labeled *F1* through *F3*.

Three interconnected /24 networks: two Ethernets and an FDDI ring.



Let us start out by seeing how a user on host 1 sends a packet to a user on host 2. Let us assume the sender knows the name of the intended receiver, possibly something like mary@eagle.cs.uni.edu. The first step is to find the IP address for host 2, known as *eagle.cs.uni.edu*. This lookup is performed by the Domain Name System. For the moment, we will just assume that DNS returns the IP address for host 2 (192.31.65.5).

The upper layer software on host 1 now builds a packet with 192.31.65.5 in the *Destination address* field and gives it to the IP software to transmit. The IP software can look at the address and see that the destination is on its own network, but it needs some way to find the destination's Ethernet address. One solution is to have a configuration file somewhere in the system that maps IP addresses onto Ethernet addresses. While this solution is certainly possible, for organizations with thousands of machines, keeping all these files up to date is an error-prone, time-consuming job.

A better solution is for host 1 to output a broadcast packet onto the Ethernet asking: Who owns IP address 192.31.65.5? The broadcast will arrive at every machine on Ethernet 192.31.65.0, and each one will check its IP address. Host 2 alone will respond with its Ethernet address (*E2*). In this way host 1 learns that IP address 192.31.65.5 is on the host with Ethernet address *E2*. The protocol used for asking this question and getting the reply is called **ARP** (**Address Resolution Protocol**). Almost every machine on the Internet runs it. ARP is defined in RFC 826. We use ARP to get a hardware (MAC) address from an IP address.

The advantage of using ARP over configuration files is the simplicity. The system manager does not have to do much except assign each machine an IP address and decide about subnet masks. ARP does the rest.

At this point, the IP software on host 1 builds an Ethernet frame addressed to *E2*, puts the IP packet (addressed to 192.31.65.5) in the payload field, and dumps it onto the Ethernet. The Ethernet board of host 2 detects this frame, recognizes it as a frame for itself, scoops it up, and causes an interrupt. The Ethernet driver extracts the IP packet from the payload and passes it to the IP software, which sees that it is correctly addressed and processes it.

Various optimizations are possible to make ARP work more efficiently. To start with, once a machine has run ARP, it caches the result in case it needs to contact the same machine shortly. Next time it will find the mapping in its own cache, thus eliminating the need for a second broadcast. In many cases host 2 will need to send back a reply, forcing it, too, to run ARP to determine the sender's Ethernet address. This ARP broadcast can be avoided by having host 1 include its IP-to-Ethernet mapping in the ARP packet. When the ARP broadcast arrives at host 2, the pair (192.31.65.7, E1) is entered into host 2's ARP cache for future use. In fact, all machines on the Ethernet can enter this mapping into their ARP caches.

Yet another optimization is to have every machine broadcast its mapping when it boots. This broadcast is generally done in the form of an ARP looking for its own IP address. There should not be a response, but a side effect of the broadcast is to make an entry in everyone's ARP cache. If a response does (unexpectedly) arrive, two machines have been assigned the same IP address. The new one should inform the system manager and not boot.

To allow mappings to change, for example, when an Ethernet board breaks and is replaced with a new one (and thus a new Ethernet address), entries in the ARP cache should time out after a few minutes.

Now let us look at our network again, only this time host 1 wants to send a packet to host 4 (192.31.63.8). Using ARP will fail because host 4 will not see the broadcast (routers do not forward Ethernet-level broadcasts). There are two solutions. First, the CS router could be configured to respond to ARP requests for network 192.31.63.0 (and possibly other local networks). In this case, host 1 will make an ARP cache entry of (192.31.63.8, E3) and happily send all traffic for host 4 to the local router. This solution is called proxy ARP. The second solution is to have host 1 immediately see that the destination is on a remote network and just send all such traffic to a default Ethernet address that handles all remote traffic, in this case E3. This solution does not require having the CS router know which remote networks it is serving.

Either way, what happens is that host 1 packs the IP packet into the payload field of an Ethernet frame addressed to *E3*. When the CS router gets the Ethernet frame, it removes the IP packet from the payload field and looks up the IP address in its routing tables. It discovers that packets for network 192.31.63.0 are supposed to go to router 192.31.60.7. If it does not already know the FDDI address of 192.31.60.7, it broadcasts an ARP packet onto the ring and learns that its ring address is *F3*. It then inserts the packet into the payload field of an FDDI frame addressed to *F3* and puts it on the ring.

At the EE router, the FDDI driver removes the packet from the payload field and gives it to the IP software, which sees that it needs to send the packet to 192.31.63.8. If this IP address is not in its ARP cache, it broadcasts an ARP request on the EE Ethernet and learns that the destination address is *E6*, so it builds an Ethernet frame addressed to *E6*, puts the packet in the payload field, and sends it over the Ethernet. When the Ethernet frame arrives at host 4, the packet is extracted from the frame and passed to the IP software for processing.

Going from host 1 to a distant network over a WAN works essentially the same way, except that this time the CS router's tables tell it to use the WAN router whose FDDI address is *F2*.

8.10 RARP, BOOTP and DHCP

ARP solves the problem of finding out which Ethernet address corresponds to a given IP address. Sometimes the reverse problem has to be solved: Given an Ethernet address, what is the corresponding IP address? In particular, this problem occurs when a diskless workstation is booted. Such a machine will normally get the binary image of its operating system from a remote file server. But how does it learn its IP address?

The first solution devised was to use RARP (**Reverse Address Resolution Protocol**) (defined in RFC 903). This protocol allows a newly-booted workstation to broadcast its Ethernet address and say: My 48-bit

Ethernet address is 14.04.05.18.01.25. Does anyone out there know my IP address? The RARP server sees this request, looks up the Ethernet address in its configuration files, and sends back the corresponding IP address.

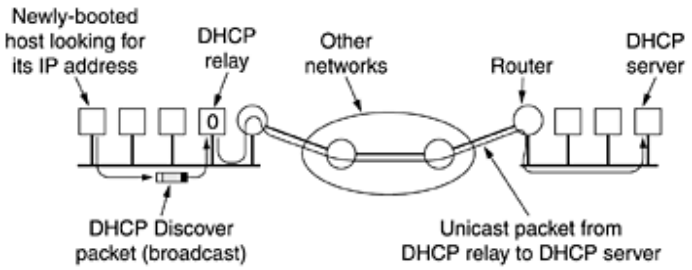
Using RARP is better than embedding an IP address in the memory image because it allows the same image to be used on all machines. If the IP address were buried inside the image, each workstation would need its own image.

A disadvantage of RARP is that it uses a destination address of all 1s (limited broadcasting) to reach the RARP server. However, such broadcasts are not forwarded by routers, so a RARP server is needed on each network. To get around this problem, an alternative bootstrap protocol called BOOTP was invented. Unlike RARP, BOOTP uses UDP messages, which are forwarded over routers. It also provides a diskless workstation with additional information, including the IP address of the file server holding the memory image, the IP address of the default router, and the subnet mask to use. BOOTP is described in RFCs 951, 1048, and 1084.

A serious problem with BOOTP is that it requires manual configuration of tables mapping IP address to Ethernet address. When a new host is added to a LAN, it cannot use BOOTP until an administrator has assigned it an IP address and entered its (Ethernet address, IP address) into the BOOTP configuration tables by hand. To eliminate this error-prone step, BOOTP was extended and given a new name: DHCP (Dynamic Host Configuration Protocol). DHCP allows both manual IP address assignment and automatic assignment. It is described in RFCs 2131 and 2132. In most systems, it has largely replaced RARP and BOOTP.

Like RARP and BOOTP, DHCP is based on the idea of a special server that assigns IP addresses to hosts asking for one. This server need not be on the same LAN as the requesting host. Since the DHCP server may not be reachable by broadcasting, a DHCP relay agent is needed on each LAN, as shown below.

Operation of DHCP



To find its IP address, a newly-booted machine broadcasts a DHCP DISCOVER packet. The DHCP relay agent on its LAN intercepts all DHCP broadcasts. When it finds a DHCP DISCOVER packet, it sends the packet as a unicast packet to the DHCP server, possibly on a distant network. The only piece of information the relay agent needs is the IP address of the DHCP server.

An issue that arises with automatic assignment of IP addresses from a pool is how long an IP address should be allocated. If a host leaves the network and does not return its IP address to the DHCP server, that address will be permanently lost. After a period of time, many addresses may be lost. To prevent that from happening, IP address assignment may be for a fixed period of time, a technique called leasing. Just before the lease expires, the host must ask the DHCP for a renewal. If it fails to make a request or the request is denied, the host may no longer use the IP address it was given earlier.

chapter 9 subnetting – practical issues

9.1 subnetting basics

In the previous chapter, you learned how to define and find the valid host ranges used in a Class A, Class B, and Class C network address by turning the host bits all off and then all on. This is very good, but here's the catch: You were only defining one network. What happens if you wanted to take one network address and create six networks from it? You would have to do something called subnetting, because that's what allows you to take one larger network and break it into a bunch of smaller networks.

There are loads of reasons in favor of subnetting. Some of the benefits include:

- **Reduced network traffic** - We all appreciate less traffic of any kind. Networks are no different. Without trusty routers, packet traffic could grind the entire network down to a near standstill. With routers, most traffic will stay on the local network; only packets destined for other networks will pass through the router. Routers create broadcast domains. The more broadcast domains you create, the smaller the broadcast domains and the less network traffic on each network segment.
- **Optimized network performance** - This is a result of reduced network traffic.
- **Simplified management** - It's easier to identify and isolate network problems in a group of smaller connected networks than within one gigantic network.
- **Facilitated spanning** of large geographical distances - Because WAN links are considerably slower and more expensive than LAN links, a single large network that spans long distances can create problems in every area listed above. Connecting multiple smaller networks makes the system more efficient.

In the following sections, I am going to move to subnetting a network address. This is the good part - ready?

9.2 how to create subnets

To create subnetworks, you take bits from the host portion of the IP address and reserve them to define the subnet address. This means fewer bits for hosts, so the more subnets, the fewer bits available for defining hosts.

Later in this chapter, you'll learn how to create subnets, starting with Class C addresses. But before you actually implement subnetting, you need to determine your current requirements as well as plan for future conditions.

Before we move on to designing and creating a subnet mask, you need to understand that in this first section we will be discussing classful routing, which means that all hosts (all nodes) in the network use the exact same subnet mask. When we move on to Variable Length Subnet Masks (VLSMs) I'll discuss classless routing, which means that each network segment can use a different subnet mask.

Follow these steps:

1. Determine the number of required network IDs:
   - One for each subnet
   - One for each wide area network connection
2. Determine the number of required host IDs per subnet:

- One for each TCP/IP host
- One for each router interface

3. Based on the above requirements, create the following:
   - One subnet mask for your entire network
   - A unique subnet ID for each physical segment
   - A range of host IDs for each subnet

9.3 subnet masks

For the subnet address scheme to work, every machine on the network must know which part of the host address will be used as the subnet address. This is accomplished by assigning a subnet mask to each machine. A subnet mask is a 32-bit value that allows the recipient of IP packets to distinguish the network ID portion of the IP address from the host ID portion of the IP address.

The network administrator creates a 32-bit subnet mask composed of 1s and 0s. The 1s in the subnet mask represent the positions that refer to the network or subnet addresses.

Not all networks need subnets, meaning they use the default subnet mask. This is basically the same as saying that a network doesn't have a subnet address. Table 3.1 shows the default subnet masks for Classes A, B, and C. These default masks cannot change. In other words, you can't make a Class B subnet mask read 255.0.0.0. If you try, the host will read that address as invalid and usually won't even let you type it in.

For a Class A network, you can't change the first byte in a subnet mask; it must read 255.0.0.0 at a minimum. Similarly, you cannot assign 255.255.255.255, as this is all 1s—a broadcast address. A Class B address must start with 255.255.0.0, and a Class C has to start with 255.255.255.0.

TABLE 3 . 1

Default Subnet Mask

Class	Format	Default Subnet Mask
A	network.node.node.node	255.0.0.0
B	network.network.node.node	255.255.0.0
C	network.network.network.node	255.255.255.0

9.4 classless inter - domain routing (CIDR)

Another term you need to familiarize yourself with is Classless Inter-Domain Routing (CIDR).

It's basically the method that ISPs (Internet Service Providers) use to allocate an amount of addresses to a company, a home - a customer. They provide addresses in a certain block size, something I'll be going into in greater detail later in this chapter.

When you receive a block of addresses from an ISP, what you get will look something like this: 192.168.10.32/28. This is telling you what your subnet mask is. The slash notation (/) means how many bits are turned on (1s). Obviously, the maximum could only be /32 because a byte is 8 bits and there are four bytes in an IP address: (4 × 8 = 32). But keep in mind that the largest subnet mask available (regardless of the class of address) can only be a /30 because you've got to keep at least 2 bits for host bits.

Take for example a Class A default subnet mask, which is 255.0.0.0. This means that the first byte of the subnet mask is all ones (1s), or 11111111. When referring to a slash notation, you need to count all the 1s bits to figure out your mask. The 255.0.0.0 is considered a /8 because it has 8 bits that are 1s—that is, 8 bits

that are turned on.

A Class B default mask would be 255.255.0.0, which is a /16 because 16 bits are ones (1s):

11111111.11111111.00000000.00000000.

Table 3.2 has a listing of every available subnet mask and its equivalent CIDR slash notation.

TABLE 3 . 2

CIDR Values

Subnet Mask	CIDR Value
255.0.0.0	/8
255.128.0.0	/9
255.192.0.0	/10
255.224.0.0	/11
255.240.0.0	/12
255.248.0.0	/13
255.252.0.0	/14
255.254.0.0	/15
255.255.0.0	/16
255.255.128.0	/17
255.255.192.0	/18
255.255.224.0	/19
255.255.240.0	/20
255.255.248.0	/21
255.255.252.0	/22
255.255.254.0	/23
255.255.255.0	/24
255.255.255.128	/25
255.255.255.192	/26
255.255.255.224	/27
255.255.255.240	/28
255.255.255.248	/29
255.255.255.252	/30

9.5 subnetting class C addresses

There are many different ways to subnet a network. The right way is the way that works best for you. First I'll show you how to use the binary method, and then we'll look at an easier way to do the same thing.

In a Class C address, only 8 bits are available for defining the hosts. Remember that subnet bits start at the left and go to the right, without skipping bits. This means that the only Class C subnet masks can be the following:

Binary	Decimal	CIDR
10000000 =	128	/25
11000000 =	192	/26
11100000 =	224	/27
11110000 =	240	/28
11111000 =	248	/29
11111100 =	252	/30

We can't use a /31 or /32 because we have to have at least 2 host bits for assigning IP addresses to hosts. In the past, I never discussed the /25 in a Class C network. Cisco always had been concerned with having at least 2 subnet bits, but now, because of the ip subnet-zero command, we can use just 1 subnet bit.

In the following sections we are going to look at the binary way of subnetting, then move into the new, improved, easy to understand and implement, subnetting method.

9.6 the binary method: subnetting a class C address

In this section, I'm going to teach you how to subnet a Class C address using the binary method.

I'll start by using the second subnet mask available with a Class C address, which borrows 2 bits for subnetting. For this example, I'll be using 255.255.255.192.

192 = 11000000

The 1s represent the subnet bits, and the 0s represent the host bits available in each subnet.

192 provides 2 bits for subnetting and 6 bits for defining the hosts in each subnet.

What are the subnets? Since we now use ip subnet-zero, we can get four subnets, instead of the two that were available without the ip subnet-zero command.

00000000 = 0 (all host bits off)

01000000 = 64 (all host bits off)

10000000 = 128 (all host bits off)

11000000 = 192 (all host bits off)

The valid hosts would be defined as the numbers between the subnets, minus the all-hostbits-off and all-host-bits-on numbers.

To find the hosts, first find your subnet: turn all the host bits off, then turn all the host bits on to find your broadcast address for the subnet. The valid hosts must be between those two numbers. The first table shows the 0 subnet, valid host range, and broadcast address. The second one shows the 64 subnet, valid host range, and broadcast address. The next table shows the 128 subnet, and the last one shows the 192 subnet (the subnet and host bits combine to form one byte).

Subnet 0

Subnet	Host	Meaning
00	000000 = 0	The network (do this first)
00	000001 = 1	The first valid host
00	111110 = 62	The last valid host

00	111111 = 63	The broadcast address (do this second)

Subnet 64

Subnet	Host	Meaning
01	000000 = 64	The network
01	000001 = 65	The first valid host
01	111110 = 126	The last valid host
01	111111 = 127	The broadcast address

Subnet 128

Subnet	Host	Meaning
10	000000 = 128	The subnet address
10	000001 = 129	The first valid host
10	111110 = 190	The last valid host
10	111111 = 191	The broadcast address

Subnet 192

Subnet	Host	Meaning
11	000000 = 192	The subnet address
11	000001 = 193	The first valid host
11	111110 = 254	The last valid host
11	111111 = 255	The broadcast address

Hopefully, you understood what I was trying to show you. The example I presented only used 2 subnet bits, so what if you had to subnet using 9, 10, or even 20 subnet bits? Try that with the binary method and see how long it takes you.

In the following section, I'm going to teach you an alternate method of subnetting that makes it easier to subnet larger numbers in no time.

Since the CCNA exam gives you just over a minute for each question, it's really important to know how much time you'll spend on a subnetting question.

That's why committing as much as possible to memory, as I suggested earlier in the chapter, is vital. Using the binary method can take you way too long and you could fail the exam even if you know the material!

9.7 the fast way: subnetting a class C address

When you've chosen a possible subnet mask for your network and need to determine the number of subnets, valid hosts, and broadcast addresses of a subnet that the mask provides, all you need to do is answer five simple questions:

- How many subnets does the chosen subnet mask produce?

- How many valid hosts per subnet are available?
- What are the valid subnets?
- What's the broadcast address of each subnet?
- What are the valid hosts in each subnet?

At this point it's important that you both understand and have memorized your powers of 2. Please refer to the sidebar earlier in this chapter if you need some help. Here's how you get the answers to those five big questions:

How many subnets?

2^x = number of subnets. *x* is the number of masked bits, or the 1s. For example, in 11000000, the number of ones gives us 2^2 subnets. In this example, there are 4 subnets.

How many hosts per subnet?

2^y – 2 = number of hosts per subnet. *y* is the number of unmasked bits, or the 0s. For example, in 11000000, the number of zeros gives us 2^6 – 2 hosts. In this example, there are 62 hosts per subnet. You need to subtract two for the subnet address and the broadcast address, which are not valid hosts.

What are the valid subnets?

256 – subnet mask = block size, or increment number. An example would be 256 – 192 = 64. The block size of a 192 mask is always 64. Start counting at zero in blocks of 64 until you reach the subnet mask value and these are your subnets. 0, 64, 128, 192. Easy, huh? Yes - that is, if you can count in the needed block size!

What's the broadcast address for each subnet?

Now here's the really easy part… Since we counted our subnets in the last section as 0, 64, 128, and 192, the broadcast address is always the number right before the next subnet. For example, the 0 subnet has a broadcast address of 63 because the next subnet is 64. The 64 subnet has a broadcast address of 127 because the next subnet is 128, etc. And remember, the broadcast of the last subnet (the subnet with the same interesting octets as the mask) is always 255 for Class C.

What are the valid hosts?

Valid hosts are the numbers between the subnets, omitting all the 0s and all 1s. For example, if 64 is the subnet number and 127 is the broadcast address, then 65–126 is the valid host range - it's *always* the numbers between the subnet address and the broadcast address.

I know this can truly seem confusing. But it really isn't as hard as it seems to be at first - just hang in there! Why not try a few and see for yourself?

9.8 subnetting practice examples: class C addresses

Here's your opportunity to practice subnetting Class C addresses using the method I just described. Exciting, isn't it! We're going to start with the first Class C subnet mask and work through every subnet that we can using a Class C address. When we're done, I'll show you how easy this is with Class A and B networks too!

9.8.1 practice example #1C: 255.255.255.192 (/26)

Let's use the Class C subnet mask from the preceding example, 255.255.255.192, to see how much simpler this method is than writing out the binary numbers. We're going to subnet the network address 192.168.10.0 and subnet mask 255.255.255.192.

192.168.10.0 = **Network address**

255.255.255.192 = **Subnet mask**

Now, let's answer the big five:

How many subnets?

Since 192 is 2 bits on (**11**000000), the answer would be 2^2.

How many hosts per subnet?

We have 6 host bits off (11**000000**), so the equation would be 2^6 – 2 = 62 hosts.

What are the valid subnets?

256 – 192 = 64. Remember, we start at zero and count in our block size, so our subnets are 0, 64, 128, and 192.

What's the broadcast address for each subnet?

The number right before the value of the next subnet is all host bits turned on and equals the broadcast address.

What are the valid hosts?

These are the numbers between the subnet and broadcast address. The easiest way to find the hosts is to write out the subnet address and the broadcast address. This way, the valid hosts are obvious. The following table shows the 0, 64, 128, and 192 subnets, the valid host ranges of each, and the broadcast address of each subnet:

The subnets (do this first) 0 64 128 192

Our first host (perform host addressing last) 1 65 129 193

Our last host 62 126 190 254

The broadcast address (do this second) 63 127 191 255

9.8.2 practice example #2C: 255.255.255.224 (/27)

This time, we'll subnet the network address 192.168.10.0 and subnet mask 255.255.255.224.

192.168.10.0 = **Network address**

255.255.255.224 = **Subnet mask**

How many subnets?

224 is 11100000, so our equation would be 2^3 = 8.

How many hosts?

2^5 – 2 = 30.

What are the valid subnets?

256 – 224 = 32. We just start at zero and count to the subnet mask value in blocks (increments) of 32: 0, 32, 64, 96, 128, 160, 192, 224.

What's the broadcast address for each subnet (always the number right before the next subnet)?

What are the valid hosts (the numbers between the subnet number and the broadcast address)?

To answer questions 4 and 5, first just write out the subnets, then write out the broadcast addresses—the number right before the next subnet. Lastly, fill in the host addresses. The following table gives you all the subnets for the 255.255.255.224 Class C subnet mask:

The subnet address 0 32 64 96 128 160 192 224

The first valid host 1 33 65 97 129 161 193 225

The last valid host 30 62 94 126 158 190 222 254

The broadcast address 31 63 95 127 159 191 223 255

9.8.3 practice example #3C: 255.255.255.240 (/28)

Let's practice on another one:

192.168.10.0 = **Network address**

255.255.255.240 = **Subnet mask**

Subnets? 240 is 11110000 in binary. 2^4 = 16.

Hosts? 4 host bits, or 2^4 – 2 = 14.

Valid subnets? 256 – 240 = 16. Start at 0. 0 + 16 = 16. 16 + 16 = 32. 32 + 16 = 48. 48 + 16 = 64. 64 + 16 = 80. 80 + 16 = 96. 96 + 16 = 112. 112 + 16 = 128. 128 + 16 = 144. 144 + 16 = 160. 160 + 16 = 176. 176 + 16 = 192. 192 + 16 = 208. 208 + 16 = 224. 224 + 16 = 240.

Broadcast address for each subnet?

Valid hosts?

To answer questions 4 and 5, check out the following table. It gives you the subnets, valid hosts, and broadcast addresses for each subnet. First, find the address of each subnet using the block size (increment). Second, find the broadcast address of each subnet increment (it's always the number right before the next valid subnet), then just fill in the host addresses. The following table shows the available subnets, hosts, and broadcast addresses provided from a Class C 255.255.255.240 mask.

Cisco has figured out the most people cannot count in sixteens and therefore have a hard time finding valid subnets, hosts, and broadcast addresses with the Class C 255.255.255.240 mask. You'd be wise to study this mask.

9.8.4 practice example #4C: 255.255.255.248 (/29)

Let's keep practicing:

192.168.10.0 = **Network address**

255.255.255.248 = **Subnet mask**

Subnets? 248 in binary = 11111000. 2^5 = 32.

Hosts? 2^3 – 2 = 6.

Valid subnets? 256 – 248 = 0, 8, 16, 24, 32, 40, 48, 56, 64, 72, 80, 88, 96, 104, 112, 120,

128, 136, 144, 152, 160, 168, 176, 184, 192, 200, 208, 216, 224, 232, 240, and 248.

***Broadcast address for each subnet*?**

Valid hosts?

Take a look at the following table. It shows some of the subnets (first four and last four only),

valid hosts, and broadcast addresses for the Class C 255.255.255.248 mask:

9.8.5 practice example #5C: 255.255.255.252 (/30)

Just a couple more:

192.168.10.0 = **Network address**

255.255.255.252 = **Subnet mask**

Subnets? 64.

Hosts? 2.

Valid subnets? 0, 4, 8, 12, etc., all the way to 252.

Broadcast address for each subnet? (always the number right before the next subnet)

Valid hosts? (the numbers between the subnet number and the broadcast address)

Subnet 0 8 16 24 … 224 232 240 248

First host 1 9 17 25 … 225 233 241 249

Last host 6 14 22 30 … 230 238 246 254

Broadcast 7 15 23 31 … 231 239 247 255

The next subnet is 64, so the broadcast address is 63. (Remember that the broadcast address of a subnet is always the number right before the next subnet.) The valid host range is 33–62. This is too easy! No, it's not?

Okay, then let's try another one. We'll subnet another Class C address:

192.168.10.33 = Node address

255.255.255.240 = Subnet mask

What subnet and broadcast address is the above IP address a member of? 256 – 240 = 16. 0, 16 + 16 = 32. 32 + 16 = 48. And bingo - the host address is between the 32 and 48 subnets.

The subnet is 192.168.10.32, and the broadcast address is 47. The valid host range is 33–46

Okay, we need to do more, just to make sure you have this down.

You have a node address of 192.168.10.174 with a mask of 255.255.255.240. What is the valid host range?

The mask is 240, so we'd do a 256 – 240 = 16. This is our block size. Just keep adding 16 until we pass the host address of 174: 0, 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176. The host address of 174 is between 160 and 176, so the subnet is 160. The broadcast address is 175, so the valid host range is 161–174. That was a tough one.

One more - just for fun. This is the easiest one of all Class C subnetting:

192.168.10.17 = **Node address**

255.255.255.252 = **Subnet mask**

What subnet and broadcast address is the above IP address a member of? 256 – 252 = 0, 4, 8, 12, 16, 20, etc. You've got it! The host address is between the 16 and 20 subnets. The subnet is 192.168.10.16, and the broadcast address is 19. The valid host range is 17–18.

Now that you're all over Class C subnetting, let's move on to Class B subnetting. But before we do, let's have a quick review.

9.8.6 what do we know?

Okay - here's where you can really apply what you've learned so far, and begin committing it all to memory. This is a very cool section that I've been using in my classes for years. It will really help you nail down subnetting!

When you see a subnet mask of slash notation (CIDR), you should know the following:

/26

What do we know about a /26?

- 192 mask
- 2 bits on and 6 bits off (11000000)
- Block size of 64
- 4subnets, each with 62 hosts

/27

What do we know about a /27?

- 224 mask
- 3 bits on and 5 bits off (11100000)
- Block size of 32
- 8 subnets, each with 30 hosts

/28

What do we know about a /28?

- 240 mask
- 4 bits on and 4 bits off
- Block size of 16
- 16 subnets, each with 14 hosts

/29

What do we know about a /29?

- 248 mask
- 5 bits on and 3 bits off
- Block size of 8
- 32 subnets, each with 6 hosts

/30

What do we know about a /30?

- 252 mask
- 6 bits on and 2 bits off
- Block size of 4
- 64 subnets, each with 2 hosts

Regardless whether you have a Class A, Class B, or Class C address, the /30 mask will only provide you with two hosts, ever. This mask is suited almost exclusively - as well as suggested by Cisco - for use on point-to-point links.

If you can memorize this section, you'll be much better off in your day-to-day job and in your studies. Try

saying it out loud, which helps you memorize things - yes, your significant other and/or coworkers will think you've lost it, but they probably already do if you are in the networking field. And if you're not yet in the networking field but are studying all this to break into it, you might as well have people start thinking you're an odd bird now, since they will eventually anyway.

It's also helpful to write these on some type of flash card and have people test your skill.

You'd be amazed at how fast you can get subnetting down if you memorize block sizes, as well as this "What Do We Know?" section.

9.9 subnetting class B addresses

Before we dive into this, let's look at all the possible Class B subnet masks first. Notice that we have a lot more possible subnet masks than we do with a Class C network address:

255.255.128.0 (/17)	255.255.255.0 (/24)
255.255.192.0 (/18)	255.255.255.128 (/25)
255.255.224.0 (/19)	255.255.255.192 (/26)
255.255.240.0 (/20)	255.255.255.224 (/27)
255.255.248.0 (/21)	255.255.255.240 (/28)
255.255.252.0 (/22)	255.255.255.248 (/29)
255.255.254.0 (/23)	255.255.255.252 (/30)

We know the Class B network address has 16 bits available for host addressing. This means we can use up to 14 bits for subnetting (because we have to leave at least 2 bits for host addressing).

By the way, do you notice anything interesting about that list of subnet values - a pattern, maybe? Ah ha! That's exactly why I had you memorize the binary-to-decimal numbers at the beginning of this section. Since subnet mask bits start on the left, move to the right, and can't skip bits, the numbers are always the same regardless of the class of address. Memorize this pattern.

The process of subnetting a Class B network is pretty much the same as it is for a Class C, except that you just have more host bits. Use the same subnet numbers for the third octet with Class B that you used for the fourth octet with Class C, but add a zero to the network portion and a 255 to the broadcast section in the fourth octet. The following table shows you an example host range of two subnets used in a Class B subnet:

Just add the valid hosts between the numbers, and you're set!

This above example is only true until you get up to /24. After that, it's numerically exactly like Class C.

9.10 subnetting practice examples: class B addresses

This section will give you an opportunity to practice subnetting Class B addresses.

9.10.1 practice example #1B: 255.255.192.0 (/18)

172.16.0.0 = **Network address**

255.255.192.0 = **Subnet mask**

Subnets? *2^2 = 4.*

Hosts? 2^14 – 2 = 16,382 (6 bits in the third octet, and 8 in the fourth).

Valid subnets? 256 – 192 = 64. 0, 64, 128, 192. Remember the subnetting is performed in the third octet, so the subnet numbers are really 0.0, 64.0, 128.0, and 192.0, as shown in the next table.

Broadcast address for each subnet?

Valid hosts?

The following table shows the four subnets available, the valid host range, and the broadcast address of each:

Subnet	0.0	64.0	128.0	192.0
First host	0.1	64.1	128.1	192.1
Last host	63.254	127.254	191.254	255.254
Broadcast	63.255	127.255	191.255	255.255

Notice that we just added the fourth octet's lowest and highest values and came up with the answers. Again, it's pretty much the same as it is for a Class C subnet—we just added 0 and 255 in the fourth octet.

9.10.2 practice example #2B: 255.255.240.0 (/20)

172.16.0.0 = **Network address**

255.255.240.0 = **Subnet mask**

Subnets? 2^4 = 16.

Hosts? 2^12 – 2 = 4094.

Valid subnets? 256 – 240 = 0, 16, 32, 48, etc., up to 240. Notice that these are the same numbers as a Class C 240 mask.

Broadcast address for each subnet?

Valid hosts?

The following table shows the first four subnets, valid hosts, and broadcast addresses in a Class B 255.255.240.0 mask:

Subnet	0.0	16.0	32.0	48.0
First host	0.1	16.1	32.1	48.1
Last host	15.254	31.254	47.254	63.254
Broadcast	15.255	31.255	47.255	63.255

9.10.3 practice example #3B: 255.255.254.0 (/23)

172.16.0.0 = **Network address**

255.255.254.0 = **Subnet mask**

Subnets? 2^7 = 128.

Hosts? 2^9 – 2 = 510.

Valid subnets? 256 – 254 = 0, 2, 4, 6, 8, etc., up to 254.

Broadcast address for each subnet?

Valid hosts?

The following table shows the first five subnets, valid hosts, and broadcast addresses in a Class B 255.255.254.0 mask:

Subnet	0.0	2.0	4.0	6.0	8.0
First host	0.1	2.1	4.1	6.1	8.1
Last host	1.254	3.254	5.254	7.254	9.254
Broadcast	1.255	3.255	5.255	7.255	9.255

In your studies, remember that it's very important for you to know your Class B /23 mask, and how many subnets and hosts it provides!

9.10.4 practice example #4B: 255.255.255.0 (/24)

Contrary to popular belief, 255.255.255.0 used with a Class B network address is not called a Class B network with a Class C subnet mask. It's amazing how many people see this mask used in a Class B network and think it's a Class C subnet mask. This is a Class B subnet mask with 8 bits of subnetting—it's considerably different from a Class C mask. Subnetting this address is fairly simple:

172.16.0.0 = **Network address**

255.255.255.0 = **Subnet mask**

Subnets? 2^8 = 256.

Hosts? 2^8 – 2 = 254.

Valid subnets? 256 – 255 = 1. 0, 1, 2, 3, etc. all the way to 255.

Broadcast address for each subnet?

Valid hosts?

The following table shows the first four subnets and the last two, valid hosts, and broadcast addresses in a Class B 255.255.255.0 mask:

Subnet	0.0	1.0	2.0	3.0 ...	254.0	255.0
First host	0.1	1.1	2.1	3.1 ...	254.1	255.1
Last host	0.254	1.254	2.254	3.254 ...	254.254	255.254
Broadcast	0.255	1.255	2.255	3.255 ...	254.255	255.255

9.10.5 practice example #5B: 255.255.255.128 (/25)

This is one of the hardest subnet masks you can play with, though. And worse, it actually is a really good

subnet to use in production because it creates over 500 subnets with 126 hosts for each subnet—a nice mixture. So, don't skip over it!

172.16.0.0 = **Network address**

255.255.255.128 = **Subnet mask**

Subnets? 2^9 = 512.

Hosts? 2^7 – 2 = 126.

Valid subnets? Okay, now for the tricky part. 256 – 255 = 1. 0, 1, 2, 3, etc., for the third octet. But you can't forget the one subnet bit used in the fourth octet. Remember when I showed you how to figure one subnet bit with a Class C mask? You figure this the same way. (Now you know why I showed you the 1-bit subnet mask in the Class C section—to make this part easier.) You actually get two subnets for each third octet value, hence the 512 subnets. For example, if the third octet is showing subnet 3, the two subnets would actually be 3.0 and 3.128.

Broadcast address for each subnet?

Valid hosts?

The following table shows how you can create subnets, valid hosts, and broadcast addresses using the Class B 255.255.255.128 subnet mask (the first eight subnets are shown, and then the last two subnets):

Subnet	0.0	0.128	1.0	1.128	2.0	2.128 ...	255.0	255.128
First host	0.1	0.129	1.1	1.129	2.1	2.129 ...	255.1	255.129
Last host	0.126	0.254	1.126	1.254	2.126	2.254 ...	255.126	255.254
Broadcast	0.127	0.255	1.127	1.255	2.127	2.255 ...	255.127	255.255

As with the /23 mask, it's also really important for you to know your Class B / 25 mask and how many subnets and hosts it provides!

9.10.6 practice example #6B: 255.255.255.192 (/26)

Now, this is where Class B subnetting gets easy. Since the third octet has a 255 in the mask section, whatever number is listed in the third octet is a subnet number. However, now that we have a subnet number in the fourth octet, we can subnet this octet just like we did with Class C subnetting. Let's try it out:

172.16.0.0 = **Network address**

255.255.255.192 = **Subnet mas**k

Subnets? 2^10 = 1024.

Hosts? 2^6 – 2 = 62.

Valid subnets? 256 – 192 = 64. The subnets are shown in the following table. Do these numbers look familiar?

Broadcast address for each subnet?

Valid hosts?

The following table shows the first eight subnet ranges, valid hosts, and broadcast addresses:

Subnet	0.0	0.64	0.128	0.192	1.0	1.64	1.128	1.192
First host	0.1	0.65	0.129	0.193	1.1	1.65	1.129	1.193
Last host	0.62	0.126	0.190	0.254	1.62	1.126	1.190	1.254

Broadcast	0.63	0.127	0.191	0.255	1.63	1.127	1.191	1.255

Notice that for each subnet value in the third octet, you get subnets 0, 64, 128, and 192 in the fourth octet.

9.10.7 practice example #7B: 255.255.255.224 (/27)

This is done the same way as the preceding subnet mask, except that we just have more subnets and fewer hosts per subnet available.

172.16.0.0 = **Network address**

255.255.255.224 = **Subnet mask**

Subnets? 2^11 = 2048.

Hosts? 2^5 – 2 = 30.

Valid subnets? 256 – 224 = 32. 0, 32, 64, 96, 128, 160, 192, 224.

Broadcast address for each subnet?

Valid hosts?

The following table shows the first eight subnets:

Subnet	0.0	0.32	0.64	0.96	0.128	0.160	0.192	0.224
First host	0.1	0.33	0.65	0.97	0.129	0.161	0.193	0.225
Last host	0.30	0.62	0.94	0.126	0.158	0.190	0.222	0.254
Broadcast	0.31	0.63	0.95	0.127	0.159	0.191	0.223	0.255

This next table shows the last eight subnets:

Subnet	255.0	255.32	255.64	255.96	255.128	255.160	255.192	255.224
First host	255.1	255.33	255.65	255.97	255.129	255.161	255.193	255.225
Last host	255.30	255.62	255.94	255.126	255.158	255.190	255.222	255.254
Broadcast	255.31	255.63	255.95	255.127	255.159	255.191	255.223	255.255

9.11 subnetting in your head: class B addresses

Are you nuts? Subnet Class B addresses in our heads? If you think easier equals crazy, then, yes, I'm a few sails short, but it's actually easier than writing it out—I'm not kidding! Let me show you how:

Q: What subnet and broadcast address is the IP address 172.16.10.33 255.255.255.224 a member of?

A: 256 – 224 = 32. 32 + 32 = 64. Bingo: 33 is between 32 and 64. However, remember that the third octet is considered part of the subnet, so the answer would be the 10.32 subnet. The broadcast is 10.63, since 10.64 is the next subnet.

Q: What subnet and broadcast address is the IP address 172.16.90.66 255.255.255.192 a member of?

A: 256 – 192 = 64. 64 + 64 = 128. The subnet is 172.16.90.64. The broadcast must be 172.16.90.127, since 90.128 is the next subnet.

Q: What subnet and broadcast address is the IP address 172.16.50.97 255.255.255.224 a member of?

A: 256 – 224 = 32, 64, 96, 128. The subnet is 172.16.50.96, and the broadcast must be 172.16.50.127 since 50.128 is the next subnet.

Q: What subnet and broadcast address is the IP address 172.16.10.10 255.255.255.192 a member of?

A: 256 – 192 = 64. This address must be in the 172.16.10.0 subnet, and the broadcast must be 172.16.10.63.

Q: What subnet and broadcast address is the IP address 172.16.10.10 255.255.255.252 a member of?

A: 256 – 252 = 4. The subnet is 172.16.10.8, with a broadcast of 172.16.10.11.

Q: What is the subnet and broadcast address of the host 172.16.88.255/20?

A: What is a /20? If you can't answer this, you can't answer this question, can you?

A /20 is 255.255.240.0, which gives us a block size of 16 in the third octet, and since no subnet bits are on in the fourth octet, the answer is always 0 and 255 in the fourth octet. 0, 16, 32, 48, 64, 80, 96…bingo. 88 is between 80 and 96, so the subnet is 80.0 and the broadcast address is 95.255.

9.12 subnetting class A addresses

Class A subnetting is not performed any differently from Classes B and C, but there are 24 bits to play with instead of the 16 in a Class B address and the 8 in a Class C address.

Let's start by listing all the Class A subnets:

255.128.0.0 (/9)	255.255.240.0 (/20)
255.192.0.0 (/10)	255.255.248.0 (/21)
255.224.0.0 (/11)	255.255.252.0 (/22)
255.240.0.0 (/12)	255.255.254.0 (/23)
255.248.0.0 (/13)	255.255.255.0 (/24)
255.252.0.0 (/14)	255.255.255.128 (/25)
255.254.0.0 (/15)	255.255.255.192 (/26)
255.255.0.0 (/16)	255.255.255.224 (/27)
255.255.128.0 (/17)	255.255.255.240 (/28)
255.255.192.0 (/18)	255.255.255.248 (/29)
255.255.224.0 (/19)	255.255.255.252 (/30)

That's it. You must leave at least 2 bits for defining hosts. And I hope you can see the pattern by now. Remember, we're going to do this the same way as a Class B or C subnet. It's just that, again, we simply have more host bits.

9.13 subnetting practice examples: class A addresses

When you look at an IP address and a subnet mask, you must be able to distinguish the bits used for subnets from the bits used for determining hosts. This is imperative. If you're still struggling with this concept,

please reread the preceding “IP Addressing” section. It shows you how to determine the difference between the subnet and host bits, and should help clear things up.

9.13.1 practice example #1A: 255.255.0.0 (/16)

Class A addresses use a default mask of 255.0.0.0, which leaves 22 bits for subnetting since you must leave 2 bits for host addressing. The 255.255.0.0 mask with a Class A address is using 8 subnet bits.

