## 1. NETWORK HARDWARE

Network hardware has 2 main dimensions : Transmission technology

Scale

**Transmission Technology**

- In transmission technology there are 2 types. **broadcast** links and **point-to-point** links.
- Point-to-point links connect individual pairs of machines. To go from the source to the destination on a network made up of point-to-point links, short messages, called **packets** is used.
- Point-to-point transmission with exactly one sender and exactly one receiver is sometimes called **unicasting**.
- On a broadcast network, the communication channel is shared by all the machines on the network; packets sent by any machine are received by all the others. An address field within each packet specifies the intended recipient. Upon receiving a packet, a machine checks the address field. If the packet is intended for the receiving machine, that machine processes the packet; if the packet is intended for some other machine, it is just ignored.
- A wireless network is a common example of a broadcast link, with communication shared over a coverage regionthat depends on the wireless channel and the transmitting machine.
- Broadcast systems usually also allow the possibility of addressing a packet to *all* destinations by using a special code in the address field. When a packet with this code is transmitted, it is received and processed by every machine on the network. This mode of operation is called **broadcasting**. Some broadcast systems also support transmission to a subset of the machines, which known as **multicasting**.

**Scale:**

Distance is important as a classification metric because different technologies are used at different scales.

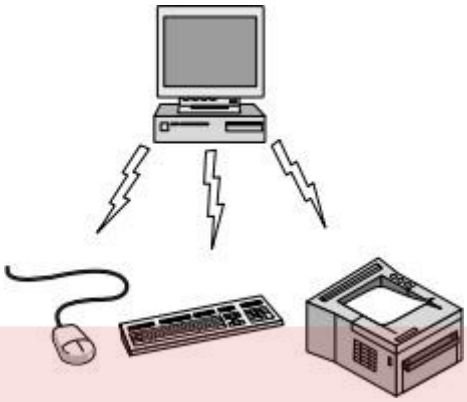
| Interprocessor distance | Processors located in same | Example |
|---|---|---|
| 1 m | Square meter | Personal area network |
| 10 m | Room | Local area network |
| 100 m | Building | |
| 1 km | Campus | |
| 10 km | City | Metropolitan area network |
| 100 km | Country | Wide area network |
| 1000 km | Continent | |
| 10,000 km | Planet | The Internet |

At the top are the personal area networks, networks that are meant for one person. Beyond these come longer-range networks. These can be divided into local, metropolitan, and wide area networks, each with increasing scale.

## 1.1 Personal Area Networks:

**PANs** (**Personal Area Networks**) let devices communicate over the range of a person.

- A common example is a wireless network that connects a computer with its peripherals. In the simplest form, Bluetooth networks use the master-slave paradigm.



- The system unit (the PC) 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, what frequencies they can use, and so on. Bluetooth can be used in other settings, too.
- It is often used to connect a headset to a mobile phone without cords and it can allow your digital music player to connect to your car merely being brought within range.
- A completely different kind of PAN is formed when an embedded medical device such as a pacemaker, PANs can also be built with other technologies that communicate over short ranges, such as RFID on smartcards insulin pump, or hearing aid talks to a user-operated remote control.

## 1.2 Local Area Network

- A LAN is a privately owned network that operates within and nearby a single building like a home, office or factory. LANs are widely used to connect personal computers and consumer electronics to let them share resources (e.g., printers) and exchange information. When LANs are used by companies, they are called **enterprise networks**.
- Each computer talks to a device in the ceiling through a device called AP(Access Point) or base station relays packets between the wireless computers and also between them and the Internet. There is a standard for wireless LANs called **IEEE 802.11**, popularly known as **WiFi**, which has become very widespread. It runs at speeds anywhere from 11 to hundreds of Mbps.

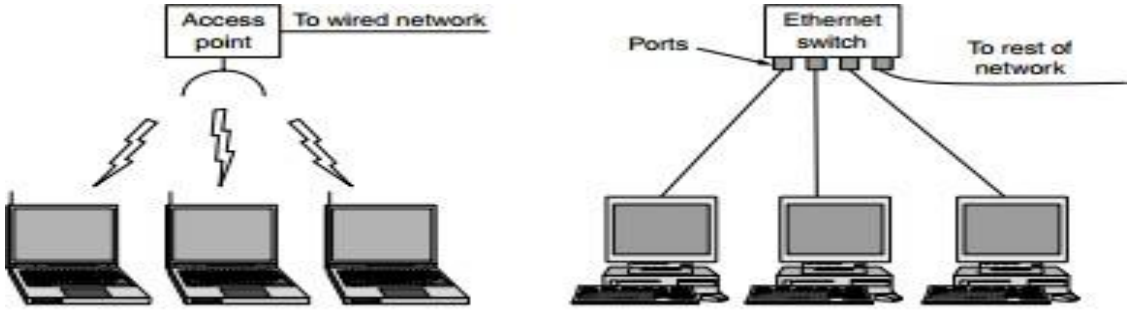



**Figure 1-8.** Wireless and wired LANs. (a) 802.11. (b) Switched Ethernet.

- Wired LANs use a range of different transmission technologies. Most of them use copper wires, but some use optical fiber. LANs are restricted in size, which means that the worst-case transmission time is bounded and known in advance. The most commonly used wired LANs is IEEE802.3, called as **Ethernet.**

**Switched Ethernet**

- Each computer speaks the Ethernet protocol and connects to a box called a **switch** with a point-to-point link. A switch has multiple **ports**, each of which can connect to one computer.
- The job of the switch is to relay packets between computers that are attached to it, using the address in each packet to determine which computer to send it to.
- To build larger LANs, switches can be plugged into each other using their ports. It is also possible to divide one large physical LAN into two smaller logical LANs.
- Depending on the how the channel is allocated, it is divided into static and dynamic designs.
  - ❖ **Static allocation:** A typical static allocation would be to divide time into discrete intervals and use a round-robinalgorithm, allowing each machine to broadcast only when its time slot comes up.
  - ❖ **Dynamic allocation:** Dynamic allocation can be done by either centralized or decentralized.

    Centralized channel: There is a single entity, for example, the base station in cellular

    networks.
- Decentralized channel: There is no central entity; each machine must decide for itself whether to transmit.
- Many devices are already capable of being **networked**. These include computers, entertainment devices such as TVs and DVDs, phones and other consumer electronics such as cameras, appliances like clock radios, and infrastructure like utility meters and thermostats.

**Home Network:**

- Home networks can be just as another LAN
  - ❖ First, the networked devices have to be very easy to install. Wireless routers are the most returned consumerelectronic item.
  - ❖ Second, the network and devices have to be foolproof in operation.

- ❖ Third, low price is essential for success.
- ❖ Fourth, it must be possible to start out with one or two devices and expand the reach of the network gradually.
- ❖ Fifth, security and reliability will be very important

- According to the convenience and cost favours, the home networks can be wired or wireless. Security favours thewired networking.
- **Power-line networks** let devices that plug into outlets broadcast information throughout the house.

## 1.3 Metropolitan Area Networks:

- A **MAN** (**Metropolitan Area Network**) covers a city. The best-known examples of MANs are the cabletelevision networks available in many cities.
- Initially these were designed locally, and then companies started to expand to wire up the entire cities. The nextstep was television programming and even entire channels designed for cable only.
- When the Internet began attracting a mass audience, the cable TV network operators began to realize that withsome changes to the system, they could provide two-way Internet service in unused parts of the spectrum.

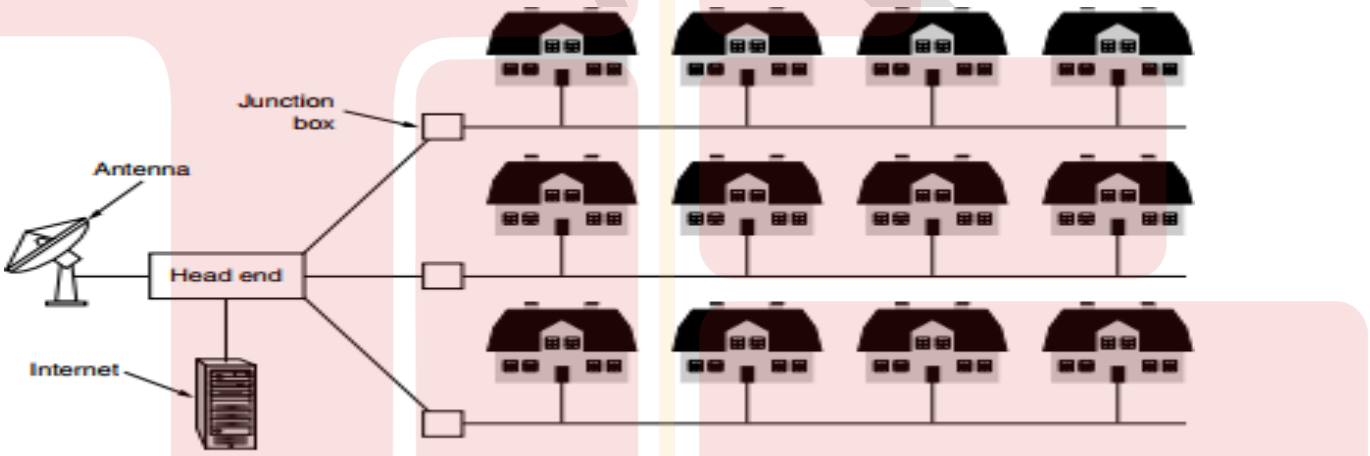
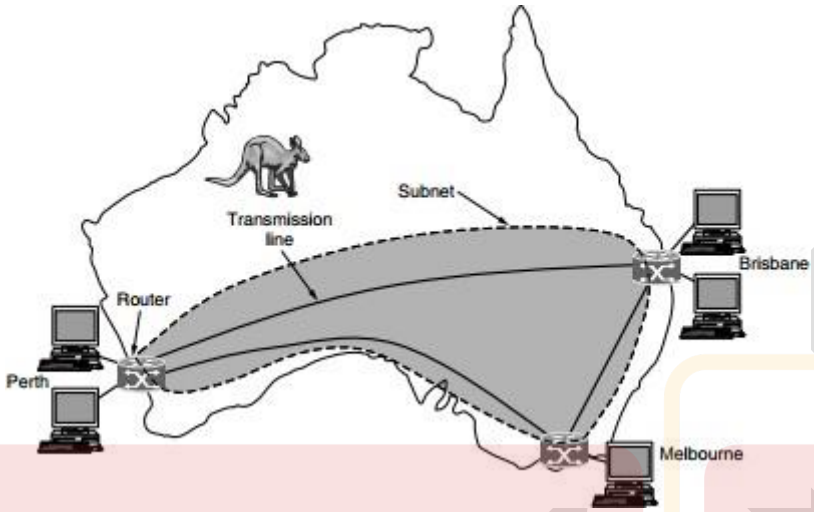



**Figure 1-9.** A metropolitan area network based on cable TV.

- Both the television signals and Internet is being fed into the centralized **cable headend** for subsequentdistribution to people's homes.
- Recent developments in high speed wireless Internet access have resulted in another MAN, which has beenstandardized as IEEE 802.16 and is popularly known as **WiMAX**.

## 1.4 Wide Area Networks

A **WAN** (**Wide Area Network**) spans a large geographical area, often a country or continent.




The fig shows the network that connects offices in Perth, Melbourne and Brisbane. Each offices contain the computers and these are called machine **hosts** in the traditional usage. The rest of the network that connects these hosts is then called the **communication subnet**, or just **subnet.** The job of the subnet is to carry messages from host to host.

- The subnet 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**, or just **switches**, are specialized computers that connect two 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.
- The two varieties of WAN

i) WAN using a VPN(virtual private network)
ii) WAN using ISP network

- ❖ **WAN using a VPN(virtual private network)**

  A company might connect its offices to the Internet This allows connections to be made between the offices as virtual links that use the underlying capacity of the Internet. This is called VPN.

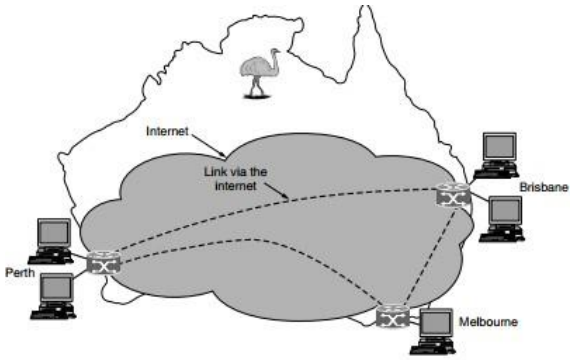



**Figure 1-11.** WAN using a virtual private network.

Compared to the dedicated arrangement, a VPN has the usual advantage of virtualization, which is that it provides flexible reuse of a resource.

❖ **WAN using ISP network**

The subnet operator is known as a **network service provider** and the offices are its customers.. The subnet operator will connect to other customers too, as long as they can pay and it can provide service. Such a subnet operator is called an **ISP** (**Internet Service Provider**) and the subnet is an **ISP network.**

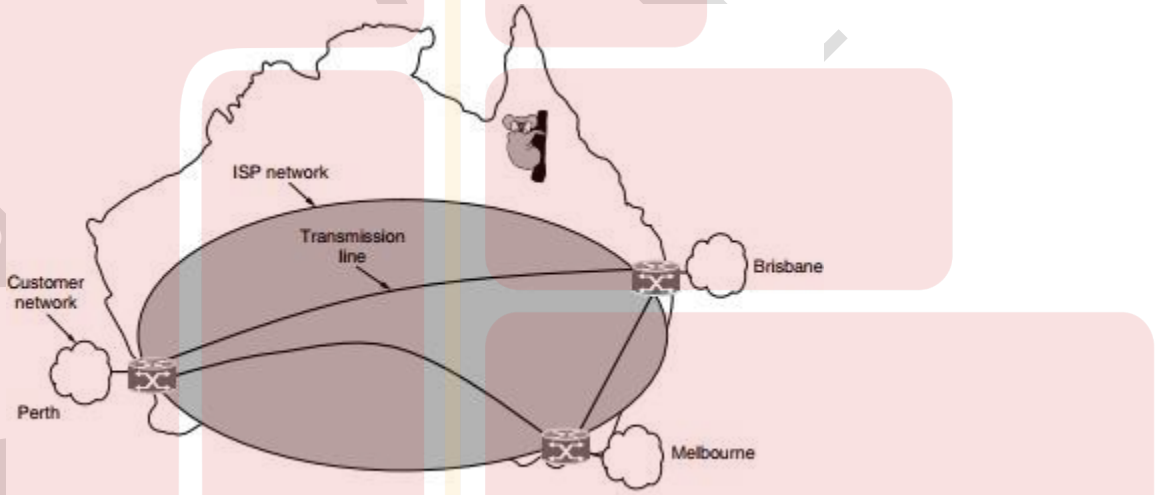



**Figure 1-12.** WAN using an ISP network.

- In most WANs, the network contains many transmission lines, each 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. The network makes the decision as to which path to use is called the **routing algorithm**. Each router makes the decision as to where to send a packet next is called the **forwarding algorithm**.

**Satellite System**

- It is one of wireless WAN technologies. Each computer on the ground has an antenna through which it can send data to and receive data from to a satellite in orbit. All computers can hear the output *from* the

satellite, and in some cases they can also hear the upward transmissions of their fellow computers *to* the satellite as well. Satellite networks are inherently broadcast and are most useful when the broadcast property is important.

**Cellular Telephone Network**

- Another example of a WAN that uses wireless technology. There are so many generations in the cellular telephonenetwork.
- The first generation was analog and for voice only. The second generation was digital and for voice only. The third generation is digital and is for both voice and data. Each cellular base station covers a distance much larger than a wireless LAN, with a range measured in kilometers rather than tens of meters.
- The base stations are connected to each other by a backbone network that is usually wired. The data rates of cellular networks are often on the order of 1 Mbps, much smaller than a wireless LAN that can range up to on the order of 100 Mbps.

## 1.5 Internetworks:

- A collection of interconnected networks is called an **internetwork** or **internet.** The Internet uses ISP networks to connect enterprise networks, home networks, and many other networks.
- The term ‘‘subnet’’ refers to the collection of routers and communication lines owned by the network operator. A network is formed by the combination of a subnet and its hosts. Connecting a LAN and a WAN or connecting two LANs is the usual way to form an internetwork.
- The general name for a machine that makes a connection between two or more networks and provides the necessary translation, both in terms of hardware and software, is a **gateway.** Gateways are distinguished by the layer at which they operate in the protocol hierarchy.

## 1.2 NETWORK SOFTWARE

### 1.2.1 Protocol hierarchies

- To reduce the design complexity, most networks are organized as a stack of **layers** or **levels**, each one built upon the one below it. The number of layers, the name of each layer, the contents of each layer, and the function of each layer differ from network to network.
- The purpose of each layer is to offer certain services to the higher layers while shielding those layers from the details of how the offered services are actually implemented.
- A **protocol** is an agreement between the communicating parties on how communication is to proceed.

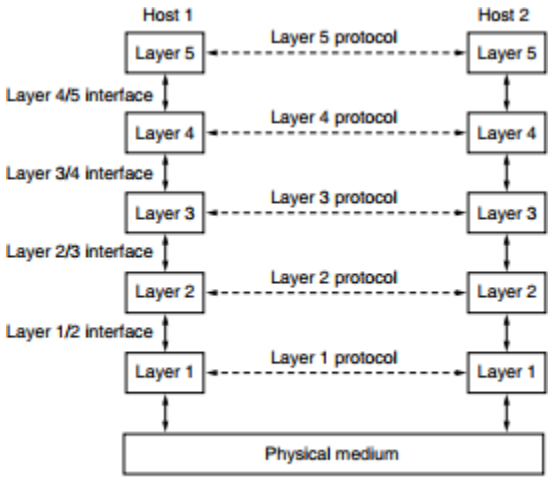
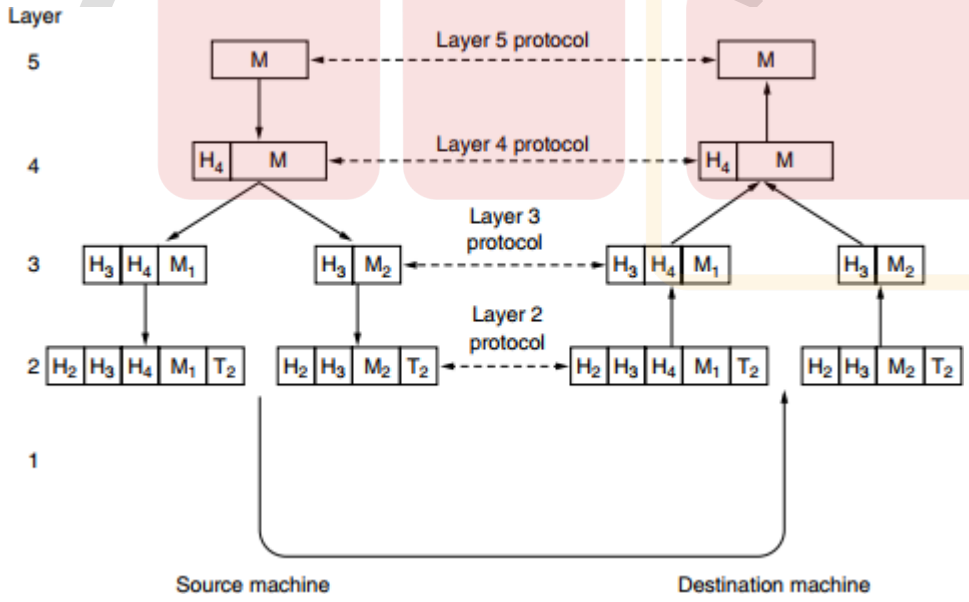



**Figure 1-13.** Layers, protocols, and interfaces.

- The entities comprising the corresponding layers on different machines are called **peers**. The peers may be software processes, hardware devices, or even human beings. No data are directly transferred from layer *n* on one machine to layer *n* on another machine. It is transferred through the physical medium( actual communication occurs). Virtual communication is shown by dotted lines and physical communication by solid lines.
- Between each pair of adjacent layers is an **interface**. The interface defines which primitive operations and services the lower layer makes available to the upper one.
- A set of layers and protocols is called a **network architecture**. The specification of an architecture must contain enough information to allow an implementer to write the program or build the hardware for each layer so that it will correctly obey the appropriate protocol. A list of the protocols used by a certain system, one protocol per layer, is called a **protocol stack.**



- The above diagram is the example of providing communication to the top layer of 5 layer network.
- A message, *M*, is produced by an application process running in layer 5 and given to layer 4 for transmission.
- Layer 4 puts a **header** in front of the message to identify the message and passes the result to layer 3. The header includes control information, such as addresses, to allow layer 4 on the destination machine to deliver the message.
- Layer 3 decides which of the outgoing lines to use and passes the packets to layer 2. Layer 2 adds to each piece not only a header but also a trailer, and gives the resulting unit to layer 1 for physical transmission.
- At the receiving machine the message moves upward, from layer to layer, with headers being stripped off as it progresses.

### 1.2.2 Design Issues For the Layers

Some of the design issues are

i) Reliability ii) Evolution of the network iii) Resource Allocation iv) Security

- **Reliability**
  - ❖ One mechanism for finding errors in received information uses codes for **error detection**. Information that is incorrectly received can then be retransmitted until it is received correctly. **error correction**, where the correct message is recovered from the possibly incorrect bits that were originally received.
  - ❖ Another reliability issue is finding a working path through a network. The process of selecting a path for traffic in a network or between or across multiple networks is called **routing.**
- **Evolution of network**
  - ❖ Since the networks grow larger and new designs emerge in the existing networks, key structuring mechanism used to support the change by dividing the overall problem called, **protocol layering** is implemented**.**
  - ❖ Every layer needs a mechanism for identifying the senders and receivers that are involved in a particular message. This mechanism is called **addressing** or **naming**, in the low and high layers, respectively.
  - ❖ The mechanism which involves in doing disassembling, transmitting, and then reassembling messages in overall network is called **Internetwork.** Designs that continue to work well when the network gets large are said to be **scalable**.
- **Resource Allocation**
  - ❖ Networks provide a service to hosts from their underlying resources, such as the capacity of transmission lines(bandwidth).

**Statistical Multiplexing**

- Sharing the resources based on the statistics of demand. It can be applied at low layers for a single link, or at high layers for a network or even applications that use the network.

- **Flow Control**
  - Flow control is a technique used to regulate data transfer between computers or other nodes in a network. Flowcontrol ensures that the transmitting device does not send more data to the receiving device than it can handle.
- **Congestion**
  - When too many computers want to send the data and the network cannot handle it all. Thos overloading ofcomputers are called congestion. This makes the end users' network slow.
- **Quality of service(QoS)**
  - QoS is the use of mechanisms or technologies that work on a network to control traffic and ensure theperformance of critical applications with limited network capacity.
- **Security**
  - **Confidentiality:** The data is only available to authorized parties. When information has been kept confidential, itmeans that it has not been compromised by other parties.
  - **Authentication:** Authentication is used by a client when the client need to know that the server is system itclaims to be. In authentication, the user or computer has to prove its identity to the server or client.
  - **Integrity:** Integrity means guarding against improper information modification or destruction and includesensuring information non repudiation and authencity.

### 1.2.3 Connection Oriented Versus Connectionless Service:

- **Connection oriented:** The service user first establishes a connection, uses the connection, and then releases the connection.
  - The essential aspect of a connection is that it acts like a tube: the sender pushes objects (bits) in at one end, and the receiver takes them out at the other end.
  - A **circuit** is another name for a connection with associated resources, such as a fixed bandwidth.
  - Reliable connection-oriented service has two minor variations: message sequences and byte streams. In the former variant, the message boundaries are preserved. When two 1024-byte messages are sent, they arrive as two distinct 1024- byte messages, never as one 2048-byte message.
  - In the latter, the connection is simply a stream of bytes, with no message boundaries. When 2048 bytes arrive at the receiver, there is no way to tell if they were sent as one 2048-byte message, two 1024-byte messages, or 2048 1-byte messages.
- **Connectionless Service:** Each message (letter) carries the full destination address, and each one is routed

throughthe intermediate nodes inside the system independent of all the subsequent messages.