Subnets? 2^8 = 256.

Hosts? 2^16 – 2 = 65,534.

Valid subnets? 256 – 255 = 1. 0, 1, 2, 3, etc. (all in the second octet). The subnets would be 10.0.0.0, 10.1.0.0, 10.2.0.0, 10.3.0.0, etc., up to 10.255.0.0.

Broadcast address for each subnet?

Valid hosts?

The following table shows the first two and last two subnets, valid host range, and broadcast addresses for the private Class A 10.0.0.0 network:

Subnet	10.0.0.0	10.1.0.0	…	10.254.0.0	10.255.0.0
First host	10.0.0.1	10.1.0.1	…	10.254.0.1	10.255.0.1
Last host	10.0.255.254	10.1.255.254	...	10.254.255.254	10.255.255.254
Broadcast	10.0.255.255	10.1.255.255	…	10.254.255.255	10.255.255.255

9.13.2 Practice Example #2A: 255.255.240.0 (/20)

255.255.240.0 gives us 12 bits of subnetting and leaves us 12 bits for host addressing.

Subnets? 2^12 = 4096.

Hosts? 2^12 – 2 = 4094.

Valid subnets? 256 – 240 = 16. The subnets in the second octet are a block size of 1 and the subnets in the third octet are 0, 16, 32, etc.

Broadcast address for each subnet?

Valid hosts?

The following table shows some examples of the host ranges—the first three and the last subnets:

Subnet	10.0.0.0	10.0.16.0	10.0.32.0	…	10.0.240.0
First host	10.0.0.1	10.0.16.1	10.0.32.1	…	10.0.240.1
Last host	10.0.15.254	10.0.31.254	10.0.47.254	…	10.0.255.254
Broadcast	10.0.15.255	10.0.31.255	10.0.47.255	…	10.0.255.255

9.13.3 practice example #3A: 255.255.255.192 (/26)

Let's do one more example using the second, third, and fourth octets for subnetting.

Subnets? 2^18 = 262,144.

Hosts? 2^6 – 2 = 62.

Valid subnets? In the second and third octet, the block size is 1 and in the fourth octet the block size is 64.

Broadcast address for each subnet?

Valid hosts?

The following table shows the first four subnets and their valid hosts and broadcast addresses in the Class A 255.255.255.192 mask:

Subnet	10.0.0.0	10.0.0.64	10.0.0.128	10.0.0.192
First host	10.0.0.1	10.0.0.65	10.0.0.129	10.0.0.193
Last host	10.0.0.62	10.0.0.126	10.0.0.190	10.0.0.254
Broadcast	10.0.0.63	10.0.0.127	10.0.0.191	10.0.0.255

The following table shows the last four subnets and their valid hosts and broadcast addresses:

Subnet	10.255.255.0	10.255.255.64	10.255.255.128	10.255.255.192
First host	10.255.255.1	10.255.255.65	10.255.255.129	10.255.255.193
Last host	10.255.255.62	10.255.255.126	10.255.255.190	10.255.255.254
Broadcast	10.255.255.63	10.255.255.127	10.255.255.191	10.255.255.255

9.14 subnetting in your head: class A addresses

This sounds hard, but as with Class C and Class B, the numbers are the same; we just start in the second octet. What makes this easy? You only need to worry about the octet that has the largest block size (typically called the interesting octet; one that is something other than 0 or 255)—for example, 255.255.240.0 (/20) with a Class A network. The second octet has a block size of 1, so any number listed in that octet is a subnet. The third octet is a 240 mask, which means we have a block size of 16 in the third octet. If your host ID is 10.20.80.30, what is your subnet, broadcast address, and valid host range?

The subnet in the second octet is 20, but the third octet is in block sizes of 16, so we'll just count them out: 0, 16, 32, 48, 64, 80, 96... bingo! (By the way, you can count by sixteens by now, right?) This makes our subnet 10.20.80.0, with a broadcast of 10.20.95.255 because the next subnet is 10.20.96.0. The valid host range is 10.20.80.1 through 10.20.95.254. Yes, you can do this in your head if you just know your block sizes!

9.15 subnetting – a faster approach

Most subnetting problems sound like this:

Given an IP address and a CIDR number (like **172.16.10.33/27**), find:

1. the subnet (the subnet ID)
2. the broadcast address
3. the valid host range

In other problems, we have to find, as well:

4. the number of subnets within a network
5. the number of hosts within a subnet

How do we start?

step 1. Find n, c, x, y

Remember that the 32 bits of an IPv4 address are split into 3 parts: **n + x + y = 32**.

a) Find **n** first. The most significant n bits are used to identify the network class, and can have the values 8, 16, 24 for class A, B and C, respectively.

How do we know which class? We look at the first byte of the IP address. Class A IP addresses have the first byte in the range 0-127 (1-126 in real life), class B in the range 128-191 and class C in the range 192-223.

In our case the first byte is 172, class B, this means that **n = 16**.

b) **c** is the CIDR number, the number after the slash. In our case, **c = 27**.

c) because, by definition, c = n + x, we have that **x = c – n**. In our case, **x = 27-16 = 11**.

d) finally, **y = 32 – c**. In our case, **y = 32 – 27 = 5**.

step 2. Write the CIDR byte in binary format (base 2)

The **CIDR byte** is the byte where the c(+1)-th bit lies (counting from left to right). For example, if c = CIDR = 22, that would be the third byte. In our case, when CIDR = 27, that is the fourth (the last) byte.

Ok, so let's write the last byte of the given IP address in binary format, keeping the other bytes in the existing format (decimal). We get, since 33 = 32 + 1:

172.16. 10. **0010 0001**

step 3. Find the subnet ID

For that, we set all host bits (the last y bits, that's the last 5 in our case) equal to 0. We get:

172.16.10. 001**0 0000**

So, the **subnet ID** is **172.16.10.32**.

step 4. Find the broadcast address

For that, we set all host bits (the last y bits, that's the last 5 in our case) equal to 1. We get:

172.16.10. 001**1 1111**

So, the **broadcast address** is **172.16.10.63**.

step 5. **Find the valid host range**

The valid host range is the open interval (both ends excluded) between the subnet ID and the broadcast

address. In our case:

The valid host rage is **172.16.10.33 – 172.16.10.62**.

steps 6, 7. If required, find the number of subnets and the number of hosts in each subnet

Number of subnets = 2^x. In our case **2^11 = 2048**.

Number of hosts = 2^y - 2. In our case **2^5 – 2 = 30**.

9.16 subnetting fast, 3 examples

These 3 examples are taken from the CCNA study guide, ed 6. Written Lab 3.3, page 160/200.

Example 1. 192.168.20.123/28.

step 1. n = 24 (class C), x = c – n = 28 – 24 = 4, y = 32 – c = 32 – 28 = 4.

step 2. bit 28 is in the last byte, rewrite the IP address as: 192.168.20. **0111 1011**

step 3. the subnet ID is 192.168.20. 0111 **0000** = **192.168.20.112**

step 4. the broadcast address is 192.168.20. 0111 **1111** = **192.168.20.127**

step 5. the valid host range is **192.168.20.113 - 192.168.20.126**

step 6. the number of subnets is 2^x = 2^4 = **16**.

step 7. the number of hosts (within a subnet) is 2^y - 2 = 2^4 - 2 = **14**.

Example 2. 172.31.254.12/24.

step 1. n = 16 (class B), x = c – n = 24 – 16 = 8, y = 32 – c = 32 – 24 = 8.

step 2. bit 24 is in the third byte, but since this byte will not change, let's choose the next byte, where the (c+1)-th bit lies), rewrite the IP address as: 172.31.254. **0000 1100**

step 3. the subnet ID is 172.31.254. **0000 0000** = **172.31.254.0**

step 4. the broadcast address is 172.31.254. **1111 1111** = **172.31.254.255**

step 5. the valid host range is **172.31.254.1 - 172.31.254.254**

step 6. the number of subnets is 2^x = 2^8 = **256**.

step 7. the number of hosts (within a subnet) is 2^y - 2 = 2^8 - 2 = **254**.

Example 3. 63.24.89.21/18

step 1. n = 8 (class A), x = c – n = 18 – 8 = 10, y = 32 – c = 32 – 18 = 14.

step 2. bit 18 is in the third byte, rewrite the IP address as: 63.24. **0101 1001** .21

step 3. the subnet ID is 63.24. 01**00 0000** .**0 = 63.24.64.0**

step 4. the broadcast address is 63.24. 01**11 1111 .255 = 63.24.127.255**

step 5. the valid host range is **63.24.64.1 - 63.24.127.254**

step 6. the number of subnets is 2^x = 2^10 = **1024**.

step 7. the number of hosts (within a subnet) is 2^y - 2 = 2^14 - 2 = **16384**.

chapter 10 routing

10.1 routing principles

The main function of the network layer is routing packets from the source to destination. In most subnets, 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. The algorithms that choose the routes and the data structures used 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 subnet 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 subnet uses virtual circuits internally, routing decisions are made only when a new virtual circuit is being set up. Thereafter, data packets just follow the previously-established route. The latter case is sometimes called **session routing** because a route remains in force for an entire user session (e.g., a login session at a terminal or a file transfer).

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.

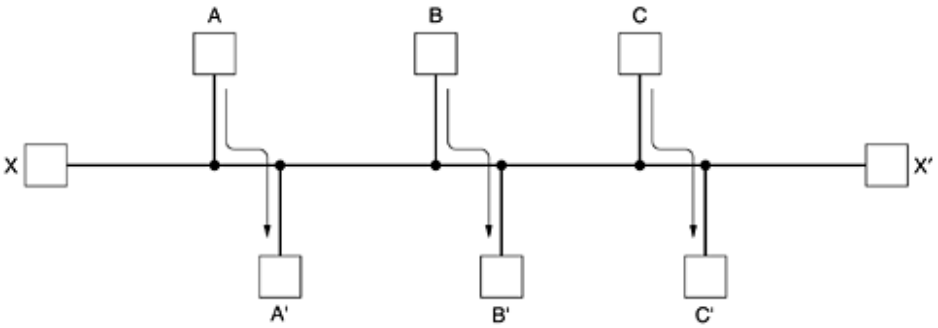
Regardless of whether routes are chosen independently for each packet or only when new connections are established, certain properties are desirable in a routing algorithm:

- correctness
- simplicity
- robustness
- stability
- fairness
- optimality.

Correctness and **simplicity** hardly require comment, but the need for **robustness** may be less obvious at first. Once a network comes on the air, it may be expected to run continuously for years without systemwide failures. The routing algorithm should cope with changes in the topology and traffic without requiring all jobs in all hosts to be aborted and the network to be rebooted every time some router crashes.

Stability is also an important goal for the routing algorithm. There exist routing algorithms that never converge to equilibrium, no matter how long they run. A stable algorithm reaches equilibrium and stays there. Fairness and optimality may sound obvious - surely no reasonable person would oppose them, but they are often contradictory goals. As a simple example of this conflict, look at the figure below.

Conflict between fairness and optimality



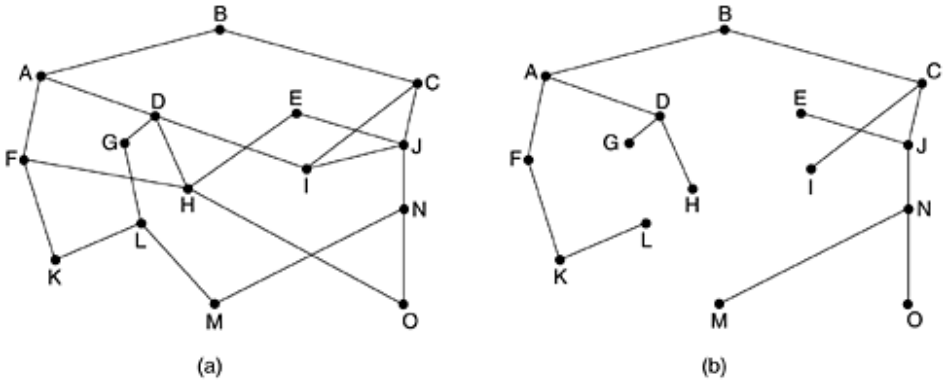
Suppose that there is enough traffic between *A* and *A*', between *B* and *B*', and between *C* and *C*' to saturate the horizontal links. To maximize the total flow, the *X* to *X*' traffic should be shut off altogether. Unfortunately, *X* and *X*' may not see it that way. Evidently, some compromise between global efficiency and fairness to individual connections is needed.

10.2 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**. 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 below, where the distance metric is the number of hops. Note that a sink tree is not necessarily unique; other trees with the same path lengths may exist. The goal of all routing algorithms is to discover and use the sink trees for all routers.

(a) A subnet. (b) A sink tree for router B



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.

10.3 routing protocols

A **routing protocol** is a protocol that specifies how routers communicate with each other, disseminating information that enables them to select routes between any two nodes on a computer network, the choice of the route being done by routing algorithms. Each router has *a priori* knowledge only of networks attached to it directly. A routing protocol shares this information first among immediate neighbors, and then throughout the

network. This way, routers gain knowledge of the topology of the network.

The term **routing protocol** may refer specifically to one operating at layer three of the OSI model, which similarly disseminates topology information between routers.

Although there are many types of routing protocols, three major classes are in widespread use on IP networks:

- **Interior gateway routing** via link-state routing protocols, such as OSPF and IS-IS
- Interior gateway routing via path vector or distance vector protocols, such as RIP, IGRP and EIGRP
- **Exterior gateway routing**. BGP v4 is the routing protocol used by the public Internet.

The specific characteristics of routing protocols include

- the manner in which they either prevent routing loops from forming or break them up if they do
- the manner in which they select preferred routes, using information about hop costs
- the time they take to converge
- how well they scale up
- many other factors

10.3.1 interior routing protocols

Interior Gateway Protocols (IGPs) exchange routing information within a single routing domain. A given autonomous system can contain multiple routing domains, or a set of routing domains can be coordinated without being an Internet-participating autonomous system. Common examples include:

- IGRP (Interior Gateway Routing Protocol)
- EIGRP (Enhanced Interior Gateway Routing Protocol)
- OSPF (Open Shortest Path First)
- RIP (Routing Information Protocol)
- IS-IS (Intermediate System to Intermediate System)

Note that IGRP, a Cisco proprietary routing protocol, is no longer supported. EIGRP accepts IGRP configuration commands, but the internals of IGRP and EIGRP are completely different.

10.3.2 exterior routing protocols

The **Exterior Gateway Protocol** (**EGP**) is a now obsolete routing protocol for the Internet originally specified in 1982 by Eric C. Rosen of Bolt, Beranek and Newman, and David L. Mills. It was first described in RFC 827 and formally specified in RFC 904 (1984). EGP is a simple reachability protocol, and, unlike modern distance-vector and path-vector protocols, it is limited to tree-like topologies.

During the early days of the Internet, an exterior gateway protocol, EGP version 3 (EGP3), was used to interconnect autonomous systems. EGP3 should not be confused with EGPs in general. Currently, **Border Gateway Protocol** (BGP) is the accepted standard for Internet routing and has essentially replaced the more limited EGP3.

10.3.3 adaptability

Routing protocols can be classified using another criterion, as well – **adaptability**.

Nonadaptive algorithms do not base their routing decisions on measurements or estimates of the current traffic and topology. Instead, the choice of the route to use to get from *I* to *J* (for all *I* and J) is computed in advance, off-line, and downloaded to the routers when the network is booted. This procedure is sometimes called **static routing**.

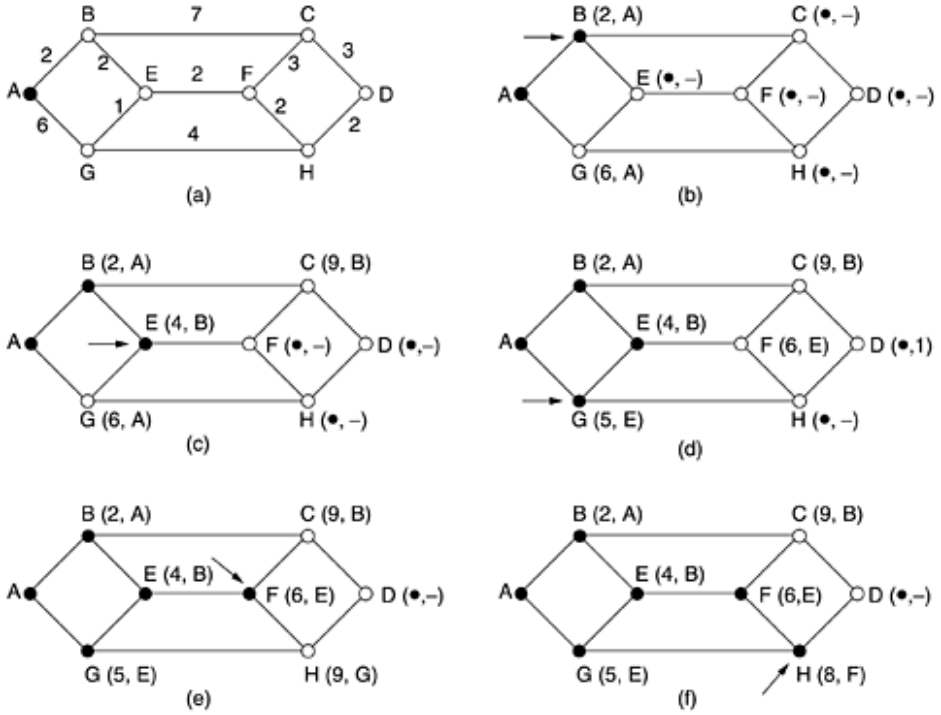
Adaptive algorithms, in contrast, change their routing decisions to reflect changes in the topology, and usually the traffic as well. Adaptive algorithms differ in where they get their information (e.g., locally, from adjacent routers, or from all routers), when they change the routes (e.g., every ΔT sec, when the load changes or when the topology changes), and what metric is used for optimization (e.g., distance, number of hops, or estimated transit time).

10.4 shortest path routing

Let us begin our study of feasible routing algorithms with a technique that is widely used in many forms because it is simple and easy to understand. The idea is to build a graph of the subnet, with each node of the graph representing a router and each arc of the graph representing a communication line (often called a 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 the next diagram 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).

However, many other metrics besides hops and physical distance are also possible. For example, each arc could be labeled with the mean queueing and transmission delay for some standard test packet as determined by hourly test runs. With this graph labeling, the shortest path is the fastest path rather than the path with the fewest arcs or kilometers.

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



In the general case, the labels on the arcs could be computed as a function of the distance, bandwidth, average traffic, communication cost, mean queue length, 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). Each node is labeled (in parentheses) with its distance from the source node along the best known path. 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 part (a), 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. 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 part (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. The diagram hows the first five steps of the algorithm.

To see why the algorithm works, look at part (c). At that point we have just made *E* permanent. Suppose that there were a shorter path than *ABE*, say *AXYZE*. 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. Either the label at *Z* is greater than or equal to that at *E*, in which case *AXYZE* cannot be a shorter path than *ABE*, or it is less than that of *E*, in which case

Z and not *E* will become permanent first, allowing *E* to be probed from *Z*.

10.5 flooding

Another static algorithm 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, which 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 subnet.

An alternative technique for damming the flood is to keep track of which packets have been flooded, to avoid sending them out a second time. 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.

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 is a duplicate; if so, it is discarded. Furthermore, the full list below *k* is not needed, since *k* effectively summarizes it.

A variation of flooding that is slightly more practical is **selective flooding**. In this algorithm the routers do not send every incoming packet out on every line, only on those lines that are going approximately in the right direction. There is usually little point in sending a westbound packet on an eastbound line unless the topology is extremely peculiar and the router is sure of this fact.

Flooding is not practical in most applications, but it does have some uses. For example, in military applications, where large numbers of routers may be blown to bits at any instant, the tremendous robustness of flooding is highly desirable. In distributed database applications, it is sometimes necessary to update all the databases concurrently, in which case flooding can be useful. 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. A fourth possible use of flooding is 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.

10.6 distance vector routing

Modern computer networks generally use dynamic routing algorithms rather than the static ones described above because static algorithms do not take the current network load into account. 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.

Distance vector routing algorithms operate by having each router maintain a table (i.e, a vector) giving the best known distance to each destination and which line to use to get there. These tables are updated by exchanging information with the neighbors.

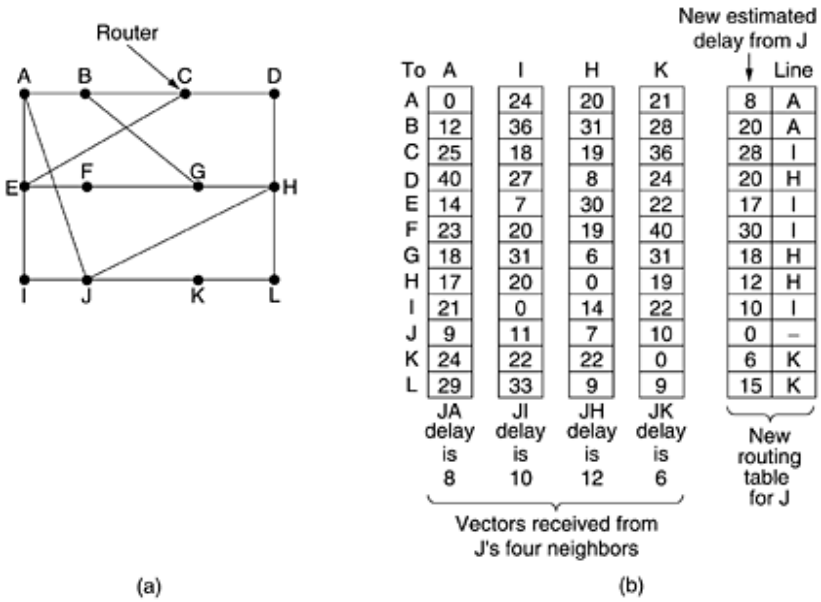
The distance vector routing algorithm is sometimes called by other names, most commonly the distributed Bellman-Ford routing algorithm and the Ford-Fulkerson algorithm. It was the original ARPANET routing algorithm and was also used in the Internet under the name RIP.

In distance vector routing, each router maintains a routing table indexed by, and containing one entry for, each router in the subnet. This entry contains two parts: the preferred outgoing line to use for that destination and an estimate of the time or distance to that destination. 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 queue length, the router simply examines each queue. If the metric is 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* msec 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 X_i 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* msec, it also knows that it can reach router *i* via *X* in $X_i + m$ 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 line in its new routing table. Note that the old routing table is not used in the calculation.

This updating process is illustrated below. Part (a) shows a subnet. 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 msec, respectively.



Consider how *J* computes its new route to router *G*. It knows that it can get to *A* in 8 msec, and *A* claims to be able to get to *G* in 18 msec, so *J* knows it can count on a delay of 26 msec 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) msec, respectively. The best of these values is 18, so it makes an entry in its routing table that the delay to *G* is 18 msec 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.

10.7 link state routing

Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing. Two primary problems caused its demise. First, since the delay metric was queue length, it did not take line bandwidth into account when choosing routes. Initially, all the lines were 56 kbps, so line bandwidth was not an issue, but after some lines had been upgraded to 230 kbps and others to 1.544 Mbps, not taking bandwidth into account was a major problem. Of course, it would have been possible to change the delay metric to factor in line bandwidth, but a second problem also existed, namely, the algorithm often took too long to converge (the count-to-infinity problem). For these reasons, it was replaced by an entirely new

algorithm, now called **link state routing**. Variants of link state routing are now widely used.

The idea behind link state routing can be stated as five parts. Each router must do the following:

1. Discover its neighbors and learn their network addresses.
2. Measure the delay or cost to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to all other routers.
5. Compute the shortest path to every other router.

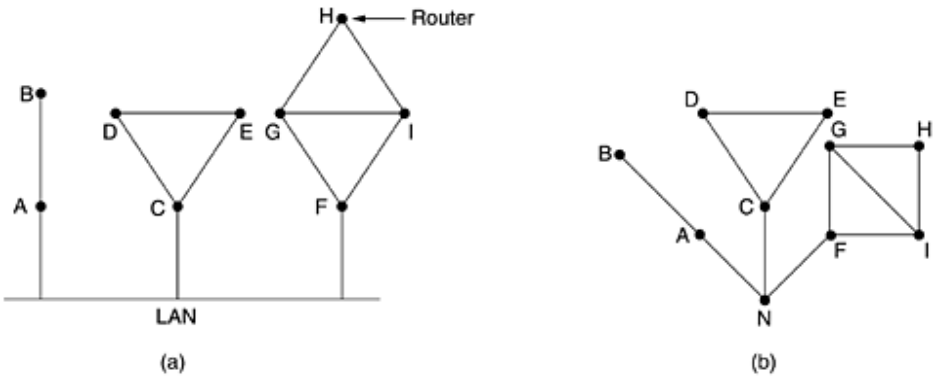
In effect, the complete topology and all delays are experimentally measured and distributed to every router. Then Dijkstra's algorithm can be run to find the shortest path to every other router. Below we will consider each of these five steps in more detail.

10.7.1 learning about the neighbors

When a router is booted, its first task is to learn who its neighbors are. It accomplishes this goal by sending a special HELLO packet on each point-to-point line. The router on the other end is expected to send back a reply telling who it is. These names must be globally unique because when a distant router later hears that three routers are all connected to F, it is essential that it knows whether all mean the same F.

When two or more routers are connected by a LAN, the situation is slightly more complicated. Part (a) illustrates a LAN to which three routers, A, C, and F, are directly connected. Each of these routers is connected to one or more additional routers, as shown.

(a) Nine routers and a LAN. (b) A graph model of (a)



One way to model the LAN is to consider it as a node itself, as shown in part (b). Here we have introduced a new, artificial node, *N*, to which *A*, *C*, and *F* are connected. The fact that it is possible to go from *A* to *C* on the LAN is represented by the path *ANC* here.

10.7.2 measuring line cost

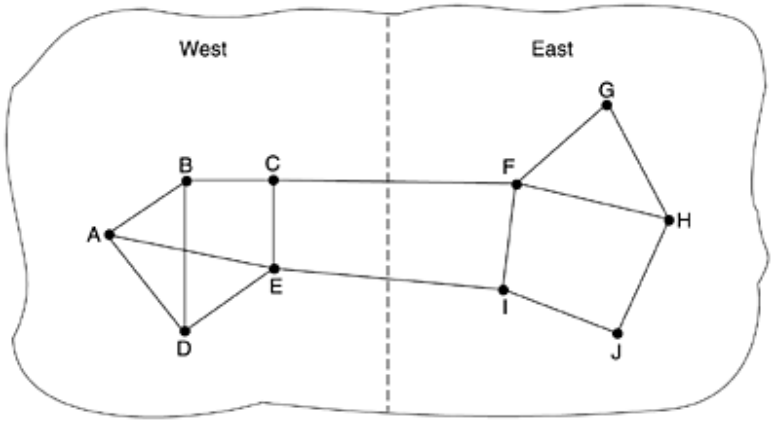
The link state routing algorithm requires each router to know, or at least have a reasonable estimate of, the delay to each of its neighbors. The most direct way to determine this delay is to send over the line a special ECHO packet that the other side is required to send back immediately. By measuring the round-trip time and dividing it by two, the sending router can get a reasonable estimate of the delay. For even better results, the test can be conducted several times, and the average used. Of course, this method implicitly

assumes the delays are symmetric, which may not always be the case.

An interesting issue is whether to take the load into account when measuring the delay. To factor the load in, the round-trip timer must be started when the ECHO packet is queued. To ignore the load, the timer should be started when the ECHO packet reaches the front of the queue. Arguments can be made both ways. Including traffic-induced delays in the measurements means that when a router has a choice between two lines with the same bandwidth, one of which is heavily loaded all the time and one of which is not, the router will regard the route over the unloaded line as a shorter path. This choice will result in better performance.

Unfortunately, there is also an argument against including the load in the delay calculation. Consider the subnet below, which is divided into two parts, East and West, connected by two lines, *CF* and *EI*.

A subnet in which the East and West parts are connected by two lines

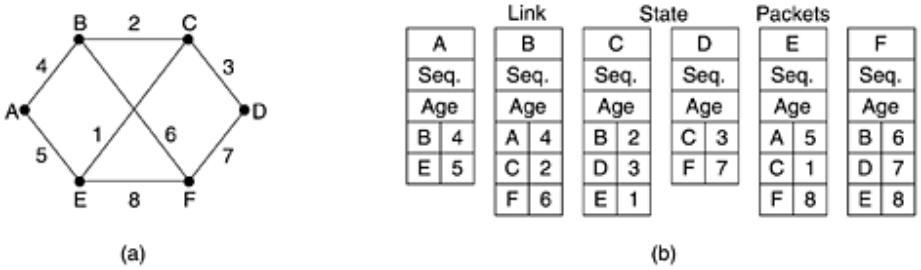


Suppose that most of the traffic between East and West is using line *CF*, and as a result, this line is heavily loaded with long delays. Including queueing delay in the shortest path calculation will make *EI* more attractive. After the new routing tables have been installed, most of the East-West traffic will now go over *EI*, overloading this line. 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. If load is ignored and only bandwidth is considered, this problem does not occur. Alternatively, the load can be spread over both lines, but this solution does not fully utilize the best path. To avoid oscillations in the choice of best path, it may be wise to distribute the load over multiple lines, with some known fraction going over each line.

10.7.3 building link state packets

Once the information needed for the exchange has been collected, the next step is for each router to build a packet containing all the data. The packet starts with the identity of the sender, followed by a sequence number and age (to be described later), and a list of neighbors. For each neighbor, the delay to that neighbor is given.

(a) A subnet (b) The link state packets for this subnet



An example subnet is given part (a) with delays shown as labels on the lines. The corresponding link state packets for all six routers are shown in part (b).

Building the link state packets is easy. The hard part is determining when to build them. One possibility is to build them periodically, that is, at regular intervals. Another possibility is to build them when some significant event occurs, such as a line or neighbor going down or coming back up again or changing its properties appreciably.

10.7.4 distributing the link state packets

The trickiest part of the algorithm is distributing the link state packets reliably. As the packets are distributed and installed, the routers getting the first ones will change their routes. Consequently, the different routers may be using different versions of the topology, which can lead to inconsistencies, loops, unreachable machines, and other problems.

First we will describe the basic distribution algorithm. The idea is to use flooding to distribute the link state packets. To keep the flood in check, each packet contains a sequence number that is incremented for each new packet sent. Routers keep track of all the (source router, sequence) pairs they see. When a new link state packet comes in, it is checked against the list of packets already seen. If it is new, it is forwarded on all lines except the one it arrived on. If it is a duplicate, it is discarded. If a packet with a sequence number lower than the highest one seen so far ever arrives, it is rejected as being obsolete.

This algorithm has a few problems, but they are manageable. First, if the sequence numbers wrap around, confusion will reign. The solution here is to use a 32-bit sequence number. With one link state packet per second, it would take 137 years to wrap around, so this possibility can be ignored.

Second, if a router ever crashes, it will lose track of its sequence number. If it starts again at 0, the next packet will be rejected as a duplicate.

Third, if a sequence number is ever corrupted and 65,540 is received instead of 4 (a 1-bit error), packets 5 through 65,540 will be rejected as obsolete, since the current sequence number is thought to be 65,540.

The solution to all these problems is to include the age of each packet after the sequence number and decrement it once per second. When the age hits zero, the information from that router is discarded. Normally, a new packet comes in, say, every 10 sec, so router information only times out when a router is down (or six consecutive packets have been lost, an unlikely event). The *Age* field is also decremented by each router during the initial flooding process, to make sure no packet can get lost and live for an indefinite period of time (a packet whose age is zero is discarded).

Some refinements to this algorithm make it more robust. When a link state packet comes in to a router for flooding, it is not queued for transmission immediately. Instead it is first put in a holding area to wait a short while. If another link state packet from the same source comes in before the first packet is transmitted, their sequence numbers are compared. If they are equal, the duplicate is discarded. If they are different, the older one is thrown out. To guard against errors on the router-router lines, all link state packets are acknowledged. When a line goes idle, the holding area is scanned in round-robin order to select a packet or acknowledgment to send.

The data structure used by router *B* for the subnet shown above is depicted below.

The packet buffer for router B

			Send flags			ACK flags			
Source	Seq.	Age	A	C	F	A	C	F	Data
A	21	60	0	1	1	1	0	0	
F	21	60	1	1	0	0	0	1	
E	21	59	0	1	0	1	0	1	
C	20	60	1	0	1	0	1	0	
D	21	59	1	0	0	0	1	1	

Each row here corresponds to a recently-arrived, but as yet not fully-processed, link state packet. The table records where the packet originated, its sequence number and age, and the data. In addition, there are send and acknowledgment flags for each of *B*'s three lines (to *A*, *C*, and *F*, respectively). The send flags mean that the packet must be sent on the indicated line. The acknowledgment flags mean that it must be acknowledged there.

In the picture, the link state packet from *A* arrives directly, so it must be sent to *C* and *F* and acknowledged to *A*, as indicated by the flag bits. Similarly, the packet from *F* has to be forwarded to *A* and *C* and acknowledged to *F*. However, the situation with the third packet, from *E*, is different. It arrived twice, once via *EAB* and once via *EFB*. Consequently, it has to be sent only to *C* but acknowledged to both *A* and *F*, as indicated by the bits.

If a duplicate arrives while the original is still in the buffer, bits have to be changed. For example, if a copy of *C*'s state arrives from *F* before the fourth entry in the table has been forwarded, the six bits will be changed to 100011 to indicate that the packet must be acknowledged to *F* but not sent there.

10.7.5 computing the new routes

Once a router has accumulated a full set of link state packets, it can construct the entire subnet graph because every link is represented. Every link is, in fact, represented twice, once for each direction. The two values can be averaged or used separately.

Now Dijkstra's algorithm can be run locally to construct the shortest path to all possible destinations. The results of this algorithm can be installed in the routing tables, and normal operation resumed.

For a subnet with *n* routers, each of which has *k* neighbors, the memory required to store the input data is proportional to *kn*. For large subnets, this can be a problem. Also, the computation time can be an issue. Nevertheless, in many practical situations, link state routing works well.

However, problems with the hardware or software can wreak havoc with this algorithm (also with other ones). For example, if a router claims to have a line it does not have or forgets a line it does have, the subnet graph will be incorrect. If a router fails to forward packets or corrupts them while forwarding them, trouble will arise. Finally, if it runs out of memory or does the routing calculation wrong, bad things will happen. As the subnet grows into the range of tens or hundreds of thousands of nodes, the probability of some router failing occasionally becomes nonnegligible. The trick is to try to arrange to limit the damage when the inevitable happens. Perlman (1988) discusses these problems and their solutions in detail.

Link state routing is widely used in actual networks, so a few words about some example protocols using it are in order. The **OSPF** (**Open Shortest Path First**) protocol, which is widely used in the Internet, uses a link state algorithm.

Another link state protocol is **IS-IS** (**Intermediate System-Intermediate System**), which was designed for DECnet and later adopted by ISO for use with its connectionless network layer protocol, CLNP. Since

then it has been modified to handle other protocols as well, most notably, IP. IS-IS is used in some Internet backbones (including the old NSFNET backbone) and in some digital cellular systems such as CDPD. Novell NetWare uses a minor variant of IS-IS (NLSP) for routing IPX packets.

Basically IS-IS distributes a picture of the router topology, from which the shortest paths are computed. Each router announces, in its link state information, which network layer addresses it can reach directly. These addresses can be IP, IPX, AppleTalk, or any other addresses. IS-IS can even support multiple network layer protocols at the same time.

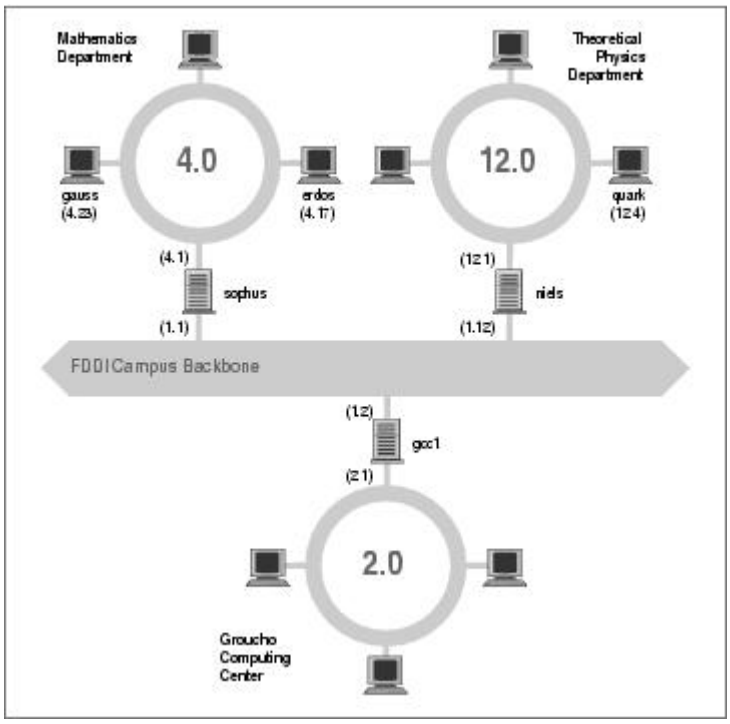
Many of the innovations designed for IS-IS were adopted by OSPF (OSPF was designed several years after IS-IS). These include a self-stabilizing method of flooding link state updates, the concept of a designated router on a LAN, and the method of computing and supporting path splitting and multiple metrics. As a consequence, there is very little difference between IS-IS and OSPF. The most important difference is that IS-IS is encoded in such a way that it is easy and natural to simultaneously carry information about multiple network layer protocols, a feature OSPF does not have.

10.8 gateways

Subnetting is not only a benefit to the organization; it is frequently a natural consequence of hardware boundaries. The viewpoint of a host on a given physical network, such as an Ethernet, is a very limited one: it can only talk to the host of the network it is on. All other hosts can be accessed only through special-purpose machines called *gateways*. A **gateway** is a host that is connected to two or more physical networks simultaneously and is configured to switch packets between them.

The figure below shows part of the network topology at Groucho Marx University (GMU). Hosts that are on two subnets at the same time are shown with both addresses.

A part of the net topology at Groucho Marx University



Different physical networks have to belong to different IP networks for IP to be able to recognize if a host is on a local network. For example, the network number 149.76.4.0 is reserved for hosts on the mathematics LAN. When sending a datagram to quark, the network software on erdos immediately sees from the IP address 149.76.12.4 that the destination host is on a different physical network, and therefore can be reached only through a gateway (sophus by default).

sophus itself is connected to two distinct subnets: the Mathematics department and the campus backbone. It accesses each through a different interface, `eth0` and `fddi0`, respectively. Now, what IP address do we assign it? Should we give it one on subnet 149.76.1.0, or on 149.76.4.0?

The answer is: “both.” sophus has been assigned the address 149.76.1.1 for use on the 149.76.1.0 network and address 149.76.4.1 for use on the 149.76.4.0 network. A gateway must be assigned one IP address for each network it belongs to. These addresses - along with the corresponding netmask - are tied to the interface through which the subnet is accessed. Thus, the interface and address mapping for sophus would look like this:

Interface	*Address*	*Netmask*
`eth0`	149.76.4.1	255.255.255.0
`fddi0`	149.76.1.1	255.255.255.0
`lo`	127.0.0.1	255.0.0.0

The last entry describes the loopback interface `lo`.

Generally, you can ignore the subtle difference between attaching an address to a host or its interface. For hosts that are on one network only, like erdos, you would generally refer to the host as having this-and-that IP address, although strictly speaking, it's the Ethernet interface that has this IP address. The distinction is really important only when you refer to a gateway.

10.9 routing tables

We now focus our attention on how IP chooses a gateway to use to deliver a datagram to a remote network.

We have seen that erdos, when given a datagram for quark, checks the destination address and finds that it is not on the local network. erdos therefore sends the datagram to the default gateway sophus, which is now faced with the same task. sophus recognizes that quark is not on any of the networks it is connected to directly, so it has to find yet another gateway to forward it through. The correct choice would be niels, the gateway to the Physics department. sophus thus needs information to associate a destination network with a suitable gateway.

IP uses a table for this task that associates networks with the gateways by which they may be reached. A catch-all entry (the *default route*) must generally be supplied too; this is the gateway associated with network 0.0.0.0. All destination addresses match this route, since none of the 32 bits are required to match, and therefore packets to an unknown network are sent through the default route. On sophus, the table might look like this:

Network	*Netmask*	*Gateway*	*Interface*
149.76.1.0	255.255.255.0	-	`fddi0`
149.76.2.0	255.255.255.0	149.76.1.2	`fddi0`
149.76.3.0	255.255.255.0	149.76.1.3	`fddi0`
149.76.4.0	255.255.255.0	-	`eth0`
149.76.5.0	255.255.255.0	149.76.1.5	`fddi0`
…	…	…	…
0.0.0.0	0.0.0.0	149.76.1.2	`fddi0`

If you need to use a route to a network that sophus is directly connected to, you don't need a gateway; the gateway column here contains a hyphen.

The process for identifying whether a particular destination address matches a route is a mathematical operation. The process is quite simple, but it requires an understanding of binary arithmetic and logic: A route matches a destination if the network address logically ANDed with the netmask precisely equals the destination address logically ANDed with the netmask.

Translation: a route matches if the number of bits of the network address specified by the netmask (starting from the left-most bit, the high order bit of byte one of the address) match that same number of bits in the destination address.

When the IP implementation is searching for the best route to a destination, it may find a number of routing entries that match the target address. For example, we know that the default route matches every destination, but datagrams destined for locally attached networks will match their local route, too. How does IP know which route to use? It is here that the netmask plays an important role. While both routes match the destination, one of the routes has a larger netmask than the other. We previously mentioned that the netmask was used to break up our address space into smaller networks. The larger a netmask is, the more specifically a target address is matched; when routing datagrams, we should always choose the route that has the largest netmask. The default route has a netmask of zero bits, and in the configuration presented above, the locally attached networks have a 24-bit netmask. If a datagram matches a locally attached network, it will be routed to the appropriate device in preference to following the default route because the local network route matches with a greater number of bits. The only datagrams that will be routed via the default route are those that don't match any other route.

You can build routing tables by a variety of means. For small LANs, it is usually most efficient to construct them by hand and feed them to IP using the **route** command at boot time. For larger networks, they are built and adjusted at runtime by *routing daemons*; these daemons run on central hosts of the network and exchange routing information to compute “optimal” routes between the member networks.

Depending on the size of the network, you'll need to use different routing protocols. For routing inside autonomous systems (such as the Groucho Marx campus), the *internal routing protocols* are used. The most prominent one of these is the *Routing Information Protocol* (RIP), which is implemented by the BSD **routed** daemon. For routing between autonomous systems, *external routing protocols* like *External Gateway Protocol* (EGP) or *Border Gateway Protocol* (BGP) have to be used; these protocols, including RIP, have been implemented in the University of Cornell's **gated** daemon.

10.10 traceroute

Traceroute is the program that shows you the route over the network between two systems, listing all the intermediate routers a connection must pass through to get to its destination. It can help you determine why your connections to a given server might be poor, and can often help you figure out where exactly the problem is. It also shows you how systems are connected to each other, letting you see how your ISP connects to the Internet as well as how the target system is connected.

10.10.1 running traceroute or its variants

The *traceroute* program is available on most computers which support networking, including most Unix systems, Mac OS X, and Windows 95 and later.

On a Unix system, including Mac OS X, run a traceroute at the command line like this:

```
traceroute server.name
```

If the traceroute command is not found, it may be present but not in your shell's search path. On some

systems, traceroute can be found in /usr/sbin, which is often not in the default user path. In this case, run it with the full path:

```
/usr/sbin/traceroute server.name
```

On Mac OS X, if you would rather not open a terminal and use the command line, a GUI front-end for traceroute (and several other utilities) called *Network Utility* can be found in the Utilities folder within the Applications folder. Run it, click the “Traceroute” tab, and enter an address to run a trace to.

MTR is an alternate implementation of traceroute for Unix. It combines a trace with continuing pings of each hop to provide a more complete report all at once. It is available here.

If you're stuck with Windows, the command is called *tracert*. Open a DOS window and enter the command:

```
tracert server.name
```

You can also download VisualRoute, a graphical traceroute program available for Windows, Sparc Solaris, and Linux. VisualRoute helps you analyze the traceroute, and provides a nifty world map showing you where your packets are going (it's not always geographically accurate).

10.10.2 the output

Here is some example traceroute output, from a Unix system:

```
traceroute to library.airnews.net (206.66.12.202), 30 hops max, 40 byte packets
 1  rbrt3 (208.225.64.50)  4.867 ms  4.893 ms  3.449 ms
 2  519.Hssi2-0-0.GW1.EWR1.ALTER.NET (157.130.0.17) 6.918 ms 8.721 ms 16.476 ms
 3  113.ATM3-0.XR2.EWR1.ALTER.NET (146.188.176.38)  6.323 ms 6.123 ms  7.011 ms
 4  192.ATM2-0.TR2.EWR1.ALTER.NET (146.188.176.82)  6.955 ms 15.400 ms 6.684 ms
 5  105.ATM6-0.TR2.DFW4.ALTER.NET (146.188.136.245)49.105 ms 49.92 ms 47.371 ms
 6  298.ATM7-0.XR2.DFW4.ALTER.NET (146.188.240.77) 48.162 ms 48.05 ms 47.565 ms
 7  194.ATM9-0.GW1.DFW1.ALTER.NET (146.188.240.45) 47.886 ms 47.38 ms 50.690 ms
 8  iadfw3-gw.customer.ALTER.NET  (137.39.138.74)  69.827 ms 68.11 ms 66.859 ms
 9  library.airnews.net           (206.66.12.202) 174.853 ms 163.9 ms 147.50 ms
```

Here, I am tracing the route to library.airnews.net, the news server name at Airnews. The first line of output is information about what I'm doing; it shows the target system, that system's IP address, the maximum number of hops that will be allowed, and the size of the packets being sent.

Then we have one line for each system or router in the path between me and the target system. Each line shows the name of the system (as determined from DNS), the system's IP address, and three *round trip times* in milliseconds. The round trip times (or RTTs) tell us how long it took a packet to get from me to that system and back again, called the *latency* between the two systems. By default, three packets are sent to each system along the route, so we get three RTTs.

Sometimes, a line in the output may have one or two of the times missing, with an asterisk where it should be:

```
 9  host230-142.uuweb.com (208.229.230.142)  12.619 ms * *
```

In this case, the machine is up and responding, but for whatever reason it did not respond to the second and third packets. This does not necessarily indicate a problem; in fact, it is usually normal, and just means that the system discarded the packet for some reason. Many systems do this normally. These are most often computers, rather than dedicated routers. Systems running Solaris routinely show an asterisk instead of the second RTT.

It's important to remember that timeouts are *not* necessarily an indication of packet loss. This is a

common misconception, but since there are only three probes, dropping one response is no big deal.

Sometimes you will see an entry with just an IP address and no name:

```
 1  207.126.101.2 (207.126.101.2)  0.858 ms  1.003 ms  1.152 ms
```

This simply means that a reverse DNS lookup on the address failed, so the name of the system could not be determined.

If your trace ends in all timeouts, like this:

```
12  al-fa3-0-0.austtx.ixcis.net (216.140.128.242) 84.585 ms 92.399 ms 87.805 ms
13  * * *
14  * * *
15  * * *
```

This means that the target system could not be reached. More accurately, it means that the packets could not make it there and back; they may actually be reaching the target system but encountering problems on the return trip (more on this later). This is possibly due to some kind of problem, but it may also be an intentional block due to a firewall or other security measures, and the block may affect traceroute but not actual server connections.

A trace can end with one of several error indications indicating why the trace cannot proceed. In this example, the router is indicating that it has no route to the target host:

```
 4  rbrt3.exit109.com (208.225.64.50)  35.931 ms !H *  39.970 ms !H
```

The **!H** is a “host unreachable” error message (it indicates that an ICMP error message was received). The trace will stop at this point. Possible ICMP error messages of this nature include:

!H - Host unreachable. The router has no route to the target system.

!N - Network unreachable.

!P - Protocol unreachable.

!S - Source route failed. You tried to use source routing, but the router is configured to block source-routed packets.

!F - Fragmentation needed. This indicates that the router is misconfigured.

!X - Communication administratively prohibited. The network administrator has blocked traceroute at this router.

Sometimes, with some versions of traceroute, you will see TTL warnings after the times:

```
 6  qwest-nyc-oc12.above.net (208.185.156.26)  90.0 ms (ttl=251!)  90.0 ms
(ttl=251!)  90.0 ms (ttl=251!)
```

This merely indicates that the TTL (time-to-live) value on the reply packet was different from what was expected. This probably means that your route is asymmetric (see below). This is not shown by all versions of traceroute, and can be safely ignored.

The output of the Windows version of traceroute is slightly different from the Unix examples (I have censored my router's name and IP address from the listing):

```
Tracing route to news-east.usenetserver.com [63.211.125.90]
over a maximum of 30 hops:

  1     3 ms     3 ms     2 ms  my.router [xxx.xxx.xx.xxx]
  2    35 ms    36 ms    35 ms  rbtserv5.exit109.com [208.225.64.56]
```

```
  3    36 ms    37 ms    36 ms  rbrt3.exit109.com [208.225.64.50]
  4    41 ms    40 ms    41 ms  571.Hssi5-0.GW1.EWR1.ALTER.NET [157.130.3.205]
  5    42 ms    44 ms    52 ms  113.ATM2-0.XR1.EWR1.ALTER.NET [146.188.176.34]
  6    43 ms    41 ms    41 ms  193.at-1-0-0.XR1.NYC9.ALTER.NET [152.63.17.218]
  7    61 ms    41 ms    41 ms  181.ATM6-0.BR2.NYC9.ALTER.NET [152.63.22.225]
  8    41 ms    42 ms    47 ms  137.39.52.10
  9    47 ms    42 ms    42 ms  so-6-0-0.mp2.NewYork1.level3.net [209.247.10.45]
 10    65 ms    63 ms    68 ms  loopback0.hsipaccess1.Atlanta1.Level3.net
[209.244.3.2]
 11   104 ms    68 ms    80 ms  news-east.usenetserver.com [63.211.125.90]

Trace complete.
```

The Windows version does not show ICMP error messages in the manner described above. Errors are shown as (possibly ambiguous or confusing) text. For example, a “host unreachable” error will be shown as “Destination net unreachable” on Windows.

10.10.3 the reverse route

Any connection over the Internet actually depends on two routes: the route from your system to the server, and the route from that server back to your system. These routes may be (and often are) completely different (asymmetric). If they differ, a problem in your connection could be a problem with either the route to the server, or with the route back from the server. A problem reflected in a traceroute output may actually not lie with the obvious system in your trace; it may rather be with some other system on the reverse route back *from* the system that looks, from the trace, to be the cause of the problem.

So a traceroute from you to the server is only showing you half of the picture. The other half is the *return route* or *reverse route.* So how can you see that route?

In the good old days, you could use *source routing* with traceroute to see the reverse trace back to you from a host. The idea is to specify what is called a *loose source route,* which specifies a system your packets should pass through before proceeding on to their destination.

The ability to use loose source routing to see the reverse route could be pretty handy. Unfortunately, source routing has a great potential for abuse, and therefore most network administrators block all source-routed packets at their border routers. So, in practice, loose source routes aren't going to work.

These days, the only hope you likely have of running a reverse traceroute is if the system you want to trace from has a traceroute facility on their web site. Many systems, and Usenet providers in particular, have a web page where you can run a traceroute from their system back to yours. In combination with your trace to their system, this can give you the other half of the picture. I have a list of Usenet provider traceroute pages here.

10.10.4 how does it work?

You don't need to worry about the low-level details of how traceroute works in order to use it. But, if you're interested, here they are.

Traceroute works by causing each router along a network path to return an ICMP (Internet Control Message Protocol) error message. An IP packet contains a *time-to-live* (TTL) value which specifies how long it can go on its search for a destination before being discarded. Each time a packet passes through a router, its TTL value is decremented by one; when it reaches zero, the packet is dropped, and an ICMP *Time-To-Live Exceeded* error message is returned to the sender.

The traceroute program sends its first group of packets with a TTL value of one. The first router along

the path will therefore discard the packet (its TTL is decremented to zero) and return the TTL Exceeded error. Thus, we have found the first router on the path. Packets can then be sent with a TTL of two, and then three, and so on, causing each router along the path to return an error, identifying it to us. Eventually either the final destination is reached, or the maximum value (default is 30) is reached and the traceroute ends.

At the final destination, a different error is returned. Most traceroute programs work by sending UDP datagrams to some random high-numbered port where nothing is likely to be listening. When that final system is reached, since nothing is answering on that port, an ICMP Port Unreachable error message is returned, and we are finished.

The Windows version of traceroute uses ICMP Echo Request packets (ping packets) rather than UDP datagrams. In practice, this seems to make little difference in the outcome, unless a system along the route is blocking one type of traffic but not the other.

In the unlikely even that some program happens to be listening on the UDP port that traceroute is trying to contact, the trace will fail at the last hop. You can run another trace using ICMP Echo Requests, which will probably succeed, or specify a different target port for the UDP datagrams.

A few versions of traceroute, such as the one on Solaris, allow you to choose either method (high-port UDP or ICMP echo requests).

chapter 11 the data link layer – switching

In this chapter we will study the design principles for layer 2, the data link layer. This study deals with the algorithms for achieving reliable, efficient communication between two adjacent machines at the data link layer. By adjacent, we mean that the two machines are connected by a communication channel that acts conceptually like a wire (e.g., a coaxial cable, telephone line, or point-to-point wireless channel). The essential property of a channel that makes it "wirelike" is that the bits are delivered in exactly the same order in which they are sent.

At first you might think this problem is so trivial that there is no software to study - machine A just puts the bits on the wire, and machine B just takes them off. Unfortunately, communication circuits make errors occasionally. Furthermore, they have only a finite data rate, and there is a nonzero propagation delay between the time a bit is sent and the time it is received. These limitations have important implications for the efficiency of the data transfer. The protocols used for communications must take all these factors into consideration. These protocols are the subject of this chapter.

After an introduction to the key design issues present in the data link layer, we will start our study of its protocols by looking at the nature of errors, their causes, and how they can be detected and corrected. Then we will study a series of increasingly complex protocols, each one solving more and more of the problems present in this layer. Finally, we will conclude with an examination of protocol modeling and correctness and give some examples of data link protocols.

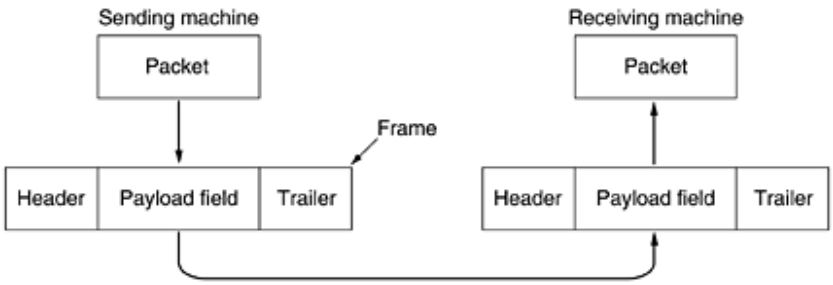
11.1 data link layer design issues

The data link layer has a number of specific functions it can carry out. These functions include:

1. Providing a well-defined service interface to the network layer.
2. Dealing with transmission errors.
3. Regulating the flow of data so that slow receivers are not swamped by fast senders.

To accomplish these goals, the data link layer takes the packets it gets from the network layer and encapsulates them into frames for transmission. Each frame contains a frame header, a payload field for holding the packet, and a frame trailer, as illustrated in below. Frame management forms the heart of what the data link layer does. In the following sections we will examine all the above-mentioned issues in detail.

Relationship between packets and frames



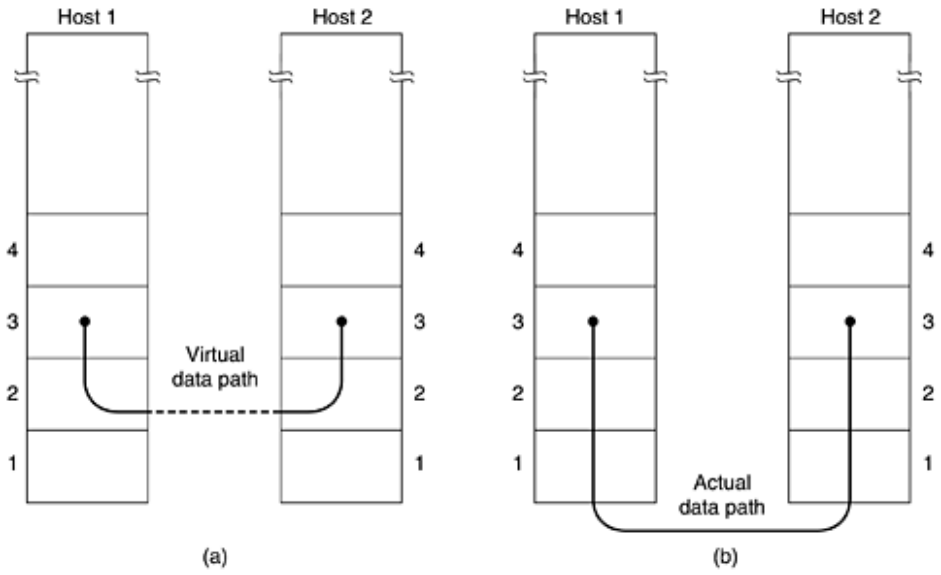
Although this chapter is explicitly about the data link layer and the data link protocols, many of the principles we will study here, such as error control and flow control, are found in transport and other protocols as well. In fact, in many networks, these functions are found only in the upper layers and not in the data link

layer. However, no matter where they are found, the principles are pretty much the same, so it does not really matter where we study them. In the data link layer they often show up in their simplest and purest forms, making this a good place to examine them in detail.

11.2 services provided to the network layer

The function of the data link layer is to provide services to the network layer. The principal service is transferring data from the network layer on the source machine to the network layer on the destination machine. On the source machine is an entity, call it a process, in the network layer that hands some bits to the data link layer for transmission to the destination. The job of the data link layer is to transmit the bits to the destination machine so they can be handed over to the network layer there, as shown in part (a). The actual transmission follows the path of part (b), but it is easier to think in terms of two data link layer processes communicating using a data link protocol. For this reason, we will implicitly use the model of part (a) throughout this chapter.

(a) Virtual communication (b) Actual communication



The data link layer can be designed to offer various services. The actual services offered can vary from system to system. Three reasonable possibilities that are commonly provided are

1. Unacknowledged connectionless service.
2. Acknowledged connectionless service.
3. Acknowledged connection-oriented service.

Let us consider each of these in turn.

Unacknowledged connectionless service consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them. No logical connection is established beforehand or released afterward. If a frame is lost due to noise on the line, no attempt is made to detect the loss or recover from it in the data link layer. This class of service is appropriate when the error rate is very low so that recovery is left to higher layers. It is also appropriate for real-time traffic, such as voice, in which late data are worse than bad data. Most LANs use unacknowledged

connectionless service in the data link layer.

The next step up in terms of reliability is **acknowledged connectionless** service. When this service is offered, there are still no logical connections used, but each frame sent is individually acknowledged. In this way, the sender knows whether a frame has arrived correctly. If it has not arrived within a specified time interval, it can be sent again. This service is useful over unreliable channels, such as wireless systems.

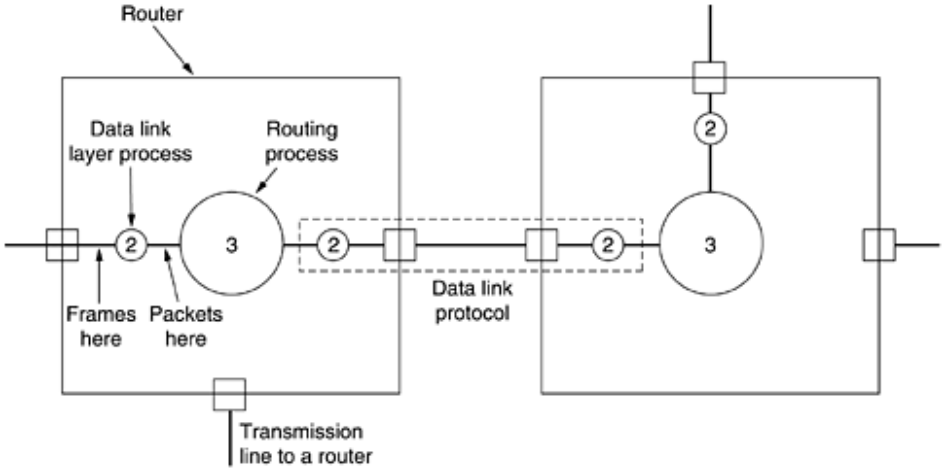
It is perhaps worth emphasizing that providing acknowledgments in the data link layer is just an optimization, never a requirement. The network layer can always send a packet and wait for it to be acknowledged. If the acknowledgment is not forthcoming before the timer expires, the sender can just send the entire message again. The trouble with this strategy is that frames usually have a strict maximum length imposed by the hardware and network layer packets do not. If the average packet is broken up into, say, 10 frames, and 20 percent of all frames are lost, it may take a very long time for the packet to get through. If individual frames are acknowledged and retransmitted, entire packets get through much faster. On reliable channels, such as fiber, the overhead of a heavyweight data link protocol may be unnecessary, but on wireless channels, with their inherent unreliability, it is well worth the cost.

Getting back to our services, the most sophisticated service the data link layer can provide to the network layer is **connection-oriented** service. With this service, the source and destination machines establish a connection before any data are transferred. Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is indeed received. Furthermore, it guarantees that each frame is received exactly once and that all frames are received in the right order. With connectionless service, in contrast, it is conceivable that a lost acknowledgment causes a packet to be sent several times and thus received several times. Connection-oriented service, in contrast, provides the network layer processes with the equivalent of a reliable bit stream.

When connection-oriented service is used, transfers go through three distinct phases. In the first phase, the connection is established by having both sides initialize variables and counters needed to keep track of which frames have been received and which ones have not. In the second phase, one or more frames are actually transmitted. In the third and final phase, the connection is released, freeing up the variables, buffers, and other resources used to maintain the connection.

Consider a typical example: a WAN subnet consisting of routers connected by point-to-point leased telephone lines. When a frame arrives at a router, the hardware checks it for errors (using techniques we will study late in this chapter), then passes the frame to the data link layer software (which might be embedded in a chip on the network interface board). The data link layer software checks to see if this is the frame expected, and if so, gives the packet contained in the payload field to the routing software. The routing software then chooses the appropriate outgoing line and passes the packet back down to the data link layer software, which then transmits it. The flow over two routers is shown below.

Placement of the data link protocol



The routing code frequently wants the job done right, that is, with reliable, sequenced connections on each of the point-to-point lines. It does not want to be bothered too often with packets that got lost on the way. It is up to the data link protocol, shown in the dotted rectangle, to make unreliable communication lines look perfect or, at least, fairly good. As an aside, although we have shown multiple copies of the data link layer software in each router, in fact, one copy handles all the lines, with different tables and data structures for each one.

11.3 framing

To provide service to the network layer, the data link layer must use the service provided to it by the physical layer. What the physical layer does is accept a raw bit stream and attempt to deliver it to the destination. This bit stream is not guaranteed to be error free. The number of bits received may be less than, equal to, or more than the number of bits transmitted, and they may have different values. It is up to the data link layer to detect and, if necessary, correct errors.

The usual approach is for the data link layer to break the bit stream up into discrete frames and compute the **checksum** for each frame. When a frame arrives at the destination, the checksum is recomputed. If the newly-computed checksum is different from the one contained in the frame, the data link layer knows that an error has occurred and takes steps to deal with it (e.g., discarding the bad frame and possibly also sending back an error report).

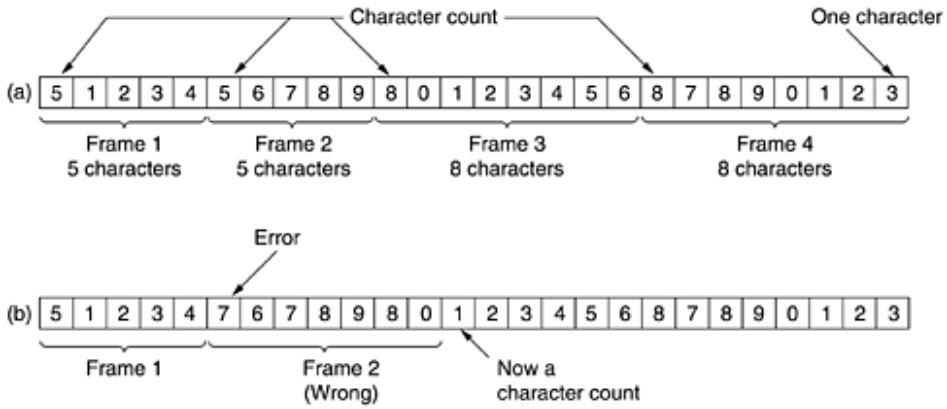
Breaking the bit stream up into frames is more difficult than it at first appears. One way to achieve this framing is to insert time gaps between frames, much like the spaces between words in ordinary text. However, networks rarely make any guarantees about timing, so it is possible these gaps might be squeezed out or other gaps might be inserted during transmission.

Since it is too risky to count on timing to mark the start and end of each frame, other methods have been devised. In this section we will look at four methods:

1. Character count.
2. Flag bytes with byte stuffing.
3. Starting and ending flags, with bit stuffing.
4. Physical layer coding violations.

The **first** framing method uses a field in the header to specify the number of characters in the frame. When the data link layer at the destination sees the character count, it knows how many characters follow and hence where the end of the frame is. This technique is shown in part (a) for four frames of sizes 5, 5, 8, and 8 characters, respectively.

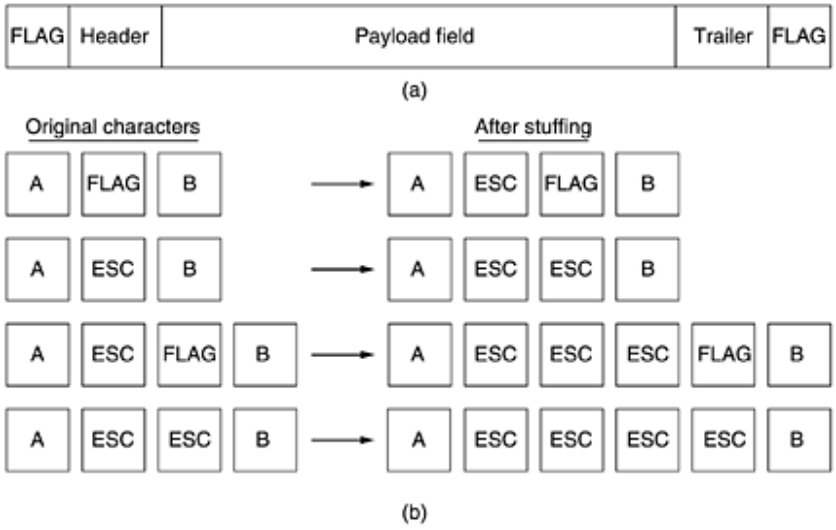
A character stream. (a) Without errors. (b) With one error



The trouble with this algorithm is that the count can be garbled by a transmission error. For example, if the character count of 5 in the second frame of part (b) becomes a 7, the destination will get out of synchronization and will be unable to locate the start of the next frame. Even if the checksum is incorrect so the destination knows that the frame is bad, it still has no way of telling where the next frame starts. Sending a frame back to the source asking for a retransmission does not help either, since the destination does not know how many characters to skip over to get to the start of the retransmission. For this reason, the character count method is rarely used anymore.

The **second** framing method gets around the problem of resynchronization after an error by having each frame start and end with **special bytes**. In the past, the starting and ending bytes were different, but in recent years most protocols have used the same byte, called a **flag byte**, as both the starting and ending delimiter, as shown in part (a) as FLAG. In this way, if the receiver ever loses synchronization, it can just search for the flag byte to find the end of the current frame. Two consecutive flag bytes indicate the end of one frame and start of the next one.

(a) A frame delimited by flag bytes. (b) Four examples of byte sequences before and after byte stuffing.



A serious problem occurs with this method when binary data, such as object programs or floating-point numbers, are being transmitted. It may easily happen that the flag byte's bit pattern occurs in the data. This situation will usually interfere with the framing. One way to solve this problem is to have the sender's data link layer insert a special escape byte (ESC) just before each "accidental" flag byte in the data. The data link layer on the receiving end removes the escape byte before the data are given to the network layer. This technique is called byte stuffing or character stuffing. Thus, a framing flag byte can be distinguished from one

in the data by the absence or presence of an escape byte before it.

Of course, the next question is: What happens if an escape byte occurs in the middle of the data? The answer is that it, too, is stuffed with an escape byte. Thus, any single escape byte is part of an escape sequence, whereas a doubled one indicates that a single escape occurred naturally in the data. Some examples are shown above, in part (b). In all cases, the byte sequence delivered after destuffing is exactly the same as the original byte sequence.

The byte-stuffing scheme depicted above is a slight simplification of the one used in the PPP protocol that most home computers use to communicate with their Internet service provider. We will discuss PPP later in this chapter.

A major disadvantage of using this framing method is that it is closely tied to the use of 8-bit characters. Not all character codes use 8-bit characters. For example. UNICODE uses 16-bit characters, As networks developed, the disadvantages of embedding the character code length in the framing mechanism became more and more obvious, so a new technique had to be developed to allow arbitrary sized characters.

The new technique allows data frames to contain an arbitrary number of bits and allows character codes with an arbitrary number of bits per character. It works like this. Each frame begins and ends with a special bit pattern, 01111110 (in fact, a flag byte). Whenever the sender's data link layer encounters five consecutive 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream. This bit stuffing is analogous to byte stuffing, in which an escape byte is stuffed into the outgoing character stream before a flag byte in the data.

When the receiver sees five consecutive incoming 1 bits, followed by a 0 bit, it automatically destuffs (i.e., deletes) the 0 bit. Just as byte stuffing is completely transparent to the network layer in both computers, so is bit stuffing. If the user data contain the flag pattern, 01111110, this flag is transmitted as 011111010 but stored in the receiver's memory as 01111110. Figure 3-6 gives an example of bit stuffing.

Bit stuffing. (a) The original data. (b) The data as they appear on the line. (c) The data as they are stored in the receiver's memory after destuffing

With bit stuffing, the boundary between two frames can be unambiguously recognized by the flag pattern. Thus, if the receiver loses track of where it is, all it has to do is scan the input for flag sequences, since they can only occur at frame boundaries and never within the data.

The last method of framing is only applicable to networks in which the encoding on the physical medium contains some redundancy. For example, some LANs encode 1 bit of data by using 2 physical bits. Normally, a 1 bit is a high-low pair and a 0 bit is a low-high pair. The scheme means that every data bit has a transition in the middle, making it easy for the receiver to locate the bit boundaries. The combinations high-high and low-low are not used for data but are used for delimiting frames in some protocols.

As a final note on framing, many data link protocols use a combination of a character count with one of the other methods for extra safety. When a frame arrives, the count field is used to locate the end of the frame. Only if the appropriate delimiter is present at that position and the checksum is correct is the frame accepted as valid. Otherwise, the input stream is scanned for the next delimiter.

11.4 error control

Having solved the problem of marking the start and end of each frame, we come to the next problem:

how to make sure all frames are eventually delivered to the network layer at the destination and in the proper order. Suppose that the sender just kept outputting frames without regard to whether they were arriving properly. This might be fine for unacknowledged connectionless service, but would most certainly not be fine for reliable, connection-oriented service.

The usual way to ensure reliable delivery is to provide the sender with some feedback about what is happening at the other end of the line. Typically, the protocol calls for the receiver to send back special control frames bearing positive or negative acknowledgments about the incoming frames. If the sender receives a positive acknowledgment about a frame, it knows the frame has arrived safely. On the other hand, a negative acknowledgment means that something has gone wrong, and the frame must be transmitted again.

An additional complication comes from the possibility that hardware troubles may cause a frame to vanish completely (e.g., in a noise burst). In this case, the receiver will not react at all, since it has no reason to react. It should be clear that a protocol in which the sender transmits a frame and then waits for an acknowledgment, positive or negative, will hang forever if a frame is ever lost due to, for example, malfunctioning hardware.

This possibility is dealt with by introducing **timers** into the data link layer. When the sender transmits a frame, it generally also starts a timer. The timer is set to expire after an interval long enough for the frame to reach the destination, be processed there, and have the acknowledgment propagate back to the sender. Normally, the frame will be correctly received and the acknowledgment will get back before the timer runs out, in which case the timer will be canceled.

However, if either the frame or the acknowledgment is lost, the timer will go off, alerting the sender to a potential problem. The obvious solution is to just transmit the frame again. However, when frames may be transmitted multiple times there is a danger that the receiver will accept the same frame two or more times and pass it to the network layer more than once. To prevent this from happening, it is generally necessary to assign **sequence numbers** to outgoing frames, so that the receiver can distinguish retransmissions from originals.

The whole issue of managing the timers and sequence numbers so as to ensure that each frame is ultimately passed to the network layer at the destination exactly once, no more and no less, is an important part of the data link layer's duties. Later in this chapter, we will look at a series of increasingly sophisticated examples to see how this management is done.

11.5 flow control

Another important design issue that occurs in the data link layer (and higher layers as well) is what to do with a sender that systematically wants to transmit frames faster than the receiver can accept them. This situation can easily occur when the sender is running on a fast (or lightly loaded) computer and the receiver is running on a slow (or heavily loaded) machine. The sender keeps pumping the frames out at a high rate until the receiver is completely swamped. Even if the transmission is error free, at a certain point the receiver will simply be unable to handle the frames as they arrive and will start to lose some. Clearly, something has to be done to prevent this situation.

Two approaches are commonly used. In the first one, **feedback-based flow control**, the receiver sends back information to the sender giving it permission to send more data or at least telling the sender how the receiver is doing. In the second one, **rate-based flow control**, the protocol has a built-in mechanism that limits the rate at which senders may transmit data, without using feedback from the receiver.

Various feedback-based flow control schemes are known, but most of them use the same basic principle. The protocol contains well-defined rules about when a sender may transmit the next frame. These rules often prohibit frames from being sent until the receiver has granted permission, either implicitly or explicitly. For example, when a connection is set up, the receiver might say: "You may send me n frames now, but after they have been sent, do not send any more until I have told you to continue.

11.6 error detection and correction

The telephone system has three parts: the switches, the interoffice trunks, and the local loops. The first two are now almost entirely digital in most developed countries. The local loops are still analog twisted copper pairs and will continue to be so for years due to the enormous expense of replacing them. While errors are rare on the digital part, they are still common on the local loops. Furthermore, wireless communication is becoming more common, and the error rates here are orders of magnitude worse than on the interoffice fiber trunks. The conclusion is: transmission errors are going to be with us for many years to come. We have to learn how to deal with them.

As a result of the physical processes that generate them, errors on some media (e.g., radio) tend to come in bursts rather than singly. Having the errors come in bursts has both advantages and disadvantages over isolated single-bit errors. On the advantage side, computer data are always sent in blocks of bits. Suppose that the block size is 1000 bits and the error rate is 0.001 per bit. If errors were independent, most blocks would contain an error. If the errors came in bursts of 100 however, only one or two blocks in 100 would be affected, on average. The disadvantage of burst errors is that they are much harder to correct than are isolated errors.

11.7 error-correcting codes

Network designers have developed two basic strategies for dealing with errors. One way is to include enough redundant information along with each block of data sent, to enable the receiver to deduce what the transmitted data must have been. The other way is to include only enough redundancy to allow the receiver to deduce that an error occurred, but not which error, and have it request a retransmission. The former strategy uses error-correcting codes and the latter uses error-detecting codes. The use of error-correcting codes is often referred to as forward error correction.

Each of these techniques occupies a different ecological niche. On channels that are highly reliable, such as fiber, it is cheaper to use an error detecting code and just retransmit the occasional block found to be faulty. However, on channels such as wireless links that make many errors, it is better to add enough redundancy to each block for the receiver to be able to figure out what the original block was, rather than relying on a retransmission, which itself may be in error.

To understand how errors can be handled, it is necessary to look closely at what an error really is. Normally, a frame consists of m data (i.e., message) bits and r redundant, or check, bits. Let the total length be n (i.e., n = m + r). An n-bit unit containing data and check bits is often referred to as an n-bit codeword.

Given any two codewords, say, 10001001 and 10110001, it is possible to determine how many corresponding bits differ. In this case, 3 bits differ. To determine how many bits differ, just exclusive OR the two codewords and count the number of 1 bits in the result, for example:

```
10001001
10110001
00111000
```

The number of bit positions in which two codewords differ is called the Hamming distance (Hamming, 1950). Its significance is that if two codewords are a Hamming distance d apart, it will require d single-bit errors to convert one into the other.

In most data transmission applications, all 2^m possible data messages are legal, but due to the way the check bits are computed, not all of the 2^n possible codewords are used. Given the algorithm for computing the check bits, it is possible to construct a complete list of the legal codewords, and from this list find the two codewords whose Hamming distance is minimum. This distance is the Hamming distance of the complete code.

The error-detecting and error-correcting properties of a code depend on its Hamming distance. To detect d errors, you need a distance d + 1 code because with such a code there is no way that d single-bit errors can change a valid codeword into another valid codeword. When the receiver sees an invalid codeword, it can tell that a transmission error has occurred. Similarly, to correct d errors, you need a distance 2d + 1 code because that way the legal codewords are so far apart that even with d changes, the original codeword is still closer than any other codeword, so it can be uniquely determined.

As a simple example of an error-detecting code, consider a code in which a single parity bit is appended to the data. The parity bit is chosen so that the number of 1 bits in the codeword is even (or odd). For example, when 1011010 is sent in even parity, a bit is added to the end to make it 10110100. With odd parity 1011010 becomes 10110101. A code with a single parity bit has a distance 2, since any single-bit error produces a codeword with the wrong parity. It can be used to detect single errors.

As a simple example of an error-correcting code, consider a code with only four valid codewords:

0000000000, 0000011111, 1111100000, and 1111111111

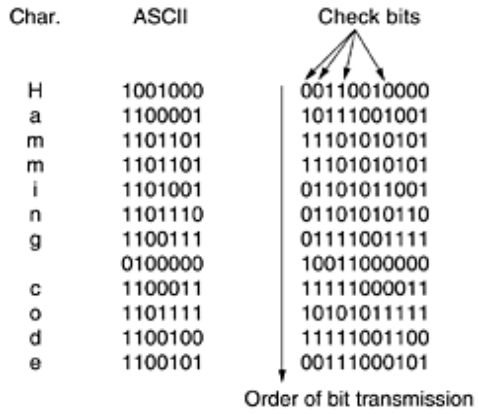
This code has a distance 5, which means that it can correct double errors. If the codeword 0000000111 arrives, the receiver knows that the original must have been 0000011111. If, however, a triple error changes 0000000000 into 0000000111, the error will not be corrected properly.

Imagine that we want to design a code with m message bits and r check bits that will allow all single errors to be corrected. Each of the 2^m legal messages has n illegal codewords at a distance 1 from it. These are formed by systematically inverting each of the n bits in the n-bit codeword formed from it. Thus, each of the 2^m legal messages requires n + 1 bit patterns dedicated to it. Since the total number of bit patterns is 2^n, we must have $(n + 1)2^m < 2^n$. Using n = m + r, this requirement becomes $(m + r + 1) < 2^r$. Given m, this puts a lower limit on the number of check bits needed to correct single errors.

This theoretical lower limit can, in fact, be achieved using a method due to Hamming (1950). The bits of the codeword are numbered consecutively, starting with bit 1 at the left end, bit 2 to its immediate right, and so on. The bits that are powers of 2 (1, 2, 4, 8, 16, etc.) are check bits. The rest (3, 5, 6, 7, 9, etc.) are filled up with the m data bits. Each check bit forces the parity of some collection of bits, including itself, to be even (or odd). A bit may be included in several parity computations. To see which check bits the data bit in position k contributes to, rewrite k as a sum of powers of 2. For example, 11 = 1 + 2 + 8 and 29 = 1 + 4 + 8 + 16. A bit is checked by just those check bits occurring in its expansion (e.g., bit 11 is checked by bits 1, 2, and 8).

When a codeword arrives, the receiver initializes a counter to zero. It then examines each check bit, k (k = 1, 2, 4, 8, ...), to see if it has the correct parity. If not, the receiver adds k to the counter. If the counter is zero after all the check bits have been examined (i.e., if they were all correct), the codeword is accepted as valid. If the counter is nonzero, it contains the number of the incorrect bit. For example, if check bits 1, 2, and 8 are in error, the inverted bit is 11, because it is the only one checked by bits 1, 2, and 8. Figure 3-7 shows some 7-bit ASCII characters encoded as 11-bit codewords using a Hamming code. Remember that the data are found in bit positions 3, 5, 6, 7, 9, 10, and 11.

Use of a Hamming code to correct burst errors



Hamming codes can only correct single errors. However, there is a trick that can be used to permit Hamming codes to correct burst errors. A sequence of k consecutive codewords are arranged as a matrix, one codeword per row. Normally, the data would be transmitted one codeword at a time, from left to right. To correct burst errors, the data should be transmitted one column at a time, starting with the leftmost column. When all k bits have been sent, the second column is sent, and so on, as indicated in Fig. 3-7. When the frame arrives at the receiver, the matrix is reconstructed, one column at a time. If a burst error of length k occurs, at most 1 bit in each of the k codewords will have been affected, but the Hamming code can correct one error per codeword, so the entire block can be restored. This method uses kr check bits to make blocks of km data bits immune to a single burst error of length k or less.