- A **packet** is a message at the network layer. When the intermediate nodes receive a message in full before sending it on to the next node, this is called **store-and-forward s witching.** The alternative, in which the onward transmission of a message at a node starts before it is completely received by the node, is called **cut-through s witching.**
- **Voice over IP:** For some applications, the transit delays introduced by acknowledgements are unacceptable. One such application is digitized voice traffic for **voice over IP**. It is less disruptive for telephone users to hear a bit of noise on the line from time to time than to experience a delay waiting for acknowledgements.
- Unreliable (meaning not acknowledged) connectionless service is often called **datagram** service. The six different types of service are

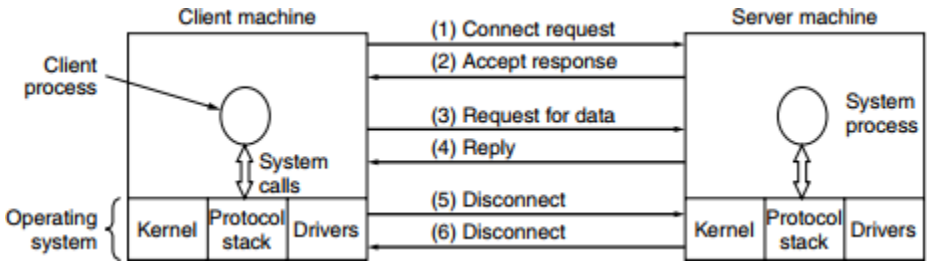
| | Service | Example |
|---|---|---|
| Connection-oriented | Reliable message stream | Sequence of pages |
| | Reliable byte stream | Movie download |
| | Unreliable connection | Voice over IP |
| Connection-less | Unreliable datagram | Electronic junk mail |
| | Acknowledged datagram | Text messaging |
| | Request-reply | Database query |

## 1.2.4 Service Primitivies

- A service is formally specified by a set of **primitives** (operations) available to user processes to access the service. These primitives tell the service to perform some action or report on an action taken by a peer entity. The set of primitives available depends on the nature of the service being provided. The primitives for connection- oriented service are different from those of connectionless service.

| Primitive | Meaning |
|---|---|
| LISTEN | Block waiting for an incoming connection |
| CONNECT | Establish a connection with a waiting peer |
| ACCEPT | Accept an incoming connection from a peer |
| RECEIVE | Block waiting for an incoming message |
| SEND | Send a message to the peer |
| DISCONNECT | Terminate a connection |

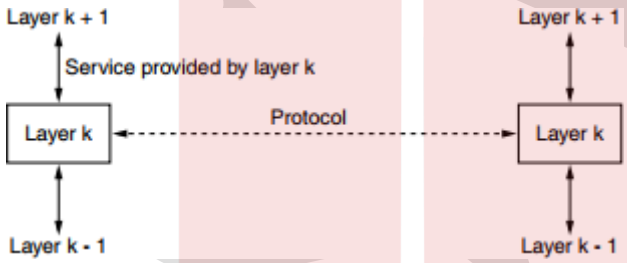
- First, the server executes LISTEN to indicate that it is prepared to accept incoming connections. Next, the client process executes CONNECT to establish a connection with the server. The CONNECT call needs to specify who to connect to.




- Then the client executes SEND to transmit its request (3) followed by the execution of RECEIVE to get the reply. The arrival of the request packet at the server machine unblocks the server so it can handle the request. After it has done the work, the server uses SEND to return the answer to the client (4). When the client is done, it executes DISCONNECT to terminate the connection (5). When the server gets the packet, it also issues a DISCONNECT of its own, acknowledging the client and releasing the connection (6).

### 1.2.5 The Relationship of Services to Protocols

- **Service:** A *service* is a set of primitives (operations) that a layer provides to the layer above it. The service defines what operations the layer is prepared to perform on behalf of its users, but it says nothing at all about how these operations are implemented. A service relates to an interface between two layers, with the lower layer being the service provider and the upper layer being the service user.
- **Protocol:** A Protocol is a set of rules governing the format and meaning of the packets, or messages that are exchanged by the peer entities within a layer. Entities use protocols to implement their service definitions.

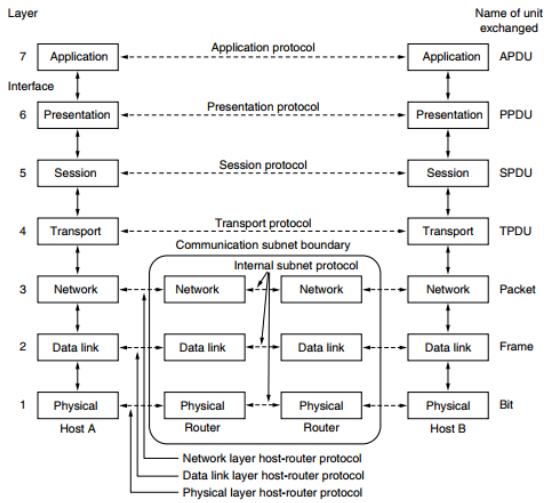



## 1.3 Reference Models

There are two important network architectures.

i) OSI reference model
ii) TCP/IP reference model

### 1.3.1 The OSI Reference Model




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

1. A layer should be created where a different abstraction is needed.

2. Each layer should perform a well-defined function.

3. The function of each layer should be chosen with an eye toward defining internationally standardized protocols.

4. The layer boundaries should be chosen to minimize the information flow across the interfaces.
5. 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.

- **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.
- **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. The sender breaks up the input data into **data frame** and transmit the frames sequentially.
  - ❖ If the service is reliable, the receiver confirms correct receipt of each frame by sending back an **acknowledgement frame**. The medium access control sublayer deals how to control access to the shared channel.
- **The Network Layer**
  - ❖ The **network layer** controls the operation of the subnet. A key design issue is determining how packets are routedfrom source to destination. Routes can be based on static tables that are ‘‘wired into’’ the network

and rarely changed, or more often they can be updated automatically to avoid failed components. Handling congestion is also the responsibility of the network layer.

- **The Transport Layer**

❖ The basic function of the **transport layer** is to accept data from above it, split it up into smaller units if need be, pass these to the network layer, and ensure that the pieces all arrive correctly at the other end. The transport layer also determines what type of service to provide to the session layer, and, ultimately, to the users of the network. The transport layer is a true end-to-end layer; it carries data all the way from the source to the destination.

- **The Session Layer**

❖ The session layer allows users on different machines to establish **sessions** between them. Sessions offer various services, including **dialog control** , **token management** (preventing two parties from attempting the same criticaloperation simultaneously), and **synchronization.**

- **The Presentation Layer**

❖ The **presentation layer** is concerned with the syntax and semantics of the information transmitted. The presentation layer manages these abstract data structures and allows higher-level data structures (e.g., banking records) to be defined and exchanged.

- **The Application Layer**

❖ The **application layer** contains a variety of protocols that are commonly needed by users. One widely used application protocol is **HTTP** (**HyperText Transfer Protocol**),

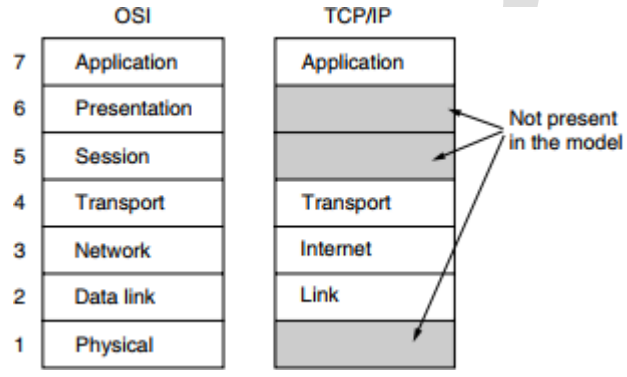
### 1.3.2 The TCP/IP Reference Model

- **The Link Layer**

❖ The **link layer** describes what links such as serial lines and classic Ethernet must do to meet the needs of this connectionless internet layer.

- **The Internet Layer**

❖ The **internet layer** is the linchpin that holds the whole architecture together.



- ❖ Its job is to permit hosts to inject packets into any network and have them travel independently to the destination (potentially on a different network). They may even arrive in a completely different order than they were sent.
- ❖ The internet layer defines an official packet format and protocol called **IP** (**Internet Protocol**), plus a companion protocol called **ICMP** (**Internet Control Message Protocol**) that helps it function.

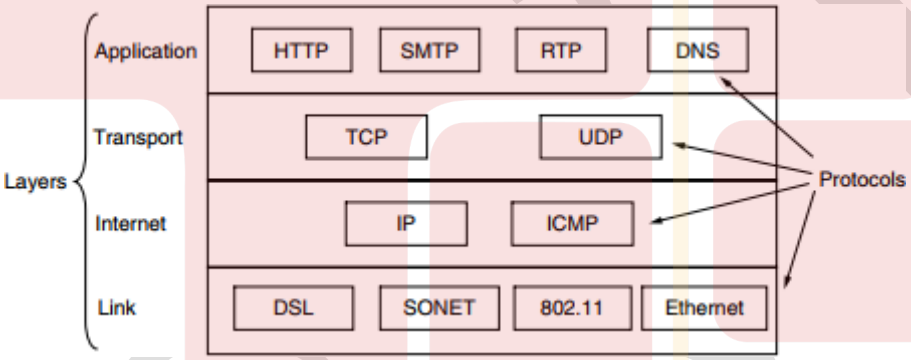
- **The Transport Layer**

❖ The layer above the internet layer in the TCP/IP model is now usually called the **transport layer**. Two end-to- end transport protocols have been defined here. The first one, **TCP** (**Transmission Control Protocol**), is a reliable connection-oriented protocol that allows a byte stream originating on one machine to be delivered without error on any other machine in the internet.

❖ The second protocol in this layer, **UDP** (**User Datagram Protocol**), is an unreliable, connectionless protocol for applications that do not want TCP's sequencing or flow control and wish to provide their own.

- **The Application Layer**

❖ On top of the transport layer is the **application layer**. It contains all the higher-level protocols.




## 1.3.3 A Comparison of OSI and TCP/IP Reference Models

| Parameters | OSI Model | TCP/IP Model |
|---|---|---|
| Full Form | Open Systems Interconnection | Transmission Control Protocol/ Internet Protocol |
| Layers | It has 7 layers | It has 4 layers |
| Usage | It is low in usage | It is mostly used. |
| Approach | It is vertically approached | It is horizontally approached |
| Delivery | Delivery of package is guaranteed. | Delivery is not guaranteed |
| Replacement | Changes can be easily done | Replacing the tools is not easy. |
| Reliability model | Less reliable than TCP/IP model | It is more reliable than OSI |

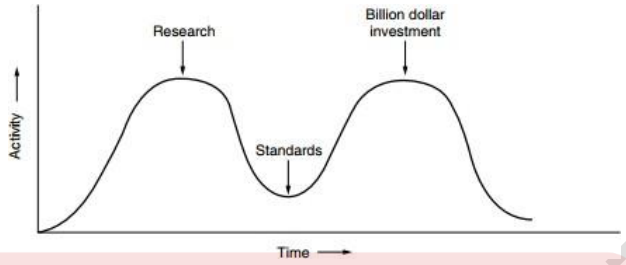
## 1.3.4 A Critique of OSI Model and Protocols

A critique of OSI Model and Protocols are

a) Bad Timing b) Bad Technology c) Bad Implementation d) Bad politics

**Bad Timing**

- The time at which a standard is established is absolutely critical to its success.




- When the subject is first discovered, there is a burst of research activity in the form of discussions, papers, and meetings. After a while this activity subsides, corporations discover the subject, and the billion-dollar wave of investment hits.
- If they are written too early (before the research results are well established), the subject may still be poorly understood; the result is a bad standard. If they are written too late, so many companies may have already made major investments in different ways of doing things that the standards are effectively ignored.

**Bad Technology**

The choice of seven layers was a bad one as two of the layers (session and presentation) are nearly empty, whereas two other ones (data link and network) are overfull. Another problem with OSI is that some functions, such as addressing, flow control, and error control, reappear again and again in each layer.

**Bad Implementation**

The initial implementations were huge, unwieldy, and slow.

**Bad Politics**

OSI, was widely thought to be the creature of the European telecommunication ministries, the European Community, and later the U.S. Government.

## 1.3.6 A Critique of TCP/IP Reference Model

1) The model does not clearly distinguish the concepts of services, interfaces, and protocols.
2) The TCP/IP model is not at all general and is poorly suited to describing any protocol stack other than TCP/IP.
3) The link layer is not really a layer at all in the normal sense of the term as used in the context of layered protocols.

4) The TCP/IP model does not distinguish between the physical and data link layers.

## 1.4 The Physical Layer

The physical layer is the first and lowest layer of the OSI communication model.

Its function is to transport data using electrical, mechanical or procedural interfaces.

### 1.4.1 Guided Transmission Medium

The purpose of the physical layer is to transport bits from one machine to another. Various physical media can be used for the actual transmission.

#### 1.4.1.1 Magnetic Media

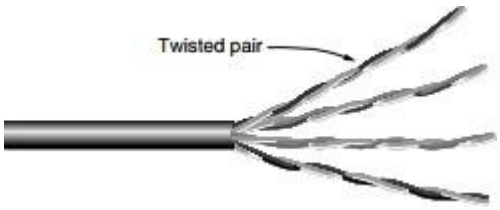
One of the most common ways to transport data from one computer to another is to write them onto magnetic tape or removable media (e.g., recordable DVDs), physically transport the tape or disks to the destination machine, and read them back in again.

Common types of magnetic media are, audio reel to reel cassettes tapes, hard disk drives, floppy disks etc..

#### 1.4.1.2 Twisted Pairs

- 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. A signal is usually carried as the difference in voltage between the two wires in the pair. This provides better immunity to external noise because the noise tends to affect both wires the same, leaving the differential unchanged.
- 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. Both telephone calls and ADSL Internet access run over these lines.
- Twisted pairs can be used for transmitting either analog or digital information. The bandwidth depends on the thickness of the wire and the distance traveled, but several megabits/sec can be achieved for a few kilometres

**Varieties of Twisted pair Cabling**



The garden variety deployed in many office buildings is called **Category 5** cabling, or ‘‘Cat 5.

A category 5 twisted pair consists 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. To reach higher speeds, 1-Gbps Ethernet uses all four pairs in both directions simultaneously.

**Full duplex**

Links that can be used in both directions at the same time, like a two-lane road, are called **full-duplex** links.

**Half duplex**

Links that can be used in either direction, but only one way at a time, like a single-track railroad line, called **half-duplex** links.

**Simplex**

Links that allow traffic in only one direction, like a one-way street are called **simplex** links.

**Category 3**

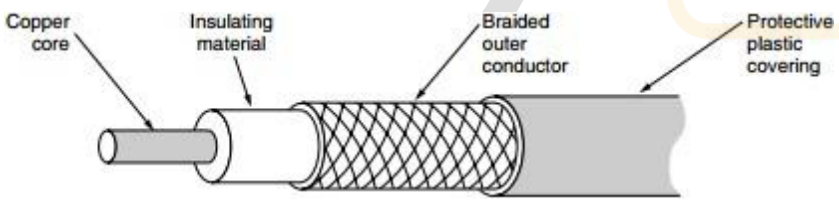
Cable uses a same connector, but has more twists per meter. More twists result in less crosstalk and a better-quality signal over longer distances, making the cables more suitable for high-speed computer communication, especially 100-Mbps and 1-Gbps Ethernet LANs.

**Category 6 and Category 7**

New wiring use these categories. It has more specifications to handle signals with greater bandwidths. In category 6, wiring types are referred to as UTP( Unshielded Twisted Pair). Category 7 cables have shielding on individual twisted pair. Shielding reduces the susceptibility to external interference and crosstalk with other nearby cables to meet demanding performance specifications.

**1.4.1. 3 Coaxial Cable**

- Coaxial cable has better shielding and greater bandwidth than unshielded 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.

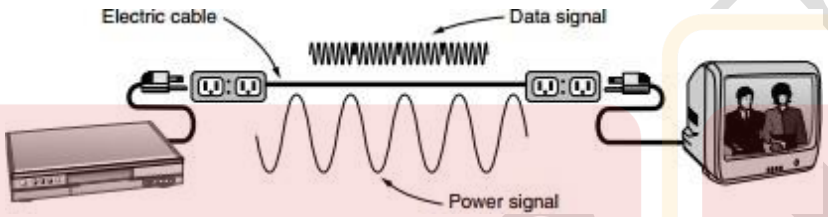



- A coaxial cable consists of a stiff copper wire as the core, surrounded by an insulating material. The insulator is encased by a cylindrical conductor, often as a closely woven braided mesh. The outer conductor is covered in a protective plastic sheath.

- The bandwidth possible depends on the cable quality and length. Coaxial cables used to be widely used within thetelephone system for long-distance lines but have now largely been replaced by fiber optics on longhaul routes. Coax is still widely used for cable television and metropolitan area networks.

### 1.4.1.4 Power Lines

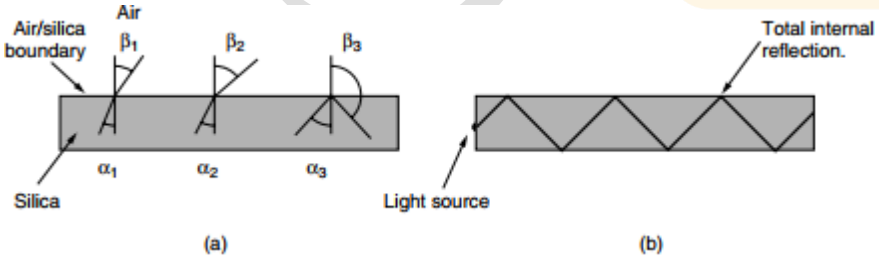
- Power lines deliver electrical power to houses, and electrical wiring within houses distributes the power to electrical outlets.
- Power lines have been used by electricity companies for low-rate communication such as remote metering. The convenience of using power lines for networking should be like simply plug a TV and a receiver into the wall, because they need power, and they can send and receive movies over the electrical wiring.




- Electrical signals are sent at 50–60 Hz and the wiring attenuates the much higher frequency (MHz) signals needed for high-rate data communication. The electrical properties of the wiring vary from one house to the next and change as appliances are turned on and off, which causes data signals to bounce around the wiring.

### 1.4.1.5 Fiber Optics

- Fiber optics are used for long-haul transmission in network backbones, highspeed LANs (although so far, copper has always managed catch up eventually), and high-speed Internet access such as **FttH** (**Fiber to the Home**).
- An optical transmission system has three key components: the light source, the transmission medium, and the detector. The transmission medium is an ultra-thin fiber of glass. The detector generates an electrica l 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.




Fig(a): 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. A light ray incident on the boundary at an angle α1 emerging at an angle β1. The amount of refraction depends on the properties of the two media (in particular, their indices of refraction)

Fig (b): 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. The fig shows the only one trapped ray, but since any light ray incident on the boundary above the critical angle will be reflected internally, many different rays will be bouncing around at different angles. Each ray is said to have a different mode, so a fiber having this property is called a **multimode fiber**.

**Single mode fiber**

if the fiber's diameter is reduced to a few wavelengths of light the fiber acts like a wave guide and the light can propagate only in a straight line, without bouncing, yielding a **single-mode fiber.** Single- mode fibers are more expensive but are widely used for longer distances.

**Transmission of Light Through Fiber**

Optical fibers are made of glass, which, in turn, is made from sand, an inexpensive raw material available in unlimited amounts. The attenuation of light through glass depends on the wavelength of the light. It is defined as the ratio of input to output signal power.

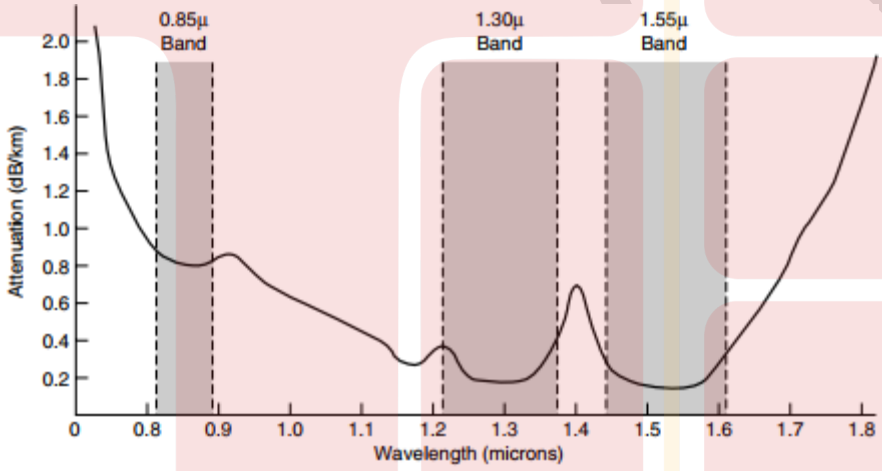



**Figure 2-7.** Attenuation of light through fiber in the infrared region.

The figure shows the near- infrared part of the spectrum. Visible light has slightly shorter wavelengths, from 0.4 to 0.7 microns. The true metric purist would refer to these wavelengths as 400 nm to 700 nm. Three wavelength bands are most commonly used at present for optical communication. They are centered at 0.85, 1.30, and 1.55 microns, respectively. All three bands are 25,000 to 30,000 GHz wide. The 0.85- micron band was used first. It has higher attenuation and so is used for shorter distances. The last two bands have good attenuation properties (less than 5% loss per kilometer). The 1.55- micron band is now widely used with erbium-doped amplifiers that work directly in the optical domain.

**Chromatic Dispersion**

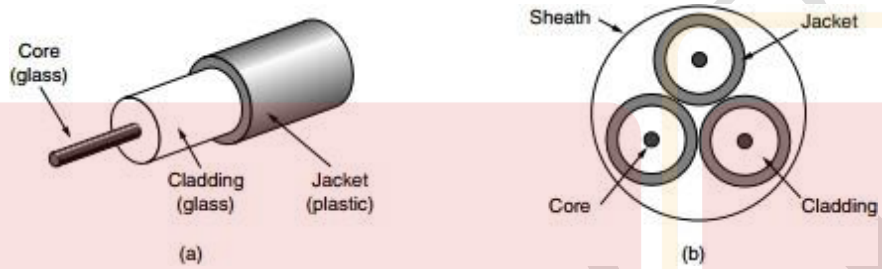
Light pulses sent down a fiber spread out in length as they propagate.

**Soliton:**

A soliton is a pulse that can collide with another similar pulse and still retain its shape after the collision, again in the presence of both dispersion and non-linearities.

### 1.4.1.6 Fiber Cables

- Fiber optic cables are similar to coax, except without the braid. 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.
- Fibers can be connected in three different ways. First, they can terminate in connectors and be plugged into fiber sockets. Connectors lose about 10 to 20% of the light, but they make it easy to reconfigure systems.
- Second, they can be spliced mechanically. Mechanical splices just lay the two carefully cut ends next to each other in a special sleeve and clamp them in place. Alignment can be improved by passing light through the junction and then making small adjustments to maximize the signal.
- Third, two pieces of fiber can be fused (melted) to form a solid connection. A fusion splice is almost as good as a single drawn fiber, but even here, a small amount of attenuation occurs. For all three kinds of splices, reflections can occur at the point of the splice, and the reflected energy can interfere with the signal.

| Item | LED | Semiconductor laser |
|---|---|---|
| Data rate | Low | High |
| Fiber type | Multi-mode | Multi-mode or single-mode |
| Distance | Short | Long |
| Lifetime | Long life | Short life |
| Temperature sensitivity | Minor | Substantial |
| Cost | Low cost | Expensive |

**Figure 2-9.** A comparison of semiconductor diodes and LEDs as light sources.

**Comparison of Fiber Optics and Copper Wire**

**Advantages of Fiber Optics:**

- It can handle higher bandwidths than copper.
- Fiber optic cables have low attenuation.
- Fiber optics cables are not affected by electromagnetic interferences and power fluctuations.
- Fiber optic cables are much more secured.
- Fiber cables are thin and light weight.
- The life cycle of fiber cables is 30 to 50 years, which is much higher than copper cables.

**Disadvantages of Fiber Optics:**

- It is a newer technology with not much expertise.
- Copper cables and connectors are much cheaper than fiber optic cables and connectors.
- Propagation of signals in fiber optic cables is unidirectional.

## 1.5 WIRELESS TRANSMISSION

Wireless communications has many important applications besides providing connectivity to users from any place.

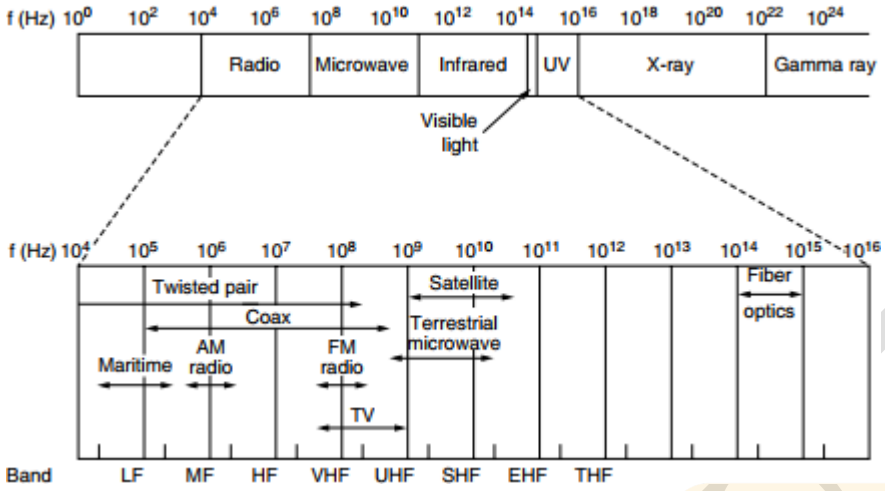
### 1.5.1 The Electromagnetic Spectrum

- When electrons move, they create electromagnetic waves that can propagate through space.
- The number of oscillations per second of a wave is called its **frequency**, *f*, and is measured in **Hz.** The distance between two consecutive maxima (or minima) is called the **wavelength,** represented by lambda.
- In a 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 × 108 m/sec, or about 1 foot (30 cm) per nanosecond.

The fundamental relation between $f$, $\lambda$, and $c$ (in a vacuum) is

$$\lambda f = c$$

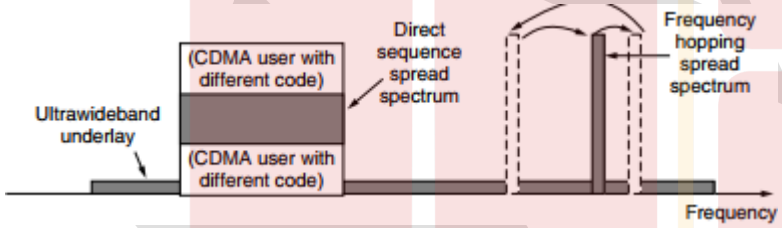
- The radio, microwave, infrared, and visible light portions of the spectrum can all be used for transmitting information by modulating the amplitude, frequency, or phase of the waves. Ultraviolet light, X-rays, and gamma rays would be even better, due to their higher frequencies, but they are hard to produce and modulate, do not propagate well. The terms LF, MF, and HF refer to Low, Medium, and High Frequency, respectively.