11.8 error-detecting codes

Error-correcting codes are widely used on wireless links, which are notoriously noisy and error prone when compared to copper wire or optical fibers. Without error-correcting codes, it would be hard to get anything through. However, over copper wire or fiber, the error rate is much lower, so error detection and retransmission is usually more efficient there for dealing with the occasional error.

As a simple example, consider a channel on which errors are isolated and the error rate is 10^{-6} per bit. Let the block size be 1000 bits. To provide error correction for 1000-bit blocks, 10 check bits are needed; a megabit of data would require 10,000 check bits. To merely detect a block with a single 1-bit error, one parity bit per block will suffice. Once every 1000 blocks, an extra block (1001 bits) will have to be transmitted. The total overhead for the error detection + retransmission method is only 2001 bits per megabit of data, versus 10,000 bits for a Hamming code.

If a single parity bit is added to a block and the block is badly garbled by a long burst error, the probability that the error will be detected is only 0.5, which is hardly acceptable. The odds can be improved considerably if each block to be sent is regarded as a rectangular matrix n bits wide and k bits high, as described above. A parity bit is computed separately for each column and affixed to the matrix as the last row. The matrix is then transmitted one row at a time. When the block arrives, the receiver checks all the parity bits. If any one of them is wrong, the receiver requests a retransmission of the block. Additional retransmissions are requested as needed until an entire block is received without any parity errors. This method can detect a single burst of length n, since only 1 bit per column will be changed. A burst of length n + 1 will pass undetected, however, if the first bit is inverted, the last bit is inverted, and all the other bits are correct. (A burst error does not imply that all the bits are wrong; it just implies that at least the first and last are wrong.) If the block is badly garbled by a long burst or by multiple shorter bursts, the probability that any of the n columns will have the correct parity, by accident, is 0.5, so the probability of a bad block being accepted when it should not be is 2.

11.9 data link protocols

To introduce the subject of protocols, we will begin by looking at three protocols of increasing complexity. For interested readers, a simulator for these and subsequent protocols is available via the Web (see the preface). Before we look at the protocols, it is useful to make explicit some of the assumptions underlying the model of communication. To start with, we assume that in the physical layer, data link layer, and network layer are independent processes that communicate by passing messages back and forth. In many cases, the physical and data link layer processes will be running on a processor inside a special network I/O chip and the network layer code will be running on the main CPU. However, other implementations are also possible (e.g., three processes inside a single I/O chip; or the physical and data link layers as procedures called by the network layer process). In any event, treating the three layers as separate processes makes the discussion conceptually cleaner and also serves to emphasize the independence of the layers.

Another key assumption is that machine A wants to send a long stream of data to machine B, using a reliable, connection-oriented service. Later, we will consider the case where B also wants to send data to A simultaneously. A is assumed to have an infinite supply of data ready to send and never has to wait for data to be produced. Instead, when A's data link layer asks for data, the network layer is always able to comply immediately. (This restriction, too, will be dropped later.)

We also assume that machines do not crash. That is, these protocols deal with communication errors, but not the problems caused by computers crashing and rebooting.

As far as the data link layer is concerned, the packet passed across the interface to it from the network layer is pure data, whose every bit is to be delivered to the destination's network layer. The fact that the destination's network layer may interpret part of the packet as a header is of no concern to the data link layer.

When the data link layer accepts a packet, it encapsulates the packet in a frame by adding a data link header and trailer to it (see Fig. 3-1). Thus, a frame consists of an embedded packet, some control information (in the header), and a checksum (in the trailer). The frame is then transmitted to the data link layer on the other machine. We will assume that there exist suitable library procedures to_physical_layer to send a frame and from_physical_layer to receive a frame. The transmitting hardware computes and appends the checksum (thus creating the trailer), so that the datalink layer software need not worry about it. The polynomial algorithm discussed earlier in this chapter might be used, for example.

Initially, the receiver has nothing to do. It just sits around waiting for something to happen. In the example protocols of this chapter we will indicate that the data link layer is waiting for something to happen by the procedure call wait_for_event(&event). This procedure only returns when something has happened (e.g., a frame has arrived). Upon return, the variable event tells what happened. The set of possible events differs for the various protocols to be described and will be defined separately for each protocol. Note that in a more realistic situation, the data link layer will not sit in a tight loop waiting for an event, as we have suggested, but will receive an interrupt, which will cause it to stop whatever it was doing and go handle the incoming frame. Nevertheless, for simplicity we will ignore all the details of parallel activity within the data link layer and assume that it is dedicated full time to handling just our one channel.

When a frame arrives at the receiver, the hardware computes the checksum. If the checksum is incorrect (i.e., there was a transmission error), the data link layer is so informed (event = cksum_err). If the inbound frame arrived undamaged, the data link layer is also informed (event = frame_arrival) so that it can acquire the frame for inspection using from_physical_layer. As soon as the receiving data link layer has acquired an undamaged frame, it checks the control information in the header, and if everything is all right, passes the packet portion to the network layer. Under no circumstances is a frame header ever given to a network layer.

There is a good reason why the network layer must never be given any part of the frame header: to keep the network and data link protocols completely separate. As long as the network layer knows nothing at all about the data link protocol or the frame format, these things can be changed without requiring changes to the network layer's software. Providing a rigid interface between network layer and data link layer greatly simplifies the software design because communication protocols in different layers can evolve independently.

11.10 data link sublayers

The IEEE Ethernet Data Link layer has two sublayers:

11.10.1 Media Access Control (MAC) 802.3

Defines how packets are placed on the media. Contention media access is “first come/first served” access where everyone shares the same bandwidth - hence the name. Physical addressing is defined here, as well as logical topologies. What’s a logical topology?

It’s the signal path through a physical topology. Line discipline, error notification (not correction), ordered delivery of frames, and optional flow control can also be used at this sublayer.

11.10.2 Logical Link Control (LLC) 802.2

Responsible for identifying Network layer protocols and then encapsulating them. An LLC header tells the Data Link layer what to do with a packet once a frame is received. It works like this: A host will receive a frame and look in the LLC header to find out where the packet is destined for—say, the IP protocol at the Network layer. The LLC can also provide flow control and sequencing of control bits.

The switches and bridges I talked about near the beginning of the chapter both work at the Data Link layer and filter the network using hardware (MAC) addresses.

11.11 switches and bridges at the data link layer

Layer 2 switching is considered hardware-based bridging because it uses specialized hardware called an *application-specific integrated circuit (ASIC)*. ASICs can run up to gigabit speeds with very low latency rates.

Bridges and switches read each frame as it passes through the network. The layer 2 device then puts the source hardware address in a filter table and keeps track of which port the frame was received on. This information (logged in the bridge’s or switch’s filter table) is what helps the machine determine the location of the specific sending device.

The real estate business is all about location, location, location, and it’s the same way for both layer 2 and layer 3 devices. Though both need to be able to negotiate the network, it’s crucial to remember that they’re concerned with very different parts of it. Primarily, layer 3 machines (such as routers) need to locate specific networks, whereas layer 2 machines (switches and bridges) need to eventually locate specific devices. So, networks are to routers as individual devices are to switches and bridges. And routing tables that “map” the internetwork are for routers, as filter tables that “map” individual devices are for switches and bridges.

After a filter table is built on the layer 2 device, it will only forward frames to the segment where the destination hardware address is located. If the destination device is on the same segment as the frame, the layer 2 device will block the frame from going to any other segments. If the destination is on a different segment, the frame can only be transmitted to that segment. This is called *transparent bridging*.

When a switch interface receives a frame with a destination hardware address that isn’t found in the device’s filter table, it will forward the frame to all connected segments. If the unknown device that was sent the “mystery frame” replies to this forwarding action, the switch updates its filter table regarding that device’s location. But in the event the destination address of the transmitting frame is a broadcast address, the switch will forward all broadcasts to every connected segment by default.

All devices that the broadcast is forwarded to are considered to be in the same broadcast domain. This can be a problem; layer 2 devices propagate layer 2 broadcast storms that choke performance, and the only way to stop a broadcast storm from propagating through an internetwork is with a layer 3 device—a router.

The biggest benefit of using switches instead of hubs in your internetwork is that each switch port is actually its own collision domain. (Conversely, a hub creates one large collision domain.)

But even armed with a switch, you still can't break up broadcast domains. Neither switches nor bridges will do that. They'll typically simply forward all broadcasts instead.

Another benefit of LAN switching over hub-centered implementations is that each device on every segment plugged into a switch can transmit simultaneously. At least, they can as long as there is only one host on each port and a hub isn't plugged into a switch port. As you might have guessed, hubs only allow one device per network segment to communicate at a time.

Each network segment connected to the switch must have the same type of devices attached. What this means to you and me is that you can connect an Ethernet hub into a switch port and then connect multiple Ethernet hosts into the hub, but you can't mix Token Ring hosts in with the Ethernet gang on the same segment. Mixing hosts in this manner is called *media translation*, and Cisco says you've just got to have a router around if you need to provide this service, although I have found this not to be true in reality—but remember, we're studying for the CCNA exam here, right?

chapter 12 the mac sublayer - ethernet

12.1 channel access

Networks can be divided into two categories: those using point-to-point connections and those using broadcast channels. This chapter deals with broadcast networks and their protocols.

In any broadcast network, the key issue is how to determine who gets to use the channel when there is competition for it. To make this point clearer, consider a conference call in which six people, on six different telephones, are all connected so that each one can hear and talk to all the others. It is very likely that when one of them stops speaking, two or more will start talking at once, leading to chaos. In a face-to-face meeting, chaos is avoided by external means, for example, at a meeting, people raise their hands to request permission to speak. When only a single channel is available, determining who should go next is much harder. Many protocols for solving the problem are known and form the contents of this chapter. In the literature, broadcast channels are sometimes referred to as multiaccess channels or random access channels.

The protocols used to determine who goes next on a multiaccess channel belong to a sublayer of the data link layer called the MAC (Medium Access Control) sublayer. The MAC sublayer is especially important in LANs, many of which use a multiaccess channel as the basis for communication. WANs, in contrast, use point-to-point links, except for satellite networks. Because multiaccess channels and LANs are so closely related, in this chapter we will discuss LANs in general, including a few issues that are not strictly part of the MAC sublayer.

Technically, the MAC sublayer is the bottom part of the data link layer, so logically we should have studied it before examining all the point-to-point protocols. Nevertheless, for most people, understanding protocols involving multiple parties is easier after two-party protocols are well understood. For that reason we have deviated slightly from a strict bottom-up order of presentation.

The central theme of this chapter is how to allocate a single broadcast channel among competing users. We will first look at static and dynamic schemes in general. Then we will examine a number of specific algorithms.

12.2 static channel allocation in LANs and MANs

The traditional way of allocating a single channel, such as a telephone trunk, among multiple competing users is Frequency Division Multiplexing (FDM). If there are N users, the bandwidth is divided into *N* equal-sized portions, each user being assigned one portion. Since each user has a private frequency band, there is no interference between users. When there is only a small and constant number of users, each of which has a heavy (buffered) load of traffic, FDM is a simple and efficient allocation mechanism.

However, when the number of senders is large and continuously varying or the traffic is bursty, FDM presents some problems. If the spectrum is cut up into *N* regions and fewer than *N* users are currently interested in communicating, a large piece of valuable spectrum will be wasted. If more than *N* users want to communicate, some of them will be denied permission for lack of bandwidth, even if some of the users who have been assigned a frequency band hardly ever transmit or receive anything.

However, even assuming that the number of users could somehow be held constant at *N*, dividing the single available channel into static subchannels is inherently inefficient. The basic problem is that when some users are quiescent, their bandwidth is simply lost. They are not using it, and no one else is allowed to use it either. Furthermore, in most computer systems, data traffic is extremely bursty (peak traffic to mean traffic ratios of 1000:1 are common). Consequently, most of the channels will be idle most of the time.

The poor performance of static FDM can easily be seen from a simple queueing theory calculation. Let us start with the mean time delay, T, for a channel of capacity C bps, with an arrival rate of λ frames/sec, each frame having a length drawn from an exponential probability density function with mean $1/\mu$ bits/frame. With these parameters the arrival rate is λ frames/sec and the service rate is μC frames/sec. From queueing theory it can be shown that for Poisson arrival and service times,

$$T = \frac{1}{\mu C - \lambda}$$

For example, if C is 100 Mbps, the mean frame length, $1/\mu$, is 10,000 bits, and the frame arrival rate, λ, is 5000 frames/sec, then T = 200 μsec. Note that if we ignored the queueing delay and just asked how long it takes to send a 10,000 bit frame on a 100-Mbps network, we would get the (incorrect) answer of 100 μsec. That result only holds when there is no contention for the channel.

Now let us divide the single channel into N independent subchannels, each with capacity C/N bps. The mean input rate on each of the subchannels will now be λ/N. Recomputing T we get

$$T_{\text{FDM}} = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = NT$$

The mean delay using FDM is N times worse than if all the frames were somehow magically arranged orderly in a big central queue.

Precisely the same arguments that apply to FDM also apply to time division multiplexing (TDM). Each user is statically allocated every Nth time slot. If a user does not use the allocated slot, it just lies fallow. The same holds if we split up the networks physically. Using our previous example again, if we were to replace the 100-Mbps network with 10 networks of 10 Mbps each and statically allocate each user to one of them, the mean delay would jump from 200 μsec to 2 msec.

Since none of the traditional static channel allocation methods work well with bursty traffic, we will now explore dynamic methods.

12.3 dynamic channel allocation in LANs and MANs

Before we get into the first of the many channel allocation methods to be discussed in this chapter, it is worthwhile carefully formulating the allocation problem. Underlying all the work done in this area are five key assumptions, described below.

Station Model. The model consists of N independent stations (e.g., computers, telephones, or personal communicators), each with a program or user that generates frames for transmission. Stations are sometimes called terminals. The probability of a frame being generated in an interval of length Δt is $\lambda \Delta t$, where λ is a constant (the arrival rate of new frames). Once a frame has been generated, the station is blocked and does nothing until the frame has been successfully transmitted.

Single Channel Assumption. A single channel is available for all communication. All stations can transmit on it and all can receive from it. As far as the hardware is concerned, all stations are equivalent, although protocol software may assign priorities to them.

Collision Assumption. If two frames are transmitted simultaneously, they overlap in time and the resulting signal is garbled. This event is called a collision. All stations can detect collisions. A collided frame must be transmitted again later. There are no errors other than those generated by collisions.

4a. Continuous Time. Frame transmission can begin at any instant. There is no master clock dividing time into discrete intervals.

4b. Slotted Time. Time is divided into discrete intervals (slots). Frame transmissions always begin at the start of a slot. A slot may contain 0, 1, or more frames, corresponding to an idle slot, a successful

transmission, or a collision, respectively.

5a. Carrier Sense. Stations can tell if the channel is in use before trying to use it. If the channel is sensed as busy, no station will attempt to use it until it goes idle.

5b. No Carrier Sense. Stations cannot sense the channel before trying to use it. They just go ahead and transmit. Only later can they determine whether the transmission was successful.

Some discussion of these assumptions is in order. The first one says that stations are independent and that work is generated at a constant rate. It also implicitly assumes that each station only has one program or user, so while the station is blocked, no new work is generated. More sophisticated models allow multiprogrammed stations that can generate work while a station is blocked, but the analysis of these stations is much more complex.

The single channel assumption is the heart of the model. There are no external ways to communicate. Stations cannot raise their hands to request that the teacher call on them.

The collision assumption is also basic, although in some systems (notably spread spectrum), this assumption is relaxed, with surprising results. Also, some LANs, such as token rings, pass a special token from station to station, possession of which allows the current holder to transmit a frame. But in the coming sections we will stick to the single channel with contention and collisions model.

Two alternative assumptions about time are possible. Either it is continuous (4a) or it is slotted (4b). Some systems use one and some systems use the other, so we will discuss and analyze both. For a given system, only one of them holds.

Similarly, a network can either have carrier sensing (5a) or not have it (5b). LANs generally have carrier sense. However, wireless networks cannot use it effectively because not every station may be within radio range of every other station. Stations on wired carrier sense networks can terminate their transmission prematurely if they discover that it is colliding with another transmission. Collision detection is rarely done on wireless networks, for engineering reasons.

Many algorithms for allocating a multiple access channel are known. In the following sections we will study a small sample of the more interesting ones and give some examples of their use.

12.4 ALOHA

In the 1970s, Norman Abramson and his colleagues at the University of Hawaii devised a new and elegant method to solve the channel allocation problem. Their work has been extended by many researchers since then (Abramson, 1985). Although Abramson's work, called the ALOHA system, used ground-based radio broadcasting, the basic idea is applicable to any system in which uncoordinated users are competing for the use of a single shared channel.

We will discuss two versions of ALOHA here: pure and slotted. They differ with respect to whether time is divided into discrete slots into which all frames must fit. Pure ALOHA does not require global time synchronization; slotted ALOHA does.

12.4.1 pure ALOHA

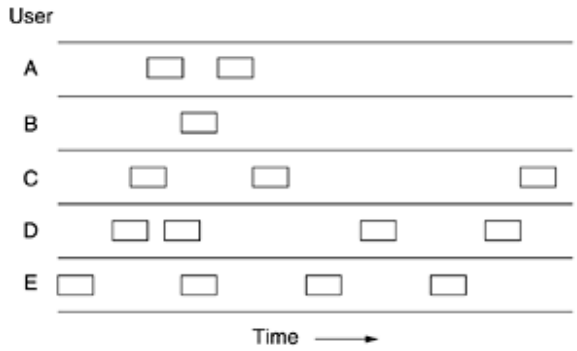
The basic idea of an ALOHA system is simple: let users transmit whenever they have data to be sent. There will be collisions, of course, and the colliding frames will be damaged. However, due to the feedback property of broadcasting, a sender can always find out whether its frame was destroyed by listening to the channel, the same way other users do. With a LAN, the feedback is immediate; with a satellite, there is a delay of 270 msec before the sender knows if the transmission was successful. If listening while transmitting is not possible for some reason, acknowledgments are needed. If the frame was destroyed, the sender just waits a random amount of time and sends it again. The waiting time must be random or the same frames will collide over and over, in lockstep. Systems in which multiple users share a common channel in a way that

can lead to conflicts are widely known as contention systems.

A sketch of frame generation in an ALOHA system is given below. We have made the frames all the same length because the throughput of ALOHA systems is maximized by having a uniform frame size rather than by allowing variable length frames.

Whenever two frames try to occupy the channel at the same time, there will be a collision and both will be garbled. If the first bit of a new frame overlaps with just the last bit of a frame almost finished, both frames will be totally destroyed and both will have to be retransmitted later. The checksum cannot (and should not) distinguish between a total loss and a near miss. Bad is bad.

In pure ALOHA, frames are transmitted at completely arbitrary times.



An interesting question is: What is the efficiency of an ALOHA channel? In other words, what fraction of all transmitted frames escape collisions under these chaotic circumstances? Let us first consider an infinite collection of interactive users sitting at their computers (stations). A user is always in one of two states: typing or waiting. Initially, all users are in the typing state. When a line is finished, the user stops typing, waiting for a response. The station then transmits a frame containing the line and checks the channel to see if it was successful. If so, the user sees the reply and goes back to typing. If not, the user continues to wait and the frame is retransmitted over and over until it has been successfully sent.

Let the "frame time" denote the amount of time needed to transmit the standard, fixed-length frame (i.e., the frame length divided by the bit rate). At this point we assume that the infinite population of users generates new frames according to a Poisson distribution with mean N frames per frame time. (The infinite-population assumption is needed to ensure that N does not decrease as users become blocked.) If $N > 1$, the user community is generating frames at a higher rate than the channel can handle, and nearly every frame will suffer a collision. For reasonable throughput we would expect $0 < N < 1$.

In addition to the new frames, the stations also generate retransmissions of frames that previously suffered collisions. Let us further assume that the probability of k transmission attempts per frame time, old and new combined, is also Poisson, with mean G per frame time. Clearly, $G \geq N$. At low load (i.e., $N \sim 0$), there will be few collisions, hence few retransmissions, so $G \approx N$. At high load there will be many collisions, so $G > N$. Under all loads, the throughput, S, is just the offered load, G, times the probability, P_0, of a transmission succeeding—that is, $S = GP_0$, where P_0 is the probability that a frame does not suffer a collision.

The probability that k frames are generated during a given frame time is given by the Poisson distribution:

Equation 4

$$\Pr[k] = \frac{G^k e^{-G}}{k!}$$

so the probability of zero frames is just e^{-G}. In an interval two frame times long, the mean number of frames generated is $2G$. The probability of no other traffic being initiated during the entire vulnerable period is thus given by $P_0 = e^{-2G}$. Using $S = GP_0$, we get

$$S = Ge^{-2G}$$

The relation between the offered traffic and the throughput is shown in the figure below. The maximum throughput occurs at $G = 0.5$, with $S = 1/2e$, which is about 0.184. In other words, the best we can hope for is a channel utilization of 18 percent. This result is not very encouraging, but with everyone transmitting at will, we could hardly have expected a 100 percent success rate.

12.4.2 slotted ALOHA

In 1972, Roberts published a method for doubling the capacity of an ALOHA system (Roberts, 1972). His proposal was to divide time into discrete intervals, each interval corresponding to one frame. This approach requires the users to agree on slot boundaries. One way to achieve synchronization would be to have one special station emit a pip at the start of each interval, like a clock.

In Roberts' method, which has come to be known as slotted ALOHA, in contrast to Abramson's pure ALOHA, a computer is not permitted to send whenever a carriage return is typed. Instead, it is required to wait for the beginning of the next slot. Thus, the continuous pure ALOHA is turned into a discrete one. Since the vulnerable period is now halved, the probability of no other traffic during the same slot as our test frame is e^{-G} which leads to

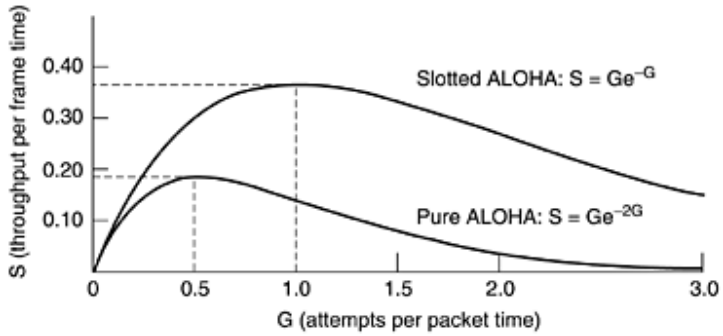
Equation 4

$$S = Ge^{-G}$$

As you can see from the figure below, slotted ALOHA peaks at G = 1, with a throughput of S =1/e or about 0.368, twice that of pure ALOHA. If the system is operating at G = 1, the probability of an empty slot is 0.368. The best we can hope for using slotted ALOHA is 37 percent of the slots empty, 37 percent successes, and 26 percent collisions. Operating at higher values of G reduces the number of empties but increases the number of collisions exponentially. To see how this rapid growth of collisions with G comes about, consider the transmission of a test frame. The probability that it will avoid a collision is e^{-G}, the probability that all the other users are silent in that slot. The probability of a collision is then just 1 - e^{-G}. The probability of a transmission requiring exactly k attempts, (i.e., k - 1 collisions followed by one success) is

$$P_k = e^{-G}(1 - e^{-G})^{k-1}$$

Throughput versus offered traffic for ALOHA systems



The expected number of transmissions, E, per carriage return typed is then

$$E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1 - e^{-G})^{k-1} = e^{G}$$

As a result of the exponential dependence of *E* upon *G*, small increases in the channel load can drastically reduce its performance.

Slotted Aloha is important for a reason that may not be initially obvious. It was devised in the 1970s, used in a few early experimental systems, then almost forgotten. When Internet access over the cable was invented, all of a sudden there was a problem of how to allocate a shared channel among multiple competing users, and slotted Aloha was pulled out of the garbage can to save the day. It has often happened that protocols that are perfectly valid fall into disuse for political reasons (e.g., some big company wants everyone to do things its way), but years later some clever person realizes that a long-discarded protocol solves his current problem. For this reason, in this chapter we will study a number of elegant protocols that are not currently in widespread use, but might easily be used in future applications, provided that enough network designers are aware of them. Of course, we will also study many protocols that are in current use as well.

12.5 carrier sense multiple access protocols

With slotted ALOHA the best channel utilization that can be achieved is 1/*e*. This is hardly surprising, since with stations transmitting at will, without paying attention to what the other stations are doing, there are bound to be many collisions. In local area networks, however, it is possible for stations to detect what other stations are doing, and adapt their behavior accordingly. These networks can achieve a much better utilization than 1/*e*. In this section we will discuss some protocols for improving performance.

Protocols in which stations listen for a carrier (i.e., a transmission) and act accordingly are called carrier sense protocols. A number of them have been proposed. Kleinrock and Tobagi (1975) have analyzed several such protocols in detail. Below we will mention several versions of the carrier sense protocols.

12.5.1 persistent and nonpersistent CSMA

The first carrier sense protocol that we will study here is called 1-persistent CSMA (Carrier Sense Multiple Access). When a station has data to send, it first listens to the channel to see if anyone else is transmitting at that moment. If the channel is busy, the station waits until it becomes idle. When the station detects an idle channel, it transmits a frame. If a collision occurs, the station waits a random amount of time and starts all over again. The protocol is called 1-persistent because the station transmits with a probability of 1 when it finds the channel idle.

The propagation delay has an important effect on the performance of the protocol. There is a small chance that just after a station begins sending, another station will become ready to send and sense the channel. If the first station's signal has not yet reached the second one, the latter will sense an idle channel and will also begin sending, resulting in a collision. The longer the propagation delay, the more important this effect becomes, and the worse the performance of the protocol.

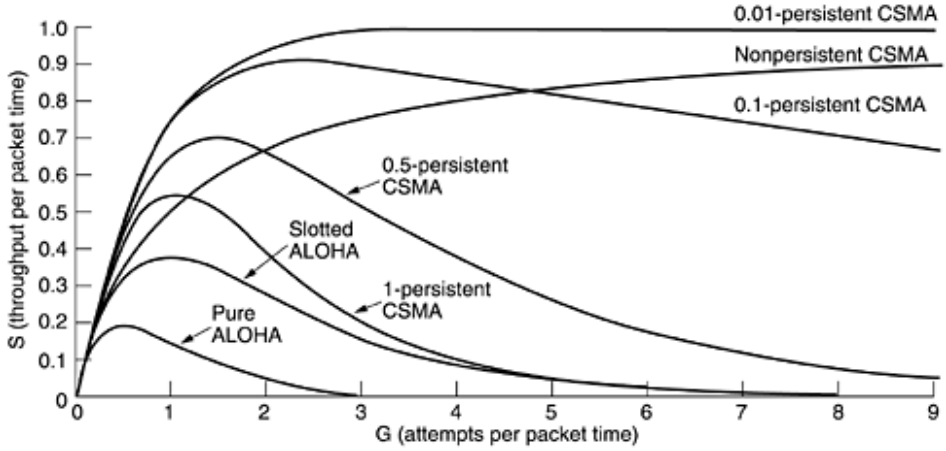
Even if the propagation delay is zero, there will still be collisions. If two stations become ready in the middle of a third station's transmission, both will wait politely until the transmission ends and then both will begin transmitting exactly simultaneously, resulting in a collision. If they were not so impatient, there would be fewer collisions. Even so, this protocol is far better than pure ALOHA because both stations have the decency to desist from interfering with the third station's frame. Intuitively, this approach will lead to a higher performance than pure ALOHA. Exactly the same holds for slotted ALOHA.

A second carrier sense protocol is nonpersistent CSMA. In this protocol, a conscious attempt is made to be less greedy than in the previous one. Before sending, a station senses the channel. If no one else is

sending, the station begins doing so itself. However, if the channel is already in use, the station does not continually sense it for the purpose of seizing it immediately upon detecting the end of the previous transmission. Instead, it waits a random period of time and then repeats the algorithm. Consequently, this algorithm leads to better channel utilization but longer delays than 1-persistent CSMA.

The last protocol is p-persistent CSMA. It applies to slotted channels and works as follows. When a station becomes ready to send, it senses the channel. If it is idle, it transmits with a probability p. With a probability q = 1 - p, it defers until the next slot. If that slot is also idle, it either transmits or defers again, with probabilities p and q. This process is repeated until either the frame has been transmitted or another station has begun transmitting. In the latter case, the unlucky station acts as if there had been a collision (i.e., it waits a random time and starts again). If the station initially senses the channel busy, it waits until the next slot and applies the above algorithm. Figure 4-4 shows the computed throughput versus offered traffic for all three protocols, as well as for pure and slotted ALOHA.

Comparison of the channel utilization versus load for various random access protocols



12.6 ethernet

We have now finished our general discussion of channel allocation protocols in the abstract, so it is time to see how these principles apply to real systems, in particular, LANs. IEEE has standardized a number of local area networks and metropolitan area networks under the name of IEEE 802. A few have survived but many have not. The most important of the survivors are 802.3 (Ethernet) and 802.11 (wireless LAN). With 802.15 (Bluetooth) and 802.16 (wireless MAN).

Since Ethernet and IEEE 802.3 are identical except for two minor differences that we will discuss shortly, many people use the terms "Ethernet" and "IEEE 802.3" interchangeably, and we will do so, too. For more information about Ethernet, see (Breyer and Riley, 1999 ; Seifert, 1998; and Spurgeon, 2000).

12.7 ethernet cabling

Since the name "Ethernet" refers to the cable (the ether), let us start our discussion there. Four types of cabling are commonly used, as shown below.

The most common kinds of Ethernet cabling

Name	Cable	Max. seg.	Nodes/seg.	Advantages
10Base5	Thick coax	500 m	100	Original cable; now obsolete
10Base2	Thin coax	185 m	30	No hub needed
10Base-T	Twisted pair	100 m	1024	Cheapest system
10Base-F	Fiber optics	2000 m	1024	Best between buildings

Historically, 10Base5 cabling, popularly called thick Ethernet, came first. It resembles a yellow garden hose, with markings every 2.5 meters to show where the taps go. (The 802.3 standard does not actually require the cable to be yellow, but it does suggest it.) Connections to it are generally made using vampire taps, in which a pin is very carefully forced halfway into the coaxial cable's core. The notation 10Base5 means that it operates at 10 Mbps, uses baseband signaling, and can support segments of up to 500 meters. The first number is the speed in Mbps. Then comes the word "Base" (or sometimes "BASE") to indicate baseband transmission. There used to be a broadband variant, 10Broad36, but it never caught on in the marketplace and has since vanished. Finally, if the medium is coax, its length is given rounded to units of 100 m after "Base."

Historically, the second cable type was 10Base2, or thin Ethernet, which, in contrast to the garden-hose-like thick Ethernet, bends easily. Connections to it are made using industry-standard BNC connectors to form T junctions, rather than using vampire taps. BNC connectors are easier to use and more reliable. Thin Ethernet is much cheaper and easier to install, but it can run for only 185 meters per segment, each of which can handle only 30 machines.

Detecting cable breaks, excessive length, bad taps, or loose connectors can be a major problem with both media. For this reason, techniques have been developed to track them down. Basically, a pulse of known shape is injected into the cable. If the pulse hits an obstacle or the end of the cable, an echo will be generated and sent back. By carefully timing the interval between sending the pulse and receiving the echo, it is possible to localize the origin of the echo. This technique is called time domain reflectometry.

The problems associated with finding cable breaks drove systems toward a different kind of wiring pattern, in which all stations have a cable running to a central hub in which they are all connected electrically (as if they were soldered together). Usually, these wires are telephone company twisted pairs, since most office buildings are already wired this way, and normally plenty of spare pairs are available. This scheme is called 10Base-T. Hubs do not buffer incoming traffic. We will discuss an improved version of this idea (switches), which do buffer incoming traffic later in this chapter.

A fourth cabling option for Ethernet is 10Base-F, which uses fiber optics. This alternative is expensive due to the cost of the connectors and terminators, but it has excellent noise immunity and is the method of choice when running between buildings or widely-separated hubs. Runs of up to km are allowed. It also offers good security since wiretapping fiber is much more difficult than wiretapping copper wire.

12.8 the ethernet MAC sublayer protocol

The original DIX (DEC, Intel, Xerox) frame structure is shown below. Each frame starts with a *Preamble* of 8 bytes, each containing the bit pattern 10101010. The Manchester encoding of this pattern produces a 10-MHz square wave for 6.4 μsec to allow the receiver's clock to synchronize with the sender's. They are required to stay synchronized for the rest of the frame, using the Manchester encoding to keep track of the bit boundaries.

However, they are not displayed by packet sniffing software because these bits are stripped away at OSI Layer 1 by the Ethernet adapter before being passed on to the OSI Layer 2 which is where packet sniffers collect their data from. There are OSI Physical Layer sniffers which can capture and display the Preamble and Start Frame but they are expensive and mainly used to detect physical related problems.

The table below shows the complete Ethernet frame, as transmitted, for the MTU of 1500 bytes (some

implementations of gigabit Ethernet and higher speeds support larger jumbo frames). Note that the bit patterns in the preamble and start of frame delimiter are written as bit strings, with the first bit transmitted on the left (*not* as byte values, which in Ethernet are transmitted least significant bit first). This notation matches the one used in the IEEE 802.3 standard. One octet is eight bits of data (i.e., a byte on most modern computers).

10/100M transceiver chips (MII PHY) work with four bits (one nibble) at a time. Therefore the preamble will be 7 instances of 0101 + 0101, and the Start Frame Delimiter will be 0101 + 1101. 8-bit values are sent low 4-bit and then high 4-bit. 1000M transceiver chips (GMII) work with 8 bits at a time, and 10 Gbit/s (XGMII) PHY works with 32 bits at a time.

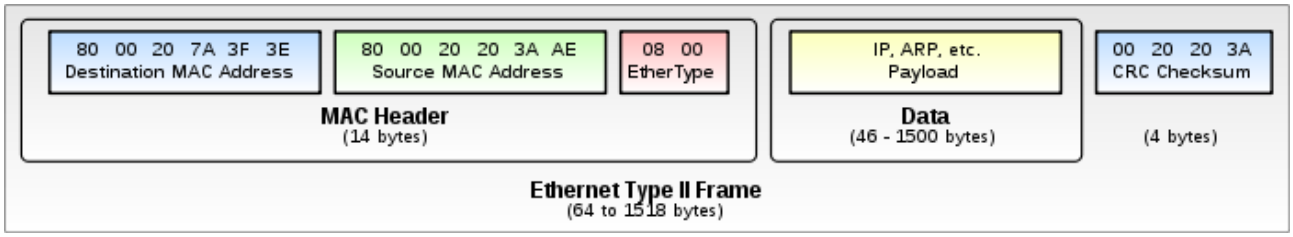
802.3 MAC Frame

Preamble	***Start-of-Frame-Delimiter***	***MAC destination***	***MAC source***	***802.1Q header (optional)***	***Ethertype/ Length***	***Payload (Data and padding)***	***CRC32***	***Interframe gap***
7 octets of 10101010	1 octet of 10101011	6 octets	6 octets	(4 octets)	2 octets	46–1500 octets	4 octets	12 octets
			64–1522 octets					
			72–1530 octets					
			84–1542 octets					

After a frame has been sent transmitters are required to transmit 12 octets of idle characters before transmitting the next frame.

12.9 ethernet II

Ethernet **II framing** (also known as **DIX Ethernet**, named after DEC, Intel and Xerox, the major participants in its design) defines the two-octet EtherType field in an Ethernet frame, preceded by destination and source MAC addresses, that identifies an upper layer protocol encapsulating the frame data.



The most common Ethernet Frame format, type II

For example, an EtherType value of 0x0800 signals that the packet contains an IPv4 datagram. Likewise, an EtherType of 0x0806 indicates an ARP frame, 0x8100 indicates an IEEE 802.1Q frame and 0x86DD indicates an IPv6 frame.

As this industry-developed standard went through a formal IEEE standardization process, the EtherType field was changed to a (data) length field in the new 802.3 standard. (Original Ethernet packets define their length with the framing that surrounds it, rather than with an explicit length count.) Since the packet recipient still needs to know how to interpret the packet, the standard required an IEEE 802.2 header to follow the length and specify the packet type. Many years later, the 802.3x-1997 standard, and later versions of the 802.3 standard, formally approved of both types of framing. In practice, both formats are in wide use, with original Ethernet framing the most common in Ethernet local area networks, due to its simplicity and lower

overhead.

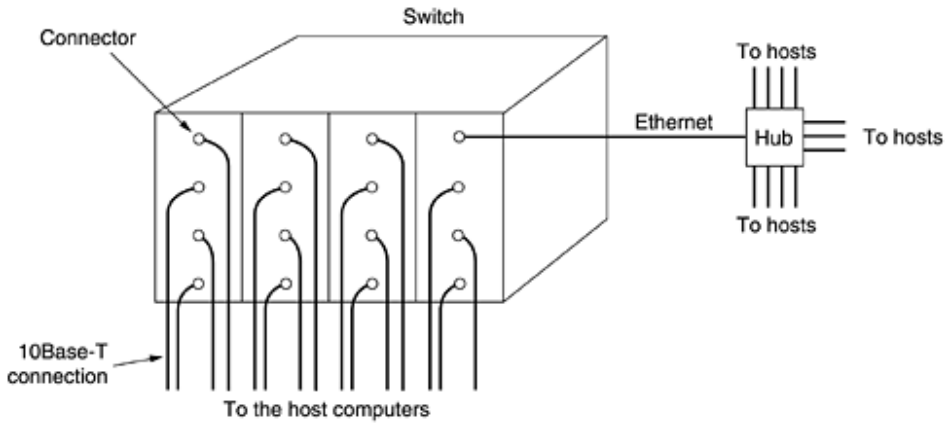
In order to allow some packets using Ethernet v2 framing and some packets using the original version of 802.3 framing to be used on the same Ethernet segment, EtherType values must be greater than or equal to 1536 (0x0600). That value was chosen because the maximum length of the data field of an Ethernet 802.3 frame is 1500 bytes (0x05DC). Thus if the field's value is greater than or equal to 1536, the frame must be an Ethernet v2 frame, with that field being a type field. If it's less than or equal to 1500, it must be an IEEE 802.3 frame, with that field being a length field. Values between 1500 and 1536, exclusive, are undefined.[1]

12.10 switched ethernet

As more and more stations are added to an Ethernet, the traffic will go up. Eventually, the LAN will saturate. One way out is to go to a higher speed, say, from 10 Mbps to 100 Mbps. But with the growth of multimedia, even a 100-Mbps or 1-Gbps Ethernet can become saturated.

Fortunately, there is an additional way to deal with increased load: switched Ethernet, as shown below. The heart of this system is a switch containing a high-speed backplane and room for typically 4 to 32 plug-in line cards, each containing one to eight connectors. Most often, each connector has a 10Base-T twisted pair connection to a single host computer.

A simple example of switched Ethernet



When a station wants to transmit an Ethernet frame, it outputs a standard frame to the switch. The plug-in card getting the frame may check to see if it is destined for one of the other stations connected to the same card. If so, the frame is copied there. If not, the frame is sent over the high-speed backplane to the destination station's card. The backplane typically runs at many Gbps, using a proprietary protocol.

What happens if two machines attached to the same plug-in card transmit frames at the same time? It depends on how the card has been constructed. One possibility is for all the ports on the card to be wired together to form a local on-card LAN. Collisions on this on-card LAN will be detected and handled the same as any other collisions on a CSMA/CD network—with retransmissions using the binary exponential backoff algorithm. With this kind of plug-in card, only one transmission per card is possible at any instant, but all the cards can be transmitting in parallel. With this design, each card forms its own collision domain, independent of the others. With only one station per collision domain, collisions are impossible and performance is improved.

With the other kind of plug-in card, each input port is buffered, so incoming frames are stored in the card's on-board RAM as they arrive. This design allows all input ports to receive (and transmit) frames at the same time, for parallel, full-duplex operation, something not possible with CSMA/CD on a single channel.

Once a frame has been completely received, the card can then check to see if the frame is destined for another port on the same card or for a distant port. In the former case, it can be transmitted directly to the destination. In the latter case, it must be transmitted over the backplane to the proper card. With this design, each port is a separate collision domain, so collisions do not occur. The total system throughput can often be increased by an order of magnitude over 10Base5, which has a single collision domain for the entire system.

Since the switch just expects standard Ethernet frames on each input port, it is possible to use some of the ports as concentrators. In Fig. 4-20, the port in the upper-right corner is connected not to a single station, but to a 12-port hub. As frames arrive at the hub, they contend for the ether in the usual way, including collisions and binary backoff. Successful frames make it to the switch and are treated there like any other incoming frames: they are switched to the correct output line over the high-speed backplane. Hubs are cheaper than switches, but due to falling switch prices, they are rapidly becoming obsolete. Nevertheless, legacy hubs still exist.

12.11 fast ethernet

At first, 10 Mbps seemed like heaven, just as 1200-bps modems seemed like heaven to the early users of 300-bps acoustic modems. But the novelty wore off quickly. As a kind of corollary to Parkinson's Law (''Work expands to fill the time available for its completion''), it seemed that data expanded to fill the bandwidth available for their transmission. To pump up the speed, various industry groups proposed two new ring-based optical LANs. One was called **FDDI** (**Fiber Distributed Data Interface**) and the other was called Fibre Channel. To make a long story short, while both were used as backbone networks, neither one made the breakthrough to the desktop. In both cases, the station management was too complicated, which led to complex chips and high prices. The lesson that should have been learned here was KISS (Keep It Simple, Stupid).

In any event, the failure of the optical LANs to catch fire left a gap for garden-variety Ethernet at speeds above 10 Mbps. Many installations needed more bandwidth and thus had numerous 10-Mbps LANs connected by a maze of repeaters, bridges, routers, and gateways, although to the network managers it sometimes felt that they were being held together by bubble gum and chicken wire.

It was in this environment that IEEE reconvened the 802.3 committee in 1992 with instructions to come up with a faster LAN. One proposal was to keep 802.3 exactly as it was, but just make it go faster. Another proposal was to redo it totally to give it lots of new features, such as real-time traffic and digitized voice, but just keep the old name (for marketing reasons). After some wrangling, the committee decided to keep 802.3 the way it was, but just make it go faster. The people behind the losing proposal did what any computer-industry people would have done under these circumstances—they stomped off and formed their own committee and standardized their LAN anyway (eventually as 802.12). It flopped miserably.

The 802.3 committee decided to go with a souped-up Ethernet for three primary reasons:

1. The need to be backward compatible with existing Ethernet LANs.
2. The fear that a new protocol might have unforeseen problems.
3. The desire to get the job done before the technology changed.

The work was done quickly (by standards committees' norms), and the result, 802.3u, was officially approved by IEEE in June 1995. Technically, 802.3u is not a new standard, but an addendum to the existing 802.3 standard (to emphasize its backward compatibility). Since practically everyone calls it fast Ethernet, rather than 802.3u, we will do that, too.

The basic idea behind fast Ethernet was simple: keep all the old frame formats, interfaces, and procedural rules, but just reduce the bit time from 100 nsec to 10 nsec. Technically, it would have been possible to copy either 10Base-5 or 10Base-2 and still detect collisions on time by just reducing the maximum cable length by a factor of ten. However, the advantages of 10Base-T wiring were so overwhelming that fast Ethernet is based entirely on this design. Thus, all fast Ethernet systems use hubs and switches; multidrop cables with vampire taps or BNC connectors are not permitted.

Nevertheless, some choices still had to be made, the most important being which wire types to support.

One contender was category 3 twisted pair. The argument for it was that practically every office in the Western world has at least four category 3 (or better) twisted pairs running from it to a telephone wiring closet within 100 meters. Sometimes two such cables exist. Thus, using category 3 twisted pair would make it possible to wire up desktop computers using fast Ethernet without having to rewire the building, an enormous advantage for many organizations.

The main disadvantage of category 3 twisted pair is its inability to carry 200 megabaud signals (100 Mbps with Manchester encoding) 100 meters, the maximum computer-to-hub distance specified for 10Base-T. In contrast, category 5 twisted pair wiring can handle 100 meters easily, and fiber can go much farther. The compromise chosen was to allow all three possibilities, as shown below, but to pep up the category 3 solution to give it the additional carrying capacity needed.

The original fast Ethernet cabling

Name	Cable	Max. segment	Advantages
100Base-T4	Twisted pair	100 m	Uses category 3 UTP
100Base-TX	Twisted pair	100 m	Full duplex at 100 Mbps (Cat 5 UTP)
100Base-FX	Fiber optics	2000 m	Full duplex at 100 Mbps; long runs

The category 3 UTP scheme, called 100Base-T4, uses a signaling speed of 25 MHz, only 25 percent faster than standard Ethernet's 20 MHz (remember that Manchester encoding, requires two clock periods for each of the 10 million bits each second). However, to achieve the necessary bandwidth, 100Base-T4 requires four twisted pairs. Since standard telephone wiring for decades has had four twisted pairs per cable, most offices are able to handle this. Of course, it means giving up your office telephone, but that is surely a small price to pay for faster e-mail.

Of the four twisted pairs, one is always to the hub, one is always from the hub, and the other two are switchable to the current transmission direction. To get the necessary bandwidth, Manchester encoding is not used, but with modern clocks and such short distances, it is no longer needed. In addition, ternary signals are sent, so that during a single clock period the wire can contain a 0, a 1, or a 2. With three twisted pairs going in the forward direction and ternary signaling, any one of 27 possible symbols can be transmitted, making it possible to send 4 bits with some redundancy. Transmitting 4 bits in each of the 25 million clock cycles per second gives the necessary 100 Mbps. In addition, there is always a 33.3-Mbps reverse channel using the remaining twisted pair. This scheme, known as 8B/6T (8 bits map to 6 trits), is not likely to win any prizes for elegance, but it works with the existing wiring plant.

For category 5 wiring, the design, 100Base-TX, is simpler because the wires can handle clock rates of 125 MHz. Only two twisted pairs per station are used, one to the hub and one from it. Straight binary coding is not used; instead a scheme called used4B/5Bis It is taken from FDDI and compatible with it. Every group of five clock periods, each containing one of two signal values, yields 32 combinations. Sixteen of these combinations are used to transmit the four bit groups 0000, 0001, 0010, ..., 1111. Some of the remaining 16 are used for control purposes such as marking frames boundaries. The combinations used have been carefully chosen to provide enough transitions to maintain clock synchronization. The 100Base-TX system is full duplex; stations can transmit at 100 Mbps and receive at 100 Mbps at the same time. Often 100Base-TX and 100Base-T4 are collectively referred to as 100Base-T.

The last option, 100Base-FX, uses two strands of multimode fiber, one for each direction, so it, too, is full duplex with 100 Mbps in each direction. In addition, the distance between a station and the hub can be up to 2 km.

In response to popular demand, in 1997 the 802 committee added a new cabling type, 100Base-T2, allowing fast Ethernet to run over two pairs of existing category 3 wiring. However, a sophisticated digital signal processor is needed to handle the encoding scheme required, making this option fairly expensive. So far, it is rarely used due to its complexity, cost, and the fact that many office buildings have already been rewired with category 5 UTP.

Two kinds of interconnection devices are possible with 100Base-T: hubs and switches. In a hub, all the incoming lines (or at least all the lines arriving at one plug-in card) are logically connected, forming a single

collision domain. All the standard rules, including the binary exponential backoff algorithm, apply, so the system works just like old-fashioned Ethernet. In particular, only one station at a time can be transmitting. In other words, hubs require half-duplex communication.

In a switch, each incoming frame is buffered on a plug-in line card and passed over a high-speed backplane from the source card to the destination card if need be. The backplane has not been standardized, nor does it need to be, since it is entirely hidden deep inside the switch. If past experience is any guide, switch vendors will compete vigorously to produce ever faster backplanes in order to improve system throughput. Because 100Base-FX cables are too long for the normal Ethernet collision algorithm, they must be connected to switches, so each one is a collision domain unto itself. Hubs are not permitted with 100Base-FX.

As a final note, virtually all switches can handle a mix of 10-Mbps and 100-Mbps stations, to make upgrading easier. As a site acquires more and more 100-Mbps workstations, all it has to do is buy the necessary number of new line cards and insert them into the switch. In fact, the standard itself provides a way for two stations to automatically negotiate the optimum speed (10 or 100 Mbps) and duplexity (half or full). Most fast Ethernet products use this feature to autoconfigure themselves.

12.12 gigabit ethernet

The ink was barely dry on the fast Ethernet standard when the 802 committee began working on a yet faster Ethernet (1995). It was quickly dubbed **gigabit Ethernet** and was ratified by IEEE in 1998 under the name 802.3z. This identifier suggests that gigabit Ethernet is going to be the end of the line unless somebody quickly invents a new letter after z. Below we will discuss some of the key features of gigabit Ethernet. More information can be found in (Seifert, 1998).

The 802.3z committee's goals were essentially the same as the 802.3u committee's goals: make Ethernet go 10 times faster yet remain backward compatible with all existing Ethernet standards. In particular, gigabit Ethernet had to offer unacknowledged datagram service with both unicast and multicast, use the same 48-bit addressing scheme already in use, and maintain the same frame format, including the minimum and maximum frame sizes. The final standard met all these goals.

All configurations of gigabit Ethernet are point-to-point rather than multidrop as in the original 10 Mbps standard, now honored as **classic Ethernet**. In the simplest gigabit Ethernet configuration, two computers are directly connected to each other. The more common case, however, is having a switch or a hub connected to multiple computers and possibly additional switches or hubs. In both configurations each individual Ethernet cable has exactly two devices on it, no more and no fewer.

Gigabit Ethernet supports two different modes of operation: full-duplex mode and half-duplex mode. The "normal" mode is full-duplex mode, which allows traffic in both directions at the same time. This mode is used when there is a central switch connected to computers (or other switches) on the periphery. In this configuration, all lines are buffered so each computer and switch is free to send frames whenever it wants to. The sender does not have to sense the channel to see if anybody else is using it because contention is impossible. On the line between a computer and a switch, the computer is the only possible sender on that line to the switch and the transmission succeeds even if the switch is currently sending a frame to the computer (because the line is full duplex). Since no contention is possible, the CSMA/CD protocol is not used, so the maximum length of the cable is determined by signal strength issues rather than by how long it takes for a noise burst to propagate back to the sender in the worst case. Switches are free to mix and match speeds. Autoconfiguration is supported just as in fast Ethernet.

The other mode of operation, half-duplex, is used when the computers are connected to a hub rather than a switch. A hub does not buffer incoming frames. Instead, it electrically connects all the lines internally, simulating the multidrop cable used in classic Ethernet. In this mode, collisions are possible, so the standard CSMA/CD protocol is required. Because a minimum (i.e., 64-byte) frame can now be transmitted 100 times faster than in classic Ethernet, the maximum distance is 100 times less, or 25 meters, to maintain the essential property that the sender is still transmitting when the noise burst gets back to it, even in the worst case. With a 2500-meter-long cable, the sender of a 64-byte frame at 1 Gbps would be long done before the frame got even a tenth of the way to the other end, let alone to the end and back.

The 802.3z committee considered a radius of 25 meters to be unacceptable and added two features to the standard to increase the radius. The first feature, called carrier extension, essentially tells the hardware to add its own padding after the normal frame to extend the frame to 512 bytes. Since this padding is added by the sending hardware and removed by the receiving hardware, the software is unaware of it, meaning that no changes are needed to existing software. Of course, using 512 bytes worth of bandwidth to transmit 46 bytes of user data (the payload of a 64-byte frame) has a line efficiency of 9%.

The second feature, called frame bursting, allows a sender to transmit a concatenated sequence of multiple frames in a single transmission. If the total burst is less than 512 bytes, the hardware pads it again. If enough frames are waiting for transmission, this scheme is highly efficient and preferred over carrier extension. These new features extend the radius of the network to 200 meters, which is probably enough for most offices.

In all fairness, it is hard to imagine an organization going to the trouble of buying and installing gigabit Ethernet cards to get high performance and then connecting the computers with a hub to simulate classic Ethernet with all its collisions. While hubs are somewhat cheaper than switches, gigabit Ethernet interface cards are still relatively expensive. To then economize by buying a cheap hub and slash the performance of the new system is foolish. Still, backward compatibility is sacred in the computer industry, so the 802.3z committee was required to put it in.

Gigabit Ethernet supports both copper and fiber cabling, as listed in Fig. 4-23. Signaling at or near 1 Gbps over fiber means that the light source has to be turned on and off in under 1 nsec. LEDs simply cannot operate this fast, so lasers are required. Two wavelengths are permitted: 0.85 microns (Short) and 1.3 microns (Long). Lasers at 0.85 microns are cheaper but do not work on single-mode fiber.

Gigabit Ethernet cabling

Name	Cable	Max. segment	Advantages
1000Base-SX	Fiber optics	550 m	Multimode fiber (50, 62.5 microns)
1000Base-LX	Fiber optics	5000 m	Single (10 μ) or multimode (50, 62.5 μ)
1000Base-CX	2 Pairs of STP	25 m	Shielded twisted pair
1000Base-T	4 Pairs of UTP	100 m	Standard category 5 UTP

Three fiber diameters are permitted: 10, 50, and 62.5 microns. The first is for single mode and the last two are for multimode. Not all six combinations are allowed, however, and the maximum distance depends on the combination used. The numbers given above are for the best case. In particular, 5000 meters is only achievable with 1.3 micron lasers operating over 10 micron fiber in single mode, but this is the best choice for campus backbones and is expected to be popular, despite its being the most expensive choice.

The 1000Base-CX option uses short shielded copper cables. Its problem is that it is competing with high-performance fiber from above and cheap UTP from below. It is unlikely to be used much, if at all.

The last option is bundles of four category 5 UTP wires working together. Because so much of this wiring is already installed, it is likely to be the poor man's gigabit Ethernet.

Gigabit Ethernet uses new encoding rules on the fibers. Manchester encoding at 1 Gbps would require a 2 Gbaud signal, which was considered too difficult and also too wasteful of bandwidth. Instead a new scheme, called 8B/10B, was chosen, based on fibre channel. Each 8-bit byte is encoded on the fiber as 10 bits, hence the name 8B/10B. Since there are 1024 possible output codewords for each input byte, some leeway was available in choosing which codewords to allow. The following two rules were used in making the choices:

1. No codeword may have more than four identical bits in a row.
2. No codeword may have more than six 0s or six 1s.

These choices were made to keep enough transitions in the stream to make sure the receiver stays in sync with the sender and also to keep the number of 0s and 1s on the fiber as close to equal as possible. In addition, many input bytes have two possible codewords assigned to them. When the encoder has a choice

of codewords, it always chooses the codeword that moves in the direction of equalizing the number of 0s and 1s transmitted so far. This emphasis of balancing 0s and 1s is needed to keep the DC component of the signal as low as possible to allow it to pass through transformers unmodified. While computer scientists are not fond of having the properties of transformers dictate their coding schemes, life is like that sometimes.

Gigabit Ethernets using 1000Base-T use a different encoding scheme since clocking data onto copper wire in 1 nsec is too difficult. This solution uses four category 5 twisted pairs to allow four symbols to be transmitted in parallel. Each symbol is encoded using one of five voltage levels. This scheme allows a single symbol to encode 00, 01, 10, 11, or a special value for control purposes. Thus, there are 2 data bits per twisted pair or 8 data bits per clock cycle. The clock runs at 125 MHz, allowing 1-Gbps operation. The reason for allowing five voltage levels instead of four is to have combinations left over for framing and control purposes.

A speed of 1 Gbps is quite fast. For example, if a receiver is busy with some other task for even 1 msec and does not empty the input buffer on some line, up to 1953 frames may have accumulated there in that 1 ms gap. Also, when a computer on a gigabit Ethernet is shipping data down the line to a computer on a classic Ethernet, buffer overruns are very likely. As a consequence of these two observations, gigabit Ethernet supports flow control (as does fast Ethernet, although the two are different).

The flow control consists of one end sending a special control frame to the other end telling it to pause for some period of time. Control frames are normal Ethernet frames containing a type of 0x8808. The first two bytes of the data field give the command; succeeding bytes provide the parameters, if any. For flow control, PAUSE frames are used, with the parameter telling how long to pause, in units of the minimum frame time. For gigabit Ethernet, the time unit is 512 nsec, allowing for pauses as long as 33.6 msec.

As soon as gigabit Ethernet was standardized, the 802 committee got bored and wanted to get back to work. IEEE told them to start on 10-gigabit Ethernet. After searching hard for a letter to follow z, they abandoned that approach and went over to two-letter suffixes. They got to work and that standard was approved by IEEE in 2002 as 802.3ae. Can 100-gigabit Ethernet be far behind?

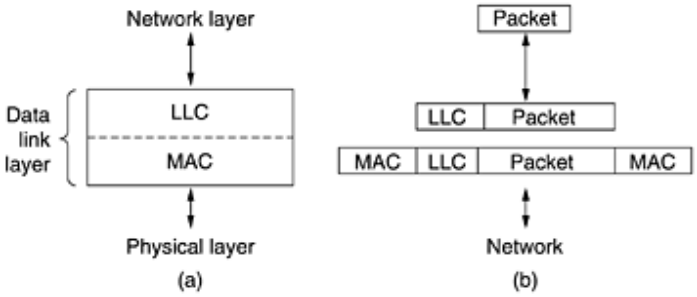
12.13 IEEE 802.2: Logical Link Control

It is now perhaps time to step back and compare what we have learned in this chapter with what we studied in the previous one. We saw in a previous chapter how two machines could communicate reliably over an unreliable line by using various data link protocols. These protocols provided error control (using acknowledgments) and flow control (using a sliding window).

In contrast, in this chapter, we have not said a word about reliable communication. All that Ethernet and the other 802 protocols offer is a best-efforts datagram service. Sometimes, this service is adequate. For example, for transporting IP packets, no guarantees are required or even expected. An IP packet can just be inserted into an 802 payload field and sent on its way. If it gets lost, so be it.

Nevertheless, there are also systems in which an error-controlled, flow-controlled data link protocol is desired. IEEE has defined one that can run on top of Ethernet and the other 802 protocols. In addition, this protocol, called LLC (Logical Link Control), hides the differences between the various kinds of 802 networks by providing a single format and interface to the network layer. This format, interface, and protocol are all closely based on the HDLC protocol. LLC forms the upper half of the data link layer, with the MAC sublayer below it, as shown below.

(a) Position of LLC. (b) Protocol formats



Typical usage of LLC is as follows. The network layer on the sending machine passes a packet to LLC, using the LLC access primitives. The LLC sublayer then adds an LLC header, containing sequence and acknowledgment numbers. The resulting structure is then inserted into the payload field of an 802 frame and transmitted. At the receiver, the reverse process takes place.

LLC provides three service options: unreliable datagram service, acknowledged datagram service, and reliable connection-oriented service. The LLC header contains three fields: a destination access point, a source access point, and a control field. The access points tell which process the frame came from and where it is to be delivered, replacing the DIX *Type* field. The control field contains sequence and acknowledgment numbers, very much in the style of HDLC, but not identical to it. These fields are primarily used when a reliable connection is needed at the data link level, in which case other protocol would be used. For the Internet, best-efforts attempts to deliver IP packets is sufficient, so no acknowledgments at the LLC level are required.

12.14 retrospective on ethernet

Ethernet has been around for over 20 years and has no serious competitors in sight, so it is likely to be around for many years to come. Few CPU architectures, operating systems, or programming languages have been king of the mountain for two decades going on three. Clearly, Ethernet did something right. What?

Probably the main reason for its longevity is that Ethernet is simple and flexible. In practice, simple translates into reliable, cheap, and easy to maintain. Once the vampire taps were replaced by BNC connectors, failures became extremely rare. People hesitate to replace something that works perfectly all the time, especially when they know that an awful lot of things in the computer industry work very poorly, so that many so-called "upgrades" are appreciably worse than what they replaced.

Simple also translates into cheap. Thin Ethernet and twisted pair wiring is relatively inexpensive. The interface cards are also low cost. Only when hubs and switches were introduced were substantial investments required, but by the time they were in the picture, Ethernet was already well established.

Ethernet is easy to maintain. There is no software to install (other than the drivers) and there are no configuration tables to manage (and get wrong). Also, adding new hosts is as simple as just plugging them in.

Another point is that Ethernet interworks easily with TCP/IP, which has become dominant. IP is a connectionless protocol, so it fits perfectly with Ethernet, which is also connectionless. IP fits much less well with ATM, which is connection oriented. This mismatch definitely hurt ATM's chances.

Lastly, Ethernet has been able to evolve in certain crucial ways. Speeds have gone up by several orders of magnitude and hubs and switches have been introduced, but these changes have not required changing the software. When a network salesman shows up at a large installation and says: "I have this fantastic new network for you. All you have to do is throw out all your hardware and rewrite all your software," he has a problem. FDDI, Fibre Channel, and ATM were all faster than Ethernet when introduced, but they were incompatible with Ethernet, far more complex, and harder to manage. Eventually, Ethernet caught up with them in terms of speed, so they had no advantages left and quietly died off except for ATM's use deep within the core of the telephone system.

chapter 13 the physical layer

13.1 guided transmission media

The purpose of the physical layer is to transport a raw bit stream from one machine to another. Various physical media can be used for the actual transmission. Each one has its own niche in terms of bandwidth, delay, cost, and ease of installation and maintenance. Media are roughly grouped into **guided media**, such as copper wire and fiber optics, and **unguided media**, such as radio and lasers through the air. We will look at all of these in the following sections.

13.1.1 magnetic media

One of the most common ways to transport data from one computer to another is to write them onto magnetic tape or removable media (e.g., recordable DVDs), physically transport the tape or disks to the destination machine, and read them back in again. Although this method is not as sophisticated as using a geosynchronous communication satellite, it is often more cost effective, especially for applications in which high bandwidth or cost per bit transported is the key factor.

A simple calculation will make this point clear. An industry standard Ultrium tape can hold 200 gigabytes. A box 60 x 60 x 60 cm can hold about 1000 of these tapes, for a total capacity of 200 terabytes, or 1600 terabits (1.6 petabits). A box of tapes can be delivered anywhere in the United States in 24 hours by Federal Express and other companies. The effective bandwidth of this transmission is 1600 terabits/86,400 sec, or 19 Gbps. If the destination is only an hour away by road, the bandwidth is increased to over 400 Gbps. No computer network can even approach this.

For a bank with many gigabytes of data to be backed up daily on a second machine (so the bank can continue to function even in the face of a major flood or earthquake), it is likely that no other transmission technology can even begin to approach magnetic tape for performance. Of course, networks are getting faster, but tape densities are increasing, too.

If we now look at cost, we get a similar picture. The cost of an Ultrium tape is around $40 when bought in bulk. A tape can be reused at least ten times, so the tape cost is maybe $4000 per box per usage. Add to this another $1000 for shipping (probably much less), and we have a cost of roughly $5000 to ship 200 TB. This amounts to shipping a gigabyte for under 3 cents. No network can beat that. The moral of the story is:

Never underestimate the bandwidth of a station wagon full of tapes hurtling down the highway.

13.1.2 twisted pair

Although the bandwidth characteristics of magnetic tape are excellent, the delay characteristics are poor. Transmission time is measured in minutes or hours, not milliseconds. For many applications an on-line connection is needed. One of the oldest and still most common transmission media is **twisted pair**. A twisted pair consists of two insulated copper wires, typically about 1 mm thick. The wires are twisted together in a helical form, just like a DNA molecule. Twisting is done because two parallel wires constitute a fine antenna. When the wires are twisted, the waves from different twists cancel out, so the wire radiates less effectively.

The most common application of the twisted pair is the telephone system. Nearly all telephones are connected to the telephone company (telco) office by a twisted pair. Twisted pairs can run several kilometers without amplification, but for longer distances, repeaters are needed. When many twisted pairs run in parallel for a substantial distance, such as all the wires coming from an apartment building to the telephone company

office, they are bundled together and encased in a protective sheath. The pairs in these bundles would interfere with one another if it were not for the twisting. In parts of the world where telephone lines run on poles above ground, it is common to see bundles several centimeters in diameter.

Twisted pairs can be used for transmitting either analog or digital signals. The bandwidth depends on the thickness of the wire and the distance traveled, but several megabits/sec can be achieved for a few kilometers in many cases. Due to their adequate performance and low cost, twisted pairs are widely used and are likely to remain so for years to come.

Twisted pair cabling comes in several varieties, two of which are important for computer networks. Category 3 twisted pairs consist of two insulated wires gently twisted together. Four such pairs are typically grouped in a plastic sheath to protect the wires and keep them together. Prior to about 1988, most office buildings had one category 3 cable running from a central wiring closet on each floor into each office. This scheme allowed up to four regular telephones or two multiline telephones in each office to connect to the telephone company equipment in the wiring closet.

Starting around 1988, the more advanced category 5 twisted pairs were introduced. They are similar to category 3 pairs, but with more twists per centimeter, which results in less crosstalk and a better-quality signal over longer distances, making them more suitable for high-speed computer communication. Up-and-coming categories are 6 and 7, which are capable of handling signals with bandwidths of 250 MHz and 600 MHz, respectively (versus a mere 16 MHz and 100 MHz for categories 3 and 5, respectively).

All of these wiring types are often referred to as UTP (Unshielded Twisted Pair), to contrast them with the bulky, expensive, shielded twisted pair cables IBM introduced in the early 1980s, but which have not proven popular outside of IBM installations. Twisted pair cabling is illustrated below.

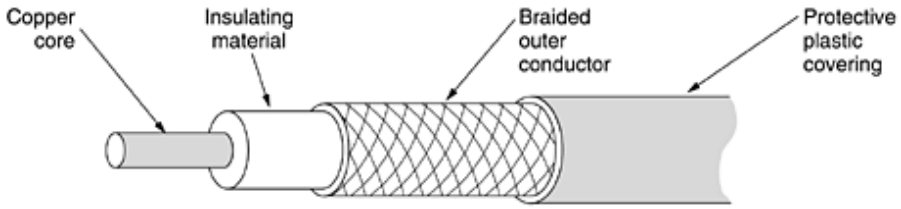
(a) Category 3 UTP. (b) Category 5 UTP

13.1.3 coaxial cable

Another common transmission medium is the coaxial cable (known to its many friends as just "coax" and pronounced "co-ax"). It has better shielding than twisted pairs, so it can span longer distances at higher speeds. Two kinds of coaxial cable are widely used. One kind, 50-ohm cable, is commonly used when it is intended for digital transmission from the start. The other kind, 75-ohm cable, is commonly used for analog transmission and cable television but is becoming more important with the advent of Internet over cable. This distinction is based on historical, rather than technical, factors (e.g., early dipole antennas had an impedance of 300 ohms, and it was easy to use existing 4:1 impedance matching transformers).

A coaxial cable consists of a stiff copper wire as the core, surrounded by an insulating material. The insulator is encased by a cylindrical conductor, often as a closely-woven braided mesh. The outer conductor is covered in a protective plastic sheath. A cutaway view of a coaxial cable is shown below.

A coaxial cable



The construction and shielding of the coaxial cable give it a good combination of high bandwidth and excellent noise immunity. The bandwidth possible depends on the cable quality, length, and signal-to-noise ratio of the data signal. Modern cables have a bandwidth of close to 1 GHz. Coaxial cables used to be widely used within the telephone system for long-distance lines but have now largely been replaced by fiber optics on long-haul routes. Coax is still widely used for cable television and metropolitan area networks, however.

13.1.4 fiber optics

Many people in the computer industry take enormous pride in how fast computer technology is improving. The original (1981) IBM PC ran at a clock speed of 4.77 MHz. Twenty years later, PCs could run at 2 GHz, a gain of a factor of 20 per decade. Not too bad.

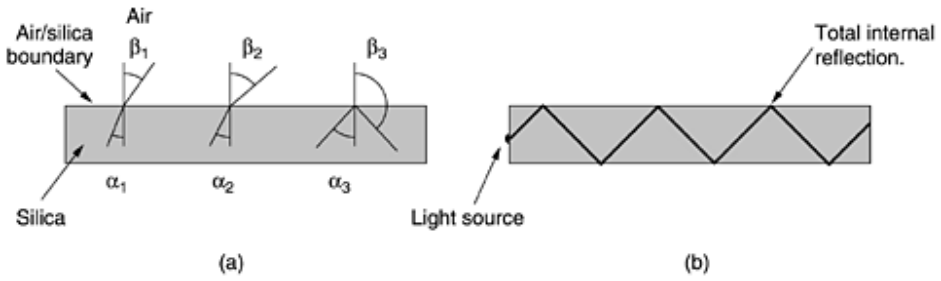
In the same period, wide area data communication went from 56 kbps (the ARPANET) to 1 Gbps (modern optical communication), a gain of more than a factor of 125 per decade, while at the same time the error rate went from 10^{-5} per bit to almost zero.

Furthermore, single CPUs are beginning to approach physical limits, such as speed of light and heat dissipation problems. In contrast, with current fiber technology, the achievable bandwidth is certainly in excess of 50,000 Gbps (50 Tbps) and many people are looking very hard for better technologies and materials. The current practical signaling limit of about 10 Gbps is due to our inability to convert between electrical and optical signals any faster, although in the laboratory, 100 Gbps has been achieved on a single fiber.

In the race between computing and communication, communication won. The full implications of essentially infinite bandwidth (although not at zero cost) have not yet sunk in to a generation of computer scientists and engineers taught to think in terms of the low Nyquist and Shannon limits imposed by copper wire. The new conventional wisdom should be that all computers are hopelessly slow and that networks should try to avoid computation at all costs, no matter how much bandwidth that wastes. In this section we will study fiber optics to see how that transmission technology works.

An optical transmission system has three key components: the light source, the transmission medium, and the detector. Conventionally, a pulse of light indicates a 1 bit and the absence of light indicates a 0 bit. The transmission medium is an ultra-thin fiber of glass. The detector generates an electrical pulse when light falls on it. By attaching a light source to one end of an optical fiber and a detector to the other, we have a unidirectional data transmission system that accepts an electrical signal, converts and transmits it by light pulses, and then reconverts the output to an electrical signal at the receiving end.

This transmission system would leak light and be useless in practice except for an interesting principle of physics. When a light ray passes from one medium to another, for example, from fused silica to air, the ray is refracted (bent) at the silica/air boundary, as shown below, in part (a). Here we see a light ray incident on the boundary at an angle a_1 emerging at an angle b_1. The amount of refraction depends on the properties of the two media (in particular, their indices of refraction). For angles of incidence above a certain critical value, the light is refracted back into the silica; none of it escapes into the air. Thus, a light ray incident at or above the critical angle is trapped inside the fiber, as shown below, in part (b), and can propagate for many kilometers with virtually no loss.

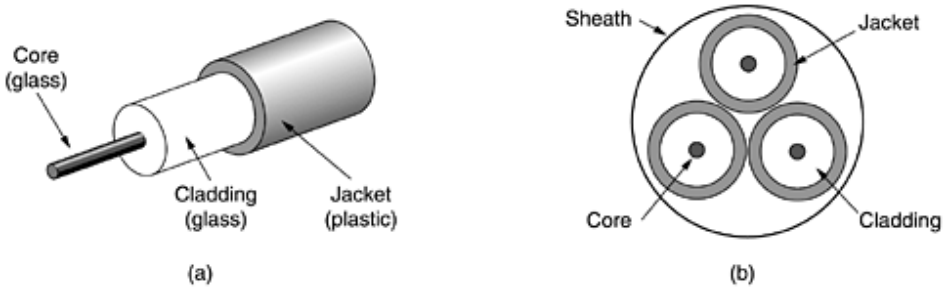


13.1.5 fiber cables

Fiber optic cables are similar to coax, except without the braid. The next figure shows a single fiber viewed from the side. At the center is the glass core through which the light propagates. In multimode fibers, the core is typically 50 microns in diameter, about the thickness of a human hair. In single-mode fibers, the core is 8 to 10 microns.

The core is surrounded by a glass cladding with a lower index of refraction than the core, to keep all the light in the core. Next comes a thin plastic jacket to protect the cladding. Fibers are typically grouped in bundles, protected by an outer sheath. The figure below shows a sheath with three fibers.

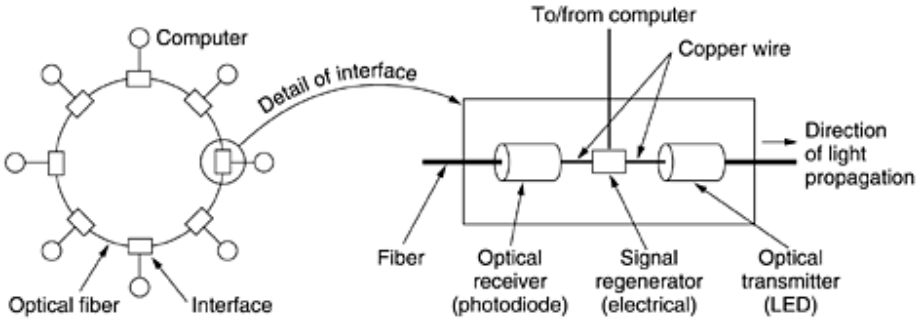
(a) Side view of a single fiber. (b) End view of a sheath with three fibers



13.1.6 Fiber Optic Networks

Fiber optics can be used for LANs as well as for long-haul transmission, although tapping into it is more complex than connecting to an Ethernet. One way around the problem is to realize that a ring network is really just a collection of point-to-point links, as shown below. The interface at each computer passes the light pulse stream through to the next link and also serves as a T junction to allow the computer to send and accept messages.

A fiber optic ring with active repeaters



13.2 wireless transmission

Our age has given rise to information junkies: people who need to be on-line all the time. For these mobile users, twisted pair, coax, and fiber optics are of no use. They need to get their hits of data for their laptop, notebook, shirt pocket, palmtop, or wristwatch computers without being tethered to the terrestrial communication infrastructure. For these users, wireless communication is the answer. In the following sections, we will look at wireless communication in general, as it has many other important applications besides providing connectivity to users who want to surf the Web from the beach.

Some people believe that the future holds only two kinds of communication: fiber and wireless. All fixed (i.e., nonmobile) computers, telephones, faxes, and so on will use fiber, and all mobile ones will use wireless.

Wireless has advantages for even fixed devices in some circumstances. For example, if running a fiber to a building is difficult due to the terrain (mountains, jungles, swamps, etc.), wireless may be better. It is noteworthy that modern wireless digital communication began in the Hawaiian Islands, where large chunks of Pacific Ocean separated the users and the telephone system was inadequate.

13.2.1 the electromagnetic spectrum

When electrons move, they create electromagnetic waves that can propagate through space (even in a vacuum). These waves were predicted by the British physicist James Clerk Maxwell in 1865 and first observed by the German physicist Heinrich Hertz in 1887. The number of oscillations per second of a wave is called its frequency, f, and is measured in Hz (in honor of Heinrich Hertz). The distance between two consecutive maxima (or minima) is called the wavelength, which is universally designated by the Greek letter l (lambda).

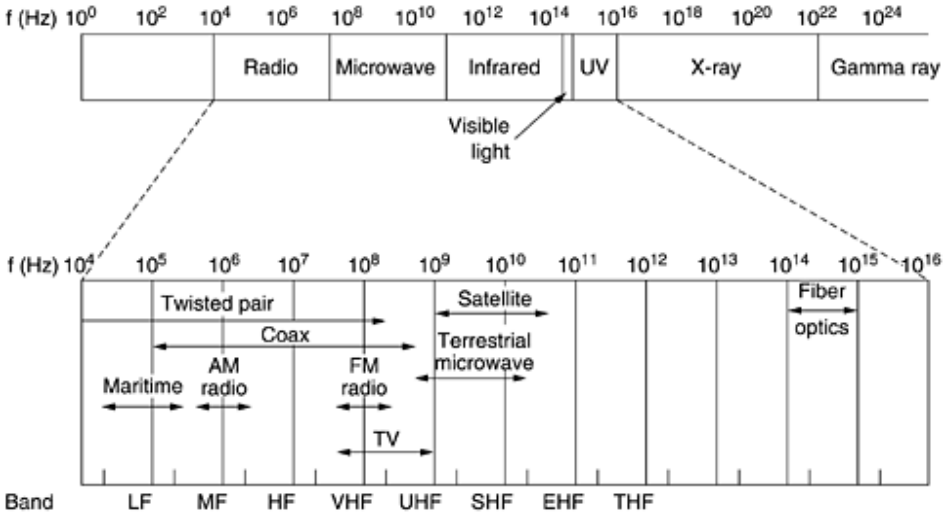
When an antenna of the appropriate size is attached to an electrical circuit, the electromagnetic waves can be broadcast efficiently and received by a receiver some distance away. All wireless communication is based on this principle.

In vacuum, all electromagnetic waves travel at the same speed, no matter what their frequency. This speed, usually called the speed of light, c, is approximately 3 x 10^8 m/sec, or about 1 foot (30 cm) per nanosecond. (A case could be made for redefining the foot as the distance light travels in a vacuum in 1 nsec rather than basing it on the shoe size of some long-dead king.) In copper or fiber the speed slows to about 2/3 of this value and becomes slightly frequency dependent. The speed of light is the ultimate speed limit. No object or signal can ever move faster than it.

The electromagnetic spectrum is shown below. The radio, microwave, infrared, and visible light portions of the spectrum can all be used for transmitting information by modulating the amplitude, frequency, or phase of the waves. Ultraviolet light, X-rays, and gamma rays would be even better, due to their higher frequencies, but they are hard to produce and modulate, do not propagate well through buildings, and are dangerous to living things. The bands listed at the bottom of the figure are the official ITU names and are based on the

wavelengths, so the LF band goes from 1 km to 10 km (approximately 30 kHz to 300 kHz). The terms LF, MF, and HF refer to low, medium, and high frequency, respectively. Clearly, when the names were assigned, nobody expected to go above 10 MHz, so the higher bands were later named the Very, Ultra, Super, Extremely, and Tremendously High Frequency bands. Beyond that there are no names, but Incredibly, Astonishingly, and Prodigiously high frequency (IHF, AHF, and PHF) would sound nice.

The electromagnetic spectrum and its uses for communication



In terms of frequency, the **visible light** spectrum corresponds to a band in the vicinity of 400-792 THz.

13.2.2 radio transmission

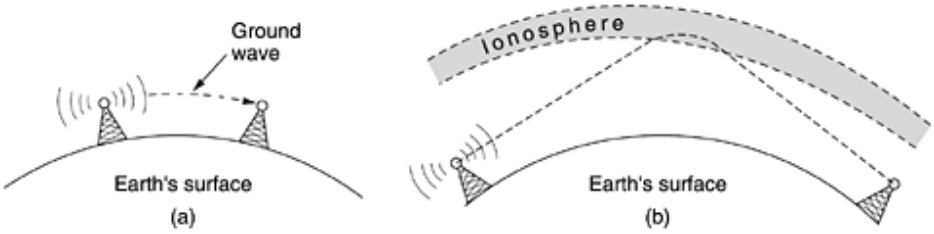
Radio waves are easy to generate, can travel long distances, and can penetrate buildings easily, so they are widely used for communication, both indoors and outdoors. Radio waves also are omnidirectional, meaning that they travel in all directions from the source, so the transmitter and receiver do not have to be carefully aligned physically.

The properties of radio waves are frequency dependent. At low frequencies, radio waves pass through obstacles well, but the power falls off sharply with distance from the source, roughly as $1/r^2$ in air. At high frequencies, radio waves tend to travel in straight lines and bounce off obstacles. They are also absorbed by rain. At all frequencies, radio waves are subject to interference from motors and other electrical equipment.

Due to radio's ability to travel long distances, interference between users is a problem. For this reason, all governments tightly license the use of radio transmitters, with one exception, discussed below.

In the VLF, LF, and MF bands, radio waves follow the ground, as illustrated below. These waves can be detected for perhaps 1000 km at the lower frequencies, less at the higher ones. AM radio broadcasting uses the MF band, which is why the ground waves from Boston AM radio stations cannot be heard easily in New York. Radio waves in these bands pass through buildings easily, which is why portable radios work indoors. The main problem with using these bands for data communication is their low bandwidth.

(a) In the VLF, LF, and MF bands, radio waves follow the curvature of the earth. (b) In the HF band, they bounce off the ionosphere



In the HF and VHF bands, the ground waves tend to be absorbed by the earth. However, the waves that reach the ionosphere, a layer of charged particles circling the earth at a height of 100 to 500 km, are refracted by it and sent back to earth, as shown above. Under certain atmospheric conditions, the signals can bounce several times. Amateur radio operators (hams) use these bands to talk long distance. The military also communicate in the HF and VHF bands.

13.2.3 microwave transmission

Above 100 MHz, the waves travel in nearly straight lines and can therefore be narrowly focused. Concentrating all the energy into a small beam by means of a parabolic antenna (like the familiar satellite TV dish) gives a much higher signal-to-noise ratio, but the transmitting and receiving antennas must be accurately aligned with each other. In addition, this directionality allows multiple transmitters lined up in a row to communicate with multiple receivers in a row without interference, provided some minimum spacing rules are observed. Before fiber optics, for decades these microwaves formed the heart of the long-distance telephone transmission system. In fact, MCI, one of AT&T's first competitors after it was deregulated, built its entire system with microwave communications going from tower to tower tens of kilometers apart. Even the company's name reflected this (MCI stood for Microwave Communications, Inc.). MCI has since gone over to fiber and merged with WorldCom.

Since the microwaves travel in a straight line, if the towers are too far apart, the earth will get in the way (think about a San Francisco to Amsterdam link). Consequently, repeaters are needed periodically. The higher the towers are, the farther apart they can be. The distance between repeaters goes up very roughly with the square root of the tower height. For 100-meter-high towers, repeaters can be spaced 80 km apart.