- Most transmissions use a relatively narrow frequency band (i.e., $\Delta f / f << 1$). In **frequency hopping spread spectrum**, the transmitter hops from frequency to frequency hundreds of times per second. It is pop ular for military communication because it makes transmissions hard to detect and next to impossible to jam. This technique is used commercially, for example, in Bluetooth and older versions of 802.11.

**Direct Sequence spread spectrum**

A second form of spread spectrum, **direct sequence spread spectrum**, uses a code sequence to spread the data signal over a wider frequency band. It is widely used commercially as a spectrally efficient way to let multiple signals share the same frequency band. These signals can be given different codes, a method called **CDMA** (**Code Division Multiple Access**)




It forms the basis of 3G mobile phone networks and is also used in GPS (Global Positioning System).

**UWB( UltraWideBand)**

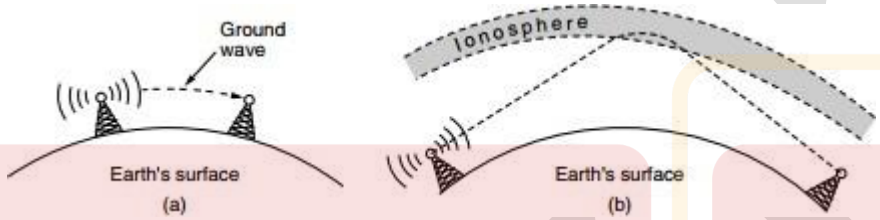
UWB sends a series of rapid pulses, varying their positions to communicate information. The rapid transitions lead to a signal that is spread thinly over a very wide frequency band. UWB is defined as signals that have a bandwidth of at least 500 MHz or at least 20% of the center frequency of their frequency band. It can tolerate a substantial amount of relatively strong interference from other narrowband signals, because it is spread across wide band of frequencies.

### 1.5.2 Radio Transmission

- Radio frequency (RF) 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 omni

directional, 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—at least as fast as $1/r^2$ in air—asthe signal energy is spread more thinly over a larger surface. This attenuation is called **path loss.**
- At high frequencies, radio waves tend to travel in straight lines and bounce off obstacles. Path loss still reduces power, though the received signal can depend strongly on reflections as well. High- frequency radio waves are also absorbed by rain and other obstacles to a larger extent than are low-frequency ones. At all frequencies, radio waves are subject to interference from motors and other electrical equipment.




From fig(a): In the VLF, LF, and MF bands, radio waves follow the ground. 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.

Fig(b) 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.

### 1.5.3 Microwave Transmission

- Microwaves travel in a straight line. Thus, repeaters are needed periodically. The distance between repeaters is square root of the tower height. For 100- meter-high towers, repeaters can be 80 km apart. Microwaves do not pass through buildings well.
- Some waves may be refracted off low- lying atmospheric layers and may take slightlylonger 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.**
- Bands up to 10 GHz are now in routine use. These waves are only a few centimetres long and are absorbed by rain. Microwave communication is so widely used for long-distance telephone communication, mobile phones, television distribution. It does not require to lay down cables. By buying a small plot of ground every 50 km and putting a microwave tower on it, one can bypass the telephone system entirely. Microwave is also relatively inexpensive.

**Politics of ElectroMagnetic Spectrum**

National governments allocate spectrum for AM and FM radio, television, and mobile phones, as well as for telephone companies, police, maritime, navigation, military, government, and many other competing users.

3 algorithms are widely used.

**Beauty Contest**

Oldest algorithm requires each carrier to explain why its proposal serves the public interest best. Government officials then decide which of the nice stories they enjoy most.

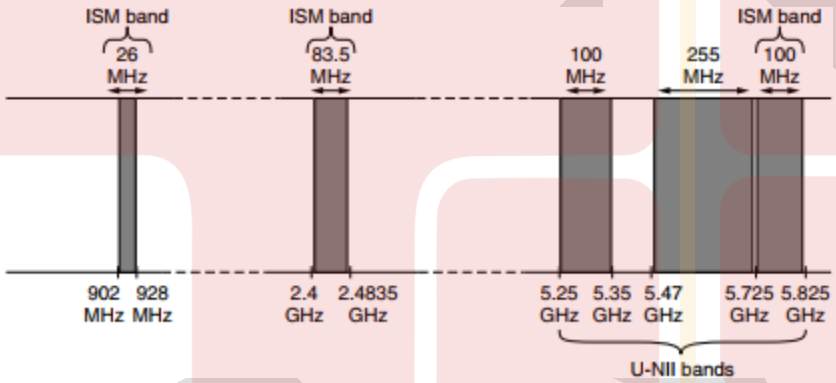
**Lottery**

Second algorithm which holds lottery among the interested companies. The problem with that idea is that companies with no interest in using the spectrum can enter the lottery.

**Auction**

Random companies has been severely criticized by many, which led to algorithm 3: **auction** off the bandwidth to the highest bidder.

A completely different approach to allocating frequencies is to not allocate them at all. Accordingly, most governments have set aside some frequency bands, called the **ISM** (**Industrial, Scientific, Medical**) bands for unlicensed usage. To minimize interference between these uncoordinated devices, the FCC mandates that all devices in the ISM bands limit their transmit power.




The 900-MHz band was used for early versions of 802.11, but it is crowded. The 2.4-GHz band is available in most countries and widely used for 802.11b/g and Bluetooth, though it is subject to interference from microwave ovens and radar installations. The 5-GHz part of the spectrum includes **U-NII** (**Unlicensed National Information Infrastructure**) bands.

One exciting development in the U.S. is the FCC decision in 2009 to allow unlicensed use of **white spaces** around 700 MHz. White spaces are frequency bands that have been allocated but are not being used locally. The only difficulty to use the white spaces, unlicensed devices must be able to detect any nearby licensed transmitters, including wireless microphones.

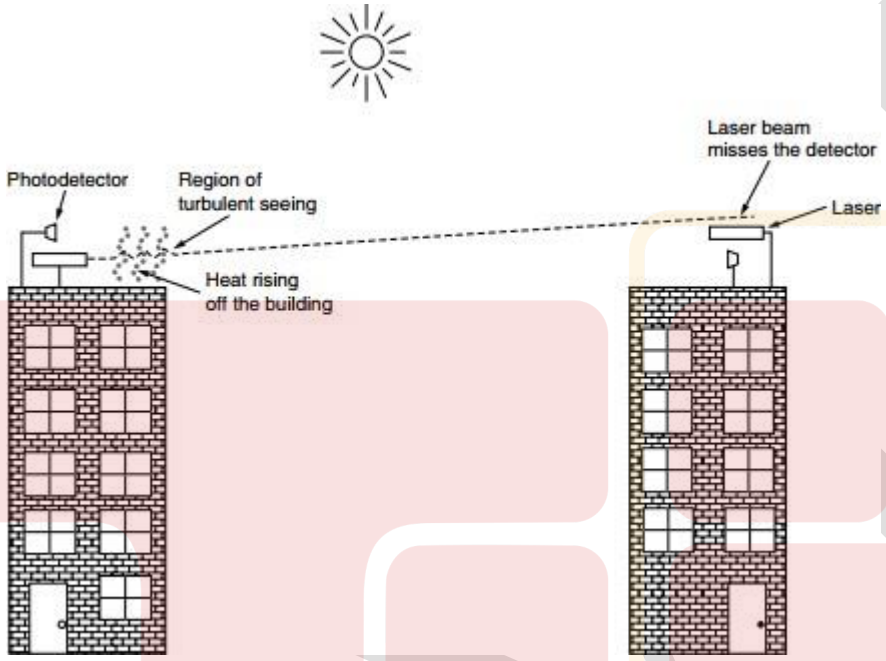
### 1.5.4 InfraRed Transmission

- Unguided infrared waves are widely used for short-range communication. The remote controls used for 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. It means that an infrared system in one room of a building will not interfere with a similar system in adjacent rooms or buildings.

- Infrared communication has a limited use on the desktop, for example, to connect notebook computers and printers with the **IrDA** (**Infrared Data Association**) standard, but it is not a major player in the communication game.

### 1.5.5 Light Transmission

A more modern application is to connect the LANs in two buildings via lasers mounted on their rooftops. Optical signaling using lasers is inherently unidirectional, so each end needs its own laser and its own photo detector.




To avoid unnecessary wiring for only limited days, the organizers placed the laser on the roof, and tested in the night which worked perfectly. At 9 A.M. on a bright, sunny day, the link failed completely and stayed down all day. The problem is that because heat from the sun during the daytime caused convection currents to rise up from the roof of the building. This turbulent air diverted the beam and made it dance around the detector, much like a shimmering road on a hot day. Communicating with visible light in this way is inherently safe and creates a low-speed network in the immediate vicinity of the display.

# DATA LINK LAYER DESIGN ISSUES

The data link layer uses the services of the physical layer to send and receive bits over communication channels. It has a number of functions, including:

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 swampedby 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 Fig. 3-1. Frame management forms the heart of what the data link layer does.

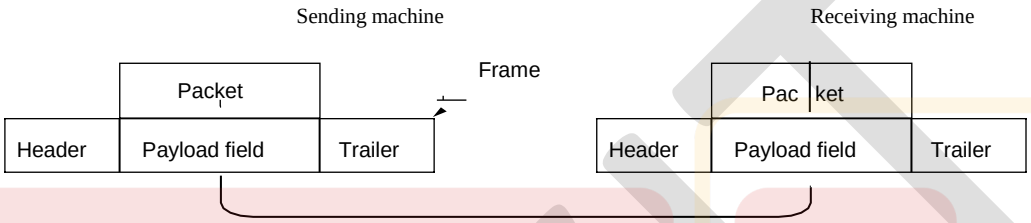



**Figure 3-1.** Relationship between packets and frames

*In fact, in many networks, these functions are found mostly in the upper layers, with the data link layer doing the minimal job that is ‘‘good enough.’’ However, no matter where they are found, the principles are pretty much the same.

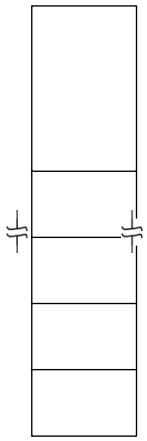
*.They often show up in their simplest and purest forms in the data link layer, making this a good place to examine them in detail.

## ❖ 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 ma- chine 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 net-work layer there, as shown in Fig. 3-2(a).

The data link layer can be designed to offer various services. The actual services that are offered vary from protocol to protocol. Three reasonable possibilities that we will consider in turn are:

1. Unacknowledged connectionless service.
2. Acknowledged connectionless service.
3. Acknowledged connection-oriented service.



## ❖ Framing

- Breaking up the bit stream into frames is more difficult than it at first appears. A good design must make it easy for a receiver to find the start of new frames while using little of the channel bandwidth. We will look at four methods:

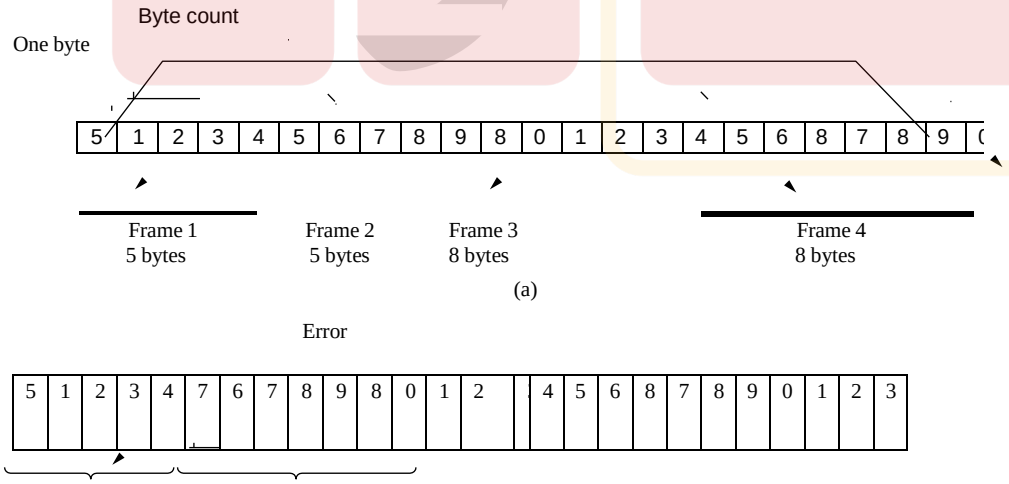
(a) Byte count.

(b) Flag bytes with byte stuffing.

(c) Flag bits with bit stuffing.

(d) Physical layer coding violations.

- The usual approach is for the data link layer to break up the bit stream into discrete frames, compute a short token called a checksum for each frame, and in- clude the checksum in the frame when it is transmitted. (Checksum algorithms will be discussed later in this chapter.)
- 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 oc- curred and takes steps to deal with it (e.g., discarding the bad frame and possibly also sending back an error report).



Frame 1 Frame 2 Now a byte count

(Wrong)

(b)

- The first framing method uses a field in the header to specify the number of bytes in the frame. When the data link layer at the destination sees the byte count, it knows how many bytes follow and hence where the end of the frame is. This technique is shown in Fig. 3-3(a)

- The trouble with this algorithm is that the count can be garbled by a transmis- sion error. For example, if the byte count of 5 in the second frame of Fig. 3-3(b) becomes a 7 due to a single bit flip, the destination will get out of synchroniza- tion. It will then be unable to locate the correct start of the next frame.

- The second framing method gets around the problem of resynchronization after an error by having each frame start and end with special bytes. Often the same byte, called a flag byte, is used as both the starting and ending delimiter. This byte is shown in Fig. 3-4(a) as FLAG

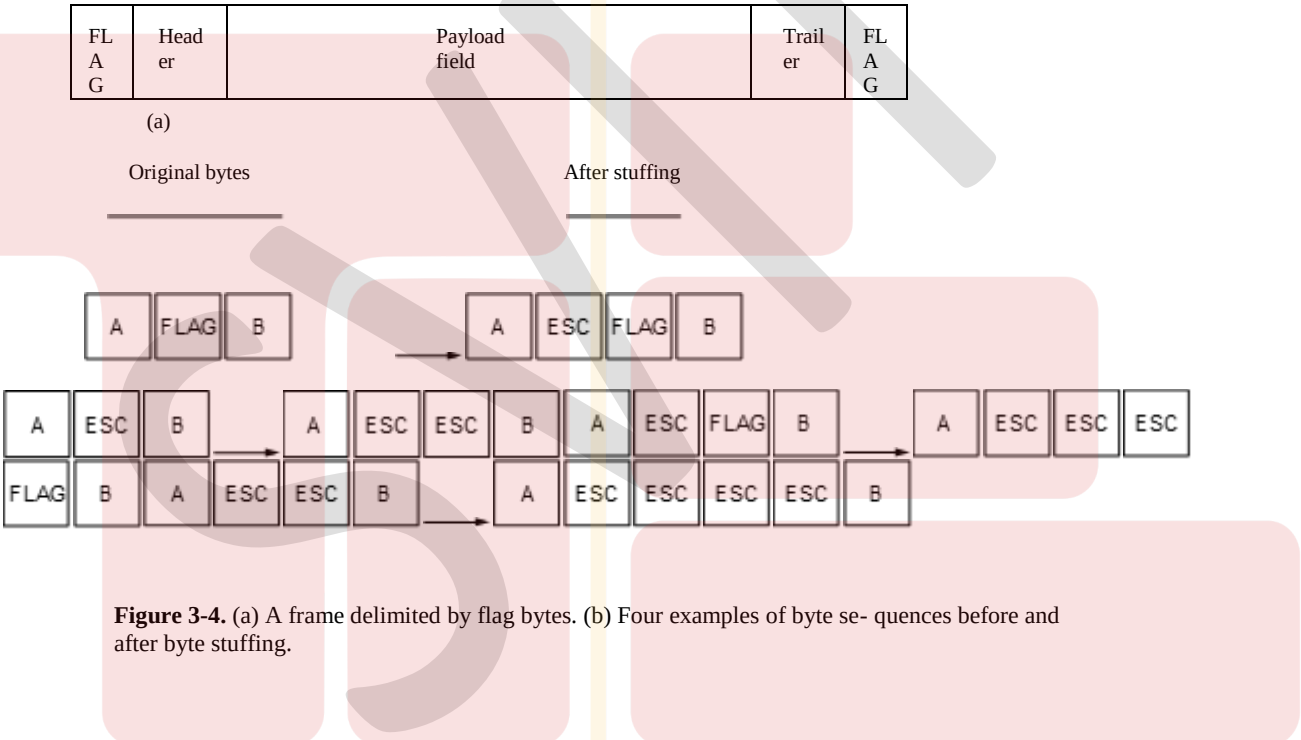



**Figure 3-4.** (a) A frame delimited by flag bytes. (b) Four examples of byte se- quences before and after byte stuffing.

- The data link layer on the receiving end re- moves the escape bytes before giving the data to the network layer. This techni- que is called **byte stuffing**.

- The byte-stuffing scheme depicted in Fig. 3-4 is a slight simplification of the one used in **PPP** (**Point-to-Point Protocol**), which is used to carry packets over communications links

Framing can be also be done at the bit level, so frames can contain an arbitrary number of bits made up of units of any size. It was developed for the once very popular **HDLC** (**High- level Data Link Control**) protocol.

- This **bit stuffing** is anal- ogous to byte stuffing, in which an escape byte is stuffed into the

outgoing charac- ter 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 3-5 gives an example of bit stuffing.

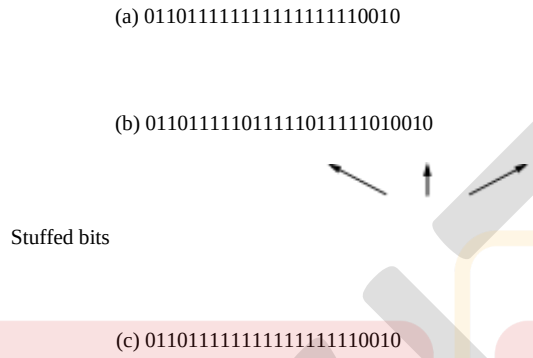



**Figure 3-5.** Bit stuffing. (a) The original data. (b) The data as they appear on the line. (c) The data as they are stored in the receiver's memory after destuf- fing.

- Many data link protocols use a combination of these methods for safety. A common pattern used for Ethernet and 802.11 is to have a frame begin with a well-defined pattern called a **preamble**.

## ❖ Error Control

- 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 pro- tocol calls for the receiver to send back special control frames bearing positive or negative acknowledgements about the incoming frames

- If the sender receives a positive acknowledgement about a frame, it knows the frame has arrived safely, On the other hand, a negative acknowledgement 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. Similarly, if the ac- knowledgement frame is lost, the sender will not know how to proceed

- It should be clear that a protocol in which the sender transmits a frame and then waits for an acknowledgement, positive or negative, will hang forever if a frame is ever lost due to, for example, malfunctioning hardware or a faulty communication channel

- 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, However, if either the frame or the acknowledgement 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 hap- pening, it is generally necessary to assign sequence numbers to outgoing frames, so that the receiver can distinguish retransmissions from originals.

## ❖ Flow Control

- The 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, is situation can occur when the sender is running on a fast, powerful computer and the receiver is running on a slow, low-end machine
- A common situation is when a smart phone requests a Web page from a far more powerful server, which then turns on the fire hose and blasts the data at the poor helpless phone until it is completely swamped. Even if the transmission is error free, the receiver may be unable to handle the frames as fast as they arrive and will 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.
- For example, hardware implementations of the link layer as **NICs** (**Network Interface Cards**) are some- times said to run at ‘‘wire speed,’’ meaning that they can handle frames as fast as they can arrive on the link. Any overruns are then not a link problem, so they are handled by higher layers.

## ❖ Error Detection and Correction

- The optical fiber in telecommunications networks, have tiny error rates so that transmission errors are a rare occurrence. But other channels, especially wireless links and aging local loops, have error rates that are orders of magnitude larger. For these links, transmission errors are the norm. They cannot be avoided at a reasonable expense or cost in terms of performance. The conclusion is that transmission errors are here to stay
- One strategy is to include enough redundant information to enable the receiver to deduce what the transmitted data must have been. The other is to include only enough redundancy to allow the receiver to deduce that an error has 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 **FEC** (**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, FEC is used on noisy channels because retransmissions are just as likely to be in error as the first transmission.
- A key consideration for these codes is the type of errors that are likely to oc- cur. Neither error-correcting codes nor error-detecting codes can handle all pos- sible errors since the

redundant bits that offer protection are as likely to be re- ceived in error as the data bits (which can compromise their protection).

- Other types of errors also exist. Sometimes, the location of an error will be known, perhaps because the physical layer received an analog signal that was far from the expected value for a 0 or 1 and declared the bit to be lost. This situation is called an **erasure channel**
- It is easier to correct errors in erasure channels than in channels that flip bits because even if the value of the bit has been lost, at least we know which bit is in error, Error-correcting codes are also seen in the physical layer, particularly for noisy channels, and in higher layers, particularly for eal-time media and content distribution. Error-detecting codes are commonly used in link, network, and transport layers.

## ❖ Error-Correcting Codes

➢ We will examine four different error-correcting codes:

1. Hamming codes.
2. Binary convolutional codes.
3. Reed-Solomon codes.
4. Low-Density Parity Check codes.

- All of these codes add redundancy to the information that is sent. A frame con- sists of *m* data (i.e., message) bits and *r* redundant (i.e. check) bits. In a **block code**, the *r* check bits are computed solely as a function of the *m* data bits with which they are associated, as though the *m* bits were looked up in a large table to find their corresponding *r* check bits.
- In a **systematic code**, the *m* data bits are sent directly, along with the check bits, rather than being encoded themselves before they are sent.
- In a **linear code**, the *r* check bits are computed as a linear function of the *m* data bits. Exclusive OR (XOR) or modulo 2 addition is a popu- lar choice. This means that encoding can be done with operations such as matrix multiplications or simple logic circuits.
- Let the total length of a block be *n* (i.e., *n* □ *m* □ *r*). We will describe this as an (*n*,*m* ) code. An *n*-bit unit containing data and check bits is referred to as an *n*- bit **codeword**
- The **code rate**, or simply rate, is the fraction of the codeword that carries information that is not redundant, or *m/n*. The rates used in practice vary widely.
- To understand how errors can be handled, it is necessary to first look closely at what an error really is. Given any two codewords that may be transmitted or received—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 XOR the two codewords and count the number of 1 bits in the result.

→ For example:

10001001

10110001

00111000

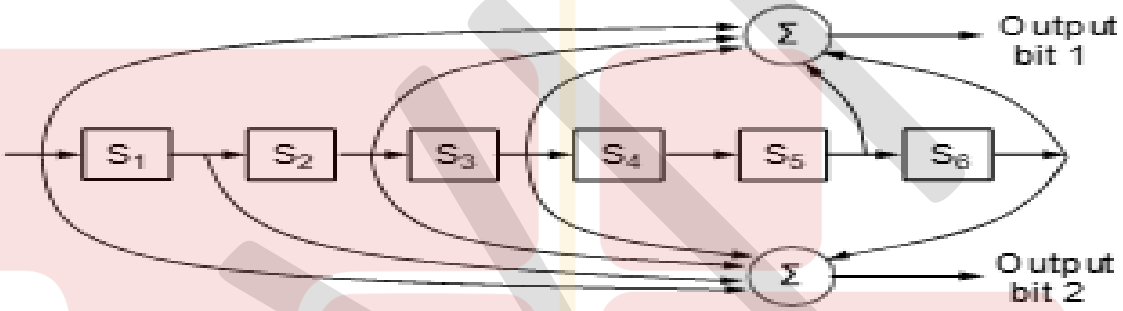
- The number of bit positions in which two codewords differ is called **Hamming distance** (Hamming, 1950). Its significance is that if two codewords are a Hamming distance *d* apart, it will require *d* single-bit errors to convert one into the other.
- In Hamming codes the bits of the codeword are 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. This pattern is shown for an (11,7) Hamming code with 7 data bits and 4 check bits in Fig. 3-6

- If the check results are not all zero, however, an error has been detected. The set of check results forms the error syndrome that is used to pinpoint and correct the error. In Fig. 3-6, a single-bit error occurred on the channel so the check re- sults are 0, 1, 0, and 1 for k = 8, 4, 2, and 1, respectively.

- Hamming distances are valuable for understanding block codes, and Ham- ming codes are used in error-correcting memory, However, most networks use stronger codes. The second code we will look at is a **convolutional code**. This code is the only one we will cover that is not a block code

- In a convolutional code, an encoder processes a sequence of input bits and generates a sequence of output bits. There is no natural message size or encoding boundary as in a block code. The output depends on the current and previous input bits. That is, the encoder has memory. The number of previous bits on which the output depends is called the **constraint length** of the code




Convolutional codes are widely used in deployed networks, for example, as part of the GSM mobile phone system, in satellite communications, and in 802.11. This code is known as the NASA convolutional code of $r = 1/2$ and $k = 7$, since it was first used for the Voyager space missions starting in 1977. Since then it has been liberally reused, for example, as part of 802.11.

In Fig. above, each input bit on the left-hand side produces two output bits on the right-hand side that are XOR sums of the input and internal state. Since it deals with bits and performs linear operations, this is a binary, linear convolutional code. Since 1 input bit produces 2 output bits, the code rate is 1/2. It is not sys- tematic since none of the output bits is simply the input bit.

- Extensions of the Viterbi algorithm can work with these uncertainties to provide stronger error correction. This approach of working with the uncertainty of a bit is called **soft-decision decoding**. Conversely, decid- ing whether each bit is a 0 or a 1 before subsequent error correction is called **hard-decision decoding**.
- The third kind of error-correcting code we will describe is the **Reed-Solomon code**. Like Hamming codes, Reed-Solomon codes are linear block codes, and they are often systematic too.
- Reed-Solomon codes are widely used in practice because of their strong error-correction properties, particularly for burst errors. They are used for DSL, data over cable, satellite communications, and perhaps most ubiquitously on CDs, DVDs, and Blu-ray discs.
- Reed-Solomon codes are often used in combination with other codes such as a convolutional code. The thinking is as follows. Convolutional codes are effective at handling isolated bit errors, but they will fail, likely with a burst of errors, if there are too many errors in the received bit stream. the Reed-Solomon decoding can mop up the error bursts, a task at which it

is very good.

- The final error-correcting code we will cover is the **LDPC** (**Low-Density Parity Check**) code. LDPC codes are linear block codes that were invented by Robert Gallagher in his doctoral thesis (Gallagher, 1962), they were promptly forgotten, only to be reinvented in 1995 when advances in comput- ing power had made them practical.

- n an LDPC code, each output bit is formed from only a fraction of the input bits. This leads to a matrix representation of the code that has a low density of 1s, hence the name for the code. The received codewords are decoded with an approximation algorithm that iteratively improves on a best fit of the received data to a legal codeword. This corrects errors.

- LDPC codes are practical for large block sizes and have excellent error-cor- rection abilities that outperform many other codes (including the ones we have looked at) in practice.

## ❖ Error-Detecting Codes

- Error-correcting codes are widely used on wireless links, which are notori- ously noisy and error prone when compared to optical fibers. However, over fiber or high-quality copper, the error rate is much lower, so error detection and retrans- mission is usually more efficient there for dealing with the occasional error.

We will examine three different error-detecting codes. They are all linear, systematic block codes:

1. Parity.
2. Checksums.
3. Cyclic Redundancy Checks (CRCs).

- Consider the first error-detecting 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).

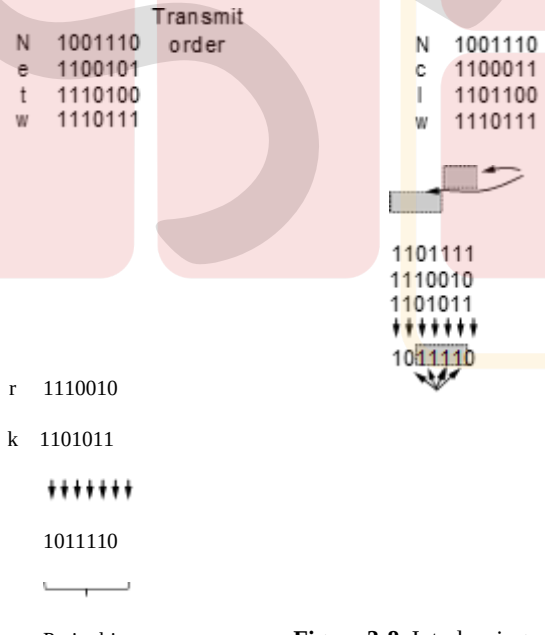



**Figure 3-8.** Interleaving of parity bits to detect a burst error

- there is something else we can do that provides better protection against burst errors: we can compute the parity bits over the data in a different order than the order in which the data bits are transmitted. Doing so is called **interleaving**.
- Interleaving is a general technique to convert a code that detects (or corrects) isolated errors into a code that detects (or corrects) burst errors. In Fig. 3-8, when a burst error of length $n$ □ 7 occurs, the bits that are in error are spread across different columns.
- The second kind of error-detecting code, the **checksum**, is closely related to groups of parity bits. The word ‘‘checksum’’ is often used to mean a group of check bits associated with a message, regardless of how are calculated
- A group of parity bits is one example of a checksum. However, there are other, stronger checksums based on a running sum of the data bits of the message. The checksum is usually placed at the end of the message, as the complement of the sum func- tion. This way, errors may be detected by summing the entire received codeword, both data bits and checksum
- A better choice is **Fletcher’s checksum** (Fletcher, 1982). It includes a posi- tional component, adding the product of the data and its position to the running sum. This provides stronger detection of changes in the position of data.
- Although the two preceding schemes may sometimes be adequate at higher layers, in practice, a third and stronger kind of error-detecting code is in wide- spread use at the link layer: the **CRC** (**Cyclic Redundancy Check**), also known as a **polynomial code**.
- Polynomial arithmetic is done modulo 2, according to the rules of algebraic field theory. It does not have carries for addition or borrows for subtraction. Both addition and subtraction are identical to exclusive OR.

| | | | |
|---|---|---|---|
| 10011011 + 11001010 | 00110011 + 11001101 | 11110000 – 10100110 | 01010101 – 10101111 |
| 01010001 | 11111110 | 01010110 | 11111010 |

Long division is carried out in exactly the same way as it is in binary except that the subtraction is again done modulo 2. A divisor is said ‘‘to go into’’ a dividend if the dividend has as many bits as the divisor.

- ♦ When the polynomial code method is employed, the sender and receiver must agree upon a **generator polynomial**, $G(x)$, in advance.
- ➢ The algorithm for computing the CRC is as follows:

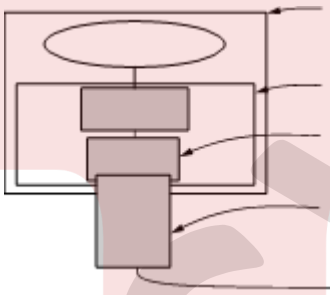
1. Let $r$ be the degree of $G(x)$. Append $r$ zero bits to the low-order end of the frame so it now contains $m$ □ $r$ bits and corresponds to the polynomial $x^r M(x)$.
2. Divide the bit string corresponding to $G(x)$ into the bit string corres- ponding to $x^r M(x)$, using modulo 2 division.
3. Subtract the remainder (which is always $r$ or fewer bits) from the bit string corresponding to $x^r M(x)$ using modulo 2 subtraction. The re- sult is the checksummed frame to be transmitted. Call its polynomial $T(x)$.

- Both the high- and low- order bits of the generator must be 1. To compute the CRC for some frame with $m$ bits corresponding to the polynomial $M(x)$, the frame must be longer than the generator polynomial.
- The idea is to append a CRC to the end of the frame in such a way that the polynomial

represented by the checksummed frame is divisi- ble by $G(x)$. When the receiver gets the checksummed frame, it tries dividing it by $G(x)$. If there is a remainder, there has been a transmission error.

# ❖ ELEMENTARY DATA LINK PROTOCOLS

- To start with, we assume that the physical layer, data link layer, and network layer are independent processes that communicate by passing messages back and forth. A common implementation is shown in Fig. 3-10
- The physical layer proc- ess and some of the data link layer process run on dedicate hardware called a **NIC** (**Network Interface Card**), three processes offloaded to ded- icated hardware called a **network accelerator**
- The rest of the link layer process and the network layer process run on the main CPU as part of the operating system, with the soft- ware for the link layer process often taking the form of a **device driver**



- 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 consid- er 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 al- ways able to comply immediately.
- As far as the data link layer is concerned, the packet passed across the inter- face 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.

```
#define MAX PKT 1024                    /* determines packet size in bytes */

typedef enum {false, true} boolean;     /* boolean type */

typedef unsigned int seq nr;            /* sequence or ack numbers */ typedef
struct {unsigned char data[MAX PKT];} packet;   /* packet definition */

typedef enum {data, ack, nak} frame kind;   /* frame kind definition */

typedef struct {                        /* frames are transported in this layer */

  frame kind kind;                      /* what kind of frame is it? */

  seq nr seq;                           /* sequence number */
```

```
  – seq nr ack;                    /* acknowledgement number */
  packet info;                     /* the network layer packet */
} frame;


––– /* Wait for an event to happen; return its type in event. */ void wait
for event(event type *event);

–– /* Fetch a packet from the network layer for transmission on the channel. */ void from
network layer(packet *p);

–– /* Deliver information from an inbound frame to the network layer. */ void to
network layer(packet *p);

–– /* Go get an inbound frame from the physical layer and copy it to r. */ void from
physical layer(frame *r);

–– /* Pass the frame to the physical layer for transmission. */ void to
physical layer(frame *s);

–– /* Start the clock running and enable the timeout event. */ void start
timer(seq nr k);

–– /* Stop the clock and disable the timeout event. */ void stop
timer(seq nr k);

––– /* Start an auxiliary timer and enable the ack timeout event. */ void start
ack timer(void);

––– /* Stop the auxiliary timer and disable the ack timeout event. */ void stop
ack timer(void);

–––– /* Allow the network layer to cause a network layer ready event. */ void
enable network layer(void);

–––– /* Forbid the network layer from causing a network layer ready event. */ void
disable network layer(void);

– /* Macro inc is expanded in-line: increment k circularly. */ #define inc(k)
if (k < MAX SEQ) k=k+ 1; else k = 0
```

**Figure 3-11.** Some definitions needed in the protocols to follow. These defini- tions are located in the file *protocol.h*.

- This procedure only returns when something has hap- pened (e.g., a frame has arrived). Upon return, the variable *event* tells what hap- pened. The set of possible events differs for the various protocols to be described and will be defined separately for each protocol.
- When a frame arrives at the receiver, the checksum is recomputed. If the checksum in the frame 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*.
- A *frame* is composed of four fields: *kind*, *seq*, *ack*, and *info*, the first three of which contain control information and the last of which may contain actual data to be transferred. These control fields are collectively called the **frame header**.

# ❖ A Utopian Simplex Protocol

- Data are transmitted in one direction only. Both the transmitting and receiving network layers are always ready. Processing time can be ignored. Infinite buffer space is available. And best of all, the communication channel between the data link lay- ers never damages or loses frames

- The protocol consists of two distinct procedures, a sender and a receiver. The sender runs in the data link layer of the source machine, and the receiver runs in the data link layer of the destination machine. No sequence numbers or acknowl- edgements are used here, so *MAX SEQ* is not needed.

- The sender is in an infinite while loop just pumping data out onto the line as fast as it can. The body of the loop consists of three actions: go fetch a packet from the (always obliging) network layer, construct an outbound frame using the variable *s*, and send the frame on its way.

- The receiver is equally simple. Initially, it waits for something to happen, the only possibility being the arrival of an undamaged frame. Eventually, the frame arrives and the procedure *wait for event* returns, with *event* set to *frame arrival* (which is ignored anyway)

```
* Protocol 1 (Utopia) provides for data transmission in one direction only, from sender to
  receiver. The communication channel is assumed to be error free and the receiver is assumed to
  be able to process all the input infinitely quickly. Consequently, the sender just sits in a loop
  pumping data out onto the line as fast as it can. */

typedef enum {frame arrival} event type;

 #include "protocol.h"

void sender1(void)

{

  frame s;                              /* buffer for an outbound frame */

  packet buffer;                        /* buffer for an outbound packet */

  while (true) {

      from network layer(&buffer);      /* go get something to send */ s.info = buffer;
                                        /* copy it into s for transmission */

      to physical layer(&s);            /* send it on its way */

  }                                     /* Tomorrow, and tomorrow, and tomorrow, Creeps in
                                           this petty pace from day to day

                                           To the last syllable of recorded time.

            – Macbeth, V, v */

}


 void receiver1(void)
```

```
{
  frame r;
  event type event;                 /* filled in by wait, but not used here */
  while (true) {
      wait for event(&event);       /* only possibility is frame arrival */ from
      physical layer(&r);           /* go get the inbound frame */
      to network layer(&r.info);    /* pass the data to the network layer */
  }
}
```

**Figure 3-12.** A utopian simplex protocol.

The utopia protocol is unrealistic because it does not handle either flow con- trol or error correction. Its processing is close to that of an unacknowledged con- nectionless service that relies on higher layers to solve these problems, though even an unacknowledged connectionless service would do some error detection.

### ❖ A Simplex Stop-and-Wait Protocol for an Error-Free Channel

- Now we will tackle the problem of preventing the sender from flooding the receiver with frames faster than the latter is able to process them. This situation can easily happen in practice so being able to prevent it is of great importance.
- One solution is to build the receiver to be powerful enough to process a con- tinuous stream of back-to-back frames, It must have sufficient buffering and processing abilities to run at the line rate and must be able to pass the frames that are received to the network layer quickly enough.
- A more general solution to this problem is to have the receiver provide feed- back to the sender. After having passed a packet to its network layer, the receiver sends a little dummy frame back to the sender which, in effect, gives the sender permission to transmit the next frame.
- Protocols in which the sender sends one frame and then waits for an acknowl- edgement before proceeding are called **stop-and-wait.**
- As in protocol 1, the sender starts out by fetching a packet from the network layer, using it to construct a frame, and sending it on its way. The only difference between *receiver1* and *receiver2* is that after delivering a packet to the network layer, *receiver2* sends an acknowledgement frame back to the sender before entering the wait loop again.

## ❖ Simplex Stop-and-Wait Protocol for a Noisy Channel

- The normal situation of a communication channel that makes errors. Frames may be either damaged or lost completely. However, we assume that if a frame is damaged in transit, the receiver hardware will detect this.
- when it computes the checksum. If the frame is damaged in such a way that the checksum is nevertheless correct—an unlikely occurrence—this protocol (and all other protocols) can fail (i.e., deliver an incorrect packet to the network layer).

- At first glance it might seem that a variation of protocol 2 would work: adding a timer. The sender could send a frame, but the receiver would only send an ac- knowledgement frame if the data were correctly received.

- If a damaged frame ar- rived at the receiver, it would be discarded. After a while the sender would time out and send the frame again. This process would be repeated until the frame finally arrived intact.

```
/* Protocol 2 (Stop-and-wait) also provides for a one-directional flow of data from sender to receiver.
   The communication channel is once again assumed to be error

   free, as in protocol 1. However, this time the receiver has only a finite buffer capacity and a finite
   processing speed, so the protocol must explicitly prevent the sender from flooding the receiver
   with data faster than it can be handled. */

--typedef enum {frame arrival} event type; #include
"protocol.h"

void sender2(void)

{
frame s; /* buffer for an outbound frame */
  packet buffer;                    /* buffer for an outbound packet */
  --event type event;               /* frame arrival is the only possibility */
  while (true) {
      --from network layer(&buffer); /* go get something to send */ s.info = buffer;
                                     /* copy it into s for transmission */
      --to physical layer(&s);       /* bye-bye little frame */
      --wait for event(&event);      /* do not proceed until given the go ahead */
  }
}
void receiver2(void)
{
  frame r, s;                       /* buffers for frames */
  --event type event;               /* frame arrival is the only possibility */ while (true)
  {
      -----wait for event(&event);   /* only possibility is frame arrival */ from
      physical layer(&r);           /* go get the inbound frame */
      --to network layer(&r.info);   /* pass the data to the network layer */
      --to physical layer(&s);       /* send a dummy frame to awaken sender */
  }
}
```

**Figure 3-13.** A simplex stop-and-wait protocol.

- The network layer on machine *A* gives a series of packets to its data link layer, which must ensure that an identical series of packets is delivered to the net- work layer on machine *B* by its data link layer.
- In particular, the network layer on *B* has no way of knowing that a packet has been lost or duplicated, so the data link layer must guarantee that no combination of transmission errors, however unlike- ly, can cause a duplicate packet to be delivered to a network layer.

➢ Consider the following scenario:

1. The network layer on *A* gives packet 1 to its data link layer. The packet is correctly received at *B* and passed to the network layer on *B*. *B* sends an acknowledgement frame back to *A*.
2. The acknowledgement frame gets lost completely. It just never ar- rives at all. Life would be a great deal simpler if the channel man- gled and lost only data frames and not control frames, but sad to say, the channel is not very discriminating.
3. The data link layer on *A* eventually times out. Not having received an acknowledgement, it (incorrectly) assumes that its data frame was lost or damaged and sends the frame containing packet 1 again.
4. The duplicate frame also arrives intact at the data link layer on *B* and is unwittingly passed to the network layer there. If *A* is sending a file to *B*, part of the file will be duplicated (i.e., the copy of the file made by *B* will be incorrect and the error will not have been detected). In other words, the protocol will fail.

- An example of this kind of protocol is shown in Fig. 3-14. Protocols in which the sender waits for a positive acknowledgement before advancing to the next data item are often called **ARQ** (**Automatic Repeat reQuest**) or **PAR** (**Positive Acknowledgement with Retransmission**). Like protocol 2, this one also trans- mits data only in one direction.
- protocol 3 differs from its predecessors in that both sender and receiver have a variable whose value is remembered while the data link layer is in the wait state. The sender remembers the sequence number of the next frame to send in *next frame to send*; the receiver remembers the sequence number of the next frame expected in *frame expected*.
- After transmitting a frame and starting the timer, the sender waits for some- thing exciting to happen. Only three possibilities exist: an acknowledgement frame arrives undamaged, a damaged acknowledgement frame staggers in, or the timer expires. If a valid acknowledgement comes in, the sender fetches the next packet from its network layer and puts it in the buffer, overwriting the previous packet.
- When a valid frame arrives at the receiver, its sequence number is checked to see if it is a duplicate. If not, it is accepted, passed to the network layer, and an acknowledgement is generated.

## ❖ SLIDING WINDOW PROTOCOLS

- The previous protocols, each using a separate link for simplex data traffic (in different directions). Each link is then comprised of a ‘‘forward’’ channel (for data) and a ‘‘reverse’’ channel (for acknowledgement data frames were transmitted in one direction only. In most practical situations, there is a need to transmit data in both directions.

- A better idea is to use the same link for data in both directions. After all, in protocols 2 and 3 it was already being used to transmit frames both ways, and the reverse channel normally has the same capacity as the forward channel.

- Although interleaving data and control frames on the same link is a big im- provement over having two separate physical links, yet another improvement is possible. When a data frame arrives, instead of immediately sending a separate control frame, the receiver restrains itself and waits until the network layer passes it the next packet.

- The acknowledgement is attached to the outgoing data frame (using the *ack* field in the frame header). In effect, the acknowledgement gets a free ride on the next outgoing data frame. The technique of temporarily delaying outgoing acknowledgements so that they can be hooked onto the next outgoing data frame is known as **piggybacking**.

- The *ack* field in the frame header costs only a few bits, whereas a separate frame would need a header, the acknowledgement, and a checksum.

- The next three protocols are bidirectional protocols that belong to a class called **sliding window** protocols. The three differ among themselves in terms of ef- ficiency, complexity, and buffer requirements, as discussed later. In these, as in all sliding window protocols, each outbound frame contains a sequence number, ranging from 0 up to some maximum.

- The essence of all sliding window protocols is that at any instant of time, the sender maintains a set of sequence numbers corresponding to frames it is permit- ted to send. These frames are said to fall within the **sending window**. Similarly, the receiver also maintains a **receiving window** corresponding to the set of frames it is permitted to accept. The sender's window and the receiver's window need not have the same lower and upper limits or even have the same size.

```
/* Protocol 3 (PAR) allows unidirectional data flow over an unreliable channel. */

----#define MAX SEQ 1                    /* must be 1 for protocol 3 */ typedef
enum {frame arrival, cksum err, timeout} event type;

#include "protocol.h"

void sender3(void)

{

 ----seq nr next frame to send;          /* seq number of next outgoing frame */ frame s;
                                         /* scratch variable */

 -packet buffer;                         /* buffer for an outbound packet */ event type
 event;

 ---next frame to send = 0;              /* initialize outbound sequence numbers */

 --from network layer(&buffer);          /* fetch first packet */ while
 (true) {

     -----s.info = buffer;               /* construct a frame for transmission */ s.seq = next
     frame to send;                      /* insert sequence number in frame */ to physical
     layer(&s);                          /* send it on its way */

     -start timer(s.seq);                /* if answer takes too long, time out */

     -----wait for event(&event);        /* frame arrival, cksum err, timeout */ if (event ==
     frame arrival) {

         -----from physical layer(&s);   /* get the acknowledgement */ if (s.ack ==
         next frame to send) {
```

```
            stop timer(s.ack);              /* turn the timer off */
            from network layer(&buffer); /* get the next one to send */
            inc(next frame to send);        /* invert next frame to send */
        }
    }
}
}

void receiver3(void)
{
  seq nr frame expected; frame r, s;
  event type event;
  frame expected = 0; while (true) {
    wait for event(&event);         /* possibilities: frame arrival, cksum err */ if (event ==
    frame arrival) {                /* a valid frame has arrived */
      from physical layer(&r);      /* go get the newly arrived frame */
      if (r.seq == frame expected) {  /* this is what we have been waiting for */ to network
        layer(&r.info);             /* pass the data to the network layer */ inc(frame
        expected);                  /* next time expect the other sequence nr */
      }
      s.ack = 1  frame expected;    /* tell which frame is being acked */ to physical
      layer(&s);                    /* send acknowledgement */
    }
  }
}
```

**Figure 3-14.** A positive acknowledgement with retransmission protocol.

## ❖ A One-Bit Sliding Window Protocol

- Before tackling the general case, let us examine a sliding window protocol with a window size of 1. Such a protocol uses stop-and-wait since the sender transmits a frame and waits for its acknowledgement before sending the next one.

- Figure 3-16 depicts such a protocol. Like the others, it starts out by defining some variables. *Next frame to send* tells which frame the sender is trying to send. Similarly, *frame expected* tells which frame the receiver is expecting. In both cases, 0 and 1 are the only possibilities.

```
/* Protocol 4 (Sliding window) is bidirectional. */

#define MAX SEQ 1                          /* must be 1 for protocol 4 */ typedef
enum {frame arrival, cksum err, timeout} event type;

#include "protocol.h"

void protocol4 (void)

{

    seq nr next frame to send;             /* 0 or 1 only */

    seq nr frame expected;                 /* 0 or 1 only */

    frame r, s;                            /* scratch variables */

    packet buffer;                         /* current packet being sent */ event type
    event;

    next frame to send = 0;                /* next frame on the outbound stream */

    frame expected = 0;                    /* frame expected next */

    from network layer(&buffer);           /* fetch a packet from the network layer */

    s.info = buffer;                       /* prepare to send the initial frame */

    s.seq = next frame to send;            /* insert sequence number into frame */

    s.ack = 1 □ frame expected;            /* piggybacked ack */

    to physical layer(&s);                 /* transmit the frame */

    start timer(s.seq);                    /* start the timer running */

    while (true) {

        wait for event(&event);            /* frame arrival, cksum err, or timeout */

        if (event == frame arrival) {      /* a frame has arrived undamaged */ from
            physical layer(&r);            /* go get it */

            if (r.seq == frame expected) { /* handle inbound frame stream */ to network
                layer(&r.info);            /* pass packet to network layer */

                inc(frame expected);       /* invert seq number expected next */

            }

            if (r.ack == next frame to send) { /* handle outbound frame stream */ stop
                timer(r.ack);              /* turn the timer off */

                from network layer(&buffer);/* fetch new pkt from network layer */ inc(next
                frame to send);            /* invert sender's sequence number */

            }

        }
```

```
        s.info = buffer;                    /* construct outbound frame */
        s.seq = next frame to send;         /* insert sequence number into it */
        s.ack = 1 - frame expected;         /* seq number of last received frame */
        to physical layer(&s);              /* transmit a frame */
        start timer(s.seq);                 /* start the timer running */
    }
}
```

**Figure 3-16.** A 1-bit sliding window protocol.

- Under normal circumstances, one of the two data link layers goes first and transmits the first frame. In other words, only one of the data link layer programs should contain the *to physical layer* and *start timer* procedure calls outside the main loop. The starting machine fetches the first packet from its network layer, builds a frame from it, and sends it.

- The acknowledgement field contains the number of the last frame received without error. If this number agrees with the sequence number of the frame the sender is trying to send, the sender knows it is done with the frame stored in *buff- er* and can fetch the next packet from its network layer.

- When the first valid frame arrives at computer *B*, it will be accepted and *frame expected* will be set to a value of 1. All the subsequent frames received will be rejected because *B* is now expecting frames with sequence number 1, not0. Furthermore, since all the duplicates will have *ack* = 1 and *B* is still waiting for an acknowledgement of 0, *B* will not go and fetch a new packet from its network layer.

- After every rejected duplicate comes in, *B* will send *A* a frame containing *seq* = 0 and *ack* = 0. Eventually, one of these will arrive correctly at *A*, causing *A* to begin sending the next packet. No combination of lost frames or premature timeouts can cause the protocol to deliver duplicate packets to either network layer, to skip a packet, or to deadlock.

- In part (a), the normal opera- tion of the protocol is shown. In (b) the peculiarity is illustrated. If *B* waits for *A*'s first frame before sending one of its own, the sequence is as shown in (a), and every frame is accepted.

- if *A* and *B* simultaneously initiate communication, their first frames cross, and the data link layers then get into situation (b). In (a) each frame arrival brings a new packet for the network layer; there are no duplicates. In (b) half of the frames contain duplicates, even though there are no transmission errors. Simi- lar situations can occur as a result of premature timeouts, even when one side clearly starts first. In fact, if multiple premature timeouts occur, frames may be sent three or more times, wasting valuable bandwidth.