Unlike radio waves at lower frequencies, microwaves do not pass through buildings well. In addition, even though the beam may be well focused at the transmitter, there is still some divergence in space. Some waves may be refracted off low-lying atmospheric layers and may take slightly longer to arrive than the direct waves. The delayed waves may arrive out of phase with the direct wave and thus cancel the signal. This effect is called multipath fading and is often a serious problem. It is weather and frequency dependent. Some operators keep 10 percent of their channels idle as spares to switch on when multipath fading wipes out some frequency band temporarily.

In summary, microwave communication is so widely used for long-distance telephone communication, mobile phones, television distribution, and other uses that a severe shortage of spectrum has developed. It has several significant advantages over fiber. The main one is that no right of way is needed, and by buying a small plot of ground every 50 km and putting a microwave tower on it, one can bypass the telephone system and communicate directly. This is how MCI managed to get started as a new long-distance telephone company so quickly. (Sprint went a completely different route: it was formed by the Southern Pacific Railroad, which already owned a large amount of right of way and just buried fiber next to the tracks.)

13.2.4 infrared and millimeter waves

Unguided infrared and millimeter waves are widely used for short-range communication. The remote controls used on televisions, VCRs, and stereos all use infrared communication. They are relatively

directional, cheap, and easy to build but have a major drawback: they do not pass through solid objects (try standing between your remote control and your television and see if it still works). In general, as we go from long-wave radio toward visible light, the waves behave more and more like light and less and less like radio.

On the other hand, the fact that infrared waves do not pass through solid walls well is also a plus. It means that an infrared system in one room of a building will not interfere with a similar system in adjacent rooms or buildings: you cannot control your neighbor's television with your remote control. Furthermore, security of infrared systems against eavesdropping is better than that of radio systems precisely for this reason. Therefore, no government license is needed to operate an infrared system, in contrast to radio systems, which must be licensed outside the ISM bands. Infrared communication has a limited use on the desktop, for example, connecting notebook computers and printers, but it is not a major player in the communication game.

13.2.5 lightwave transmission

Unguided optical signaling has been in use for centuries. Paul Revere used binary optical signaling from the Old North Church just prior to his famous ride. A more modern application is to connect the LANs in two buildings via lasers mounted on their rooftops. Coherent optical signaling using lasers is inherently unidirectional, so each building needs its own laser and its own photodetector. This scheme offers very high bandwidth and very low cost. It is also relatively easy to install and, unlike microwave, does not require an FCC license.

The laser's strength, a very narrow beam, is also its weakness here. Aiming a laser beam 1-mm wide at a target the size of a pin head 500 meters away requires the marksmanship of a latter-day Annie Oakley. Usually, lenses are put into the system to defocus the beam slightly.

13.3 communication satellites

In the 1950s and early 1960s, people tried to set up communication systems by bouncing signals off metallized weather balloons. Unfortunately, the received signals were too weak to be of any practical use. Then the U.S. Navy noticed a kind of permanent weather balloon in the sky—the moon—and built an operational system for ship-to-shore communication by bouncing signals off it.

Further progress in the celestial communication field had to wait until the first communication satellite was launched. The key difference between an artificial satellite and a real one is that the artificial one can amplify the signals before sending them back, turning a strange curiosity into a powerful communication system.

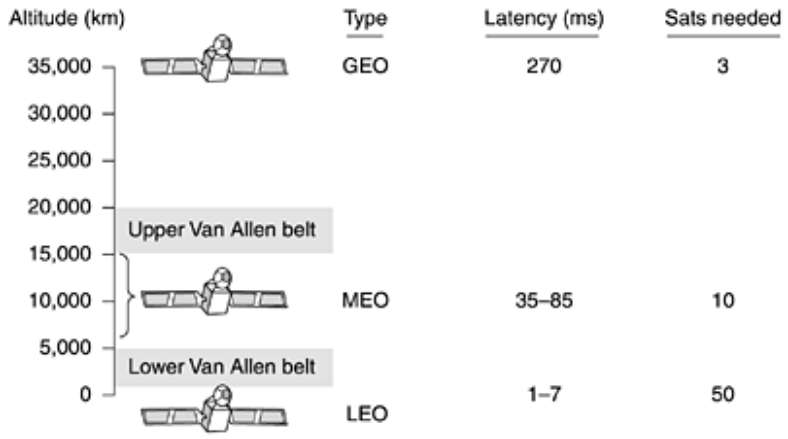
Communication satellites have some interesting properties that make them attractive for many applications. In its simplest form, a communication satellite can be thought of as a big microwave repeater in the sky. It contains several transponders, each of which listens to some portion of the spectrum, amplifies the incoming signal, and then rebroadcasts it at another frequency to avoid interference with the incoming signal. The downward beams can be broad, covering a substantial fraction of the earth's surface, or narrow, covering an area only hundreds of kilometers in diameter. This mode of operation is known as a bent pipe.

According to Kepler's law, the orbital period of a satellite varies as the radius of the orbit to the 3/2 power. The higher the satellite, the longer the period. Near the surface of the earth, the period is about 90 minutes. Consequently, low-orbit satellites pass out of view fairly quickly, so many of them are needed to provide continuous coverage. At an altitude of about 35,800 km, the period is 24 hours. At an altitude of 384,000 km, the period is about one month, as anyone who has observed the moon regularly can testify.

A satellite's period is important, but it is not the only issue in determining where to place it. Another issue is the presence of the Van Allen belts, layers of highly charged particles trapped by the earth's magnetic field. Any satellite flying within them would be destroyed fairly quickly by the highly-energetic charged particles trapped there by the earth's magnetic field. These factors lead to three regions in which satellites can be

placed safely. These regions and some of their properties are illustrated below.

Communication satellites and some of their properties, including altitude above the earth, round-trip delay time, and number of satellites needed for global coverage



13.3.1 geostationary satellites

In 1945, the science fiction writer Arthur C. Clarke calculated that a satellite at an altitude of 35,800 km in a circular equatorial orbit would appear to remain motionless in the sky. so it would not need to be tracked (Clarke, 1945). He went on to describe a complete communication system that used these (manned) geostationary satellites, including the orbits, solar panels, radio frequencies, and launch procedures. Unfortunately, he concluded that satellites were impractical due to the impossibility of putting power-hungry, fragile, vacuum tube amplifiers into orbit, so he never pursued this idea further, although he wrote some science fiction stories about it.

The invention of the transistor changed all that, and the first artificial communication satellite, Telstar, was launched in July 1962. Since then, communication satellites have become a multibillion dollar business and the only aspect of outer space that has become highly profitable. These high-flying satellites are often called GEO (Geostationary Earth Orbit) satellites.

With current technology, it is unwise to have geostationary satellites spaced much closer than 2 degrees in the 360-degree equatorial plane, to avoid interference. With a spacing of 2 degrees, there can only be 360/2 = 180 of these satellites in the sky at once. However, each transponder can use multiple frequencies and polarizations to increase the available bandwidth.

To prevent total chaos in the sky, orbit slot allocation is done by ITU. This process is highly political, with countries barely out of the stone age demanding "their" orbit slots (for the purpose of leasing them to the highest bidder). Other countries, however, maintain that national property rights do not extend up to the moon and that no country has a legal right to the orbit slots above its territory. To add to the fight, commercial telecommunication is not the only application. Television broadcasters, governments, and the military also want a piece of the orbiting pie.

Modern satellites can be quite large, weighing up to 4000 kg and consuming several kilowatts of electric power produced by the solar panels. The effects of solar, lunar, and planetary gravity tend to move them away from their assigned orbit slots and orientations, an effect countered by on-board rocket motors. This fine-tuning activity is called station keeping. However, when the fuel for the motors has been exhausted, typically in about 10 years, the satellite drifts and tumbles helplessly, so it has to be turned off. Eventually, the

orbit decays and the satellite reenters the atmosphere and burns up or occasionally crashes to earth.

13.3.2 medium-earth orbit satellites

At much lower altitudes, between the two Van Allen belts, we find the MEO (Medium-Earth Orbit) satellites. As viewed from the earth, these drift slowly in longitude, taking something like 6 hours to circle the earth. Accordingly, they must be tracked as they move through the sky. Because they are lower than the GEOs, they have a smaller footprint on the ground and require less powerful transmitters to reach them. Currently they are not used for telecommunications, so we will not examine them further here. The 24 GPS (Global Positioning System) satellites orbiting at about 18,000 km are examples of MEO satellites.

13.3.3 low-earth orbit satellites

Moving down in altitude, we come to the LEO (Low-Earth Orbit) satellites. Due to their rapid motion, large numbers of them are needed for a complete system. On the other hand, because the satellites are so close to the earth, the ground stations do not need much power, and the round-trip delay is only a few milliseconds.

13.3.4 satellites versus fiber

A comparison between satellite communication and terrestrial communication is instructive. As recently as 20 years ago, a case could be made that the future of communication lay with communication satellites. After all, the telephone system had changed little in the past 100 years and showed no signs of changing in the next 100 years. This glacial movement was caused in no small part by the regulatory environment in which the telephone companies were expected to provide good voice service at reasonable prices (which they did), and in return got a guaranteed profit on their investment. For people with data to transmit, 1200-bps modems were available. That was pretty much all there was.

The introduction of competition in 1984 in the United States and somewhat later in Europe changed all that radically. Telephone companies began replacing their long-haul networks with fiber and introduced high-bandwidth services like ADSL (Asymmetric Digital Subscriber Line). They also stopped their long-time practice of charging artificially-high prices to long-distance users to subsidize local service.

All of a sudden, terrestrial fiber connections looked like the long-term winner. Nevertheless, communication satellites have some major niche markets that fiber does not (and, sometimes, cannot) address. We will now look at a few of these.

First, while a single fiber has, in principle, more potential bandwidth than all the satellites ever launched, this bandwidth is not available to most users. The fibers that are now being installed are used within the telephone system to handle many long distance calls at once, not to provide individual users with high bandwidth. With satellites, it is practical for a user to erect an antenna on the roof of the building and completely bypass the telephone system to get high bandwidth. Teledesic is based on this idea.

A second niche is for mobile communication. Many people nowadays want to communicate while jogging, driving, sailing, and flying. Terrestrial fiber optic links are of no use to them, but satellite links potentially are. It is possible, however, that a combination of cellular radio and fiber will do an adequate job for most users (but probably not for those airborne or at sea).

A third niche is for situations in which broadcasting is essential. A message sent by satellite can be received by thousands of ground stations at once. For example, an organization transmitting a stream of stock, bond, or commodity prices to thousands of dealers might find a satellite system to be much cheaper than simulating broadcasting on the ground.

A fourth niche is for communication in places with hostile terrain or a poorly developed terrestrial

infrastructure. Indonesia, for example, has its own satellite for domestic telephone traffic. Launching one satellite was cheaper than stringing thousands of undersea cables among the 13,677 islands in the archipelago.

A fifth niche market for satellites is to cover areas where obtaining the right of way for laying fiber is difficult or unduly expensive.

Sixth, when rapid deployment is critical, as in military communication systems in time of war, satellites win easily.

In short, it looks like the mainstream communication of the future will be terrestrial fiber optics combined with cellular radio, but for some specialized uses, satellites are better. However, there is one caveat that applies to all of this: economics. Although fiber offers more bandwidth, it is certainly possible that terrestrial and satellite communication will compete aggressively on price. If advances in technology radically reduce the cost of deploying a satellite (e.g., some future space shuttle can toss out dozens of satellites on one launch) or low-orbit satellites catch on in a big way, it is not certain that fiber will win in all markets.

13.4 the public switched telephone network

When two computers owned by the same company or organization and located close to each other need to communicate, it is often easiest just to run a cable between them. LANs work this way. However, when the distances are large or there are many computers or the cables have to pass through a public road or other public right of way, the costs of running private cables are usually prohibitive. Furthermore, in just about every country in the world, stringing private transmission lines across (or underneath) public property is also illegal. Consequently, the network designers must rely on the existing telecommunication facilities.

These facilities, especially the PSTN (Public Switched Telephone Network), were usually designed many years ago, with a completely different goal in mind: transmitting the human voice in a more-or-less recognizable form. Their suitability for use in computer-computer communication is often marginal at best, but the situation is rapidly changing with the introduction of fiber optics and digital technology.

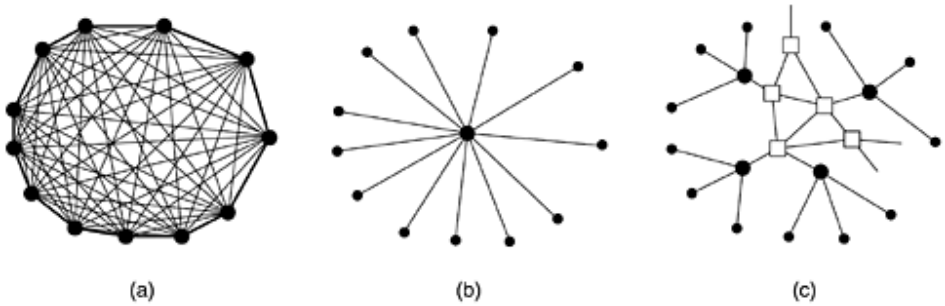
To see the order of magnitude of the problem, let us make a rough but illustrative comparison of the properties of a typical computer-computer connection via a local cable and via a dial-up telephone line. A cable running between two computers can transfer data at 10^9 bps, maybe more. In contrast, a dial-up line has a maximum data rate of 56 kbps, a difference of a factor of almost 20,000. That is the difference between a duck waddling leisurely through the grass and a rocket to the moon. If the dial-up line is replaced by an ADSL connection, there is still a factor of 1000–2000 difference.

The trouble, of course, is that computer systems designers are used to working with computer systems and when suddenly confronted with another system whose performance (from their point of view) is 3 or 4 orders of magnitude worse, they, not surprising, devoted much time and effort to trying to figure out how to use it efficiently. In the following sections we will describe the telephone system and show how it works. For additional information about the innards of the telephone system see (Bellamy, 2000).

13.5 structure of the telephone system

Soon after Alexander Graham Bell patented the telephone in 1876 (just a few hours ahead of his rival, Elisha Gray), there was an enormous demand for his new invention. The initial market was for the sale of telephones, which came in pairs. It was up to the customer to string a single wire between them. The electrons returned through the earth. If a telephone owner wanted to talk to n other telephone owners, separate wires had to be strung to all n houses. Within a year, the cities were covered with wires passing over houses and trees in a wild jumble. It became immediately obvious that the model of connecting every telephone to every other telephone, as shown below, was not going to work.

(a) Fully-interconnected network. (b) Centralized switch. (c) Two-level hierarchy



To his credit, Bell saw this and formed the Bell Telephone Company, which opened its first switching office (in New Haven, Connecticut) in 1878. The company ran a wire to each customer's house or office. To make a call, the customer would crank the phone to make a ringing sound in the telephone company office to attract the attention of an operator, who would then manually connect the caller to the callee by using a jumper cable. The model of a single switching office is illustrated above in part (b).

Pretty soon, Bell System switching offices were springing up everywhere and people wanted to make long-distance calls between cities, so the Bell system began to connect the switching offices. The original problem soon returned: to connect every switching office to every other switching office by means of a wire between them quickly became unmanageable, so second-level switching offices were invented. After a while, multiple second-level offices were needed, as illustrated above in part (c). Eventually, the hierarchy grew to five levels.

By 1890, the three major parts of the telephone system were in place: the switching offices, the wires between the customers and the switching offices (by now balanced, insulated, twisted pairs instead of open wires with an earth return), and the long-distance connections between the switching offices. While there have been improvements in all three areas since then, the basic Bell System model has remained essentially intact for over 100 years. For a short technical history of the telephone system, see (Hawley, 1991).

A variety of transmission media are used for telecommunication. Local loops consist of category 3 twisted pairs nowadays, although in the early days of telephony, uninsulated wires spaced 25 cm apart on telephone poles were common. Between switching offices, coaxial cables, microwaves, and especially fiber optics are widely used.

In the past, transmission throughout the telephone system was analog, with the actual voice signal being transmitted as an electrical voltage from source to destination. With the advent of fiber optics, digital electronics, and computers, all the trunks and switches are now digital, leaving the local loop as the last piece of analog technology in the system. Digital transmission is preferred because it is not necessary to accurately reproduce an analog waveform after it has passed through many amplifiers on a long call. Being able to correctly distinguish a 0 from a 1 is enough. This property makes digital transmission more reliable than analog. It is also cheaper and easier to maintain.

In summary, the telephone system consists of three major components:

1. Local loops (analog twisted pairs going into houses and businesses).
2. Trunks (digital fiber optics connecting the switching offices).
3. Switching offices (where calls are moved from one trunk to another).

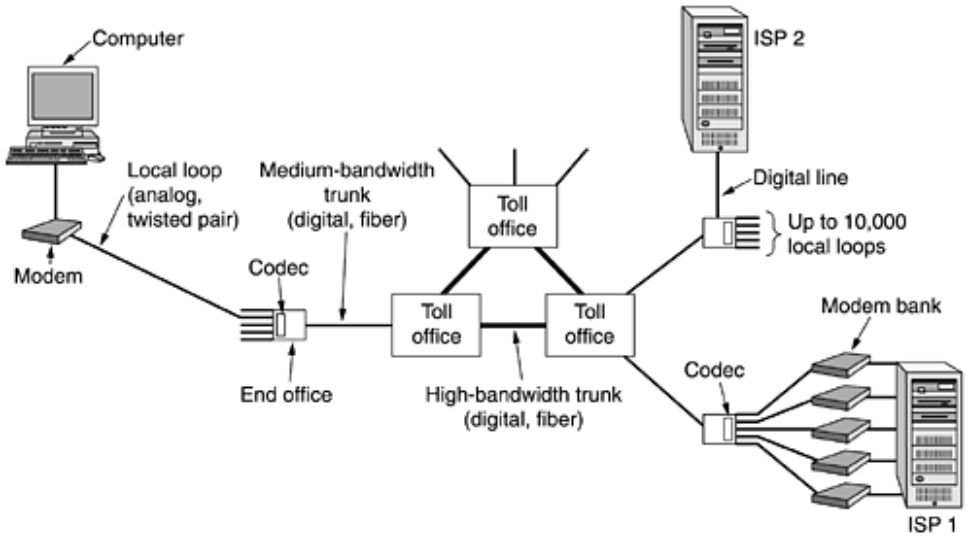
After a short digression on the politics of telephones, we will come back to each of these three components in some detail. The local loops provide everyone access to the whole system, so they are critical. Unfortunately, they are also the weakest link in the system. For the long-haul trunks, the main issue is how to collect multiple calls together and send them out over the same fiber. This subject is called multiplexing, and we will study three different ways to do it. Finally, there are two fundamentally different ways

of doing switching; we will look at both.

13.6 the local loop: modems, ADSL and wireless

It is now time to start our detailed study of how the telephone system works. The main parts of the system are illustrated below. Here we see the local loops, the trunks, and the toll offices and end offices, both of which contain switching equipment that switches calls. An end office has up to 10,000 local loops (in the U.S. and other large countries). In fact, until recently, the area code + exchange indicated the end office, so (212) 601-xxxx was a specific end office with 10,000 subscribers, numbered 0000 through 9999. With the advent of competition for local service, this system was no longer tenable because multiple companies wanted to own the end office code. Also, the number of codes was basically used up, so complex mapping schemes had to be introduced.

The use of both analog and digital transmission for a computer to computer call. Conversion is done by the modems and codecs



Let us begin with the part that most people are familiar with: the two-wire local loop coming from a telephone company end office into houses and small businesses. The local loop is also frequently referred to as the "last mile," although the length can be up to several miles. It has used analog signaling for over 100 years and is likely to continue doing so for some years to come, due to the high cost of converting to digital. Nevertheless, even in this last bastion of analog transmission, change is taking place. In this section we will study the traditional local loop and the new developments taking place here, with particular emphasis on data communication from home computers.

When a computer wishes to send digital data over an analog dial-up line, the data must first be converted to analog form for transmission over the local loop. This conversion is done by a device called a modem, something we will study shortly. At the telephone company end office the data are converted to digital form for transmission over the long-haul trunks.

If the other end is a computer with a modem, the reverse conversion—digital to analog—is needed to traverse the local loop at the destination. This arrangement is shown in Fig. 2-23 for ISP 1 (Internet Service Provider), which has a bank of modems, each connected to a different local loop. This ISP can handle as

many connections as it has modems (assuming its server or servers have enough computing power). This arrangement was the normal one until 56-kbps modems appeared, for reasons that will become apparent shortly.

Analog signaling consists of varying a voltage with time to represent an information stream. If transmission media were perfect, the receiver would receive exactly the same signal that the transmitter sent. Unfortunately, media are not perfect, so the received signal is not the same as the transmitted signal. For digital data, this difference can lead to errors.

Transmission lines suffer from three major problems: attenuation, delay distortion, and noise. Attenuation is the loss of energy as the signal propagates outward. The loss is expressed in decibels per kilometer. The amount of energy lost depends on the frequency. To see the effect of this frequency dependence, imagine a signal not as a simple waveform, but as a series of Fourier components. Each component is attenuated by a different amount, which results in a different Fourier spectrum at the receiver.

To make things worse, the different Fourier components also propagate at different speeds in the wire. This speed difference leads to distortion of the signal received at the other end.

Another problem is noise, which is unwanted energy from sources other than the transmitter. Thermal noise is caused by the random motion of the electrons in a wire and is unavoidable. Crosstalk is caused by inductive coupling between two wires that are close to each other. Sometimes when talking on the telephone, you can hear another conversation in the background. That is crosstalk. Finally, there is impulse noise, caused by spikes on the power line or other causes. For digital data, impulse noise can wipe out one or more bits.

13.6.1 modems

Due to the problems just discussed, especially the fact that both attenuation and propagation speed are frequency dependent, it is undesirable to have a wide range of frequencies in the signal. Unfortunately, the square waves used in digital signals have a wide frequency spectrum and thus are subject to strong attenuation and delay distortion. These effects make baseband (DC) signaling unsuitable except at slow speeds and over short distances.

To get around the problems associated with DC signaling, especially on telephone lines, AC signaling is used. A continuous tone in the 1000 to 2000-Hz range, called a sine wave carrier, is introduced. Its amplitude, frequency, or phase can be modulated to transmit information. In amplitude modulation, two different amplitudes are used to represent 0 and 1, respectively. In frequency modulation, also known as frequency shift keying, two (or more) different tones are used. (The term keying is also widely used in the industry as a synonym for modulation.) In the simplest form of phase modulation, the carrier wave is systematically shifted 0 or 180 degrees at uniformly spaced intervals. A better scheme is to use shifts of 45, 135, 225, or 315 degrees to transmit 2 bits of information per time interval. Also, always requiring a phase shift at the end of every time interval, makes it is easier for the receiver to recognize the boundaries of the time intervals.

13.6.2 digital subscriber lines

When the telephone industry finally got to 56 kbps, it patted itself on the back for a job well done. Meanwhile, the cable TV industry was offering speeds up to 10 Mbps on shared cables, and satellite companies were planning to offer upward of 50 Mbps. As Internet access became an increasingly important part of their business, the telephone companies (LECs) began to realize they needed a more competitive product. Their answer was to start offering new digital services over the local loop. Services with more bandwidth than standard telephone service are sometimes called broadband, although the term really is more of a marketing concept than a specific technical concept.

Initially, there were many overlapping offerings, all under the general name of xDSL (Digital Subscriber Line), for various x. Below we will discuss these but primarily focus on what is probably going to become the

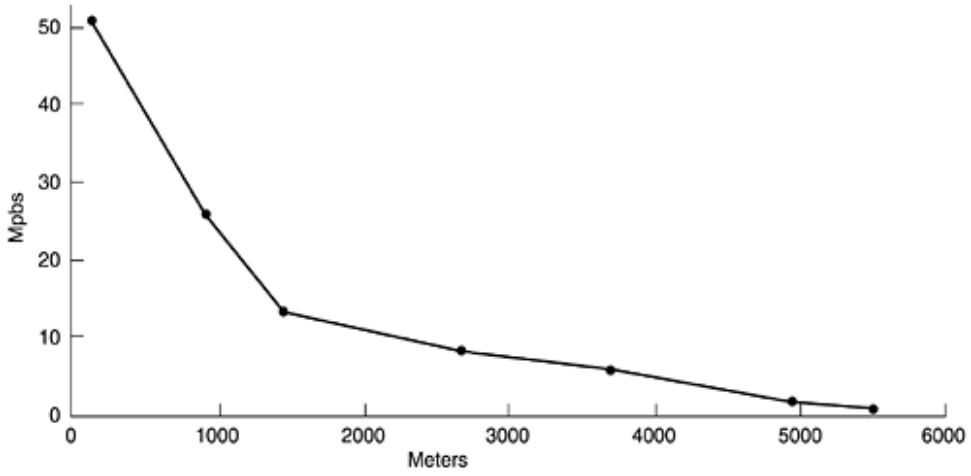
most popular of these services, ADSL (Asymmetric DSL). Since ADSL is still being developed and not all the standards are fully in place, some of the details given below may change in time, but the basic picture should remain valid. For more information about ADSL, see (Summers, 1999; and Vetter et al., 2000).

The reason that modems are so slow is that telephones were invented for carrying the human voice and the entire system has been carefully optimized for this purpose. Data have always been stepchildren. At the point where each local loop terminates in the end office, the wire runs through a filter that attenuates all frequencies below 300 Hz and above 3400 Hz. The cutoff is not sharp—300 Hz and 3400 Hz are the 3 dB points—so the bandwidth is usually quoted as 4000 Hz even though the distance between the 3 dB points is 3100 Hz. Data are thus also restricted to this narrow band.

The trick that makes xDSL work is that when a customer subscribes to it, the incoming line is connected to a different kind of switch, one that does not have this filter, thus making the entire capacity of the local loop available. The limiting factor then becomes the physics of the local loop, not the artificial 3100 Hz bandwidth created by the filter.

Unfortunately, the capacity of the local loop depends on several factors, including its length, thickness, and general quality. A plot of the potential bandwidth as a function of distance is given below. This figure assumes that all the other factors are optimal (new wires, modest bundles, etc.).

Bandwidth versus distance over category 3 UTP for DSL



13.6.3 wireless local loops

Since 1996 in the U.S. and a bit later in other countries, companies that wish to compete with the entrenched local telephone company (the former monopolist), called an ILEC (Incumbent LEC), are free to do so. The most likely candidates are long-distance telephone companies (IXCs). Any IXC wishing to get into the local phone business in some city must do the following things. First, it must buy or lease a building for its first end office in that city. Second, it must fill the end office with telephone switches and other equipment, all of which are available as off-the-shelf products from various vendors. Third, it must run a fiber between the end office and its nearest toll office so the new local customers will have access to its national network. Fourth, it must acquire customers, typically by advertising better service or lower prices than those of the ILEC.

Then the hard part begins. Suppose that some customers actually show up. How is the new local phone company, called a CLEC (Competitive LEC) going to connect customer telephones and computers to its shiny new end office? Buying the necessary rights of way and stringing wires or fibers is prohibitively expensive. Many CLECs have discovered a cheaper alternative to the traditional twisted-pair local loop: the

WLL (Wireless Local Loop).

In a certain sense, a fixed telephone using a wireless local loop is a bit like a mobile phone, but there are three crucial technical differences. First, the wireless local loop customer often wants high-speed Internet connectivity, often at speeds at least equal to ADSL. Second, the new customer probably does not mind having a CLEC technician install a large directional antenna on his roof pointed at the CLEC's end office. Third, the user does not move, eliminating all the problems with mobility and cell handoff that we will study later in this chapter. And thus a new industry is born: fixed wireless (local telephone and Internet service run by CLECs over wireless local loops).

Although WLLs began serious operation in 1998, we first have to go back to 1969 to see the origin. In that year the FCC allocated two television channels (at 6 MHz each) for instructional television at 2.1 GHz. In subsequent years, 31 more channels were added at 2.5 GHz for a total of 198 MHz.

Instructional television never took off and in 1998, the FCC took the frequencies back and allocated them to two-way radio. They were immediately seized upon for wireless local loops. At these frequencies, the microwaves are 10–12 cm long. They have a range of about 50 km and can penetrate vegetation and rain moderately well. The 198 MHz of new spectrum was immediately put to use for wireless local loops as a service called MMDS (Multichannel Multipoint Distribution Service). MMDS can be regarded as a MAN (Metropolitan Area Network), as can its cousin LMDS (discussed below).

The big advantage of this service is that the technology is well established and the equipment is readily available. The disadvantage is that the total bandwidth available is modest and must be shared by many users over a fairly large geographic area.

13.7 trunks and multiplexing

Economies of scale play an important role in the telephone system. It costs essentially the same amount of money to install and maintain a high-bandwidth trunk as a low-bandwidth trunk between two switching offices (i.e., the costs come from having to dig the trench and not from the copper wire or optical fiber). Consequently, telephone companies have developed elaborate schemes for multiplexing many conversations over a single physical trunk. These multiplexing schemes can be divided into two basic categories: FDM (Frequency Division Multiplexing) and TDM (Time Division Multiplexing). In FDM, the frequency spectrum is divided into frequency bands, with each user having exclusive possession of some band. In TDM, the users take turns (in a round-robin fashion), each one periodically getting the entire bandwidth for a little burst of time.

AM radio broadcasting provides illustrations of both kinds of multiplexing. The allocated spectrum is about 1 MHz, roughly 500 to 1500 kHz. Different frequencies are allocated to different logical channels (stations), each operating in a portion of the spectrum, with the interchannel separation great enough to prevent interference. This system is an example of frequency division multiplexing. In addition (in some countries), the individual stations have two logical subchannels: music and advertising. These two alternate in time on the same frequency, first a burst of music, then a burst of advertising, then more music, and so on. This situation is time division multiplexing.

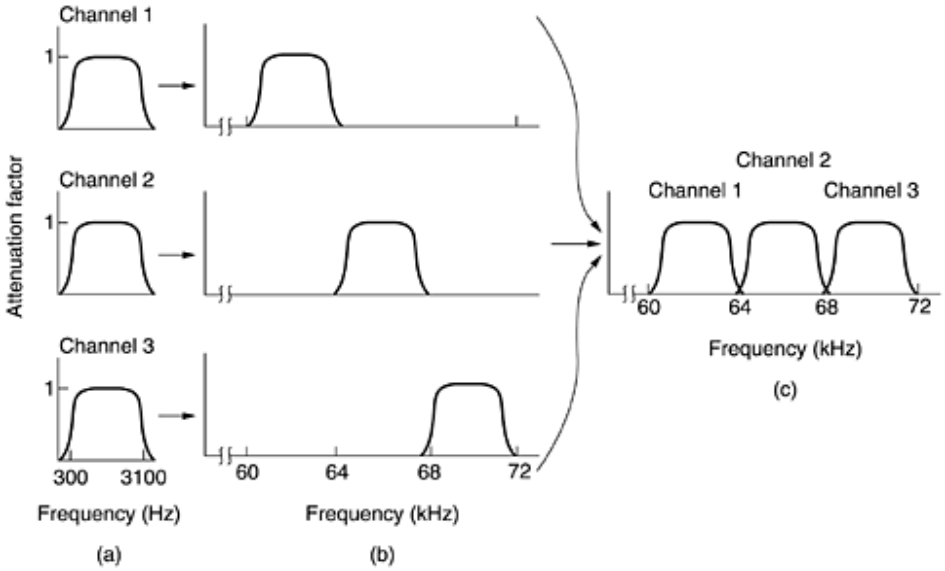
Below we will examine frequency division multiplexing. After that we will see how FDM can be applied to fiber optics (wavelength division multiplexing). Then we will turn to TDM, and end with an advanced TDM system used for fiber optics (SONET).

13.7.1 frequency division multiplexing

The figure below shows how three voice-grade telephone channels are multiplexed using FDM. Filters limit the usable bandwidth to about 3100 Hz per voice-grade channel. When many channels are multiplexed together, 4000 Hz is allocated to each channel to keep them well separated. First the voice channels are raised in frequency, each by a different amount. Then they can be combined because no two channels now

occupy the same portion of the spectrum. Notice that even though there are gaps (guard bands) between the channels, there is some overlap between adjacent channels because the filters do not have sharp edges. This overlap means that a strong spike at the edge of one channel will be felt in the adjacent one as nonthermal noise.

Frequency division multiplexing. (a) The original bandwidths. (b) The bandwidths raised in frequency. (c) The multiplexed channel

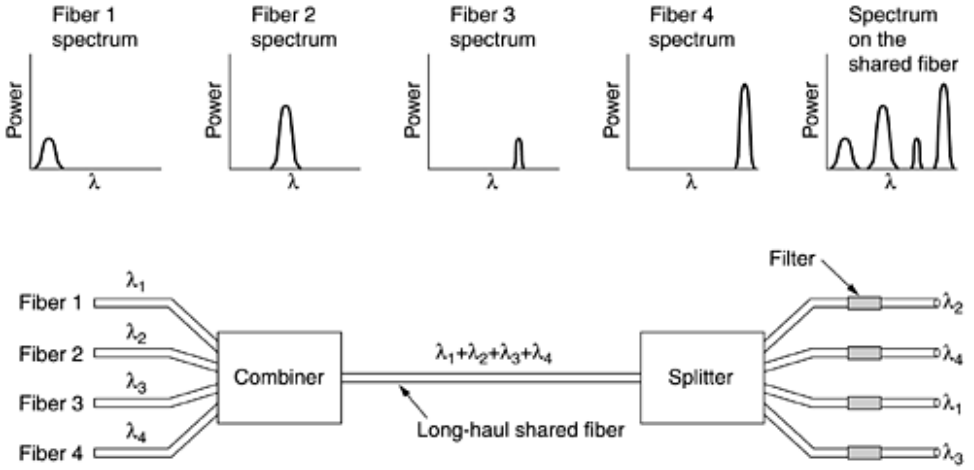


The FDM schemes used around the world are to some degree standardized. A widespread standard is twelve 4000-Hz voice channels multiplexed into the 60 to 108 kHz band. This unit is called a group. The 12-kHz to 60-kHz band is sometimes used for another group. Many carriers offer a 48- to 56-kbps leased line service to customers, based on the group. Five groups (60 voice channels) can be multiplexed to form a supergroup. The next unit is the mastergroup, which is five supergroups (CCITT standard) or ten supergroups (Bell system). Other standards of up to 230,000 voice channels also exist.

13.7.2 wavelength division multiplexing

For fiber optic channels, a variation of frequency division multiplexing is used. It is called WDM (Wavelength Division Multiplexing). The basic principle of WDM on fibers is depicted below. Here four fibers come together at an optical combiner, each with its energy present at a different wavelength. The four beams are combined onto a single shared fiber for transmission to a distant destination. At the far end, the beam is split up over as many fibers as there were on the input side. Each output fiber contains a short, specially-constructed core that filters out all but one wavelength. The resulting signals can be routed to their destination or recombined in different ways for additional multiplexed transport.

Wavelength division multiplexing



There is really nothing new here. This is just frequency division multiplexing at very high frequencies. As long as each channel has its own frequency (i.e., wavelength) range and all the ranges are disjoint, they can be multiplexed together on the long-haul fiber. The only difference with electrical FDM is that an optical system using a diffraction grating is completely passive and thus highly reliable.

WDM technology has been progressing at a rate that puts computer technology to shame. WDM was invented around 1990. The first commercial systems had eight channels of 2.5 Gbps per channel. By 1998, systems with 40 channels of 2.5 Gbps were on the market. By 2001, there were products with 96 channels of 10 Gbps, for a total of 960 Gbps. This is enough bandwidth to transmit 30 full-length movies per second (in MPEG-2). Systems with 200 channels are already working in the laboratory. When the number of channels is very large and the wavelengths are spaced close together, for example, 0.1 nm, the system is often referred to as DWDM (Dense WDM).

It should be noted that the reason WDM is popular is that the energy on a single fiber is typically only a few gigahertz wide because it is currently impossible to convert between electrical and optical media any faster. By running many channels in parallel on different wavelengths, the aggregate bandwidth is increased linearly with the number of channels. Since the bandwidth of a single fiber band is about 25,000 GHz, there is theoretically room for 2500 10-Gbps channels even at 1 bit/Hz (and higher rates are also possible).

Another new development is all optical amplifiers. Previously, every 100 km it was necessary to split up all the channels and convert each one to an electrical signal for amplification separately before reconverting to optical and combining them. Nowadays, all optical amplifiers can regenerate the entire signal once every 1000 km without the need for multiple opto-electrical conversions.

In the above example, we have a fixed wavelength system. Bits from input fiber 1 go to output fiber 3, bits from input fiber 2 go to output fiber 1, etc. However, it is also possible to build WDM systems that are switched. In such a device, the output filters are tunable using Fabry-Perot or Mach-Zehnder interferometers. For more information about WDM and its application to Internet packet switching, see (Elmirghani and Mouftah, 2000; Hunter and Andonovic, 2000; and Listani et al., 2001).

13.7.3 time division multiplexing

WDM technology is wonderful, but there is still a lot of copper wire in the telephone system, so let us turn back to it for a while. Although FDM is still used over copper wires or microwave channels, it requires analog circuitry and is not amenable to being done by a computer. In contrast, TDM can be handled entirely by digital electronics, so it has become far more widespread in recent years. Unfortunately, it can only be used for digital data. Since the local loops produce analog signals, a conversion is needed from analog to digital in the end office, where all the individual local loops come together to be combined onto outgoing trunks.

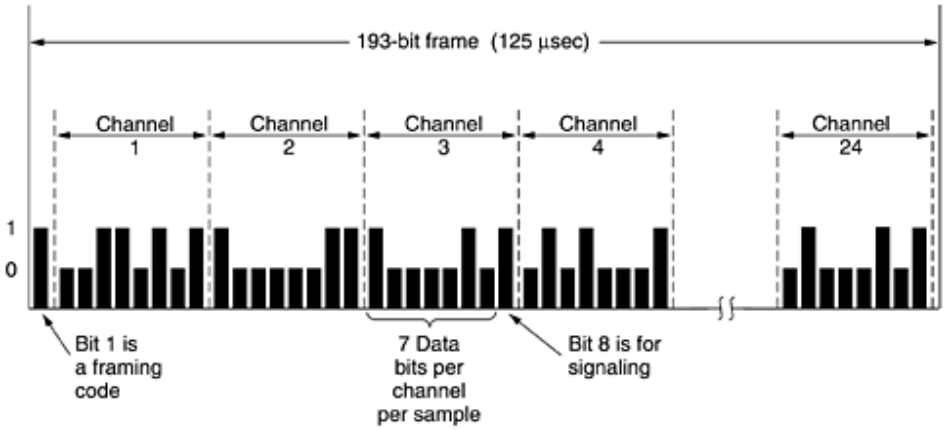
We will now look at how multiple analog voice signals are digitized and combined onto a single outgoing digital trunk. Computer data sent over a modem are also analog, so the following description also applies to

them. The analog signals are digitized in the end office by a device called a codec (coder-decoder), producing a series of 8-bit numbers. The codec makes 8000 samples per second (125 µsec/sample) because the Nyquist theorem says that this is sufficient to capture all the information from the 4-kHz telephone channel bandwidth. At a lower sampling rate, information would be lost; at a higher one, no extra information would be gained. This technique is called PCM (Pulse Code Modulation). PCM forms the heart of the modern telephone system. As a consequence, virtually all time intervals within the telephone system are multiples of 125 µsec.

When digital transmission began emerging as a feasible technology, CCITT was unable to reach agreement on an international standard for PCM. Consequently, a variety of incompatible schemes are now in use in different countries around the world.

The method used in North America and Japan is the T1 carrier, depicted in Fig. 2-33. (Technically speaking, the format is called DS1 and the carrier is called T1, but following widespread industry tradition, we will not make that subtle distinction here.) The T1 carrier consists of 24 voice channels multiplexed together. Usually, the analog signals are sampled on a round-robin basis with the resulting analog stream being fed to the codec rather than having 24 separate codecs and then merging the digital output. Each of the 24 channels, in turn, gets to insert 8 bits into the output stream. Seven bits are data and one is for control, yielding 7 x 8000 = 56,000 bps of data, and 1 x 8000 = 8000 bps of signaling information per channel.

The T1 carrier (1.544 Mbps)



A frame consists of 24 x 8 = 192 bits plus one extra bit for framing, yielding 193 bits every 125 µsec. This gives a gross data rate of 1.544 Mbps. The 193rd bit is used for frame synchronization. It takes on the pattern 0101010101 . . . . Normally, the receiver keeps checking this bit to make sure that it has not lost synchronization. If it does get out of sync, the receiver can scan for this pattern to get resynchronized. Analog customers cannot generate the bit pattern at all because it corresponds to a sine wave at 4000 Hz, which would be filtered out. Digital customers can, of course, generate this pattern, but the odds are against its being present when the frame slips. When a T1 system is being used entirely for data, only 23 of the channels are used for data. The 24th one is used for a special synchronization pattern, to allow faster recovery in the event that the frame slips.