## ❖ A Protocol Using Go-Back-N

- The transmission time re- quired for a frame to arrive at the receiver plus the transmission time for the ac- knowledgement to come back is negligible. Sometimes this assumption is clearly false. In these situations the long round-trip time can have important implications for the efficiency of the bandwidth utilization.

- The problem described here can be viewed as a consequence of the rule requiring a sender to wait for an acknowledgement before sending another frame. If we relax that restriction, much better efficiency can be achieved. Basically, the solution lies in allowing the sender to transmit up to *w* frames before blocking, in- stead of just 1.

- To find an appropriate value for *w* we need to know how many frames can fit inside the channel as they propagate from sender to receiver. This capacity is de- termined by the bandwidth in bits/sec multiplied by the one-way transit time, or the **bandwidth-delay product** of the link. We can divide this quantity by the number of bits in a frame to express it as a number of frames. Call this quantity *BD.*

- For smaller window sizes, the utilization of the link will be less than 100% since the sender will be blocked sometimes. We can write the utilization as the fraction of time that the sender is not blocked:

$$\text{link utilization} \leq \frac{w}{1 + 2BD}$$

- This value is an upper bound because it does not allow for any frame processing time and treats the acknowledgement frame as having zero length, since it is usually short. The equation shows the need for having a large window *w* when- ever the bandwidth-delay product is large.

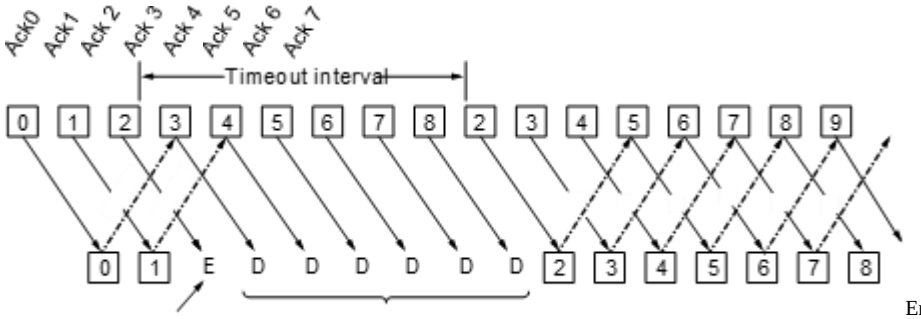
- This technique of keeping multiple frames in flight is an example of **pipelin- ing**. Pipelining frames over an unreliable communication channel raises some serious issues. First, what happens if a frame in the middle of a long stream is damaged or lost? Large numbers of succeeding frames will arrive at the receiver before the sender even finds out that anything is wrong.

- One option, called **go-back-n**, is for the receiver simply to discard all subsequent frames, sending no acknowledgements for the discarded frames. This strategy corresponds to a receive window of size 1. In other words, the data link layer refuses to accept any frame except the next one it must give to the network layer. If the sender's window fills up before the timer runs out, the pipeline will begin to empty.

- In Fig. 3-18(b) we see go-back-n for the case in which the receiver's window is large. Frames 0 and 1 are correctly received and acknowledged. Frame 2, however, is damaged or lost. The sender, unaware of this problem, continues to send frames until the timer for frame 2 expires. Then it backs up to frame 2 and starts over with it, sending 2, 3, 4, etc. all over again.

- The other general strategy for handling errors when frames are pipelined is called **selective repeat**. When it is used, a bad frame that is received is discarded, but any good frames received after it are accepted and buffered.

- Two basic approaches are available for dealing with errors in the presence of pipelining, both of which are shown in Fig. 3-18.




(a)

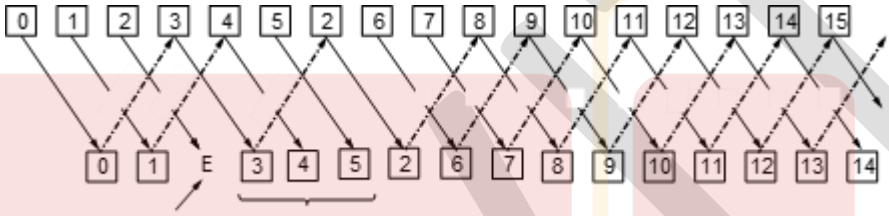



**Figure 3-18.** Pipelining and error recovery. Effect of an error when

(a) receiver's window size is 1 and (b) receiver's window size is large

- When the sender times out, only the oldest unacknowledged frame is retransmitted. If that frame arrives correctly, the receiver can deliver to the network layer, in sequence, all the frames it has buffered. Selective repeat corresponds to a receiver window larger than 1.
- Selective repeat is often combined with having the receiver send a negative acknowledgement (NAK) when it detects an error, for example, when it receives a checksum error or a frame out of sequence. NAKs stimulate retransmission be- fore the corresponding timer expires and thus improve performance.
- When that arrives, the data link layer now has 2, 3, 4, and 5 and can pass all of them to the network layer in the correct order. It can also acknowledge all frames up to and including 5, as shown in the figure. If the NAK should get lost, eventually the sender will time out for frame 2 and send it (and only it) of its own accord, but that may be a quite a while later.
- When the network layer has a packet it wants to send, it can cause a *net- work layer ready* event to happen. However, to enforce the flow control limit on the sender window or the number of unacknowledged frames that may be out- standing at any time, the data link layer must be able to keep the network layer from bothering it with more work. The library procedures *enable network layer* and *disable network layer* do this job.
- The maximum number of frames that may be outstanding at any instant is not

the same as the size of the sequence number space. For go-back-n, *MAX SEQ* frames may be outstanding at any instant, even though there are *MAX SEQ* □ 1 distinct sequence numbers (which are 0, 1, . . . , *MAX SEQ*). We will see an even tighter restriction for the next protocol, selective repeat. To see why this res- triction is required, consider the following scenario with *MAX SEQ* □ 7:

1. he sender sends frames 0 through 7.
2. A piggybacked acknowledgement for 7 comes back to the sender.
3. The sender sends another eight frames, again with sequence numbers 0 through 7.
4. Now another piggybacked acknowledgement for frame 7 comes in.

```
/* Protocol 5 (Go-back-n) allows multiple outstanding frames. The sender may transmit up to MAX SEQ
frames without waiting for an ack. In addition, unlike in the previous

protocols, the network layer is not assumed to have a new packet all the time. Instead, the network layer
causes a network layer ready event when there is a packet to send. */

#define MAX SEQ 7

typedef enum {frame arrival, cksum err, timeout, network layer ready} event type; #include
"protocol.h"


static boolean between(seq nr a, seq nr b, seq nr c)

{

/* Return true if a <= b < c circularly; false otherwise. */

 if (((a <= b) && (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a)))

      return(true); else

      return(false);

}

static void send data(seq nr frame nr, seq nr frame expected, packet buffer[ ])

{

/* Construct and send a data frame. */

 frame s;                                  /* scratch variable */

 s.info = buffer[frame nr];                /* insert packet into frame */

 s.seq = frame nr;                         /* insert sequence number into frame */ s.ack =
(frame expected + MAX SEQ) % (MAX SEQ + 1); /* piggyback ack */

 to physical layer(&s);                    /* transmit the frame */

 start timer(frame nr);                    /* start the timer running */

}


void protocol5(void)

{

 seq nr next frame to send;                /* MAX SEQ > 1; used for outbound stream */ seq nr ack
 expected;                                 /* oldest frame as yet unacknowledged */
```

```
––seq nr frame expected;                /* next frame expected on inbound stream */
frame r;                                /* scratch variable */
–packet buffer[MAX SEQ + 1];            /* buffers for the outbound stream */
–seq nr nbuffered;                      /* number of output buffers currently in use */
––seq nr i;                             /* used to index into the buffer array */ event type
event;


––––enable network layer();             /* allow network layer ready events */
–ack expected = 0;                      /* next ack expected inbound */
–––next frame to send = 0;              /* next frame going out */
–frame expected = 0;                    /* number of frame expected inbound */
nbuffered = 0;                          /* initially no packets are buffered */
while (true) {
  –––wait for event(&event);            /* four possibilities: see event type above */
  switch(event) {
    ––case network layer ready:         /* the network layer has a packet to send */
        /* Accept, save, and transmit a new frame. */
        –––––from network layer(&buffer[next frame to send]); /* fetch new packet */
        nbuffered = nbuffered + 1;      /* expand the sender's window */
        ––––––––send data(next frame to send, frame expected, buffer);/* transmit the frame */
        inc(next frame to send);        /* advance sender's upper window edge */ break;


    –––case frame arrival:              /* a data or control frame has arrived */ from physical
        layer(&r);                      /* get incoming frame from physical layer */
        –if (r.seq == frame expected) {
            /* Frames are accepted only in order. */
            ––to network layer(&r.info);    /* pass packet to network layer */
            –inc(frame expected);           /* advance lower edge of receiver's window */
        }

        /* Ack n implies n □ 1, n □ 2, etc. Check for this. */
        ––––while (between(ack expected, r.ack, next frame to send)) {
```

```
                /* Handle piggybacked ack. */
                nbuffered = nbuffered □ 1;          /* one frame fewer buffered */
                ---stop timer(ack expected);        /* frame arrived intact; stop timer */ inc(ack
                expected);                          /* contract sender's window */
            }
            break;
        case cksum err: break;          /* just ignore bad frames */
        ----case timeout:                           /* trouble; retransmit all outstanding frames */ next frame
            to send = ack expected;                 /* start retransmitting here */
            for (i = 1; i <= nbuffered; i++) {
                --------send data(next frame to send, frame expected, buffer);/* resend frame */
                inc(next frame to send);            /* prepare to send the next one */
            }
    }
    -----if (nbuffered < MAX SEQ) enable
        network layer();
    else
  }
}
```

## ❖ A Protocol Using Selective Repeat

- The go-back-n protocol works well if errors are rare, but if the line is poor it wastes a lot of bandwidth on retransmitted frames. An alternative strategy, the selective repeat protocol, is to allow the receiver to accept and buffer the frames following a damaged or lost one.
- In this protocol, both sender and receiver maintain a window of outstanding and acceptable sequence numbers, respectively. The sender's window size starts out at 0 and grows to some predefined maximum. The receiver's window, in con- trast, is always fixed in size and equal to the predetermined maximum.
- The re- ceiver has a buffer reserved for each sequence number within its fixed window. Associated with each buffer is a bit (*arrived* ) telling whether the buffer is full or empty. Whenever a frame arrives, its sequence number is checked by the function *between* to see if it falls within the window. If so and if it has not already been re- ceived, it is accepted and stored.
- This action is taken without regard to whether or not the frame contains the next packet expected by the network layer. Of course, it must be kept within the data link layer and not passed to the network layer until all the lower-numbered frames have already been delivered to the network layer in the correct order.

- Nonsequential receive introduces further constraints on frame sequence num- bers compared to protocols in which frames are only accepted in order. We can illustrate the trouble most easily with an example. Suppose that we have a 3-bit sequence number, so that the sender is permitted to transmit up to seven frames before being required to wait for an acknowledgement.
- It is at this point that disaster strikes in the form of a lightning bolt hitting the telephone pole and wiping out all the acknowledgements. The protocol should operate correctly despite this disaster

```
/* Protocol 6 (Selective repeat) accepts frames out of order but passes packets to the network layer in order.
   Associated with each outstanding frame is a timer. When the timer

   expires, only that frame is retransmitted, not all the outstanding frames, as in protocol 5. */

---#define MAX SEQ 7                          /* should be 2^n  1 */ #define
NR BUFS ((MAX SEQ + 1)/2)

------typedef enum {frame arrival, cksum err, timeout, network layer ready, ack timeout} event type; #include
"protocol.h"

-boolean no nak = true;                       /* no nak has been sent yet */

---seq nr oldest frame = MAX SEQ + 1;         /* initial value is only for the simulator */

---static boolean between(seq nr a, seq nr b, seq nr c)

{

/* Same as between in protocol 5, but shorter and more obscure. */ return ((a <= b) &&
  (b < c)) || ((c < a) && (a <= b)) || ((b < c) && (c < a));

}

------static void send frame(frame kind fk, seq nr frame nr, seq nr frame expected, packet buffer[ ])

{

/* Construct and send a data, ack, or nak frame. */

  frame s;                                    /* scratch variable */

  --s.kind = fk;                              /* kind == data, ack, or nak */ if (fk ==
  data) s.info = buffer[frame nr % NR BUFS];

  ----s.seq = frame nr;                       /* only meaningful for data frames */ s.ack =
  (frame expected + MAX SEQ) % (MAX SEQ + 1);

  -if (fk == nak) no nak = false;             /* one nak per frame, please */

  -----to physical layer(&s);                 /* transmit the frame */ if (fk ==
  data) start timer(frame nr % NR BUFS);

  --stop ack timer();                         /* no need for separate ack frame */

}


 void protocol6(void)
```

```
{
  seq nr ack expected;                    /* lower edge of sender's window */
  seq nr next frame to send;              /* upper edge of sender's window + 1 */
  seq nr frame expected;                  /* lower edge of receiver's window */
  seq nr too far;                         /* upper edge of receiver's window + 1 */
  int i;                                  /* index into buffer pool */
  frame r;                                /* scratch variable */
  packet out buf[NR BUFS];                /* buffers for the outbound stream */
  packet in buf[NR BUFS];                 /* buffers for the inbound stream */
  boolean arrived[NR BUFS];               /* inbound bit map */
  seq nr nbuffered;                       /* how many output buffers currently used */ event type
event;

  enable network layer();                 /* initialize */
  ack expected = 0;                       /* next ack expected on the inbound stream */
  next frame to send = 0;                 /* number of next outgoing frame */ frame
expected = 0;
  too far = NR BUFS;
  nbuffered = 0;                          /* initially no packets are buffered */ for (i = 0; i
< NR BUFS; i++) arrived[i] = false;
while (true) {
    wait for event(&event);               /* five possibilities: see event type above */ switch(event)
    {
      case network layer ready:           /* accept, save, and transmit a new frame */ nbuffered =
        nbuffered + 1;                    /* expand the window */
        from network layer(&out buf[next frame to send % NR BUFS]); /* fetch new
        packet */ send frame(data, next frame to send, frame expected, out buf);/* transmit the frame */ inc(next
        frame to send);                   /* advance upper window edge */
        break;
      case frame arrival:                 /* a data or control frame has arrived */
        from physical layer(&r);          /* fetch incoming frame from physical layer */ if (r.kind ==
        data) {
          /* An undamaged frame has arrived. */ if ((r.seq
          != frame expected) && no nak)
        send frame(nak, 0, frame expected, out buf); else start ack timer();
```

```
        ---if (between(frame expected,r.seq,too far) && (arrived[r.seq%NR BUFS]==false)) {
            /* Frames may be accepted in any order. */
            -arrived[r.seq % NR BUFS] = true;          /* mark buffer as full */
            ----in buf[r.seq % NR BUFS] = r.info; /* insert data into buffer */ while
            (arrived[frame expected % NR BUFS]) {
                /* Pass frames and advance window. */
                -----to network layer(&in buf[frame expected % NR BUFS]);
                -no nak = true;
                --arrived[frame expected % NR BUFS] = false;
                ----inc(frame expected); /* advance lower edge of receiver's window */ inc(too
                far);                    /* advance upper edge of receiver's window */ start ack
                timer();                 /* to see if a separate ack is needed */
            }
        }
    }
    ----------if((r.kind==nak) && between(ack expected,(r.ack+1)%(MAX SEQ+1),next frame to send))
        send frame(data, (r.ack+1) % (MAX SEQ + 1), frame expected, out buf);

    ----while (between(ack expected, r.ack, next frame to send)) { nbuffered =
        nbuffered  1;                    /* handle piggybacked ack */
        ---stop timer(ack expected % NR BUFS);     /* frame arrived intact */
        -inc(ack expected);              /* advance lower edge of sender's window */
    }
    -break; case cksum
err:
    ----if (no nak) send frame(nak, 0, frame expected, out buf); /* damaged frame */
    break;
case timeout:
    ----send frame(data, oldest frame, frame expected, out buf); /* we timed out */ break;
-case ack timeout:
    ---send frame(ack,0,frame expected, out buf);       /* ack timer expired; send ack */
}
-----if (nbuffered < NR BUFS) enable network layer(); else disable network layer();
}
```

**Figure 3-21.** A sliding window protocol using selective repeat.

- The sender is happy to learn that all its transmitted frames did actually arrive correctly, so it advances its window and immediately sends frames 7, 0, 1, 2, 3, 4, and 5. Frame 7 will be accepted by the receiver and its packet will be passed di- rectly to the network layer.

- The essence of the problem is that after the receiver advanced its window, the new range of valid sequence numbers overlapped the old one.

### ❖ EXAMPLE DATA LINK PROTOCOLS

- Within a single building, LANs are widely used for interconnection, but most wide-area network infrastructure is built up from point-to-point lines. In Chap. 4, we will look at LANs.

- The first situation is when packets are sent over SONET optical fiber links in wide-area networks. These links are widely used, for example, to connect routers in the different loca- tions of an ISP's network.

- The second situation is for ADSL links running on the local loop of the tele- phone network at the edge of the Internet. These links connect millions of individ- uals and businesses to the Internet.

- he Internet needs point-to-point links for these uses, as well as dial-up mo- dems, leased lines, and cable modems, and so on. A standard protocol called **PPP.**

# THE MEDIUM ACCESS CONTROL SUBLAYER

- 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, consider a con- ference call in which six people, on six different telephones, are all connected so that each one can hear and talk to all the others.
- When only a single channel is available, it is much harder to determine who should go next. Many protocols for solving the problem are known. They form the contents of this chap- ter. 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 be- long to a sublayer of the data link layer called the **MAC** (**Medium Access Con- trol**) sublayer. The MAC sublayer is especially important in LANs, particularly wireless ones because wireless is naturally a broadcast channel.
- WANs, in con- trast, 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 eneral, including a few issues that are not strictly part of the MAC sublayer, but the main subject here will be control of the channel.
- 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 proto- cols in Chap. 3. Nevertheless, for most people, it is easier to understand protocols involving multiple parties after two-party protocols are well understood.

### ❖ THE CHANNEL ALLOCATION PROBLEM

- The central theme of this chapter is how to allocate a single broadcast channel among competing users. The channel might be a portion of the wireless spectrum in a geographic region, or a single wire or optical fiber to which multiple nodes are connected.
- In both cases, the channel connects each user to all other users and any user who makes full use of the channel interferes with other users who also wish to use the channel.
- We will first look at the shortcomings of static allocation schemes for bursty traffic. Then, we will lay out the key assumptions used to model the dynamic schemes that we examine in the following sections.

## ❖ Static Channel Allocation

- The traditional way of allocating a single channel, such as a telephone trunk, among multiple competing users is to chop up its capacity by using one of the multiplexing schemes we described in Sec. 2.5, such as FDM (Frequency Division Multiplexing).
- When there is only a small and constant number of users, each of which has a steady stream or a heavy load of traffic, this division is a simple and efficient allocation mechanism. A wireless example is FM radio stations. Each station gets a portion of the FM band and uses it most of the time to broadcast its signal.
- when the number of senders is large and 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.
- The poor performance of static FDM can easily be seen with a simple queue- ing theory calculation. Let us start by finding the mean time delay, $T$, to send a frame onto a channel of capacity $C$ bps. We assume that the frames arrive ran- domly with an average arrival rate of □ frames/sec, and that the frames vary in length with an average length of 1/□ bits. With these parameters, the service rate of the channel is □$C$ frames/sec.

  A standard queueing theory result is: $T$ □ 1/ □$C$ □ □
- The mean delay for the divided channel is $N$ times worse than if all the frames were somehow magically arranged orderly in a big central queue. Since none of the traditional static channel allocation methods work well at all with bursty traffic, we will now explore dynamic methods.

### ❖ Assumptions for Dynamic Channel Allocation

Before we get to the first of the many channel allocation methods in this chap- ter, it is worthwhile to carefully formulate the allocation problem. Underlying all the work done in this area are the following five key assumptions:

1. **Independent Traffic**. The model consists of $N$ independent **stations** (e.g., computers, telephones), each with a program or user that gener- ates frames for transmission. The expected number of frames gener- ated in an interval of length □$t$ is □□$t$, where □ is a constant (the arri- val rate of new frames). Once a frame has been generated, the sta- tion is blocked and does nothing until the frame has been suc- cessfully transmitted.
2. **Single Channel**. A single channel is available for all communica- tion. All stations can transmit on it and all can receive from it. The stations are assumed to be equally capable, though protocols may assign them different roles (e.g., priorities).

3. **Observable Collisions**. If two frames are transmitted simultan- eously, they overlap in time and the resulting signal is garbled. This event is called a **collision**. All stations can detect that a collision has occurred. A collided frame must be transmitted again later. No er- rors other than those generated by collisions occur.
4. **Continuous or Slotted Time**. Time may be assumed continuous, in which case frame transmission can begin at any instant. Alterna- tively, time may be slotted or divided into discrete intervals (called slots). Frame transmissions must then 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.

5. **Carrier Sense or No Carrier Sense**. With the carrier sense as- sumption, stations can tell if the channel is in use before trying to use it. No station will attempt to use the channel while it is sensed as busy. If there is 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.

   - The first one says that frame arrivals are independent, both across stations and at a particular station, and that frames are generated unpredictably but at a constant rate.
   - **Poisson models**, as they are frequently called, are useful because they are mathematically tractable. They help us analyze protocols to understand roughly how performance changes over an operating range and how it compares with other designs.
   - The single-channel assumption is the heart of the model. No external ways to communicate exist. Stations cannot raise their hands to request that the teacher call on them, so we will have to come up with better solutions.
   - The collision assumption is basic. Stations need some way to detect collisions if they are to retransmit frames rather than let them be lost. For wired channels, node hardware can be designed to detect collisions when they occur. The stations can then terminate their transmissions prematurely to avoid wasting capacity.
   - The reason for the two alternative assumptions about time is that slotted time can be used to improve performance. However, it requires the stations to follow a master clock or synchronize their actions with each other to divide time into dis- crete intervals. Hence, it is not always available.
   - similarly, a network may have carrier sensing or not have it. Wired networks will generally have carrier sense. Wireless networks cannot always use it ef- fectively because not every station may be within radio range of every other sta- tion. Similarly, carrier sense will not be available in other settings in which a sta- tion cannot communicate directly with other stations, for example a cable modem in which stations must communicate via the cable headend.
   - To avoid any misunderstanding, it is worth noting that no multiaccess proto- col guarantees reliable delivery. Even in the absence of collisions, the receiver may have copied some of the frame incorrectly for various reasons. Other parts of the link layer or higher layers provide reliability.

## ❖ MULTIPLE ACCESS PROTOCOLS

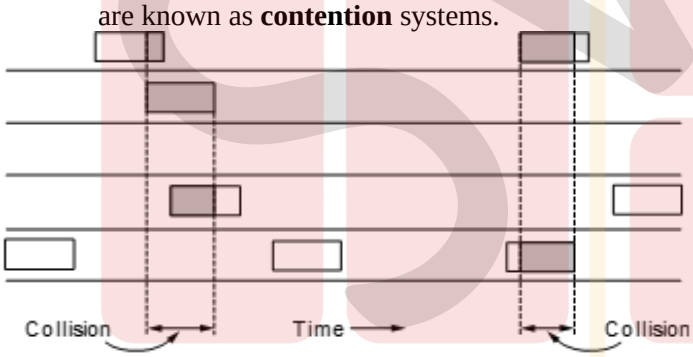
- Any algorithms for allocating a multiple access channel are known. In the following sections, we will study a small sample of the more interesting ones and give some examples of how they are commonly used in practice.

## ➢ ALOHA

- The story of our first MAC starts out in pristine Hawaii in the early 1970s. In this case, ‘‘pristine’’ can be interpreted as ‘‘not having a working telephone sys- tem.’’ This did not make life more pleasant for researcher Norman Abramson and his colleagues at the University of Hawaii who were trying to connect users on re- mote islands to the main computer in Honolulu.
- The one they found used short-range radios, with each user terminal sharing the same upstream frequency to send frames to the central computer. It included a simple and elegant method to solve the channel allocation problem.
- Their work has been extended by many researchers since then (Schwartz and Abramson, 2009). 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 continuous, as in the pure version, or divided into discrete slots into which all frames must fit.

## ➢ Pure ALOHA

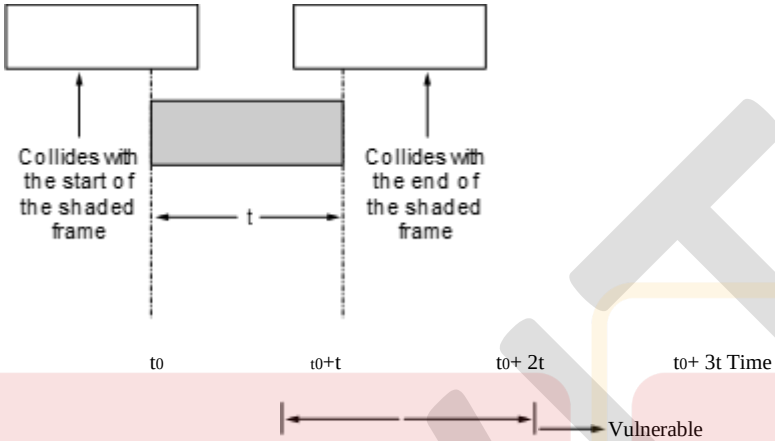
- The basic idea of an ALOHA system is simple: let users transmit whenever they have data to be sent. There will be collisions, of course, and the colliding frames will be damaged. Senders need some way to find out if this is the case. In the ALOHA system, after each station has sent its frame to the central computer, this computer rebroadcasts the frame to all of the stations
- 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 known as **contention** systems.




- A user is always in one of two states: typing or wait- ing. 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 con- taining the line over the shared channel to the central computer and checks the channel to see if it was successful.
- 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 $0 < N < 1$.
- Let us further assume that the old and new frames combined are well modeled by a Poisson distribution, with mean of $G$ frames per frame time. Clearly, $G \geq N$. At low load (i.e., $N \to 0$), there will be few collisions, hence few retransmissions, so $G \to N$. At high load, there will be

many collisions, so $G > N$.

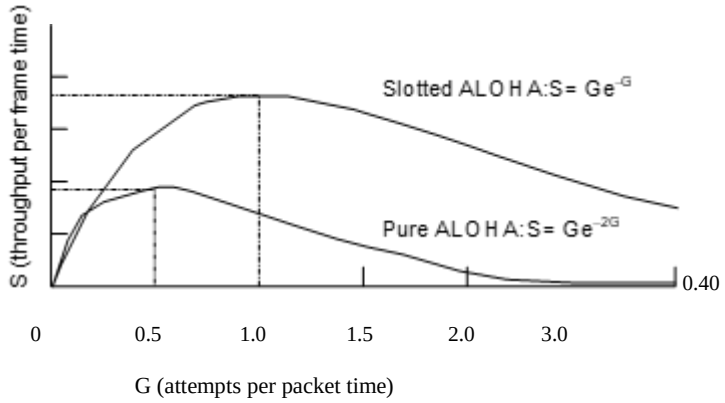
- If any other user has generated a frame between time $t_0$ and $t_0 + t$, the end of that frame will collide with the beginning of the shaded one.
- In fact, the shaded frame's fate was already sealed even before the first bit was sent, but since in pure ALOHA a station does not listen to the channel before transmitting, it has no way of knowing that another frame was already underway. Similarly, any other frame started between $t_0 + t$ and $t_0 + 2t$ will bump into the end of the shaded frame.




- The probability that $k$ frames are generated during a given frame time, in which $G$ frames are expected, is given by the Poisson distribution
- 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 frames 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 Fig. 4-3. 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%. This result is not very encouraging, but with everyone transmitting at will, we could hardly have expected a 100% success rate.

## ➢ Slotted ALOHA

- Soon after ALOHA came onto the scene, Roberts (1972) published a method for doubling the capacity of an ALOHA system. His proposal was to divide time into discrete intervals called **slots**, each interval corresponding to one frame.
- In Roberts' method, which has come to be known as **slotted ALOHA**—in contrast to Abramson's **pure ALOHA**—a station is not permitted to send when- ever the user types a line.




0.30

0.20

0.10

**Figure 4-3.** Throughput versus offered traffic for ALOHA systems.

- Instead, it is required to wait for the beginning of the next slot. Thus, the continuous time ALOHA is turned into a discrete time one. This halves the vulnerable period. To see this, look at Fig. 4-3 and imagine the collisions that are now possible. The probability of no other traffic during the same slot as our test frame is then $e^{-G}$, which leads to
- $S = Ge^{-G}$

- As you can see from Fig. 4-3, 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 (from Eq. 4-2). The best we can hope for using slotted ALOHA is 37% of the slots empty, 37% successes, and 26% 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 $e^{-G}$, which is the probabil- ity that all the other stations 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
- $P_k = e^{-G}(1 - e^{-G})^{k-1}$

## ♦ Carrier Sense Multiple Access Protocols

- This low result is hardly surprising, since with stations transmitting at will, with- out knowing what the other stations are doing there are bound to be many collis- ions.
- In LANs, however, it is often possible for stations to detect what other sta- tions are doing, and thus 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, and they were long ago analyzed in detail. For example, see Kleinrock and Tobagi (1975). Below we will look at several versions of carrier sense proto-

cols.

♦ **Persistent and Nonpersistent CSMA**

- The first carrier sense protocol that we will study here is called **1-persistent CSMA** (**Carrier Sense Multiple Access**). That is a bit of a mouthful for the sim- plest CSMA scheme. When a station has data to send, it first listens to the chan- nel to see if anyone else is transmitting at that moment.
- 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. This chance depends on the number of frames that fit on the channel, or the **bandwidth-delay product** of the channel.
- This protocol has better performance than pure ALOHA because both stations have the decency to desist from interfering with the third station's frame. 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. As before, a station senses the channel when it wants to send a frame, and if no one else is sending, the station begins doing so itself.
- 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 that the channel is busy, it waits until the next slot and applies the above algorithm. IEEE 802.11 uses a refinement of p-persistent CSMA that we will discuss in Sec. 4.4.


..

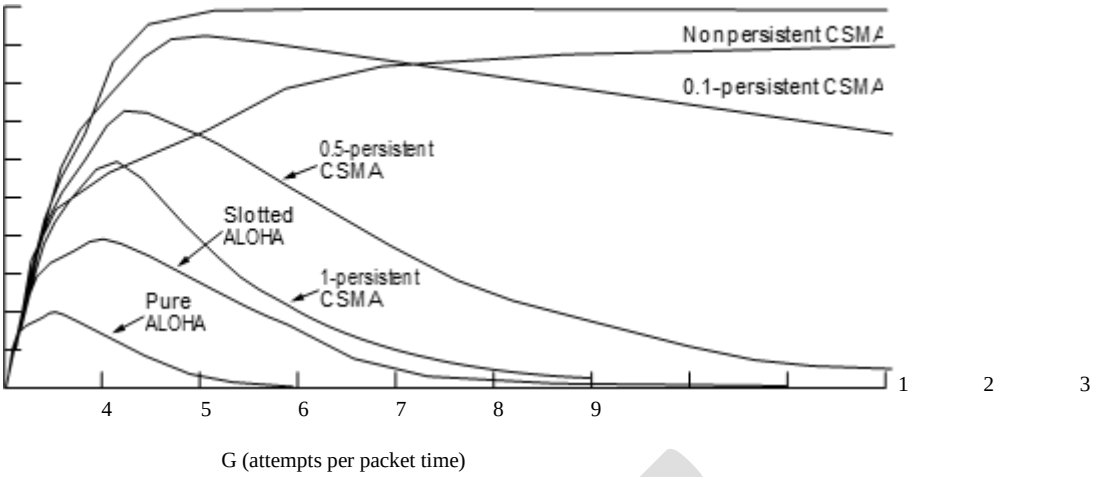



**Figure 4-4.** Comparison of the channel utilization versus load for various ran- dom access protocols.

Figure 4-4 shows the computed throughput versus offered traffic for all three protocols, as well as for pure and slotted ALOHA.

### ❖ CSMA with Collision Detection

- Persistent and nonpersistent CSMA protocols are definitely an improvement over ALOHA because they ensure that no station begins to transmit while the channel is busy.
- Another improvement is for the stations to quickly detect the collision and abruptly stop transmitting, (rather than finishing them) since they are irretrievably garbled anyway. This strategy saves time and bandwidth.
- This protocol, known as **CSMA/CD** (**CSMA with Collision Detection**), is the basis of the classic Ethernet LAN, so it is worth devoting some time to looking at it in detail. It is important to realize that collision detection is an analog proc- ess. The station's hardware must listen to the channel while it is transmitting.
- CSMA/CD, as well as many other LAN protocols, uses the conceptual model of Fig. 4-5. At the point marked $t_0$, a station has finished transmitting its frame. Any other station having a frame to send may now attempt to do so. If two or more stations decide to transmit simultaneously, there will be a collision.

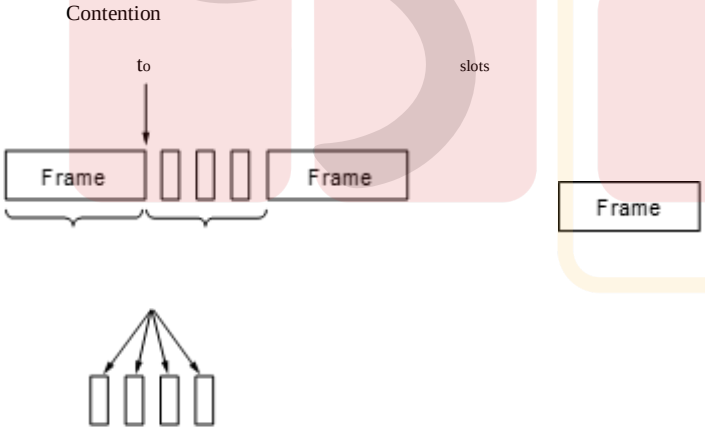



**Figure 4-5.** CSMA/CD can be in contention, transmission, or idle state.

- The minimum time to detect the collision is just the time it takes the signal to propagate from

one station to the other. Based on this information, you might think that a station that has not heard a collision for a time equal to the full cable propagation time after starting its transmission can be sure it has seized the cable.

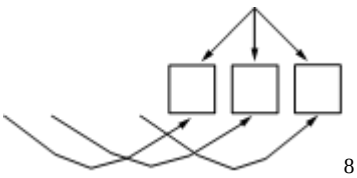
- Consider the following worst-case scenario. Let the time for a signal to pro- pagate between the two farthest stations be □. At $t_0$, one station begins trans- mitting. At $t_0$ □ □ □ □, an instant before the signal arrives at the most distant sta- tion, that station also begins transmitting.
- it detects the collision al- most instantly and stops, but the little noise burst caused by the collision does not get back to the original station until time 2□□ □. In other words, in the worst case a station cannot be sure that it has seized the channel until it has transmitted for 2□ without hearing a collision.
- The difference for CSMA/CD compared to slotted ALOHA is that slots in which only one station transmits (i.e., in which the channel is seized) are followed by the rest of a frame. This difference will greatly improve performance if the frame time is much longer than the propagation time.

### ♦ Collision-Free Protocols

- Although collisions do not occur with CSMA/CD once a station has unambi- guously captured the channel, they can still occur during the contention period. These collisions adversely affect the system performance, especially when the bandwidth-delay product is large, such as when the cable is long (i.e., large □) and the frames are short.
- Most of these protocols are not currently used in major systems, but in a rapidly changing field, having some protocols with excellent properties available for fu- ture systems is often a good thing.
- In the protocols to be described, we assume that there are exactly *N* stations, each programmed with a unique address from 0 to *N* □ 1. It does not matter that some stations may be inactive part of the time.

### ♦ A Bit-Map Protocol

- In our first collision-free protocol, the **basic bit-map method**, each con- tention period consists of exactly *N* slots. If station 0 has a frame to send, it trans- mits a 1 bit during the slot 0.
- In general, station *j* may announce that it has a frame to send by inserting a 1 bit into slot *j*. After all *N* slots have passed by, each station has complete knowledge of which stations wish to transmit.

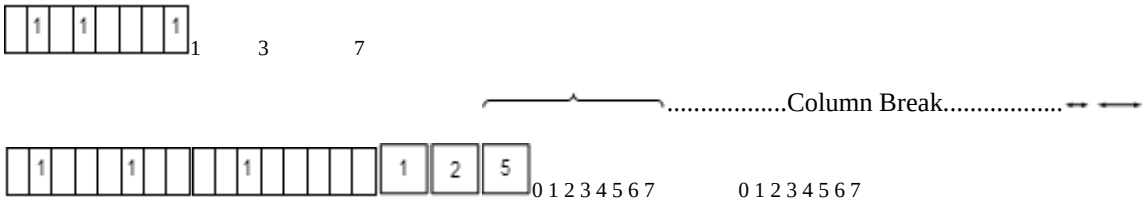


8 Contention slots Frames 8 Contention slots

1 d

0 1 2 3 4 5 6 7

.....Column Break.....




- Since everyone agrees on who goes next, there will never be any collisions. After the last ready station has transmitted its frame, an event all stations can easi- ly monitor, another *N*-bit contention period is begun.
- Protocols like this in which the desire to transmit is broadcast before the ac- tual transmission are called **reservation protocols** because they reserve channel ownership in advance and prevent collisions.
- Under conditions of low load, the bit map will simply be repeated over and over, for lack of data frames. Consider the situation from the point of view of a low-numbered station, such as 0 or 1. Typically, when it becomes ready to send, the ‘‘current’’ slot will be somewhere in the middle of the bit map.
- The prospects for high-numbered stations are brighter. Generally, these will only have to wait half a scan (*N*/2 bit slots) before starting to transmit. High- numbered stations rarely have to wait for the next scan.
- The channel efficiency at low load is easy to compute. The overhead per frame is *N* bits and the amount of data is *d* bits, for an efficiency of $d/(d + N)$.
- At high load, when all the stations have something to send all the time, the *N*- bit contention period is prorated over *N* frames, yielding an overhead of only 1 bit per frame, or an efficiency of $d/(d + 1)$.
- The mean delay for a frame is equal to the sum of the time it queues inside its station, plus an additional $(N - 1)d + N$ once it gets to the head of its internal queue. This interval is how long it takes to wait for all other stations to have their turn sending a frame and another bitmap.

### Token Passing

- The essence of the bit-map protocol is that it lets every station transmit a frame in turn in a predefined order. Another way to accomplish the same thing is to pass a small message called a **token.** from one station to the next in the same predefined order. The token represents permission to send.
- In a **token ring** protocol, the topology of the network is used to define the order in which stations send. The stations are connected one to the next in a single ring. Passing the token to the next station then simply consists of receiving the token in from one direction and transmitting it out in the other direction.

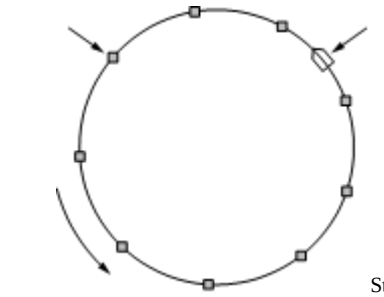


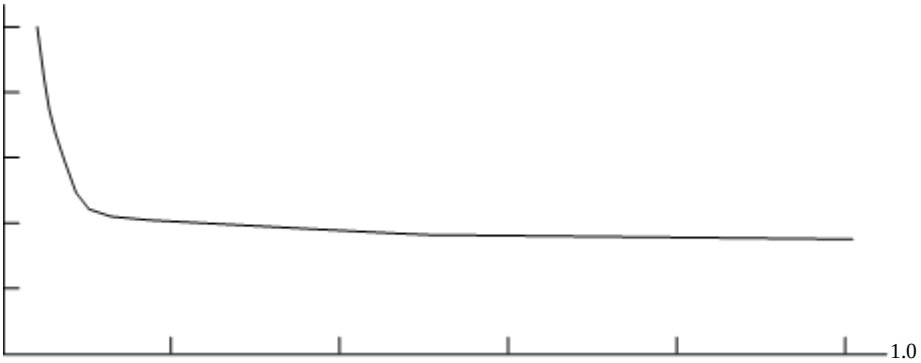

**Figure 4-7.** Token ring.

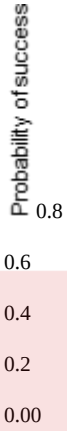
- The channel connecting the stations might instead be a single long bus. Each station then uses the bus to send the token to the next station in the predefined sequence. Possession of the token allows a station to use the bus to send one frame, as be- fore. This protocol is called **token bus**.
- In the 1990s, a much faster token ring called **FDDI** (**Fiber Distributed Data Interface**) was beaten out by switched Ethernet. In the 2000s, a token ring called **RPR** (**Resi- lient Packet Ring**) was defined as IEEE 802.17 to standardize the mix of metro- politan area rings in use by ISPs. We wonder what the 2010s will have to offer.

**Binary Countdown**

## Limited-Contention Protocols

- The contention and collision-free protocols, arriving at a new protocol that used con- tention at low load to provide low delay, but used a collision-free technique at high load to provide good channel efficiency. Such protocols, which we will call **limited-contention protocols**, do in fact exist, and will conclude our study of car- rier sense networks.
- The only contention protocols we have studied have been symmet- ric. That is, each station attempts to acquire the channel with some probability, $p$, with all stations using the same $p$.
- Each has a probability $p$ of transmitting during each slot. The probability that some station successfully acquires the channel during a given slot is the probability that any one station transmits, with probability $p$, and all other $k - 1$ stations defer, each with probability $1 - p$. This value is $kp(1 - p)^{k-1}$.




.....Column Break.....

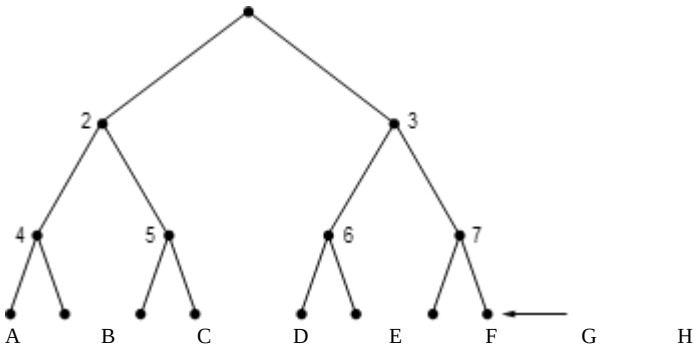
5 10 15 20 25

Number of ready stations

- They first divide the stations into (not necessarily disjoint) groups. Only the members of group 0 are permitted o compete for slot 0. If one of them succeeds, it acquires the channel and trans- mits its frame
- The limiting case is a single group containing all stations (i.e., slotted ALOHA). What we need is a way to assign stations to slots dynamically, with many stations per slot when the load is low and few (or even just one) station per slot when the load is high.

- **The Adaptive Tree Walk Protocol**

- A portion of each sample was poured into a single test tube. This mixed sample was then tested for antibodies. If none were found, all the soldiers in the group were declared healthy. If antibodies were present, two new mixed samples were prepared, one from soldiers 1 through *N*/2 and one from the rest. The proc- ess was repeated recursively until the infected soldiers were determined.




.....Column Break.....Stations

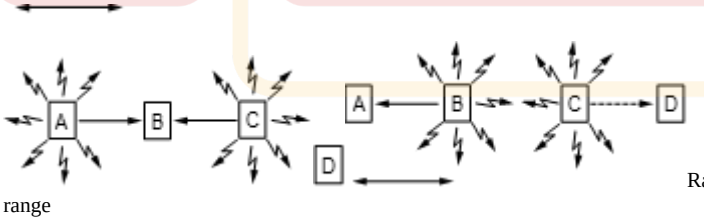
- In essence, if a collision occurs during slot 0, the entire tree is searched, depth first, to locate all ready stations. Each bit slot is associated with some particular node in the tree. If a collision occurs, the search continues recursively with the node's left and right children.
- When the load on the system is heavy, it is hardly worth the effort to dedicate slot 0 to node 1 because that makes sense only in the unlikely event that precisely one station has a frame to send. Similarly., one could argue that nodes 2 and 3 should be skipped as well for the same reason
- the expected number of them below a specific node at level $i$ is just $2^{-i}q$. Intuitively, we would expect the optimal level to begin searching the tree to be the one at which the mean number of contending stations per slot is 1, that is, the level at which $2^{-i}q \approx 1$. Solving this equation, we find that $i \approx \log_2 q$.
- Numerous improvements to the basic algorithm have been discovered and are discussed in some detail by Bertsekas and Gallager (1992). For example, consid- er the case of stations *G* and *H* being the only ones wanting to transmit.

## ♦ Wireless LAN Protocols

◊ A system of laptop computers that communicate by radio can be regarded as a wireless LAN, as we discussed in Sec. 1.5.3. Such a LAN is an example of a broadcast channel. It also has somewhat different properties than a wired LAN, which leads to different MAC protocols.

◊ There is an even more important difference between wireless LANs and wired LANs. A station on a wireless LAN may not be able to transmit frames to or re- ceive frames from all other stations because of the limited radio range of the sta- tions.

◊ The naive approach to using a wireless LAN might be to try CSMA: just listen for other transmissions and only transmit if no one else is doing so. The trouble is, this protocol is not really a good way to think about wireless because what mat- ters for reception is interference at the receiver, not at the sender.




(a) (b)

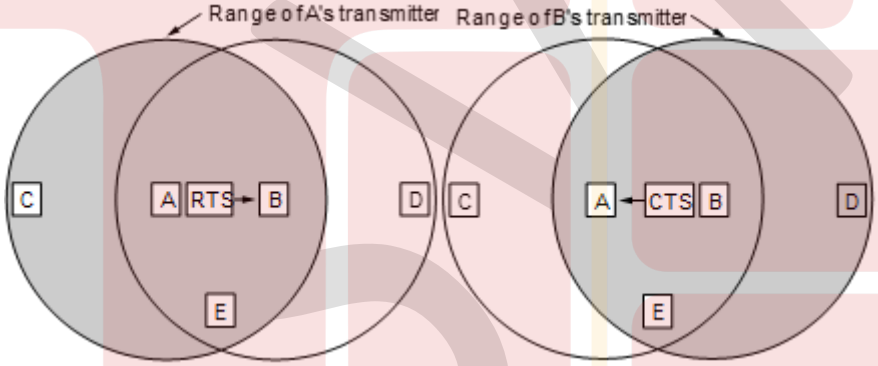
**Figure 4-11.** A wireless LAN. (a) *A* and *C* are hidden terminals when trans- mitting to *B*. (b) *B* and *C* are exposed terminals when transmitting to *A* and *D*.

- First consider what happens when *A* and *C* transmit to *B*, as depicted in Fig. 4-11(a). If *A* sends and then *C* immediately senses the medium, it will not hear *A* because *A* is out of range. Thus *C* will falsely conclude that it can transmit to *B*. If *C* does start transmitting, it will interfere at *B*, wiping out the frame from A.
- We want a MAC protocol that will prevent this kind of collision from happening because it wastes bandwidth. The problem of a station not being able to detect a potential competitor for the medium because the competitor is too far away is called the **hidden terminal problem**.
- In fact, such a transmission would cause bad reception only in the zone between *B* and *C*, where neither of the intended receivers is located. We want a MAC protocol that prevents this kind of deferral from happening because it wastes bandwidth. The problem is called the **exposed terminal problem**.
- We want this concurrency to happen as the cell gets larger and larger, in the same way that people at a party should not wait for everyone in the room to go silent before they talk; multiple conversations can take place at once in a large room as long as they are not directed to the same location.

- An early and influential protocol that tackles these problems for wireless LANs is **MACA** (**Multiple Access with Collision Avoidance**) (Karn, 1990). The basic idea behind it is for the sender to stimulate the receiver into outputting a short frame, so stations nearby can detect this transmission and avoid transmitting for the duration of the upcoming (large) data frame.

- MACA is illustrated in Fig. 4-12. Let us see how *A* sends a frame to *B*. *A* starts by sending an **RTS** (**Request To Send**) frame to *B*, as shown in Fig. 4-12(a). This short frame (30 bytes) contains the length of the data frame that will eventu- ally follow.
- Then *B* replies with a **CTS** (**Clear To Send**) frame, as shown in Fig. 4-12(b). The CTS frame contains the data length (copied from the RTS frame). Upon receipt of the CTS frame, *A* begins transmission.

receipt of the CTS frame, *A* begins transmission.




(a) (b)

**Figure 4-12.** The MACA protocol. (a) *A* sending an RTS to *B*. (b) *B* responding with a CTS to *A*.

- Therefore, it hears the RTS from *A* but not the CTS from *B*. As long as it does not interfere with the CTS, it is free to transmit while the data frame is being sent. In contrast, *D* is within range of *B* but not *A*. Station *E* hears both control messages and, like *D*, must be silent until the data frame is complete.
- Despite these precautions, collisions can still occur. For example, *B* and *C* could both send RTS frames to *A* at the same time. These will collide and be lost. In the event of a collision, an unsuccessful transmitter (i.e., one that does not hear a CTS within the expected time interval) waits a random amount of time and tries again later.

# THE NETWORK LAYER

The network layer is responsible for end-to-end transmission, requiring knowledge of the network's topology and selecting appropriate paths to avoid overloading communication lines and routers. It must also address new problems when the source and destination are in different networks, primarily using the Internet and its network layer protocol, IP. This chapter will study these issues and illustrate them using the Internet.

## 5.1 NETWORK LAYER DESIGN ISSUES

- The text introduces network layer designers' challenges, including the transport layer's service and the network's internal design.

### 5.1.1 Store-and-Forward Packet Switching

- The network layer protocols operate in a context of ISP's equipment and customers' equipment. Host H1 is connected to ISP routers, A, while H2 is on LAN with leased router F, owned by the customer. These routers are considered part of the ISP network as they run the same algorithms as the ISP's routers. This context can be seen in Fig. 5-1.