When CCITT finally did reach agreement, they felt that 8000 bps of signaling information was far too much, so its 1.544-Mbps standard is based on an 8- rather than a 7-bit data item; that is, the analog signal is quantized into 256 rather than 128 discrete levels. Two (incompatible) variations are provided. In common-channel signaling, the extra bit (which is attached onto the rear rather than the front of the 193-bit frame) takes on the values 10101010 . . . in the odd frames and contains signaling information for all the channels in the even frames.

In the other variation, channel-associated signaling, each channel has its own private signaling subchannel. A private subchannel is arranged by allocating one of the eight user bits in every sixth frame for signaling purposes, so five out of six samples are 8 bits wide, and the other one is only 7 bits wide. CCITT

also recommended a PCM carrier at 2.048 Mbps called E1. This carrier has 32 8-bit data samples packed into the basic 125-µsec frame. Thirty of the channels are used for information and two are used for signaling. Each group of four frames provides 64 signaling bits, half of which are used for channel-associated signaling and half of which are used for frame synchronization or are reserved for each country to use as it wishes. Outside North America and Japan, the 2.048-Mbps E1 carrier is used instead of T1.

13.8 switching

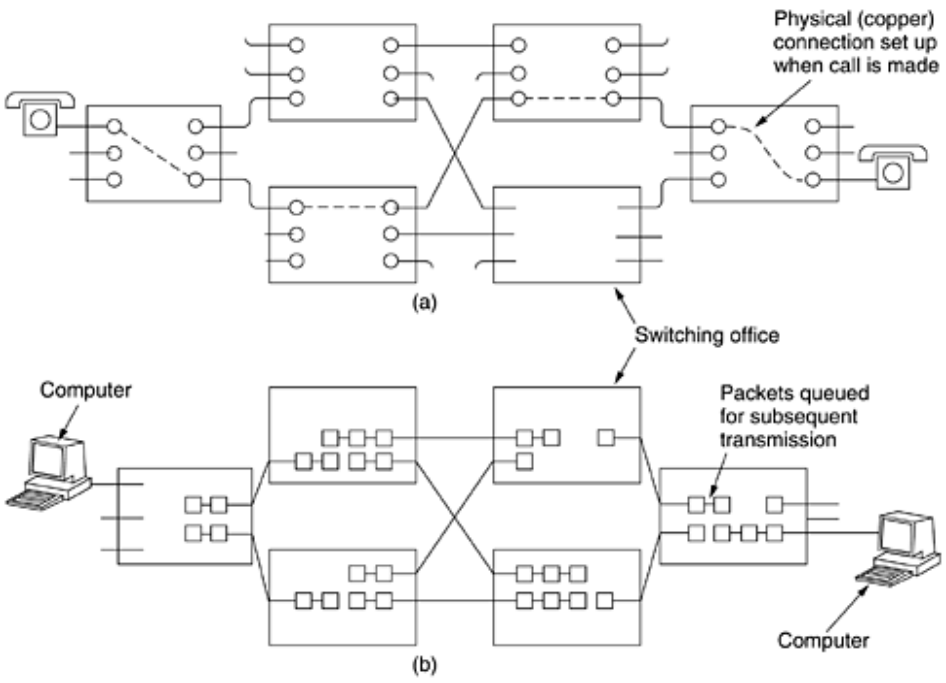
From the point of view of the average telephone engineer, the phone system is divided into two principal parts: outside plant (the local loops and trunks, since they are physically outside the switching offices) and inside plant (the switches), which are inside the switching offices. We have just looked at the outside plant. Now it is time to examine the inside plant.

Two different switching techniques are used nowadays: circuit switching and packet switching. We will give a brief introduction to each of them below. Then we will go into circuit switching in detail because that is how the telephone system works.

13.8.1 circuit switching

When you or your computer places a telephone call, the switching equipment within the telephone system seeks out a physical path all the way from your telephone to the receiver's telephone. This technique is called circuit switching and is shown schematically below. Each of the six rectangles represents a carrier switching office (end office, toll office, etc.). In this example, each office has three incoming lines and three outgoing lines. When a call passes through a switching office, a physical connection is (conceptually) established between the line on which the call came in and one of the output lines, as shown by the dotted lines.

(a) Circuit switching. (b) Packet switching



In the early days of the telephone, the connection was made by the operator plugging a jumper cable into the input and output sockets. In fact, a surprising little story is associated with the invention of automatic circuit switching equipment. It was invented by a 19th century Missouri undertaker named Almon B. Strowger. Shortly after the telephone was invented, when someone died, one of the survivors would call the town operator and say "Please connect me to an undertaker." Unfortunately for Mr. Strowger, there were two undertakers in his town, and the other one's wife was the town telephone operator. He quickly saw that either he was going to have to invent automatic telephone switching equipment or he was going to go out of business. He chose the first option. For nearly 100 years, the circuit-switching equipment used worldwide was known as Strowger gear. (History does not record whether the now-unemployed switchboard operator got a job as an information operator, answering questions such as "What is the phone number of an undertaker?").

13.8.2 message switching

An alternative switching strategy is **message switching**, illustrated also above. When this form of switching is used, no physical path is established in advance between sender and receiver. Instead, when the sender has a block of data to be sent, it is stored in the first switching office (i.e., router) and then forwarded later, one hop at a time. Each block is received in its entirety, inspected for errors, and then retransmitted. A network using this technique is called a store-and-forward network.

The first electromechanical telecommunication systems used message switching, namely, for telegrams. The message was punched on paper tape (off-line) at the sending office, and then read in and transmitted over a communication line to the next office along the way, where it was punched out on paper tape. An operator there tore the tape off and read it in on one of the many tape readers, one reader per outgoing trunk. Such a switching office was called a torn tape office. Paper tape is long gone and message switching is not used any more, so we will not discuss it further in this book.

13.8.3 packet switching

With message switching, there is no limit at all on block size, which means that routers (in a modern system) must have disks to buffer long blocks. It also means that a single block can tie up a router-router line for minutes, rendering message switching useless for interactive traffic. To get around these problems, packet switching was invented. Packet-switching networks place a tight upper limit on block size, allowing packets to be buffered in router main memory instead of on disk. By making sure that no user can monopolize any transmission line very long (milliseconds), packet-switching networks are well suited for handling interactive traffic. A further advantage of packet switching over message switching is that the first packet of a multipacket message can be forwarded before the second one has fully arrived, reducing delay and improving throughput. For these reasons, computer networks are usually packet switched, occasionally circuit switched, but never message switched.

Circuit switching and packet switching differ in many respects. To start with, circuit switching requires that a circuit be set up end to end before communication begins. Packet switching does not require any advance setup. The first packet can just be sent as soon as it is available.

The result of the connection setup with circuit switching is the reservation of bandwidth all the way from the sender to the receiver. All packets follow this path. Among other properties, having all packets follow the same path means that they cannot arrive out of order. With packet switching there is no path, so different packets can follow different paths, depending on network conditions at the time they are sent. They may arrive out of order.

Packet switching is more fault tolerant than circuit switching. In fact, that is why it was invented. If a switch goes down, all of the circuits using it are terminated and no more traffic can be sent on any of them. With packet switching, packets can be routed around dead switches.

Setting up a path in advance also opens up the possibility of reserving bandwidth in advance. If bandwidth is reserved, then when a packet arrives, it can be sent out immediately over the reserved bandwidth.

Having bandwidth reserved in advance means that no congestion can occur when a packet shows up (unless more packets show up than expected). On the other hand, when an attempt is made to establish a circuit, the attempt can fail due to congestion. Thus, congestion can occur at different times with circuit switching (at setup time) and packet switching (when packets are sent).

If a circuit has been reserved for a particular user and there is no traffic to send, the bandwidth of that circuit is wasted. It cannot be used for other traffic. Packet switching does not waste bandwidth and thus is more efficient from a system-wide perspective. Understanding this trade-off is crucial for comprehending the difference between circuit switching and packet switching. The trade-off is between guaranteed service and wasting resources versus not guaranteeing service and not wasting resources.

Another difference is that circuit switching is completely transparent. The sender and receiver can use any bit rate, format, or framing method they want to. The carrier does not know or care. With packet switching, the carrier determines the basic parameters. A rough analogy is a road versus a railroad. In the former, the user determines the size, speed, and nature of the vehicle; in the latter, the carrier does. It is this transparency that allows voice, data, and fax to coexist within the phone system.

A final difference between circuit and packet switching is the charging algorithm. With circuit switching, charging has historically been based on distance and time. For mobile phones, distance usually does not play a role, except for international calls, and time plays only a minor role (e.g., a calling plan with 2000 free minutes costs more than one with 1000 free minutes and sometimes night or weekend calls are cheaper than normal). With packet switching, connect time is not an issue, but the volume of traffic sometimes is. For home users, ISPs usually charge a flat monthly rate because it is less work for them and their customers can understand this model easily, but backbone carriers charge regional networks based on the volume of their traffic. The differences are summarized in Fig. 2-40.

A comparison of circuit-switched and packet-switched networks

Item	Circuit switched	Packet switched
Call setup	Required	Not needed
Dedicated physical path	Yes	No
Each packet follows the same route	Yes	No
Packets arrive in order	Yes	No
Is a switch crash fatal	Yes	No
Bandwidth available	Fixed	Dynamic
Time of possible congestion	At setup time	On every packet
Potentially wasted bandwidth	Yes	No
Store-and-forward transmission	No	Yes
Transparency	Yes	No
Charging	Per minute	Per packet

Both circuit switching and packet switching are important enough that we will come back to them shortly and describe the various technologies used in detail.

13.9 cable television

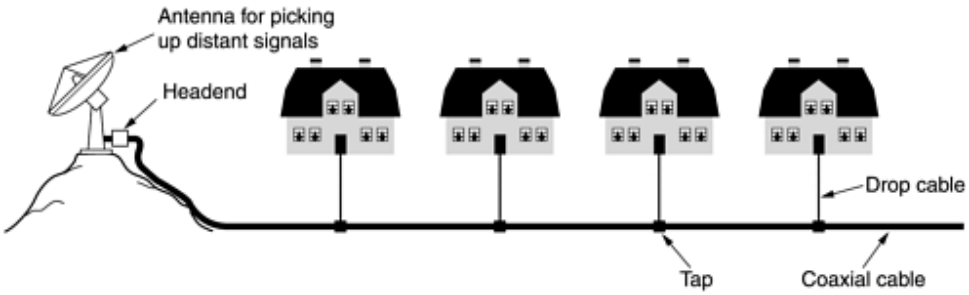
We have now studied both the fixed and wireless telephone systems in a fair amount of detail. Both will

clearly play a major role in future networks. However, an alternative available for fixed networking is now becoming a major player: cable television networks. Many people already get their telephone and Internet service over the cable, and the cable operators are actively working to increase their market share. In the following sections we will look at cable television as a networking system in more detail and contrast it with the telephone systems we have just studied. For more information about cable, see (Laubach et al., 2001; Louis, 2002; Ovadia, 2001; and Smith, 2002).

13.9.1 community antenna television

Cable television was conceived in the late 1940s as a way to provide better reception to people living in rural or mountainous areas. The system initially consisted of a big antenna on top of a hill to pluck the television signal out of the air, an amplifier, called the **head end**, to strengthen it, and a coaxial cable to deliver it to people's houses, as illustrated below.

An early cable television system



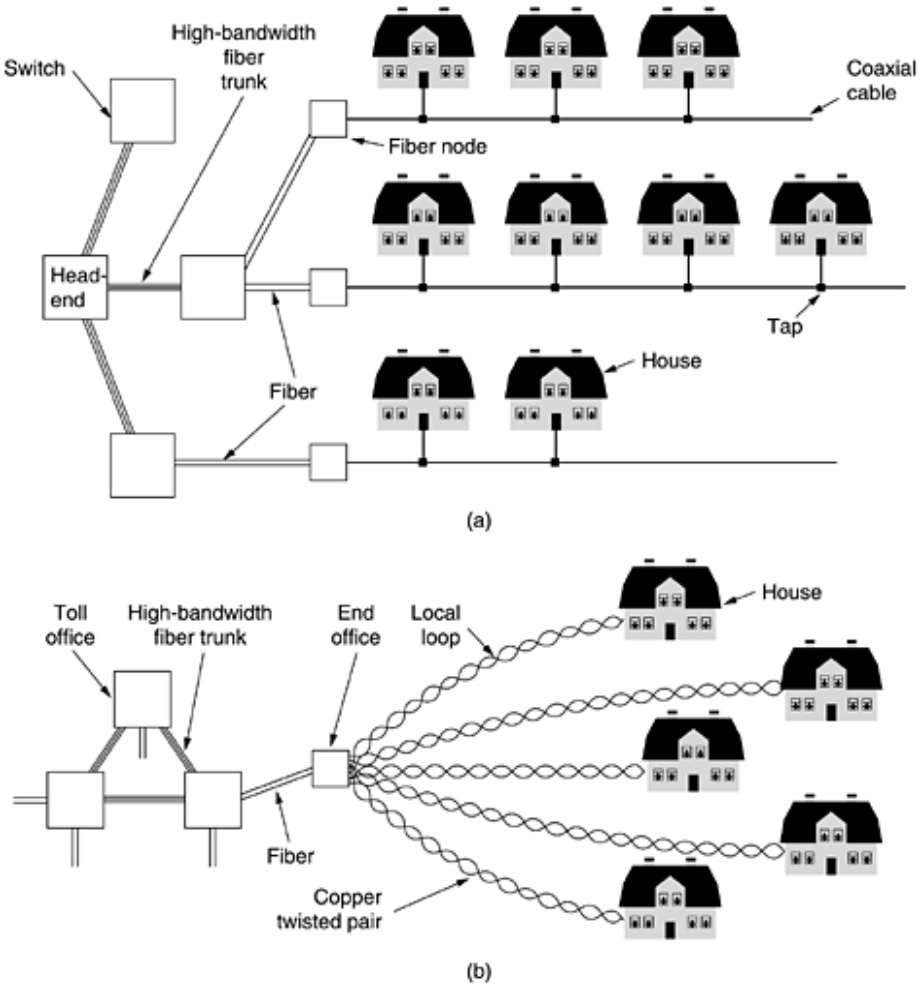
In the early years, cable television was called Community Antenna Television. It was very much a mom-and-pop operation; anyone handy with electronics could set up a service for his town, and the users would chip in to pay the costs. As the number of subscribers grew, additional cables were spliced onto the original cable and amplifiers were added as needed. Transmission was one way, from the headend to the users. By 1970, thousands of independent systems existed.

In 1974, Time, Inc., started a new channel, Home Box Office, with new content (movies) and distributed only on cable. Other cable-only channels followed with news, sports, cooking, and many other topics. This development gave rise to two changes in the industry. First, large corporations began buying up existing cable systems and laying new cable to acquire new subscribers. Second, there was now a need to connect multiple systems, often in distant cities, in order to distribute the new cable channels. The cable companies began to lay cable between their cities to connect them all into a single system. This pattern was analogous to what happened in the telephone industry 80 years earlier with the connection of previously isolated end offices to make long distance calling possible.

13.9.2 internet over cable

Over the course of the years the cable system grew and the cables between the various cities were replaced by high-bandwidth fiber, similar to what was happening in the telephone system. A system with fiber for the long-haul runs and coaxial cable to the houses is called an HFC (Hybrid Fiber Coax) system. The electro-optical converters that interface between the optical and electrical parts of the system are called fiber nodes. Because the bandwidth of fiber is so much more than that of coax, a fiber node can feed multiple coaxial cables. Part of a modern HFC system is shown below.

(a) Cable television. (b) The fixed telephone system



In recent years, many cable operators have decided to get into the Internet access business, and often the telephony business as well. However, technical differences between the cable plant and telephone plant have an effect on what has to be done to achieve these goals. For one thing, all the one-way amplifiers in the system have to be replaced by two-way amplifiers.

However, there is another difference between the HFC system and the telephone system in the picture above that is much harder to remove. Down in the neighborhoods, a single cable is shared by many houses, whereas in the telephone system, every house has its own private local loop. When used for television broadcasting, this sharing does not play a role. All the programs are broadcast on the cable and it does not matter whether there are 10 viewers or 10,000 viewers. When the same cable is used for Internet access, it matters a lot if there are 10 users or 10,000. If one user decides to download a very large file, that bandwidth is potentially being taken away from other users. The more users, the more competition for bandwidth. The telephone system does not have this particular property: downloading a large file over an ADSL line does not reduce your neighbor's bandwidth. On the other hand, the bandwidth of coax is much higher than that of twisted pairs.

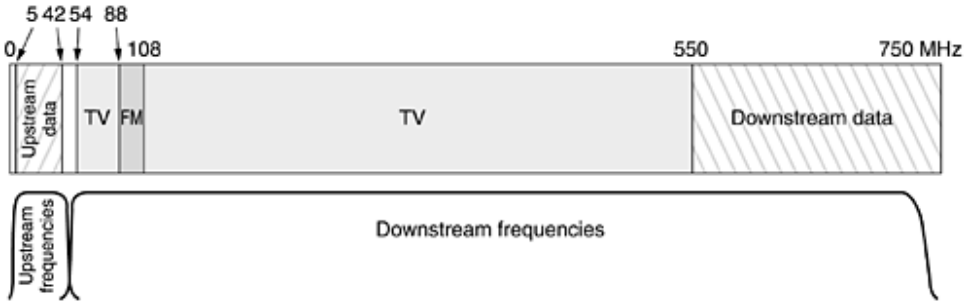
The way the cable industry has tackled this problem is to split up long cables and connect each one directly to a fiber node. The bandwidth from the headend to each fiber node is effectively infinite, so as long as there are not too many subscribers on each cable segment, the amount of traffic is manageable. Typical cables nowadays have 500–2000 houses, but as more and more people subscribe to Internet over cable, the load may become too much, requiring more splitting and more fiber nodes.

13.9.3 spectrum allocation

Throwing off all the TV channels and using the cable infrastructure strictly for Internet access would probably generate a fair number of irate customers, so cable companies are hesitant to do this. Furthermore, most cities heavily regulate what is on the cable, so the cable operators would not be allowed to do this even if they really wanted to. As a consequence, they needed to find a way to have television and Internet coexist on the same cable.

Cable television channels in North America normally occupy the 54–550 MHz region (except for FM radio from 88 to 108 MHz). These channels are 6 MHz wide, including guard bands. In Europe the low end is usually 65 MHz and the channels are 6–8 MHz wide for the higher resolution required by PAL and SECAM but otherwise the allocation scheme is similar. The low part of the band is not used. Modern cables can also operate well above 550 MHz, often to 750 MHz or more. The solution chosen was to introduce upstream channels in the 5–42 MHz band (slightly higher in Europe) and use the frequencies at the high end for the downstream. The cable spectrum is illustrated in Fig. 2-48.

Frequency allocation in a typical cable TV system used for Internet access



Note that since the television signals are all downstream, it is possible to use upstream amplifiers that work only in the 5–42 MHz region and downstream amplifiers that work only at 54 MHz and up, as shown in the figure. Thus, we get an asymmetry in the upstream and downstream bandwidths because more spectrum is available above television than below it. On the other hand, most of the traffic is likely to be downstream, so cable operators are not unhappy with this fact of life. As we saw earlier, telephone companies usually offer an asymmetric DSL service, even though they have no technical reason for doing so.

Long coaxial cables are not any better for transmitting digital signals than are long local loops, so analog modulation is needed here, too. The usual scheme is to take each 6 MHz or 8 MHz downstream channel and modulate it with QAM-64 or, if the cable quality is exceptionally good, QAM-256. With a 6 MHz channel and QAM-64, we get about 36 Mbps. When the overhead is subtracted, the net payload is about 27 Mbps. With QAM-256, the net payload is about 39 Mbps. The European values are 1/3 larger.

For upstream, even QAM-64 does not work well. There is too much noise from terrestrial microwaves, CB radios, and other sources, so a more conservative scheme—QPSK—is used. This method (shown in Fig. 2-25) yields 2 bits per baud instead of the 6 or 8 bits QAM provides on the downstream channels. Consequently, the asymmetry between upstream bandwidth and downstream bandwidth is much more than suggested by Fig. 2-48.

In addition to upgrading the amplifiers, the operator has to upgrade the headend, too, from a dumb amplifier to an intelligent digital computer system with a high-bandwidth fiber interface to an ISP. Often the name gets upgraded as well, from "headend" to CMTS (Cable Modem Termination System). In the following text, we will refrain from doing a name upgrade and stick with the traditional "headend."

13.10 ADSL versus cable

Which is better, ADSL or cable? That is like asking which operating system is better. Or which language is better. Or which religion. Which answer you get depends on whom you ask. Let us compare ADSL and cable on a few points. Both use fiber in the backbone, but they differ on the edge. Cable uses coax; ADSL uses twisted pair. The theoretical carrying capacity of coax is hundreds of times more than twisted pair. However, the full capacity of the cable is not available for data users because much of the cable's bandwidth is wasted on useless stuff such as television programs.

In practice, it is hard to generalize about effective capacity. ADSL providers give specific statements about the bandwidth (e.g., 1 Mbps downstream, 256 kbps upstream) and generally achieve about 80% of it consistently. Cable providers do not make any claims because the effective capacity depends on how many people are currently active on the user's cable segment. Sometimes it may be better than ADSL and sometimes it may be worse. What can be annoying, though, is the unpredictability. Having great service one minute does not guarantee great service the next minute since the biggest bandwidth hog in town may have just turned on his computer.

As an ADSL system acquires more users, their increasing numbers have little effect on existing users, since each user has a dedicated connection. With cable, as more subscribers sign up for Internet service, performance for existing users will drop. The only cure is for the cable operator to split busy cables and connect each one to a fiber node directly. Doing so costs time and money, so their are business pressures to avoid it.

As an aside, we have already studied another system with a shared channel like cable: the mobile telephone system. Here, too, a group of users, we could call them cellmates, share a fixed amount of bandwidth. Normally, it is rigidly divided in fixed chunks among the active users by FDM and TDM because voice traffic is fairly smooth. But for data traffic, this rigid division is very inefficient because data users are frequently idle, in which case their reserved bandwidth is wasted. Nevertheless, in this respect, cable access is more like the mobile phone system than it is like the fixed system.

Availability is an issue on which ADSL and cable differ. Everyone has a telephone, but not all users are close enough to their end office to get ADSL. On the other hand, not everyone has cable, but if you do have cable and the company provides Internet access, you can get it. Distance to the fiber node or headend is not an issue. It is also worth noting that since cable started out as a television distribution medium, few businesses have it.

Being a point-to-point medium, ADSL is inherently more secure than cable. Any cable user can easily read all the packets going down the cable. For this reason, any decent cable provider will encrypt all traffic in both directions. Nevertheless, having your neighbor get your encrypted messages is still less secure than having him not get anything at all.

The telephone system is generally more reliable than cable. For example, it has backup power and continues to work normally even during a power outage. With cable, if the power to any amplifier along the chain fails, all downstream users are cut off instantly.

Finally, most ADSL providers offer a choice of ISPs. Sometimes they are even required to do so by law. This is not always the case with cable operators.

The conclusion is that ADSL and cable are much more alike than they are different. They offer comparable service and, as competition between them heats up, probably comparable prices.

13.11 summary

The physical layer is the basis of all networks. Nature imposes two fundamental limits on all channels, and these determine their bandwidth. These limits are the Nyquist limit, which deals with noiseless channels, and the Shannon limit, which deals with noisy channels.

Transmission media can be **guided** or **unguided**. The principal guided media are **twisted pair**, **coaxial cable**, and **fiber optics**. Unguided media include **radio**, **microwaves**, **infrared**, and l**asers** through the air. An up-and-coming transmission system is **satellite communication**, especially LEO systems.

A key element in most wide area networks is the **telephone system**. Its main components are the **local loops**, **trunks**, and **switches**. Local loops are analog, twisted pair circuits, which require modems for transmitting digital data. ADSL offers speeds up to 50 Mbps by dividing the local loop into many virtual channels and modulating each one separately. Wireless local loops are another new development to watch, especially LMDS.

Trunks are digital, and can be multiplexed in several ways, including FDM, TDM, and WDM. Both circuit switching and packet switching are important.

For mobile applications, the fixed telephone system is not suitable. **Mobile phones** are currently in widespread use for voice and will soon be in widespread use for data. The first generation was analog, dominated by AMPS. The second generation was digital, with D-AMPS, GSM, and CDMA the major options. The third generation will be digital and based on broadband CDMA.

An alternative system for network access is the cable television system, which has gradually evolved from a community antenna to hybrid fiber coax. Potentially, it offers very high bandwidth, but the actual bandwidth available in practice depends heavily on the number of other users currently active and what they are doing.

chapter 14 wide area networks

14.1 introduction to Wide Area Networks

So what is it that makes something a wide area network (WAN) instead of a local area network (LAN)? Distance is the first thing that comes to mind, but these days, wireless LANs can cover some serious turf! So is it bandwidth? Here again, in many places really big pipes can be had for a price, so that's not it either. Well, what then? Perhaps one of the best ways to tell a WAN from a LAN is that you generally own a LAN infrastructure, but you generally lease WAN infrastructure from a service provider. We've already talked about a data link that you usually own (Ethernet), but now we're going to take a look at the data links you most often don't own, but instead lease from a service provider. The key to understanding WAN technologies is to be familiar with the different WAN terms and connection types often used by service providers to join your networks together.

14.1.1 defining WAN terms

Before ordering a WAN service type, it would be a good idea to understand the following terms, commonly used by service providers:

Customer premises equipment – equipment that's owned by the subscriber and located on the subscriber's premises.

Demarcation point - the spot where the service provider's responsibility ends and the CPE begins. It's generally a device in a telecommunications closet owned and installed by the telecommunications company (telco). The customer is responsible to cable (extended demarc) from this box to the CPE, which is usually a connection to a CSU/DSU or ISDN interface. (Channel/Data service Unit).

Local loop - connects the demarc to the closest switching office, called a central office.

Central office (CO) - This point connects the customers to the provider's switching network. A central office (CO) is sometimes referred to as a **point of presence** (POP).

Toll network - a trunk line inside a WAN provider's network. This network is a collection of switches and facilities owned by the ISP.

14.1.2 WAN connection types

A WAN can use a number of different connection types and this section will provide you with an introduction to the various types of WAN connections you'll find on the market today.

The following list explains the WAN connection types:

Leased lines - typically, these are referred to as a point-to-point connection or dedicated connection. A leased line is a pre-established WAN communications path from the CPE, through the DCE (Data Communication Equipment) switch, to the CPE of the remote site, allowing DTE (Data Terminal Equipment) networks to communicate at any time with no setup procedures before transmitting data. When cost is no object, it's really the best choice. It uses synchronous serial lines up to 45Mbps. HDLC and PPP encapsulations are frequently used on leased lines, and I'll go over them with you in detail in a bit.

Circuit switching - when you hear the term circuit switching, think phone call. The big advantage is cost—you only pay for the time you actually use. No data can transfer before an end-toend connection is established. Circuit switching uses dial-up modems or ISDN, and is used for low-bandwidth data transfers.

Packet switching - this is a WAN switching method that allows you to share bandwidth with other companies to save money. Packet switching can be thought of as a network that's designed to look like a leased line, yet charges you (and costs) more like circuit switching. There is a downside: If you need to transfer data constantly, forget about this option. Just get yourself a leased line. Packet switching will only work well if your data transfers are bursty in nature. Frame Relay and X.25 are packet-switching technologies. Speeds can range from 56Kbps to T3 (45Mbps).

14.2 main WAN protocols

In this section, we will define the most prominent WAN protocols used today - frame relay, ISDN, LAPB, LAPD, HDLC, PPP, and ATM. Usually, though, the only WAN protocols configured on a serial interface these days are HDLC, PPP, and Frame Relay.

14.2.1 frame relay

A packet-switched technology that emerged in the early 1990s, **Frame Relay** is a Data Link and Physical layer specification that provides high performance. Frame Relay is a successor to X.25, except that much of the technology in X.25 used to compensate for physical errors (noisy lines) has been eliminated.

Frame Relay can be more cost-effective than point-to-point links, and can typically run at speeds of 64Kbps up to 45Mbps (T3). Frame Relay provides features for dynamic bandwidth allocation and congestion control.

14.2.2 ISDN

Integrated Services Digital Network (ISDN) is a set of digital services that transmit voice and data over existing phone lines. ISDN can offer a cost-effective solution for remote users who need a higher-speed connection than analog dial-up links offer. ISDN is also a good choice as a backup link for other types of links such as Frame Relay or a T1 connection.

14.2.3 LAPB

Link Access Procedure, Balanced (LAPB) was created to be a connection-oriented protocol at the Data Link layer for use with X.25. It can also be used as a simple data link transport. LAPB causes a tremendous amount of overhead because of its strict timeout and windowing techniques.

14.2.4 LAPD

Link Access Procedure, D-Channel (LAPD) - is used with ISDN at the Data Link layer (layer 2) as a protocol for the D (signaling) channel. LAPD was derived from the Link Access Procedure, Balanced (LAPB) Protocol and is designed primarily to satisfy the signaling requirements of ISDN basic access.

14.2.5 HDLC

High-Level Data-Link Control (HDLC) - was derived from Synchronous Data Link Control (SDLC), which was created by IBM as a Data Link connection protocol. HDLC is a protocol at the Data Link layer, and it has very little overhead compared to LAPB. HDLC wasn't intended to encapsulate multiple Network layer protocols across the same link.

The HDLC header carries no identification of the type of protocol being carried inside the HDLC encapsulation. Because of this, each vendor that uses HDLC has their own way of identifying the Network layer protocol, which means that each vendor's HDLC is proprietary for their equipment.

14.2.6 PPP

Point-to-Point Protocol (PPP) - is an industry-standard protocol. Because all multiprotocol versions of HDLC are proprietary, PPP can be used to create point-to-point links between different vendors' equipment. It uses a Network Control Protocol field in the Data Link header to identify the Network layer protocol. It allows authentication and multilink connections and can be run over asynchronous and synchronous links.

14.2.7 ATM

Asynchronous Transfer Mode (ATM) was created for time-sensitive traffic, providing simultaneous transmission of voice, video, and data. ATM uses cells instead of packets that are a fixed 53 bytes long. It also can use isochronous clocking (external clocking) to help the data move faster.

14.3 cabling the Wide Area Network

There are a couple of things that you need to know in order to connect your WAN. For starters, you've got to understand the WAN Physical layer implementation provided by Cisco, and you must be familiar with the various types of WAN serial connectors.

Cisco serial connections support almost any type of WAN service. The typical WAN connections are dedicated leased lines using HDLC, PPP, Integrated Services Digital Network (ISDN), and Frame Relay. Typical speeds run at anywhere from 2400bps to 45Mbps (T3).

HDLC, PPP, and Frame Relay can use the same Physical layer specifications, but ISDN has different pinouts and specifications at the Physical layer.

14.3.1 serial transmission

WAN serial connectors use **serial transmission**, which takes place one bit at a time over a single channel. Cisco routers use a proprietary 60-pin serial connector that you must get from Cisco or a provider of Cisco equipment. Cisco also has a new, smaller proprietary serial connection that is about one-tenth the size of the 60-pin basic serial cable. This is called the "smart-serial," for some reason, and you have to make sure you have the right type of interface in your router before using this cable connector. The type of connector you have on the other end of the cable depends on your service provider or end-device requirements. The different ends available are:

- EIA/TIA-232

- EIA/TIA-449
- V.35 (used to connect to a CSU/DSU)
- X.21 (used in X.25)
- EIA-530

Serial links are described in frequency or cycles-per-second (hertz). The amount of data that can be carried within these frequencies is called bandwidth. Bandwidth is the amount of data in bits-per-second that the serial channel can carry.

14.3.2 data terminal equipment (DTE) and data communication equipment (DCE)

Router interfaces are, by default, data terminal equipment (DTE), and they connect into data communication equipment (DCE) - for example, a channel service unit/data service unit (CSU/DSU).

The CSU/DSU then plugs into a demarcation location (demarc) and is the service provider's last responsibility. Most of the time, the demarc is a jack that has an RJ-45 (8-pin modular) female connector located in a telecommunications closet.

You may have heard of demarcs if you've ever had the glorious experience of reporting a problem to your service provider—they'll always tell you that it tests fine up to the demarc, and that the problem must be the CPE, or customer premises equipment. In other words, it's your problem, not theirs.

The idea behind a WAN is to be able to connect two DTE networks together through a DCE network. The DCE network includes the CSU/DSU, through the provider's wiring and switches, all the way to the CSU/DSU at the other end. The network's DCE device (CSU/DSU) provides clocking to the DTE-connected interface (the router's serial interface).

As mentioned, the DCE network provides clocking to the router; this is the CSU/DSU. If you have a non-production network and are using a WAN crossover type of cable and do not have a CSU/DSU, then you need to provide clocking on the DCE end of the cable by using the clock rate command.

14.3.3 fixed and modular interfaces

Some of the routers Cisco sells have fixed interfaces, while others are modular. The fixed routers, such as the 2500 series, have set interfaces that can't be changed. The 2501 router has two serial connections and one 10BaseT AUI interface. If you need to add a third serial interface, you need to buy a new router—ouch! However, the 1600, 1700, 2600, 3600, and higher routers have modular interfaces that allow you to buy what you need now and add almost any type of interface you may need later. The 1600 and 1700 are limited and have both fixed and modular ports, but the 2600 and up provide many serials, Fast Ethernet, and even voice-module availability.

Let's move on and start talking about the various WAN protocols, so we can get on with configuring Cisco routers.

14.4 High-Level Data-Link Control (HDLC) protocol

The High-Level Data-Link Control (HDLC) protocol is a popular ISO-standard, bit-oriented Data Link layer protocol. It specifies an encapsulation method for data on synchronous serial data links using frame characters and checksums. HDLC is a point-to-point protocol used on leased lines. No authentication can be used with HDLC.

In byte-oriented protocols, control information is encoded using entire bytes. On the other hand, bit-oriented protocols may use single bits to represent control information. Bit-oriented protocols include SDLC, LLC, HDLC, TCP, IP, and others.

HDLC is the default encapsulation used by Cisco routers over synchronous serial links. Cisco's HDLC is proprietary—it won't communicate with any other vendor's HDLC implementation.

But don't give Cisco grief for it - *everyone's* HDLC implementation is proprietary.

The reason that every vendor has a proprietary HDLC encapsulation method is that each vendor has a different way for the HDLC protocol to encapsulate multiple Network layer protocols. If the vendors didn't have a way for HDLC to communicate the different layer 3 protocols, then HDLC would only be able to carry one protocol. This proprietary header is placed in the data field of the HDLC encapsulation.

Let's say you only have one Cisco router, and you need to connect to a Bay router because your other Cisco router is on order. What would you do? You couldn't use the default HDLC serial encapsulation because it wouldn't work. Instead, you would use something like PPP, an ISO-standard way of identifying the upper-layer protocols. In addition, you can check RFC 1661 for more information on the origins and standards of PPP.

14.5 Point-to-Point Protocol (PPP)

Point-to-Point Protocol (PPP) is a Data Link layer protocol that can be used over either asynchronous serial (dial-up) or synchronous serial (ISDN) media. It uses the LCP (Link Control Protocol) to build and maintain data-link connections. **Network Control Protocol (NCP)** is used to allow multiple Network layer protocols (routed protocols) to be used on a point-to-point connection.

Since HDLC is the default serial encapsulation on Cisco serial links and it works great, when would you choose to use PPP? The basic purpose of PPP is to transport layer 3 packets across a Data Link layer point-to-point link. It is non-proprietary, which means that if you don't have all Cisco routers, PPP would be needed on your serial interfaces—the HDLC encapsulation would not work because it is Cisco proprietary. In addition, since PPP can encapsulate several layer 3 routed protocols and provide authentication, dynamic addressing, and callback, this may be the encapsulation solution of choice for you over HDLC.

PPP contains four main components:

EIA/TIA-232-C, V.24, V.35, and ISDN - a physical layer international standard for serial communication.

HDLC - a method for encapsulating datagrams over serial links.

LCP - a method of establishing, configuring, maintaining, and terminating the point-to-point connection.

NCP - a method of establishing and configuring different Network layer protocols. NCP is designed to allow the simultaneous use of multiple Network layer protocols. Some examples of protocols here are **IPCP (Internet Protocol Control Protocol)** and I**PXCP (Internetwork Packet Exchange Control Protocol)**.

It is important to understand that the PPP protocol stack is specified at the Physical and Data Link layers only. NCP is used to allow communication of multiple Network layer protocols by encapsulating the protocols across a PPP data link.

14.5.1 Link Control Protocol (LCP) configuration options

Link Control Protocol (LCP) offers different PPP encapsulation options, including the following:

Authentication - this option tells the calling side of the link to send information that can identifythe user. The two methods are PAP and CHAP.

Compression - used to increase the throughput of PPP connections by compressing the data or

payload prior to transmission. PPP decompresses the data frame on the receiving end.

Error detection - PPP uses Quality and Magic Number options to ensure a reliable, loop-free data link.

Multilink - in IOS version 11.1, multilink is supported on PPP links with Cisco routers. This option allows several separate physical paths to appear to be one logical path at layer 3. For example, two T1s running multilink PPP would appear as a single 3Mbps path to a layer 3 routing protocol.

PPP callback - PPP can be configured to call back after successful authentication. PPP callback can be a good thing for you because you can keep track of usage based upon access charges, for accounting records, or a variety of other reasons. With callback enabled, a calling router (client) will contact a remote router (server) and authenticate as described in the previous section. Both routers must be configured for the callback feature. Once authentication is completed, the remote router will terminate the connection and then re-initiate a connection to the calling router from the remote router.

14.5.2 PPP session establishment

When PPP connections are started, the links go through three phases of session establishment, as shown below:

Link-establishment phase - LCP packets are sent by each PPP device to configure and test the link. These packets contain a field called the Configuration Option that allows each device to see the size of the data, compression, and authentication. If no Configuration Option field is present, then the default configurations are used.

Authentication phase - if required, either CHAP or PAP can be used to authenticate a link. Authentication takes place before Network layer protocol information is read. It is possible that link-quality determination may occur at this same time.

Network layer protocol phase - PPP uses the *Network Control Protocol (NCP)* to allow multiple Network layer protocols to be encapsulated and sent over a PPP data link. Each Network layer protocol (e.g., IP, IPX, AppleTalk, which are routed protocols) establishes a service with NCP.

14.5.3 PPP authentication methods

There are two methods of authentication that can be used with PPP links:

Password Authentication Protocol (PAP) - the less secure of the two methods. Passwords are sent in clear text, and PAP is only performed upon the initial link establishment. When the PPP link is first established, the remote node sends back to the originating router the username and password until authentication is acknowledged. That's it.

Challenge Handshake Authentication Protocol (CHAP) - used at the initial startup of a link and at periodic checkups on the link to make sure the router is still communicating with the same host. After PPP finishes its initial link-establishment phase, the local router sends a challenge request to the remote device. The remote device sends a value calculated using a one-way hash function called MD5. The local router checks this hash value to make sure it matches. If the values don't match, the link is immediately terminated.

14.6 Frame Relay

Frame Relay has become one of the most popular WAN services deployed over the past decade. There are good reasons for this, but primarily it has to do with cost. Frame Relay technology frequently saves money over alternatives, and very few network designs ignore the cost factor.

Frame Relay, by default, is classified as a non-broadcast multi-access (NBMA) network, which means that it does not send any broadcasts, such as RIP updates, across the network by default. We'll discuss this further later in this section.

Frame Relay has at its roots a technology called X.25. Frame Relay essentially incorporates the components of X.25 relevant to today's reliable and relatively "clean" telecommunications networks and leaves out the error-correction components that aren't needed anymore.

It's substantially more complex than the simple leased-line networks you learned about in our discussion of the HDLC and PPP protocols. These leased-line networks are easy to conceptualize. Not so with Frame Relay. It can be significantly more complex and versatile, which is why it's often represented as a "cloud" in networking graphics. I'll get to that in a minute - for right now, I'm going to introduce Frame Relay in concept and show you how it differs from simpler leased-line technologies.

14.6.1 Introduction to Frame Relay Technology

As a CCNA, you need to understand the basics of the Frame Relay technology and be able to configure it in simple scenarios. You need to realize that I'm merely introducing Frame Relay here in this chapter—this technology gets much deeper than what we'll be dealing with here. However, I will cover everything you need to know when studying the CCNA objectives.

Frame Relay is a packet-switched technology. From everything you've learned so far, just telling you this should make you immediately realize several things about it:

You won't be using the encapsulation hdlc or encapsulation ppp commands to configure it.

Frame Relay doesn't work like a point-to-point leased line (although it can be made to look like one).

Frame Relay will in many cases be less expensive than a leased line, but there are some sacrifices made in order to gain that savings.