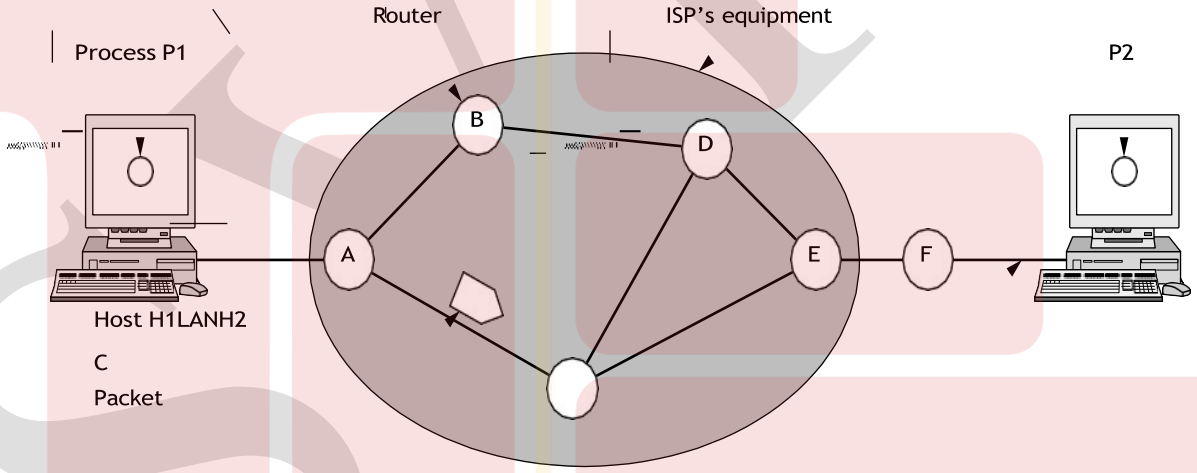



**Figure 5-1.** The environment of the network layer protocols.

- The equipment uses store-and-forward packet switching, where a host sends a packet to the nearest router, which stores it until it's processed and verified by the link, then forwards it to the destination host.

### 5.1.2 Services Provided to the Transport Layer

The network layer provides services to the transport layer at the interface, and the services must be designed with specific goals in mind.

1. The services should be independent of the router technology.
2. The transport layer should be shielded from the number, type, and topology of the routers present.

3. The network addresses made available to the transport layer should use a uniform numbering plan, even across LANs and WANs.

- The network layer designers have significant freedom in creating detailed specifications for services offered to the transport layer. However, this freedom often leads to a debate between two factions: one, represented by the Internet community, who believes routers' job is to move packets and nothing else, and the other, represented by telephone companies, who believe the network should provide a reliable, connection-oriented service. This viewpoint is based on the end-to-end argument, which has been influential in shaping the Internet.

- The debate continues, with early data networks being connection-oriented, but connectionless network layers have gained popularity since the early Internet days. The IP protocol is now a symbol of success, while connection-oriented technologies like ATM are now found in niche uses. However, the Internet is evolving connection-oriented features as quality of service becomes more important. Two examples of connection-oriented technologies are MPLS (MultiProtocol Label Switching) and VLANs, both widely used. The debate continues to evolve as the Internet evolves in response to the evolving needs of its users.

### 5.1.3 Implementation of Connectionless Service

- The network layer provides two types of service: connectionless and connection-oriented. Connectionless service injects packets into the network individually and routes them independently, forming datagram networks. No advance setup is needed, and the network is called a datagram network.
- On the other hand, connection-oriented service establishes a path from the source router to the destination router, forming a virtual circuit (VC) network. In a datagram network, a process P1 sends a long message to a transport layer, which prepares a transport header and sends the result to the network layer. This layer operates within the operating system, similar to the physical circuits set up by telephone systems.

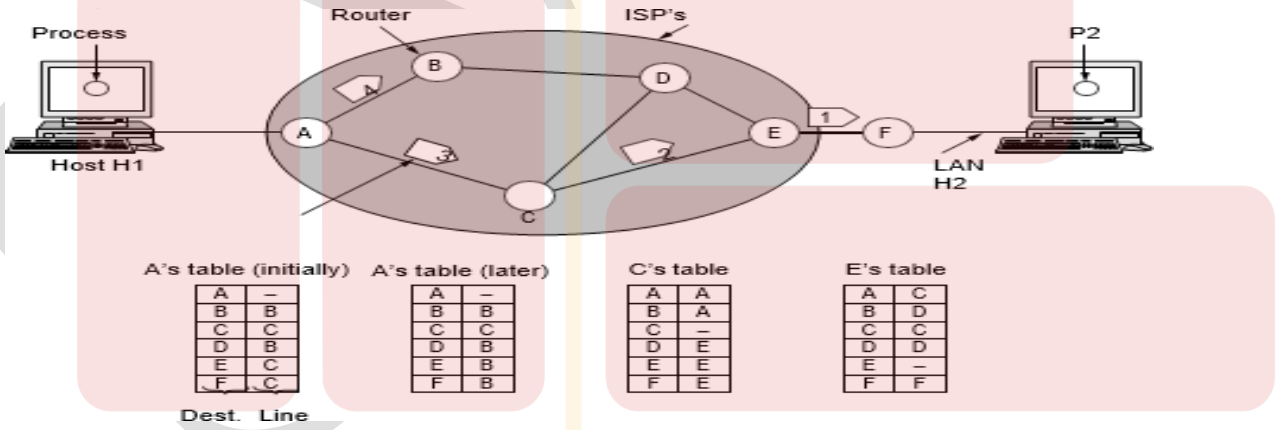



**Figure 5-2.** Routing within a datagram network.

- In this example, a message is four times longer than its maximum packet size, so the network layer breaks it into four packets, 1, 2, 3, and 4, and sends each to router A using a point-to-point protocol like PPP. The ISP takes over, and every router has an internal table indicating where to send packets for each possible destination. Each table entry is a pair consisting of a destination and the outgoing line to use for that destination.

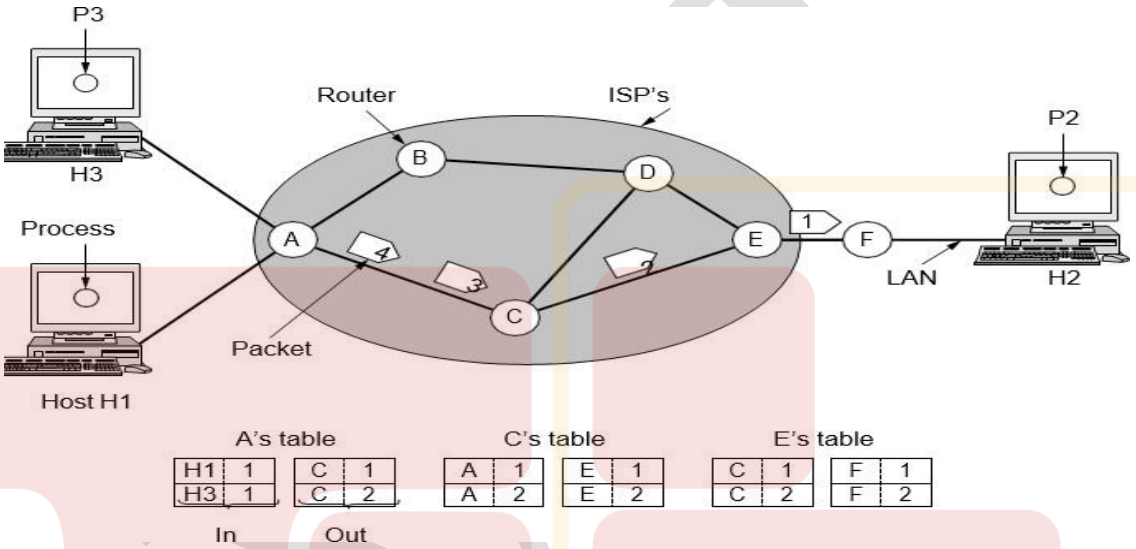
- A router has only two outgoing lines, so every incoming packet must be sent to one of these routers, even if the ultimate destination is to some other router. The router's initial routing table shows that packets 1, 2, and 3 are stored briefly at A, and each packet is

forwarded according to A's table to C within a new frame. Packet 1 is then forwarded to E and then to F, and when it gets to F, it is sent within a frame over the LAN to H2.

- However, packet 4 is sent to router B, even though it is also destined for F. A decided to send packet 4 via a different route than the first three packets, possibly due to a traffic jam along the ACE path. The routing algorithm manages the tables and makes routing decisions, and there are several different kinds of routing algorithms.

### 5.1.4 Implementation of Connection-Oriented Service

- A connection-oriented service uses a virtual-circuit network to avoid choosing a new route for every packet sent. A route is chosen during connection setup and stored in router tables. This route is used for all traffic flowing over the connection, similar to a telephone system. Each packet carries an identification indicating its virtual circuit. For example, a packet with connection identifier 1 is sent to router C and E.




- If H3 wants to establish a connection to H2, it chooses connection identifier 1 as its only connection and tells the network to establish the virtual circuit. This leads to a conflict in the tables, as A cannot distinguish connection 1 packets from H1 from connection 1 packets from H3. To avoid conflicts, routers need the ability to replace connection identifiers in outgoing packets. This process is called label switching and is used within ISP networks in the Internet.
- An example of a connection-oriented network service is MPLS (MultiProtocol Label Switching), which is used within ISP networks in the Internet with IP packets wrapped in an MPLS header with a 20-bit connection identifier or label. MPLS is often hidden from customers, but is increasingly used to help with quality of service and other ISP traffic management tasks.

### 5.1.5 Comparison of Virtual-Circuit and Datagram Networks

Both virtual circuits and datagrams have their supporters and their detractors. We will now attempt to summarize both sets of arguments. The major issues are listed in Fig. 5-4, although purists could probably find a counterexample for everything in the figure.

| Issue | Datagram network | Virtual-circuit network |
|---|---|---|
| Circuit setup | Not needed | Required |
| Addressing | Each packet contains the full source and destination address | Each packet contains a short VC number |
| State information | Routers do not hold state information about connections | Each VC requires router table space per connection |
| Routing | Each packet is routed independently | Route chosen when VC is set up; all packets follow it |
| Effect of router failures | None, except for packets lost during the crash | All VCs that passed through the failed router are terminated |
| Quality of service | Difficult | Easy if enough resources can be allocated in advance for each VC |
| Congestion control | Difficult | Easy if enough resources can be allocated in advance for each VC |

- Virtual circuits and datagrams have different advantages and disadvantages within a network. One trade-off is setup time, which takes time and resources, but is easily done with a data packet in a virtual-circuit network. In contrast, a datagram network requires more complicated lookup procedures to locate the entry for the destination.

- Datagram networks use longer destination addresses than virtual-circuit networks due to their global meaning, which can be a significant overhead and waste of bandwidth. Additionally, datagram networks require more table space in router memory, as each virtual circuit needs an entry.

- Virtual circuits offer some advantages in guaranteeing quality of service and avoiding congestion within the network, as resources can be reserved in advance when the connection is established. However, congestion avoidance is more difficult with a datagram network.

- For transaction processing systems, the overhead required to set up and clear a virtual circuit may dwarf its use. However, for long-running uses like VPN traffic between corporate offices, permanent virtual circuits may be useful.

- Virtual circuits also have a vulnerability problem, as if a router crashes and loses its memory, all virtual circuits passing through it must be aborted. Datagrams, on the other hand, allow routers to balance traffic throughout the network, as routes can be changed partway through a

long sequence of packet transmissions.

## 5.2 ROUTING ALGORITHMS

The network layer primarily handles packet routing from source to destination, with most networks requiring multiple hops. Network layer design focuses on the algorithms and data structures used to choose routes and manage network segments.

- The network layer is responsible for routing packets from the source machine to the destination machine, with most networks requiring multiple hops.
- The routing algorithm is a crucial part of network layer design, responsible for deciding which output line an incoming packet should be transmitted on. If the network uses datagrams internally, routing decisions must be made anew for every arriving data packet, while virtual circuits use established routes for entire sessions.
- A router has two processes: forwarding, which handles packets upon arrival, and updating the routing tables, where the routing algorithm plays a crucial role in deciding which routes to use.
- A routing algorithm should have certain desirable properties: correctness, simplicity, robustness, stability, fairness, and efficiency. Robustness is crucial for a network to cope with changes in topology and traffic without requiring all jobs to be aborted. Stability is also important, as a stable algorithm reaches equilibrium and stays there, ensuring communication is not disrupted until the algorithm reaches equilibrium. This is especially important for major networks that may run continuously for years without system-wide failures.
- Fairness and efficiency are often contradictory goals, as seen in Fig. 5-5. To maximize total flow, X to X′ traffic should be shut off, but X and X′ may not agree, requiring a compromise between global efficiency and fairness.
- Optimizing fairness and efficiency requires balancing packet delay and network throughput, often compromising by reducing packet distance or hops to improve delay and overall network throughput.

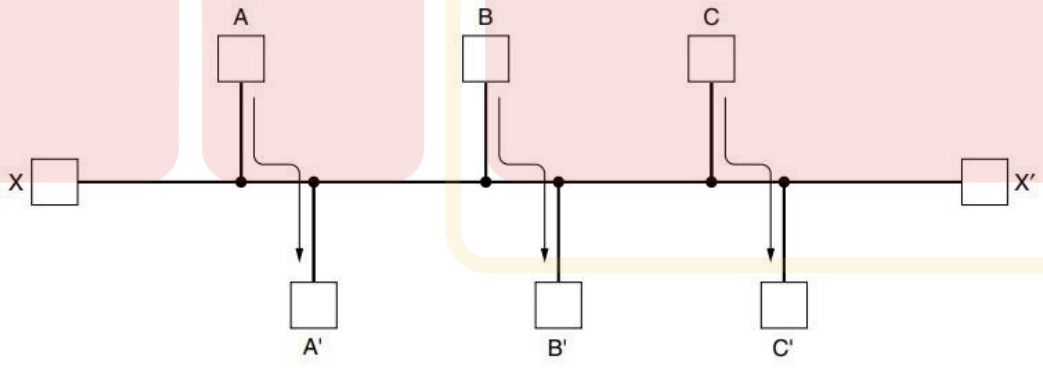



Fig: Network with a conflict between fairness and efficiency

- Static routing is a method where the choice of route is computed in advance, offline, and downloaded to routers during network booting, allowing for clear routing choices, such as sending packets to router E regardless of the final destination.
- Adaptive algorithms adapt their routing decisions to changes in topology and traffic. They

differ in information source, route change frequency, and optimization metrics. These algorithms cover delivery models and can send packets to multiple, all, or one of a set of destinations. Decisions based on topology are deferred to Sec 5.3.

### 5.2.1 The Optimality Principle

- The optimality principle, as proposed by Bellman (1957), states that if router J is on the optimal path from router I to router K, the optimal path from J to K also follows the same route. This principle can be applied to specific algorithms without considering network topology or traffic.
- The optimality principle enables the creation of a sink tree, a tree rooted at the destination, where all optimal routes from all sources are used by all router.

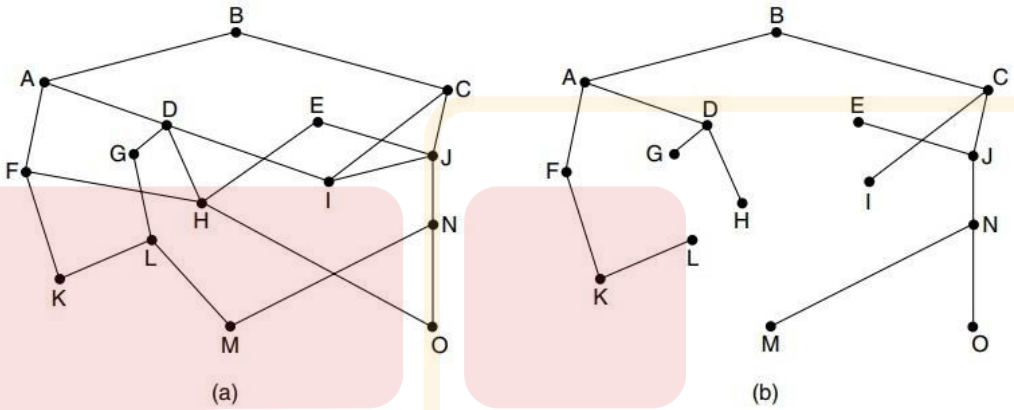



**Figure 5-6.** (a) A network. (b) A sink tree for router *B*.

- A sink tree is not unique, but if all possible paths are chosen, it forms a DAG (Directed Acyclic Graph) with no loops. Sink trees are used for both cases, depending on the assumption that paths do not interfere, such as a traffic jam on one path not causing another to divert.
- A sink tree is not unique, but if all possible paths are chosen, it forms a DAG (Directed Acyclic Graph) with no loops. Sink trees are used for both cases, depending on the assumption that paths do not interfere, such as a traffic jam on one path not causing another to divert.

### 5.2.2 Shortest Path Algorithm

- The study of routing algorithms begins with a straightforward method for computing optimal paths based on a complete network picture, despite the fact that not all routers may have all network details.

- The idea is to build a graph of the network, with each node of the graph representing a router and each edge of the graph representing a communication line, or link. To choose a route between a given pair of routers, the algorithm just finds the shortest path between

them on the graph.

- The concept of a shortest path deserves some explanation. One way of measuring path length is the number of hops. Using this metric, the paths ABC and ABE in Fig. 5-7 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).

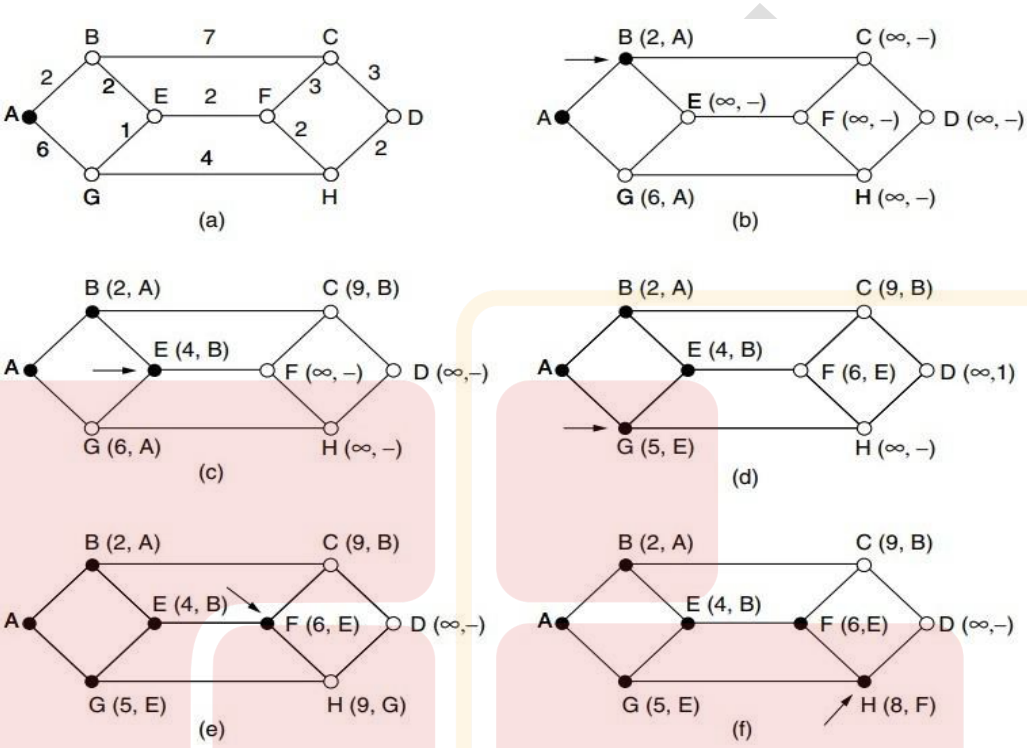



**Figure 5-7.** The first six steps used in computing the shortest path from *A* to *D*. The arrows indicate the working node.

- Dijkstra's algorithm, developed in 1959, finds the shortest paths between a source and all destinations in a network. Each node is labeled with its distance from the source node along the best known path, which must be non-negative. Initially, no paths are known, so all nodes are labeled with infinity. As paths are found, labels may change to reflect better paths, and labels can be tentative or permanent.
- The labeling algorithm is a method used to find the shortest path from a node A to a node D in a network. It starts by marking node A as permanent and then examines all nodes adjacent to it, relabeling them with distance to A.
- The node with the smallest label is made the new working node. Next, all nodes adjacent to B are examined. 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, a shorter path is considered. The entire graph is searched for the node with the smallest value, making it the working node for the next round.
- The algorithm consists of six steps. The algorithm works by determining the shortest path

from E to AXYZE, where node Z has already been made permanent. If it has, E has already been probed, so the AXYZE path cannot be a shorter path. If Z is still tentatively labeled, if the label is greater than or equal to E, then AXYZE cannot be a shorter path than ABE. If the label is less than E, Z becomes permanent first, allowing E to be probed from Z.

- The algorithm is given in Fig. 5-8, where global variables n and dist describe the graph and are initialized before shortest path is called. The shortest paths from t to s in an undirected graph are the same as the shortest paths from s to t.

### 5.2.3 Flooding

- Routers in routing algorithms rely on local knowledge rather than the entire network. A common technique is flooding, where every incoming packet is sent on every outgoing line except the one it arrived on. This generates numerous duplicate packets, unless measures are taken. One measure is having a hop counter in each packet header, which is decremented at each hop and discarded when it reaches zero. The hop counter should be initialized to the path length.

- Flooding with a hop count can lead to an exponential number of duplicate packets. To prevent this, routers should track which packets have been flooded and avoid sending them out again. This can be achieved by placing a sequence number in each packet received from its hosts, and having a list per source router detailing which sequence numbers have been seen.

```
#define MAX_NODES 1024                          /* maximum number of nodes */
#define INFINITY 1000000000                     /* a number larger than every maximum path */
int n, dist[MAX_NODES][MAX_NODES];              /* dist[i][j] is the distance from i to j */

void shortest_path(int s, int t, int path[])
{ struct state {                                /* the path being worked on */
      int predecessor;                          /* previous node */
      int length;                               /* length from source to this node */
      enum {permanent, tentative} label;        /* label state */
  } state[MAX_NODES];

  int i, k, min;
  struct state *p;

  for (p = &state[0]; p < &state[n]; p++) {     /* initialize state */
      p->predecessor = -1;
      p->length = INFINITY;
      p->label = tentative;
  }
  state[t].length = 0;  state[t].label = permanent;
  k = t;                                        /* k is the initial working node */
  do {                                          /* Is there a better path from k? */
      for (i = 0; i < n; i++)                   /* this graph has n nodes */
          if (dist[k][i] != 0 && state[i].label == tentative) {
              if (state[k].length + dist[k][i] < state[i].length) {
                  state[i].predecessor = k;
                  state[i].length = state[k].length + dist[k][i];
              }
          }

      /* Find the tentatively labeled node with the smallest label. */
      k = 0; min = INFINITY;
      for (i = 0; i < n; i++)
          if (state[i].label == tentative && state[i].length < min) {
              min = state[i].length;
              k = i;
          }
      state[k].label = permanent;
  } while (k != s);

  /* Copy the path into the output array. */
  i = 0;  k = s;
  do {path[i++] = k; k = state[k].predecessor; } while (k >= 0);
}
```

**Figure 5-8.** Dijkstra's algorithm to compute the shortest path through a graph.

- To prevent list growth without bound, each list should be augmented by a counter, k, ensuring all sequence numbers through k have been seen. When a packet arrives, it can be checked if it has been flooded and discarded.

- Flooding ensures packets are delivered to every node in the network, which is effective for broadcasting information. In wireless networks, all messages transmitted by a station can be received by all other stations within its radio range, which some algorithms utilize.
- Flooding is a robust routing algorithm that can find a path for a packet even in a war zone, and requires minimal setup. It can be used as a building block for more efficient routing algorithms and as a metric for comparison. Flooding always chooses the shortest path, making it the only algorithm that can produce a shorter delay, excluding the overhead generated by the flooding process itself.

### 5.2.4 Distance Vector Routing

- Computer networks utilize dynamic routing algorithms, such as distance vector routing and link state routing, which are more complex but more efficient in finding the shortest paths for the current topology, with a focus on the former algorithm.
- A distance vector routing algorithm operates by having each router maintain a table (i.e., a vector) giving the best known distance to each destination and which link to use to get there. These tables are updated by exchanging information with the neighbors. Eventually, every router knows the best link to reach each destination.
- The distance vector routing algorithm, also known as the Bellman-Ford routing algorithm, was the original ARPANET routing algorithm and was used in the Internet under the name RIP. It involves each router maintaining a routing table with an entry for each router, including the preferred outgoing line and distance estimate, which can be measured as hops or other metrics.
- The router is assumed to know the distance to its neighbors, either as hops or propagation delay. If the metric is hops, the distance is one hop.
- If the metric is propagation delay, the router can measure it directly with special ECHO packets. For example, if delay is used as a metric, the router sends a list of estimated delays to each neighbor every T msec. If the router knows the delay to X is m msec, it can reach router i via X in Xi + m msec. The router calculates the best estimate for each neighbor and uses it in its new routing table.
- This updating process is illustrated in Fig. 5-9. Part (a) shows a network. The first four columns of part (b) show the delay vectors received from the neighbors of router J. A claims to have a 12-msec delay to B, a 25-msec delay to C, a 40- msec delay to D, etc. Suppose that J has measured or estimated its delay to its neighbors, A, I, H, and K, as 8, 10, 12, and 6 msec, respectively.

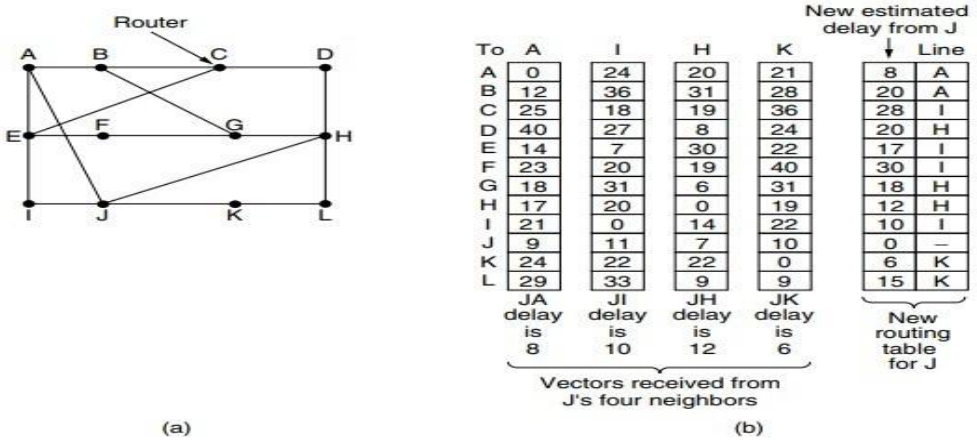



**Figure 5-9.** (a) A network. (b) Input from *A*, *I*, *H*, *K*, and the new routing table for *J*.

- Consider how J computes its new route to router G. It knows that it can get to A in 8 msec, and furthermore 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.