To help you understand, here's an example of how packet-switching versus leased-line networks work:

Let's say you have a router in Miami and a router in Denver that you want to connect. With a leased line, you pay a telecommunications company to provide a T1 line between them for you. Basically, this means that they'll provide the T1 and install a piece of equipment in each of your facilities that represents the demarcation point. You would plug this into your CSU/DSU and into your routers, select HDLC or PPP encapsulation, and proceed to configure and troubleshoot the connection.

When buying a technology like this, you can expect the telco to provide you with a full T1between your two sites. You can transmit at 1.544Mbps (full capacity) continuously if you want, and the telco has to deliver the packets. This is kind of like buying every seat to every showing of your favorite movie - you can see it whenever you want, as many times as you want - it's always available to you because, well, you paid for it.

So what happens if you skip a few showings at the theater? Do you get a refund just because you failed to show up? Nope. Likewise, if you choose to transmit at less than 1.544Mbps continuously, you don't get to enjoy any cost savings on a leased line. You're paying for the full T1whether you use it or not. Yes, the infrastructure is always there for you, but you're going to pay for that availability whether you use it or not!

Let's go back to that connection between Miami and Denver. Suppose you had a way to get a connection that looked like a T1 and acted like a T1, but allowed you to pay for whatever portion of that T1 you actually used, without charging you for data you could've sent but didn't.

Sound like a deal? That's essentially what packet-switched networks do. However, if you don't use your

bandwidth, don't think that you'll be getting a refund check! You are contractually guaranteed to able to use your agreed-upon bandwidth (called a Committed Information Rate or CIR)—you may possibly even burst to more bandwidth if someone isn't using their bandwidth, but that is not guaranteed.

This works because it's quite possible that your telecommunications company has hundreds, even thousands, of customers who have locations in both Miami and Denver. And they've installed a significant amount of infrastructure at both cities, as well as at many sites in between. So if you buy a T1 from Miami to Denver, they have to carve you out a T1 from Miami to Denver, and reserve it for you all the time, just as in our movie theater analogy. But if you're willing to share their infrastructure with some of their other customers, you get to save some serious cash.

Sometimes you'll find that you're transmitting at full T1 speeds, but other times you'll be transmitting hardly any bits at all. You may average only 25% of a full T1 over time. If the telco has 1000 customers in Miami, and each customer only averages 25% of a full T1, and these customers agree to share the telco's infrastructure (Frame Relay), the telco does not need 1000 T1s in Miami. Statistically, they can get away with a lot fewer than that because they only need enough to meet the peak demand of all their customers combined. And since probability says that all their customers will never need full capacity simultaneously, the telco needs fewer T1 lines than if each customer had their very own leased line. Fewer T1 lines = less infrastructure = big money saved.

This concept is called oversubscription. The telco sells the same infrastructure to a whole bunch of customers at a discount, knowing that it's highly unlikely that there will come a time when all their customers need simultaneous access. Telcos have been doing this for years with voice networks—ever get an All Circuits Busy message on Mother's Day? They're pretty good at installing just enough but not too much. With Frame Relay, many customers share the telco's backbone frame network, and since the customers agree to share this infrastructure (oversubscribe), they get a better price than if they each opted for dedicated leased lines. It's this win-win concept that amounts to huge savings for both customer and provider, and that makes Frame Relay so popular.

OK, so before we move on, why would you even consider using Frame Relay? Take a look at Figure 11.10 to get an idea of what a network would look like before Frame Relay.

Now, take a look below and you can see that there is now only one connection between the Corporate router and the Frame Relay switch. A total money saver!

If, for example, you had to add seven remote sites to the corporate office and had only one free serial port on your router—Frame Relay to the rescue! Of course, I guess I should mention that you now have one single point of failure, but Frame Relay is used to save money, not make a network more resilient.

In the following sections I am going to cover the Frame Relay technology information you need to know when studying the CCNA objectives.

14.6.2 frame relay technology

To better introduce you to the Frame Relay terminology, let's take a preliminary look at how the technology works. Figure 11.12 is labeled with the various terms used to describe different parts of a Frame Relay network.

The basic idea behind Frame Relay networks is to allow users to communicate between two DTE devices (in this case, routers) through DCE devices. The users shouldn't see a difference between connecting to and gathering resources from a local server and a server at a remote site connected with Frame Relay other than a potential change in speed. Figure 11.12 illustrates everything that must happen in order for two DTE devices to communicate. Here is how the process works:

1. The user's network host sends a frame out on the local area network. The hardware address of the router (default gateway) will be in the header of the frame.
2. The router picks up the frame, extracts the packet, and discards what is left of the frame. It then looks at the destination IP address within the packet and checks to see whether it knows how to get to the destination network by looking into the routing table.

3. The router next forwards the data out the interface that it thinks can find the remote network. (If it can't find the network in its routing table, it will discard the packet.) Because this will be a serial interface encapsulated with Frame Relay, the router puts the packet onto the Frame Relay network encapsulated within a Frame Relay frame.
4. The channel service unit/data service unit (CSU/DSU) receives the digital signal and encodes it into the type of digital signaling that the switch at the packet switching exchange (PSE) can understand. For example, it may alter it from the encoding used in V.35 to the encoding of the access line, which might be B8ZS over a T1. The PSE receives the digital signal and extracts the ones and zeros from the line.
5. The CSU/DSU is connected to a demarc installed by the service provider, and its location is the service provider's first point of responsibility (last point on the receiving end). The demarc is typically just an RJ-45 (8-pin modular) jack installed close to the router and CSU/DSU (sometimes called a Smart Jack).
6. The demarc is typically a twisted-pair cable that connects to the local loop. The local loop connects to the closest central office (CO), sometimes called a point of presence (POP). The local loop can connect using various physical mediums; twisted-pair or fiber is common.
7. The CO receives the frame and sends it through the Frame Relay "cloud" to its destination. This cloud can be dozens of switching offices- or more!
8. Once the frame reaches the switching office closest to the destination office, it's sent through the local loop. The frame is received at the demarc, and is then sent to the CSU/DSU. Finally, the router extracts the packet, or datagram, from the frame and puts the packet in a new LAN frame to be delivered to the destination host. The frame on the LAN will have the final destination hardware address in the header. This was found in the router's ARP cache, or an ARP broadcast was performed.

The user and server do not need to know, nor should they know, everything that happens as the frame makes its way across the Frame Relay network. The remote server should be as easy to use as a locally connected resource.

There are several things that make the Frame Relay circuit different than a leased line. With a leased line, you typically specify the bandwidth you desire (T1, fractional T1, DS3, etc.). But with Frame Relay, you specify both an access rate (port speed) and a CIR. I'll talk about this next.

14.6.3 Committed Information Rate (CIR)

Frame Relay provides a packet-switched network to many different customers at the same time.

This is a great idea because it spreads the cost of the switches, etc., among many customers. But remember, Frame Relay is based on the assumption that all customers will never need to transmit constant data all at the same time.

Frame Relay works by providing a portion of dedicated bandwidth to each user, and also allowing the user to exceed their guaranteed bandwidth if resources on the telco network are available. So basically, Frame Relay providers allow customers to buy a lower amount of bandwidth than what they really use. There are two separate bandwidth specifications with Frame Relay:

Access rate - the maximum speed at which the Frame Relay interface can transmit.

CIR - the maximum bandwidth of data guaranteed to be delivered. However, in reality, this is the average amount that the service provider will allow you to transmit. If these two values are the same, the Frame Relay connection is pretty much just like a leased line. However, they can also be set to different values. Here's an example: Let's say that you buy an access rate of T1 (1.544Mbps) and a CIR of 256Kbps. By doing this, the first 256Kbps of traffic you send is guaranteed to be delivered. Anything beyond that is called a "burst," which is a transmission that exceeds your guaranteed 256Kbps, and can be any amount up to the T1 access rate (if that amount is in your contract). If your combined committed burst (the basis for your CIR) and excess burst sizes (which when combined are known as the MBR or maximum burst rate) exceed the access rate, you can

pretty much be guaranteed that your additional traffic will be dropped, although it depends on the subscription level of the particular service provider.

In a perfect world, this always works beautifully - but remember that little word *guarantee*?

As in guaranteed rate - of 256Kbps, to be exact? This means that any burst of data you send that exceeds your guaranteed 256Kbps rate will be delivered on something called a “best effort” basis of delivery. Or maybe not - if your telco’s equipment doesn’t have the capacity to deliver at the time you transmitted, then your frames will be discarded and the DTE will be notified.

Timing is everything -you can scream data out at six times your guaranteed rate of 256Kbps (T1) *only if* your telco has the capacity available on their equipment at that moment! Remember that “oversubscription” we talked about? Well, here it is in action!

Therefore, you should choose a CIR based on realistic, anticipated traffic rates. Some Frame Relay providers allow you to purchase a CIR of zero. You can use a zero CIR to save money if retransmitting packets is acceptable to you. Doing that is a real gamble, though, because it means that every packet you send is eligible to be discarded in the provider’s network!

14.6.4 virtual circuits

Frame Relay operates using **virtual circuits**, as opposed to real circuits that leased lines use. These virtual circuits are what link together the thousands of devices connected to the provider’s “cloud.” Referring back to the Miami and Denver example, you want these routers to connect to each other. That is, you want a circuit between them. Frame Relay provides a virtual circuit to be established between your two DTE devices, making them appear to be connected via a circuit when in reality they are dumping their frames into a large, shared infrastructure. You never see the complexity of what is happening inside the cloud because you have a virtual circuit.

There are two types of virtual circuits - permanent and switched. **Permanent Virtual Circuits** (PVCs) are by far the most common type in use today. What permanent means is that the telco creates the mappings inside their gear, and as long as you pay the bill, they will remain in place.

Switched Virtual Circuits (SVCs) are more like a phone call. The virtual circuit is established when data needs to be transmitted, then is taken down when data transfer is complete.

14.6.5 Data Link Connection Identifiers (DLCIs)

Frame Relay PVCs are identified to DTE end devices using **Data Link Connection Identifiers (DLCIs)**. A Frame Relay service provider typically assigns DLCI values, which are used on Frame Relay interfaces to distinguish between different virtual circuits. Because many virtual circuits can be terminated on one multipoint Frame Relay interface, many DLCIs are often affiliated with it.

This statement bears some explanation. Suppose you have a central HQ with three branch offices. If you were to connect each branch office to HQ using a T1, you would need three serial interfaces on your router at HQ, one for each T1. Simple, huh? Well, suppose you use Frame Relay PVCs. You could have a T1 at each branch connected to a service provider and only a *single* T1 at HQ. There would be three PVCs on the single T1 at HQ, one going to each branch.

Even though there’s only a single interface and a single CSU/DSU, the three PVCs function as three separate circuits. Remember what I said about saving money? How much for two additional T1 interfaces and a pair of CSU/DSUs? Answer: A lot of money! So, just go ahead and ask for a percentage of the savings in your bonus.

Before we go on, I want to define Inverse ARP (IARP) and discuss how it is used with DLCIs in a Frame Relay network. Yes, it is somewhat similar to ARP in the fact that it maps a DLCI to an IP address (kinda like ARP does with MAC addresses to IP addresses). IARP is not configurable, but it can be disabled. It runs on a

Frame Relay router and maps the DLCI (to an IP address) for Frame Relay so it knows how to get to the Frame Relay switch. You can see the IP-to-DLCI mappings with the show frame-relay map command. If you have a non-Cisco router in your network and it does not support IARP, then you have to statically provide IP to DLCI mappings with the frame-relay map command, which I'll demonstrate in a bit.

Let's talk about DLCIs for another minute. Yes, they're locally significant - global significance requires the buy-in of the entire network to use the LMI extensions that give us global significance.

Therefore, you're likely to see global DLCIs only in private networks.

However, the DLCI does not have to be globally significant for it to be functional in getting a frame across the network. Here's how it works. When RouterA wants to send a frame to RouterB, it looks up the IARP or manual mapping of the DLCI to the IP address it's trying to get to. Armed with the DLCI, it sends the frame out with the DLCI value it found in the DLCI field of the FR header. The provider's ingress switch gets this frame and does a lookup on the DLCI/physical-port combination it observes. Associated with that combination, it finds a new "locally significant" (between it and the next-hop switch) DLCI to use in the header, and in the same entry in its table, it finds an outgoing physical port. This happens all the way to RouterB. Therefore, you actually can say that the DLCI that RouterA knows identifies the entire virtual circuit to RouterB, even though every DLCI between every pair of devices could be completely different. The point is that RouterA is unaware of these differences. That's what makes the DLCI locally significant. Note, then, that DLCIs really are used by the telco to "find" the other end of your PVC.

To demonstrate how DLCIs are considered locally significant, take a look at Figure 11.13.

In Figure 11.13, DLCI 100 is considered locally significant to RouterA and identifies the circuit between RouterA and its ingress Frame Relay switch. DLCI 200 would identify the circuit between RouterB and its ingress Frame Relay switch.

DLCI numbers, used to identify a PVC, are typically assigned by the provider and start at 16.

14.6.6 Local Management Interface (LMI)

Local Management Interface (LMI) is a signaling standard used between your router and the first Frame Relay switch it's connected to. It allows for passing information about the operation and status of the virtual circuit between the provider's network and the DTE (your router). It communicates information about the following:

Keepalives - verify that data is flowing.

Multicasting - an optional extension of the LMI specification that allows, for example, the efficient distribution of routing information and ARP requests over a Frame Relay network. Multicasting uses the reserved DLCIs from 1019 through 1022.

Global addressing - provides global significance to DLCIs, allowing the Frame Relay cloud to work exactly like a LAN.

Status of virtual circuits - provides DLCI status. These status inquiries and status messages are used as keepalives when there is no regular LMI traffic to send. Remember, LMI is not communication between your routers - it's communication between your router and the nearest Frame Relay switch. So it's entirely possible that the router on one end of a PVC is actively receiving LMI, while the router on the other end of the PVC is not. (Of course, PVCs won't work with one end down. I just said this to illustrate the local nature of LMI communications.)

Routers receive LMI information from the service provider's Frame Relay switch on a frameencapsulated interface and update the virtual circuit status to one of three different states:

Active state - everything is up, and routers can exchange information.

Inactive state - the router's interface is up and working with a connection to the switching office, but the remote router is not working.

Deleted state - no LMI information is being received on the interface from the switch. It could be a mapping problem or a line failure.

14.6.7 frame relay congestion control

Remember back to our talk about CIR? From that, it should be obvious that the lower your CIR is set, the greater the risk that your data will become toast. This can be easily avoided if you have just one key piece of information—when to transmit or not to transmit that burst! So our questions are: Is there any way for us to find out when our telco's shared infrastructure is free and clear and when it's, well, crammed and clogged? And if there is, how do we go about it? In this next section, I'm going to talk about how the Frame Relay switch notifies the DTE of congestion problems and address those very important questions.

Here are the three congestion bits and their meanings:

Discard Eligibility (DE) - as you know, when you burst (transmit packets beyond the CIR of a PVC), any packets exceeding the CIR are eligible to be discarded if the provider's network is congested at the time. Because of this, the excessive bits are marked with a *Discard Eligibility (DE)* bit in the Frame Relay header. And if the provider's network is congested, the Frame Relay switch will discard the packets with the first DE bit set. So if your bandwidth is configured with a CIR of zero, the DE will always be on.

Forward Explicit Congestion Notification (FECN) - when the Frame Relay network recognizes congestion in the cloud, the switch will set the *Forward Explicit Congestion Notification (FECN)* bit to 1 in a Frame Relay packet header. This will indicate to the destination DTE that the path the frame just traversed is congested.

Backward Explicit Congestion Notification (BECN) - when the switch detects congestion in the Frame Relay network, it'll set the *Backward Explicit Congestion Notification (BECN)* bit in a Frame Relay frame that's destined for the source router. This notifies the router that congestion is being encountered ahead. Cisco routers do not necessarily take action on this congestion information unless you tell them to.

14.7 Integrated Services Digital Network (ISDN)

Integrated Services Digital Network (ISDN) is a digital service designed to run over existing telephone networks. ISDN can support both data and voice—a telecommuter's dream. But ISDN applications require bandwidth. Typical ISDN applications and implementations include highspeed image applications (such as Group IV facsimile), high-speed file transfer, videoconferencing, and multiple links into homes of telecommuters.

ISDN is actually a set of communication protocols proposed by telephone companies that allow them to carry a group of digital services that simultaneously convey data, text, voice, music, graphics, and video to end users, and it was designed to achieve this over the telephone systems already in place. ISDN is referenced by a suite of ITU-T standards that encompass the OSI model's Physical, Data Link, and Network layers.

PPP is typically used with ISDN to provide data transfer, link integrity, and authentication. So don't think of ISDN as a replacement for PPP, HDLC, or Frame Relay, because it's really an underlying infrastructure that any of these could use. And as I said, PPP is the most common encapsulation across ISDN connections.

These are the benefits of ISDN:

- It can carry voice, video, and data simultaneously.
- Call setup is faster than with an analog modem.
- Data rates are faster than on an analog modem connection.
- Full-time connectivity across the ISDN is spoofed by the Cisco IOS routers using dial-ondemand

(DDR) routing.

- Small office and home office sites can be economically supported with ISDN BRI services.
- ISDN can be used as a backup service for a leased-line connection between the remote and central offices.

14.7.1 ISDN connections

Integrated Services Digital Network (ISDN) *Basic Rate Interface (BRI)* is two B (Bearer) channels of 64k each, and one D (Data) channel of 16k for signaling.

ISDN BRI routers come with either a U interface or something known as an S/T interface.

The difference between the two is that the U interface is already a two-wire ISDN convention that can plug right into the ISDN local loop. Conversely, the S/T interface is a four-wire interface that basically needs an adapter—a network termination type 1 (NT1)—to convert from a four-wire to the two-wire ISDN specification.

The U interface has a built-in NT1 device. If your service provider uses an NT1 device, then you need to buy a router that has an S/T interface. Most Cisco router BRI interfaces are marked with a U or an S/T. When in doubt, ask Cisco or the salesperson you bought it from.

14.7.2 ISDN components

The components used with ISDN include functions and reference points. Figure 11.17 shows how the different types of terminal and reference points can be used in an ISDN network:

In North America, ISDN uses a two-wire connection, called a U reference point, into a home or office. The NT1 device is used to convert the typical four-wire connection to a two-wire connection that is used by ISDN phones and terminal adapters (TAs). Most routers can now be purchased with a built-in NT1 (U) interface.

14.7.3 ISDN terminals

Devices connecting to the ISDN network are known as terminal equipment (TE) and **network termination (NT)** equipment. There are two types of each:

TE1 A *terminal equipment type 1 (TE1)* device refers to those terminals that understand ISDN standards and can plug right into an ISDN network.

TE2 A *terminal equipment type 2 (TE2)* device refers to those that predate ISDN standards (are not natively ISDN-compliant). To use a TE2, you have to use a terminal adapter (TA) to be able to plug into an ISDN network. An example of a TE2 device would be a serial interface on a router, a standard PC, or even the modular interface of a common analog phone.

NT1 The *network termination 1 (NT1)* device implements the ISDN Physical layer specifications and connects the user devices to the ISDN network by converting the network from a fourwire to the two-wire network used by ISDN. Basically, we'll call this a U reference point that connects into the telco.

NT2 The *network termination 2 (NT2)* device is typically a provider's equipment, such as a switch or PBX. It also provides Data Link and Network layer implementation. It's very rare to find these on a customer's premises.

TA A *terminal adapter (TA)* converts TE2 non-ISDN signaling to signaling that's used by the ISDN

switch. It connects into an NT1 device for conversion into a two-wire ISDN network.

14.7.4 ISDN Reference Points

Reference points are a series of specifications that define the connection between the various equipment used in an ISDN network. ISDN has four reference points that define logical interfaces:

R The *R reference point* defines the point between non-ISDN equipment (TE2) and a TA.

S The *S reference point* defines the point between the customer router and an NT2. Enables

calls between the different customer equipment.

T The *T reference point* defines the point between NT1 and NT2 devices. S and T reference points are electrically the same and can perform the same function. Because of this, they're sometimes referred to as an S/T reference point.

U The *U reference point* defines the point between NT1 devices and local-termination equipment in a carrier network. (This is only in North America, where the NT1 function isn't provided by the carrier network.)

14.7.5 ISDN Protocols

ISDN protocols are defined by the ITU-T, and there are several series of protocols dealing with diverse issues (see www.itu.int for more info, but it is not needed for the exam):

- Protocols beginning with the letter *E* deal with using ISDN on the existing telephone network.
- Protocols beginning with the letter *I* deal with concepts, aspects, and services.
- Protocols beginning with the letter *Q* cover switching and signaling. These are the protocols that are used to connect to an ISDN network and to troubleshoot it. The Q.921 protocol describes the ISDN Data Link process of the Link Access Procedure on the D channel (LAPD). LAPD works on the D channel to provide out-of-band signaling for ISDN. The Q.931 specifies the OSI reference model Layer 3 functions. Q.931 passes between the DTE and the provider's switch on the D channel over Q.921-specified LAPD frames to set up, maintain, and tear down calls.

14.7.6 ISDN Switch Types

We can credit AT&T and Nortel for the majority of the ISDN switches in place today, but other companies also make them.

In Table 11.1, you'll find the keyword to use along with the isdn switch-type command to configure a router for the variety of switches it's going to connect to. If you don't know which switch your provider is using at their central office, call them to find out.

Switch Type	Keyword
AT&T basic rate switch	basic-5ess
Nortel DMS-100 basic rate switch	basic-dms100
National ISDN-1 switch	basic-ni1
AT&T 4ESS (ISDN PRI only)	primary-4ess
AT&T 5ESS (ISDN PRI only)	primary-5ess

Nortel DMS-100 (ISDN PRI only) primary-dms100

14.7.7 Basic Rate Interface (BRI)

ISDN Basic Rate Interface (BRI) service, also known as 2B+D, provides two B channels and one D channel. The BRI B-channel service operates at 64Kbps and carries data, while the BRI D-channel service operates at 16Kbps and usually carries control and signaling information.

The total bandwidth for ISDN BRI is then 144Kbps (64 + 64 + 16 = 144).

The D-channel signaling protocols (Q.921 and Q.931) span the OSI reference model's Physical, Data Link, and Network layers. The D channel carries signaling information to set up and control calls. The D channel can also be used for other functions, such as an alarm system for a building or anything else that doesn't need much bandwidth, since it's only giving you a whopping 16K. D channels work with LAPD at the Data Link layer for reliable connections.

When configuring ISDN BRI, you'll need to obtain service profile identifiers (SPIDs), and you should have one SPID for each B channel. The SPID is a unique number, often based on the directory number (DN), which is the dialable number of the ISDN subscriber, of which there are normally two provided on a 2B+D implementation.

The ISDN device gives the SPID to the ISDN switch, which then allows the device to access the network for BRI service. Without a SPID, many ISDN switches don't allow an ISDN device to place a call on the network.

To set up a BRI call, four events must take place:

1. The D channel between the router and the local ISDN switch comes up.
2. The ISDN switch uses the SS7 signaling technique to set up a path to a remote switch.
3. The remote switch sets up the D-channel link to the remote router.
4. The B channels are then connected end-to-end.

14.7.8 primary rate interface (PRI)

In North America and Japan, the ISDN **Primary Rate Interface (PRI)** service - also known as 23B+D - delivers 23 64Kbps B channels and one 64Kbps D channel, for a total bit rate of 1.544Mbps.

In Europe, Australia, and other parts of the world, ISDN provides 30 64Kbps B channels and one 64Kbps D channel, for a total bit rate of 2.048Mbps.

14.8 dial-on-demand Routing (DDR)

Dial-on-demand routing (DDR) is used to allow two or more Cisco routers to dial an ISDN dial-up connection on an as-needed basis. DDR is only used for low-volume, periodic network connections using either a Plain Old Telephone Service (POTS) or ISDN connection. This was designed to reduce WAN costs if you have to pay on a per-minute or per-packet basis.

DDR works when a packet received on an interface meets the requirements of an access list defined by an administrator, which defines interesting traffic. The following five steps give a basic description of how DDR works when an interesting packet is received in a router interface:

1. The route to the destination network is determined to be across the dial-up connection.

2. Interesting packets dictate a DDR call.
3. Dialer information is looked up and the call is placed.
4. Traffic is transmitted.
5. Call is terminated when no more interesting traffic is being transmitted over a link and the idle-timeout period ends.

chapter 15 wireless networks

15.1 wireless and mobile networks

Section 7.0 in K&R.

15.2 elements of wireless networks

Section 7.1 in K&R.

15.3 wireless networks characteristics, CDMA

Section 7.2 in K&R.

15.4 WiFi: the 802.11 wireless LANs

Section 7.3 in K&R.

15.5 cellular networks

Section 7.4 in K&R.

15.6 cellular networks evolution

The presentation 5G: an advanced introduction.

15.7 mobility in cellular networks

Section 7.7 in K&R.

chapter 16 IoT – the Internet of Things

Based on the ZDNet article - https://www.zdnet.com/article/what-is-the-internet-of-things-everything-you-need-to-know-about-the-iot-right-now/

16.1 what is it?

The Internet of Things, or IoT, refers to the billions of physical devices around the world that are now connected to the internet, all collecting and sharing data. Thanks to the arrival of super-cheap computer chips and the ubiquity of wireless networks, it's possible to turn anything, from something as small as a pill to something as big as an aeroplane, into a part of the IoT. Connecting up all these different objects and adding sensors to them adds a level of digital intelligence to devices that would be otherwise dumb, enabling them to communicate real-time data without involving a human being. The Internet of Things is making the fabric of the world around us more smarter and more responsive, merging the digital and physical universes.

An alternate definition can be found in Wikipedia:

The **Internet of things** (**IoT**) is a system of interrelated computing devices, mechanical and digital machines provided with unique identifiers (UIDs) and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

16.2 example of an IoT device

A lightbulb that can be switched on using a smartphone app is an IoT device, as is a motion sensor or a smart thermostat in your office or a connected streetlight. An IoT device could be as fluffy as a child's toy or as serious as a driverless truck. Some larger objects may themselves be filled with many smaller IoT components, such as a jet engine that's now filled with thousands of sensors collecting and transmitting data back to make sure it is operating efficiently. At an even bigger scale, smart cities projects are filling entire regions with sensors to help us understand and control the environment.

The term IoT is mainly used for devices that wouldn't usually be generally expected to have an internet connection, and that can communicate with the network independently of human action. For this reason, a PC isn't generally considered an IoT device and neither is a smartphone - even though the latter is crammed with sensors. A smartwatch or a fitness band or other wearable device might be counted as an IoT device, however.

16.3 historical elements

The idea of adding sensors and intelligence to basic objects was discussed throughout the 1980s and 1990s (and there are arguably some much earlier ancestors), but apart from some early projects -- including an internet-connected vending machine - progress was slow simply because the technology wasn't ready. Chips were too big and bulky and there was no way for objects to communicate effectively.

Processors that were cheap and power-frugal enough to be all but disposable were needed before it finally became cost-effective to connect up billions of devices. The adoption of RFID tags - low-power chips that can communicate wirelessly - solved some of this issue, along with the increasing availability of broadband internet and cellular and wireless networking. The adoption of IPv6 - which, among other things,

should provide enough IP addresses for every device the world (or indeed this galaxy) is ever likely to need - was also a necessary step for the IoT to scale.

Kevin Ashton coined the phrase 'Internet of Things' in 1999, although it took at least another decade for the technology to catch up with the vision.

"The IoT integrates the interconnectedness of human culture - our 'things' - with the interconnectedness of our digital information system - 'the internet.' That's the IoT," Ashton told ZDNet.

Adding RFID tags to expensive pieces of equipment to help track their location was one of the first IoT applications. But since then, the cost of adding sensors and an internet connection to objects has continued to fall, and experts predict that this basic functionality could one day cost as little as 10 cents, making it possible to connect nearly everything to the internet.

The IoT was initially most interesting to business and manufacturing, where its application is sometimes known as machine-to-machine (M2M), but the emphasis is now on filling our homes and offices with smart devices, transforming it into something that's relevant to almost everyone. Early suggestions for internet-connected devices included 'blogjects' (objects that blog and record data about themselves to the internet), ubiquitous computing (or 'ubicomp'), invisible computing, and pervasive computing. However, it was Internet of Things and IoT that stuck.

16.4 what is expected from the future?

Big and getting bigger - there are already more connected things than people in the world.

Tech analyst company IDC predicts that in total there will be 41.6 billion connected IoT devices by 2025, or "things." It also suggests industrial and automotive equipment represent the largest opportunity of connected "things,", but it also sees strong adoption of smart home and wearable devices in the near term.

Another tech analyst, Gartner, predicts that the enterprise and automotive sectors will account for 5.8 billion devices this year, up almost a quarter on 2019. Utilities will be the highest user of IoT, thanks to the continuing rollout of smart meters. Security devices, in the form of intruder detection and web cameras will be the second biggest use of IoT devices. Building automation – like connected lighting – will be the fastest growing sector, followed by automotive (connected cars) and healthcare (monitoring of chronic conditions).

16.5 the business impact of IoT

The benefits of the IoT for business depend on the particular implementation; agility and efficiency are usually top considerations. The idea is that enterprises should have access to more data about their own products and their own internal systems, and a greater ability to make changes as a result.

Manufacturers are adding sensors to the components of their products so that they can transmit data back about how they are performing. This can help companies spot when a component is likely to fail and to swap it out before it causes damage. Companies can also use the data generated by these sensors to make their systems and their supply chains more efficient, because they will have much more accurate data about what's really going on.

Enterprise use of the IoT can be divided into two segments: industry-specific offerings like sensors in a generating plant or real-time location devices for healthcare; and IoT devices that can be used in all industries, like smart air conditioning or security systems.

While industry-specific products will make the early running, by 2020 Gartner predicts that cross-industry devices will reach 4.4 billion units, while vertical-specific devices will amount to 3.2 billion units. Consumers purchase more devices, but businesses spend more: the analyst group said that while consumer spending on IoT devices was around $725bn last year, businesses spending on IoT hit $964bn. By 2020, business and consumer spending on IoT hardware will hit nearly $3tn.

Worldwide spending on the IoT was forecast to reach $745 billion in 2019, an increase of 15.4% over the $646 billion spent in 2018, according to IDC, and pass the $1 trillion mark in 2022.

Top industries for the IoT were predicted to be discrete manufacturing ($119 billion in spending), process manufacturing ($78 billion), transportation ($71 billion), and utilities ($61 billion). For manufacturers, projects to support asset management will be key; in transportation it will be freight monitoring and fleet management taking top priority. IoT spending in the utilities industry will be dominated by smart-grid projects for electricity, gas, and water.

Consumer IoT spending was predicted to hit $108 billion, making it the second largest industry segment: smart home, personal wellness, and connected vehicle infotainment will see much of the spending.

By use case, manufacturing operations ($100 billion), production asset management ($44.2 billion), smart home ($44.1 billion), and freight monitoring ($41.7 billion) will be the largest areas of investment.

16.6 the Industrial IoT

The Industrial Internet of Things (IIoT) or the fourth industrial revolution or Industry 4.0 are all names given to the use of IoT technology in a business setting. The concept is the same as for the consumer IoT devices in the home, but in this case the aim is to use a combination of sensors, wireless networks, big data, AI and analytics to measure and optimise industrial processes.

If introduced across an entire supply chain, rather than just individual companies, the impact could be even greater with just-in-time delivery of materials and the management of production from start to finish. Increasing workforce productivity or cost savings are two potential aims, but the IIoT can also create new revenue streams for businesses; rather than just selling a standalone product – for example, like an engine – manufacturers can also sell predictive maintenance of the engine.

16.7 consumer benefits

The IoT promises to make our environment - our homes and offices and vehicles -- smarter, more measurable, and... chattier. Smart speakers like **Amazon's Echo** and **Google Home** make it easier to play music, set timers, or get information. Home security systems make it easier to monitor what's going on inside and outside, or to see and talk to visitors. Meanwhile, smart thermostats can help us heat our homes before we arrive back, and smart lightbulbs can make it look like we're home even when we're out.

Looking beyond the home, sensors can help us to understand how noisy or polluted our environment might be. Self-driving cars and smart cities could change how we build and manage our public spaces.

However, many of these innovations could have major implications for our personal privacy.

16.8 IoT and 5G

Refer to https://www.techrepublic.com/resource-library/whitepapers/5g-what-it-means-for-iot-free-pdf/post/?regId=&asset=33168534 .

16.9 IoT and the smart homes

For consumers, the smart home is probably where they are likely to come into contact with internet-enabled things, and it's one area where the big tech companies (in particular Amazon, Google, and Apple) are competing hard.

The House that Alexa Built: An Amazon showcase in London in 2017.

The most obvious of these are smart speakers like Amazon's Echo, but there are also smart plugs, lightbulbs, cameras, thermostats, and the much-mocked smart fridge. But as well as showing off your enthusiasm for shiny new gadgets, there's a more serious side to smart home applications. They may be able to help keep older people independent and in their own homes longer by making it easier for family and carers to communicate with them and monitor how they are getting on. A better understanding of how our homes operate, and the ability to tweak those settings, could help save energy - by cutting heating costs, for example.

16.10 security issues

Security is one the biggest issues with the IoT. These sensors are collecting in many cases extremely sensitive data -- what you say and do in your own home, for example. Keeping that secure is vital to consumer trust, but so far the IoT's security track record has been extremely poor. Too many IoT devices give little thought to the basics of security, like encrypting data in transit and at rest.

Flaws in software -- even old and well-used code -- are discovered on a regular basis, but many IoT devices lack the capability to be patched, which means they are permanently at risk. Hackers are now actively targeting IoT devices such as routers and webcams because their inherent lack of security makes them easy to compromise and roll up into giant botnets.

Flaws have left smart home devices like refrigerators, ovens, and dishwashers open to hackers.

Researchers found 100,000 webcams that could be hacked with ease, while some internet-connected smartwatches for children have been found to contain security vulnerabilities that allow hackers to track the wearer's location, eavesdrop on conversations, or even communicate with the user.

Governments are growing worried about the risks here. The UK government has published its own guidelines around the security of consumer IoT devices. It expects devices to have unique passwords, that companies will provide a public point of contact so anyone can report a vulnerability (and that these will be acted on), and that manufacturers will explicitly state how long devices will get security updates. It's a modest list, but a start.

When the cost of making smart objects becomes negligible, these problems will only become more widespread and intractable.

All of this applies in business as well, but the stakes are even higher. Connecting industrial machinery to IoT networks increases the potential risk of hackers discovering and attacking these devices. Industrial espionage or a destructive attack on critical infrastructure are both potential risks. That means businesses will need to make sure that these networks are isolated and protected, with data encryption with security of sensors, gateways and other components a necessity. The current state of IoT technology makes that harder to ensure, however, as does a lack of consistent IoT security planning across organisations. That's very worrying considering the documented willingness of hackers to tamper with industrial systems that have been connected to the internet but left unprotected.

The IoT bridges the gap between the digital world and the physical world, which means that hacking into devices can have dangerous real-world consequences. Hacking into the sensors controlling the temperature in a power station could trick the operators into making a catastrophic decision; taking control of a driverless car could also end in disaster.

16.11 privacy issues

With all those sensors collecting data on everything you do, the IoT is a potentially vast privacy and security headache. Take the smart home: it can tell when you wake up (when the smart coffee machine is activated) and how well you brush your teeth (thanks to your smart toothbrush), what radio station you listen to (thanks to your smart speaker), what type of food you eat (thanks to your smart oven or fridge), what your children think (thanks to their smart toys), and who visits you and passes by your house (thanks to your smart doorbell). While companies will make money from selling you the smart object in the first place, their IoT business model probably involves selling at least some of that data, too.

What happens to that data is a vitally important privacy matter. Not all smart home companies build their business model around harvesting and selling your data, but some do.

And it's worth remembering that IoT data can be combined with other bits of data to create a surprisingly detailed picture of you. It's surprisingly easy to find out a lot about a person from a few different sensor readings. In one project, a researcher found that by analysing data charting just the home's energy consumption, carbon monoxide and carbon dioxide levels, temperature, and humidity throughout the day they could work out what someone was having for dinner.

Consumers need to understand the exchange they are making and whether they are happy with that. Some of the same issues apply to business: would your executive team be happy to discuss a merger in a meeting room equipped with smart speakers and cameras, for example? One recent survey found that four out of five companies would be unable to identify all the IoT devices on their network.

Badly installed IoT products could easily open up corporate networks to attack by hackers, or simply leak data. It might seem like a trivial threat but imagine if the smart locks at your office refused to open one morning or the smart weather station in the CEO's office was used by hackers to create a backdoor into your network.

16.12 the IoT field in the cyberwarfare

The IoT makes computing physical. So if things go wrong with IoT devices, there can be major real-world consequences -- something that nations planning their cyberwarfare strategies are now taking into account.

US intelligence community briefings have warned that the country's adversaries already have the ability to threaten its critical infrastructure as well "as the broader ecosystem of connected consumer and industrial devices known as the Internet of Things". US intelligence has also warned that connected thermostats, cameras, and cookers could all be used either to spy on citizens of another country, or to cause havoc if they were hacked. Adding key elements of national critical infrastructure (like dams, bridges, and elements of the electricity grid) to the IoT makes it even more vital that security is as tight as possible.

16.13 IoT and big data

An IoT device will likely contain one or more sensors which it will use to collect data. Just what those sensors are collecting will depend on the individual device and its task. Sensors inside industrial machinery might measure temperature or pressure; a security camera might have a proximity sensor along with sound and video, while your home weather station will probably be packing a humidity sensor. All this sensor data – and much, much more – will have to be sent somewhere. That means IoT devices will need to transmit data and will do it via Wi-Fi, 4G, 5G and more.

Tech analyst IDC calculates that within five years IoT gadgets will be creating 79.4 zettabytes of data. Some of this IoT data will be "small and bursty" says IDC – a quick update like a temperature reading from a sensor or a reading from a smart meter. Other devices might create huge amounts of data traffic, like a video surveillance camera using computer vision.

IDC said the amount of data created by IoT devices will grow rapidly in the next few years. Most of the data is being generated by video surveillance, it said, but other industrial and medical uses will generate more data over time.

It said drones will also be a big driver of data creation using cameras. Looking further out, self-driving cars will also generate vast amounts of rich sensor data including audio and video, as well as more specialised automotive sensor data.

The IoT generates vast amounts of data: from sensors attached to machine parts or environment sensors, or the words we shout at our smart speakers. That means the IoT is a significant driver of big-data analytics projects because it allows companies to create vast data sets and analyse them. Giving a manufacturer vast amounts of data about how its components behave in real-world situations can help them to make improvements much more rapidly, while data culled from sensors around a city could help planners make traffic flow more efficiently.

That data will come in many different forms – voice requests, video, temperature or other sensor readings, all of which can be mined for insight. As analyst IDC notes, IoT metadata category is a growing source of data to be managed and leveraged. "Metadata is a prime candidate to be fed into NoSQL databases like MongoDB to bring structure to unstructured content or fed into cognitive systems to bring new levels of understanding, intelligence, and order to outwardly random environments," it said.

In particular, the IoT will deliver large amounts of real-time data. Cisco calculates that machine-to machine connections that support IoT applications will account for more than half of the total 27.1 billion devices and connections, and will account for 5% of global IP traffic by 2021.

16.14 IoT and the cloud

The huge amount of data that IoT applications generate means that many companies will choose to do their data processing in the cloud rather than build huge amounts of in-house capacity. Cloud computing giants are already courting these companies: Microsoft has its Azure IoT suite, while Amazon Web Services provides a range of IoT services, as does Google Cloud.

16.15 IoT and the Artificial Intelligence

IoT devices generate vast amounts of data; that might be information about an engine's temperature or whether a door is open or closed or the reading from a smart meter. All this IoT data has to be collected, stored and analysed. One way companies are making the most of this data is to feed it into artificial intelligence (AI) systems that will take that IoT data and use it to make predictions.

For example, Google has put an AI in charge of its data centre cooling system. The AI uses data pulled from thousands of IoT sensors, which is fed into deep neural networks, and which predict how different choices will affect future energy consumption. By using machine learning and AI, Google has been able to make its data centres more efficient and said the same technology could have uses in other industrial settings.

16.16 what are the next steps?

As the price of sensors and communications continue to drop, it becomes cost-effective to add more devices to the IoT – even if in some cases there's little obvious benefit to consumers. Deployments are at an early stage; most companies that are engaging with the IoT are at the trial stage right now, largely because the necessary technology – sensor technology, 5G and machine-learning powered analytics – are still themselves at a reasonably early stage of development. There are many competing platforms and standards and many different vendors, from device makers to software companies to network operators, want a slice of the pie. It's still not clear which of those will win out. But without standards, and with security an ongoing issue, we are likely to see some more big IoT security mishaps in the next few years.

As the number of connected devices continues to rise, our living and working environments will become filled with smart products – assuming we are willing to accept the security and privacy trade-offs. Some will welcome the new era of smart things. Others will pine for the days when a chair was simply a chair.

chapter 17 network security

17.1