### 5.2.5 Link State Routing

- Distance vector routing was used in the ARPANET until 1979, when it was replaced by link state routing. The primary problem that caused its demise was that the algorithm often took too long to converge after the network topology changed (due to the count-to-infinity problem). Consequently, it was replaced by an entirely new algorithm, now called link state routing. Variants of link state routing called IS-IS and OSPF are the routing algorithms that are most widely used inside large networks and the Internet today.

- The idea behind link state routing is fairly simple and can be stated as five parts. Each router must do the following things to make it work:
  1. Discover its neighbors and learn their network addresses.
  2. Set the distance or cost metric to each of its neighbors.
  3. Construct a packet telling all it has just learned.
  4. Send this packet to and receive packets from all other routers.
  5. Compute the shortest path to every other router.

- In effect, the complete topology is distributed to every router. Then Dijkstra's algorithm can be run at each router to find the shortest path to every other router. Below we will consider each of these five steps in more detail

**Learning about the Neighbors**

- A router learns its neighbors by sending a special HELLO packet on each point-to-point

line, and the router on the other end sends a reply with its name.

- These names must be globally unique to determine if all three routers are connected to the same F. When connected by a broadcast link, the situation becomes more

complicated, as shown in a broadcast LAN with three routers connected to one or more additional routers.

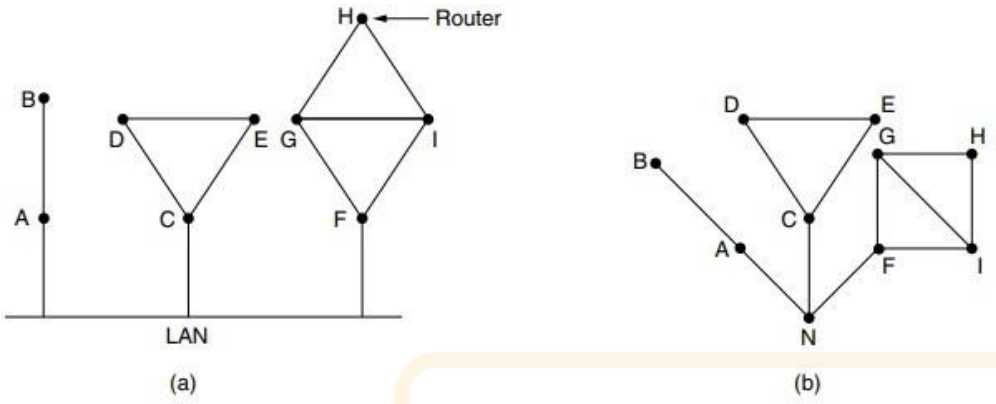



**Figure 5-11.** (a) Nine routers and a broadcast LAN. (b) A graph model of (a).

- The broadcast LAN provides connectivity between each pair of attached routers. However, modeling the LAN as many point-to-point links increases the size of the topology and leads to wasteful messages. A better way to model the LAN is to consider it as a node itself, as shown in Fig. 5-11(b). Here, we have introduced a new, artificial node, N, to which A, C, and F are connected. One designated router on the LAN is selected to play the role of N in the routing protocol. The fact that it is possible to go from A to C on the LAN is represented by the path ANC here.
- **Setting Link Costs:** The link state routing algorithm requires each link to have a distance or cost metric for finding shortest paths. The cost to reach neighbors can be set automatically, or configured by the network operator. A common choice is to make the cost inversely proportional to the bandwidth of the link. For example, 1-Gbps Ethernet may have a cost of 1 and 100-Mbps Ethernet a cost of 10. This makes higher-capacity paths better choices. If the network is geographically spread out, the delay of the links may be factored into the cost so that paths over shorter links are better choices. 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.
- **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. The cost to each neighbor is also given. An example network is presented in Fig. 5-12(a) with costs shown as labels on the lines. The corresponding link state packets for all six routers are shown in Fig. 5- 12(b).

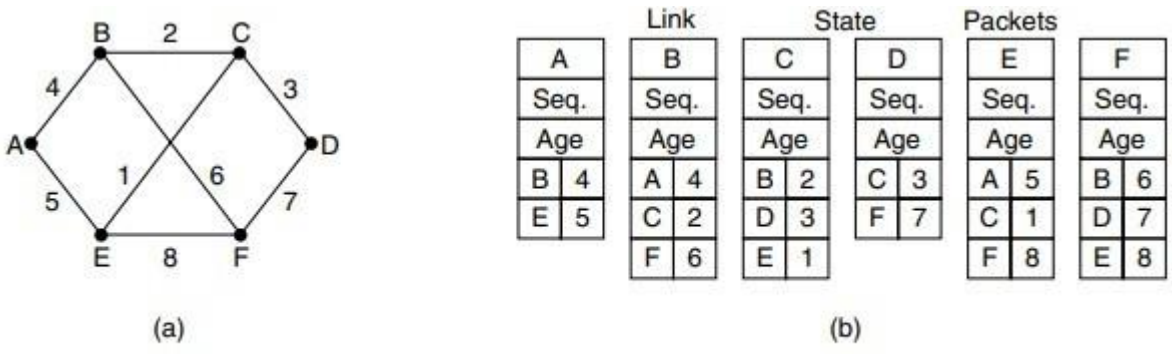
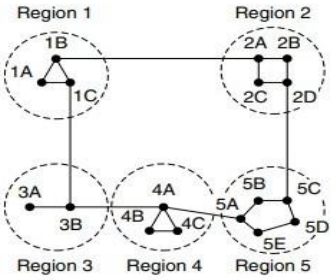



**Figure 5-12.** (a) A network. (b) The link state packets for this network.

- 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.

### 5.2.6 Hierarchical Routing

- As networks grow, router routing tables grow proportionally, consuming more memory, CPU time, and bandwidth. When the network becomes too large for every router to have an entry for every other router, hierarchical routing is used.
- Routers are divided into regions, each knowing how to route packets within its own region but not about the internal structure of other regions. This allows interconnected networks to treat each region as a separate area, allowing routers to avoid knowing the topological structure of other networks.
- For large networks, a two-level hierarchy may not be sufficient, as it may be necessary to group regions into clusters, clusters into zones, and zones into groups. For example, a packet from Berkeley to Malindi would be routed through a multilevel hierarchy.
- The Berkeley router would know the topology within California but send all out-of-state traffic to the Los Angeles router. The Los Angeles router could route traffic directly to other domestic routers but send all foreign traffic to New York. The New York router would direct all traffic to the destination country responsible for handling foreign traffic, such as Nairobi.
- The packet would then work its way down the tree in Kenya until it reached Malindi. Hierarchical routing reduces the table from 17 to 7 entries, but it also increases path length. For example, the best route from 1A to 5C is via region 2, but all traffic to region 5 goes via region 3, as it is better for most destinations in region 5.




(a)

Full table for 1A

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

(b)

Hierarchical table for 1A

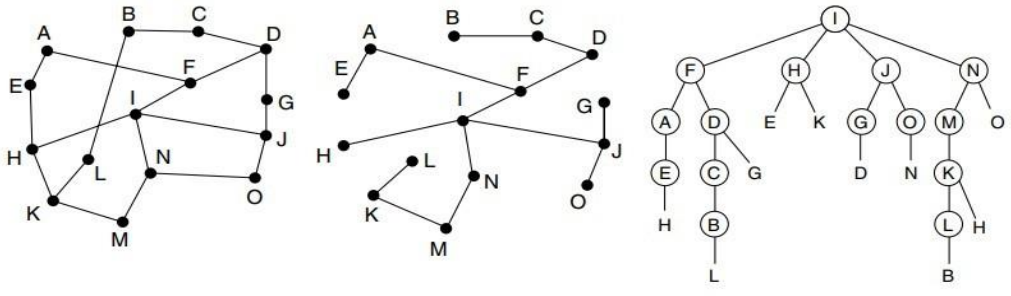
| Dest. | Line | Hops |
|---|---|---|
| 1A | – | – |
| 1B | 1B | 1 |
| 1C | 1C | 1 |
| 2 | 1B | 2 |
| 3 | 1C | 2 |
| 4 | 1C | 3 |
| 5 | 1C | 4 |

(c)

**Figure 5-14.** Hierarchical routing.

## 5.2.7 Broadcast Routing

- Broadcasting is the process of sending a packet to all destinations simultaneously, such as weather reports or stock market updates.
- This method requires no network features and requires a complete list of all destinations. However, this method is wasteful, slow, and requires a complete list of all hosts, making it not desirable in practice despite its widespread application.
- Multidestination routing is an improvement in network communication where each packet contains a list of destinations or a bit map indicating the desired destinations. When a packet arrives at a router, it checks all the destinations to determine the set of output lines needed.
- The router generates a new copy of the packet for each output line and includes only those destinations that will use the line. This partitions the destination set among the output lines, resulting in more efficient network bandwidth usage. However, this scheme still requires the source to know all destinations and requires router work.
- Flooding is a better broadcast routing technique that efficiently uses links with a simple decision rule at routers. Although it is not suitable for point-to-point communication, it is worth considering for broadcasting.
- Reverse path forwarding is an elegant and simple idea that checks if a broadcast packet arrived on the link normally used for sending packets toward the source. If so, the router forwards copies of the packet onto all links except the one it arrived on. If the packet arrived on a different link, it is discarded as a likely duplicate.



(a) (b) (c)

**Figure 5-15.** Reverse path forwarding. (a) A network. (b) A sink tree. (c) The tree built by reverse path forwarding.

- Reverse path forwarding is a network routing algorithm where router I sends packets to previously unvisited routes on the first hop. The second hop generates eight packets, two by each router that received a packet on the first hop.
- All eight packets arrive at previously unvisited routes, with five along the preferred line. The third hop generates six packets, with only three arriving on the preferred path. After five hops and 24 packets, broadcasting terminates, compared to four hops and 14 packets if the sink tree was followed exactly. Reverse path forwarding is efficient and easy to implement, sending the broadcast packet over each link only once in each direction, similar to flooding, without the need for routers to remember sequence numbers or list all destinations in the packet.
- The final broadcast algorithm enhances reverse path forwarding by using a sink tree or other spanning tree for router initiating the broadcast.
- A spanning tree is a subset of the network that includes all routers but has no loops. If each router knows which line belongs to the tree, it can copy an incoming broadcast packet onto all spanning tree lines except the one it arrived on.
- This method efficiently uses bandwidth, generating the minimum number of packets needed. However, each router must have knowledge of a spanning tree for the method to be applicable.
-

### 5.2.8 Multicast Routing

- Multicasting is a method used to send messages to large but small groups in a network. This is particularly useful for applications like multiplayer games or live sports events. Sending a distinct packet to each receiver is expensive unless the group is small.
- However, broadcasting a packet is wasteful if the group consists of numerous machines on a million-node network. Multicasting requires a routing algorithm to create and destroy groups and identify routers as members. Each group is identified by a multicast address, and routers know their group membership.
- Multicast routing schemes use spanning trees to deliver packets to members of a group efficiently, based on the group's density or sparsity. Broadcast is a good start for dense groups, as it efficiently gets the packet to all parts of the network. However, it may reach routers not members of the group, which is wasteful. Deering and Cheriton (1990) proposed pruning the broadcast spanning tree by removing links that do not lead to members, resulting in an efficient multicast spanning tree.
- For example, consider two groups, 1 and 2, with routers attached to hosts belonging to one or both groups. A spanning tree for the leftmost router can be used for broadcast but is overkill for multicast. A multicast spanning tree for the leftmost router sends packets only along this tree, which is more efficient than the broadcast tree due to the number of links. Different multicast groups have different spanning trees.
- The spanning tree can be pruned using link state routing, where each router is aware of the complete topology and can construct its own pruned tree for each sender to the group. This is an example of a link state protocol called MOSPF (Multicast OSPF).
- Distance vector routing follows a different pruning strategy, with the basic algorithm being reverse path forwarding. When a router receives a multicast message for a particular group, it responds with a PRUNE message, preventing the neighbor from

sending more multicasts. This recursive pruning of the spanning tree is also possible for routers with no group members among their hosts. DVMRP (Distance Vector Multicast Routing Protocol) is an example of this method.

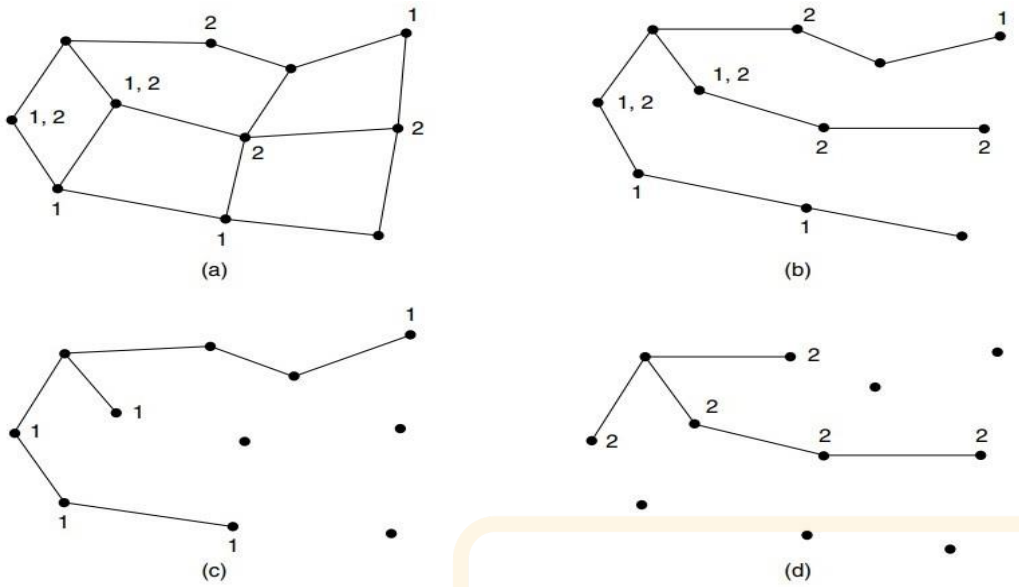



**Figure 5-16.** (a) A network. (b) A spanning tree for the leftmost router. (c) A multicast tree for group 1. (d) A multicast tree for group 2.

- Core-based trees are an alternative design for group routing, where routers agree on a root (core or rendezvous point) and build the tree by sending packets from each member to the root. The tree is the union of the paths traced by these packets. For group 1, a sender sends a packet to the core, which is forwarded down the tree. This performance optimization allows packets to be multicast without reaching the core before being forwarded up or down all branches.
- Shared trees are not optimal for all sources, as they can be inefficient depending on the location of the core and senders. However, they can be a significant savings in storage costs, messages sent, and computation. Each router maintains one tree per group, reducing the need for multiple trees. Additionally, routers not part of the tree do not support the group. This is why core-based trees are used for multicasting to sparse groups in the Internet, as part of popular protocols like PIM.

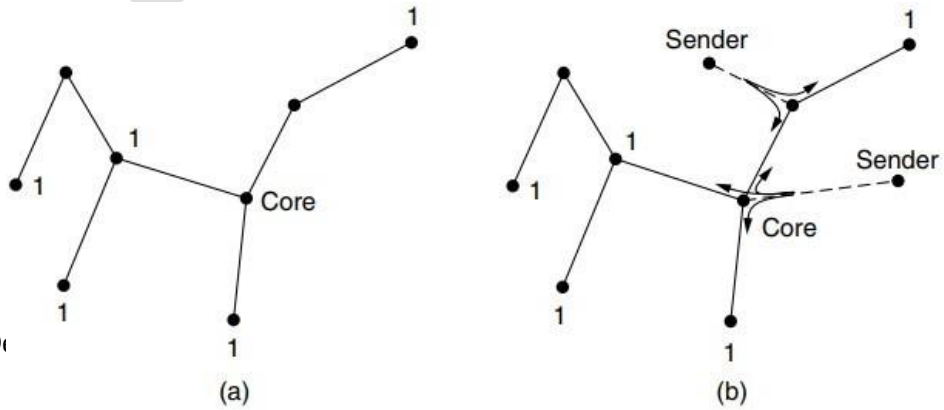



**Figure 5-17.** (a) Core-based tree for group 1. (b) Sending to group 1.

### 5.2.9 Anycast Routing

- Delivery models such as unicast, broadcast, and multicast are used to send packets to specific destinations. Anycast, a variant, delivers a packet to the nearest member of a group. Anycast routing schemes find these paths.
- This is useful when nodes provide services like time of day or content distribution, where the right information is received regardless of the contact node. Anycast is used in the Internet as part of DNS. Regular distance vector and link state routing can produce anycast routes.
- The distance vector routing procedure is used to assign addresses to group 1 members, resulting in nodes sending to the nearest instance of destination 1. This is because the routing protocol doesn't recognize multiple instances of destination 1, believing all instances of node 1 are the same.

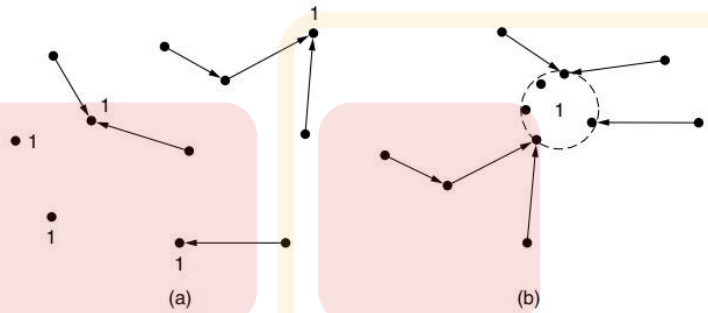



**Figure 5-18.** (a) Anycast routes to group 1. (b) Topology seen by the routing protocol.

### 5.2.10 Routing for Mobile Hosts

- Millions of people use computers on the go, with mobile hosts being either stationary or mobile. As people increasingly want to stay connected, networks must find mobile hosts to route packets.
- In a model where all hosts have a permanent home location and permanent home address, the routing goal is to send packets to mobile hosts using their fixed home addresses and efficiently reach them wherever they may be. The challenge is to find mobile hosts to ensure efficient communication and reach them efficiently.
- A different model for mobile network routing could involve recompiling routes as mobile hosts move and topology changes. This would reduce the burden of constantly computing new routes. Using home addresses can also help reduce this burden. Another option is to provide mobility above the network layer, as laptops acquire new network addresses when moved to new locations.
- This model allows a laptop to browse the web but requires a higher layer location service for other hosts to send packets. Connections cannot be maintained while the host is moving, so network-layer mobility is useful.
- Mobile routing in the Internet and cellular networks involves the mobile host indicating its current location to a home agent, who forwards packets to the mobile host's home location. This process is crucial for ensuring efficient communication and delivery of data. In a scenario where the mobile host is temporarily in San Diego, the home agent

acquires a local network address, known as a care of address.

- The mobile host sends a registration message to the home agent, which is a control message, not a data message. The sender then sends a data packet to the mobile host using its permanent address, which is routed by the network to the host's home location. In New York, the home agent intercepts the packet, wraps it with a new

header, and sends the bundle to the care of address. This tunneling mechanism is crucial in the Internet and cellular networks.

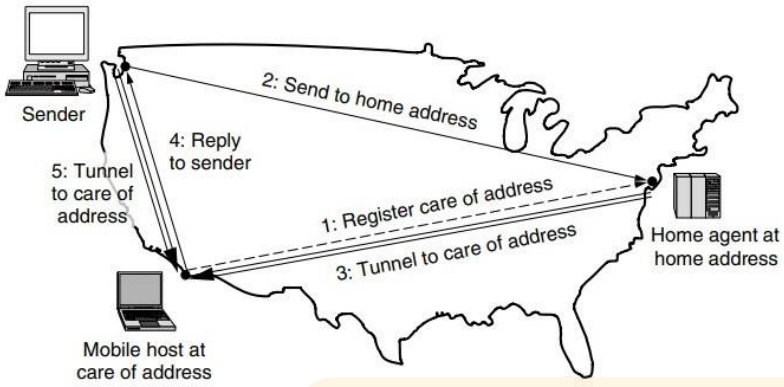



**Figure 5-19.** Packet routing for mobile hosts.

- The mobile routing system involves a triangle routing, where the mobile host retrieves the packet from the sender and sends its reply packet directly to the sender. This route may be circuitous if the remote location is far from the home location.
- The sender may learn the current care of address during step 4. Subsequent packets can be routed directly to the mobile host by tunneling them to the care of address, bypassing the home location entirely. If connectivity is lost, the home address can always be used to reach the mobile.
- Security is an important aspect, as messages may contain security information to check with cryptographic protocols. The scheme is modeled on IPv6 mobility, which is used in the Internet and IP-based cellular networks like UMTS. Extensions of the basic scheme support mobile networks without host work.

### 5.2.11 Routing in Ad Hoc Networks

- We have now seen how to do routing when the hosts are mobile but the routers are fixed. An even more extreme case is one in which the routers themselves are mobile. Among the possibilities are emergency workers at an earthquake site, military vehicles on a battlefield, a fleet of ships at sea, or a gathering of people with laptop computers in an area lacking 802.11
- In all these cases, and others, each node communicates wirelessly and acts as both a host and a router. Networks of nodes that just happen to be near each other are called ad hoc networks or MANETs (Mobile Ad hoc NETworks). Let us now examine them briefly. More information can be found in Perkins (2001).
- What makes ad hoc networks different from wired networks is that the topology is suddenly tossed out the window. Nodes can come and go or appear in new places at the drop of a bit. With a wired network, if a router has a valid path to some destination, that path continues to be valid barring failures, which are hopefully rare. With an ad hoc network, the topology may be changing all the time, so the desirability and even the validity of paths can change spontaneously without warning. Needless to say, these circumstances make routing in ad hoc networks more challenging than routing in their fixed counterparts.

- Numerous routing algorithms for ad hoc networks have been proposed, but their practical use is limited compared to mobile networks. One popular algorithm is AODV (Ad hoc On-demand Distance Vector), adapted for mobile environments with limited bandwidth

and battery lifetimes. It discovers and maintains routes, highlighting its potential in ad hoc networks.

- **Route Discovery :** The Ad Hoc Network (AODV) is a network algorithm that discovers routes to a destination on demand, saving time and effort when the topology changes before the route is used. The topology of an ad hoc network can be described by a graph of connected nodes, with two nodes connected if they can communicate directly using their radios. For simplicity, all connections are symmetric.
- The algorithm is based on a newly formed ad hoc network, where a process at node A sends a packet to node I. The algorithm maintains a distance vector table at each node, keyed by destination, providing information about the destination and its neighbors. If no entry is found for node I, the process must discover a route to it. This "on demand" approach makes the algorithm efficient and efficient.

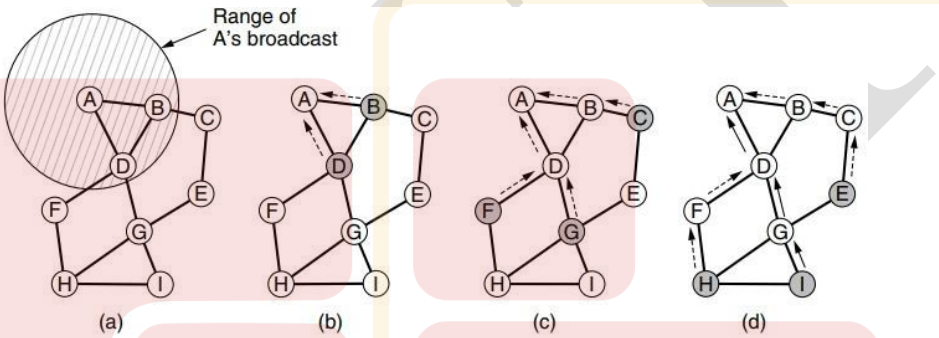



**Figure 5-20.** (a) Range of *A*'s broadcast. (b) After *B* and *D* receive it. (c) After *C*, *F*, and *G* receive it. (d) After *E*, *H*, and *I* receive it. The shaded nodes are new recipients. The dashed lines show possible reverse routes. The solid lines show the discovered route.

- A sends a ROUTE REQUEST packet to I, which is broadcasted using flooding. The packet is then rebroadcast to nodes F, G, C, H, E, and I. A sequence number is set at the source to remove duplicates during the flood.
- The request reaches node I, which constructs a ROUTE REPLY packet. This packet is unicast along the reverse of the path followed by the request. Intermediate nodes must remember the node that sent the request and increment a hop count as they forward the reply. The replies indicate which neighbor to use to reach the destination. Intermediate nodes G and D input the best route they hear into their routing tables. When the reply reaches A, a new route, ADGI, is created.
- The algorithm generates numerous broadcasts in large networks, even for nearby destinations. To reduce overhead, the scope of broadcasts is limited using the IP packet's Time to live field. If the field is 0, the packet is discarded. The route discovery process is modified by broadcasting a ROUTE REQUEST packet with a Time to live set to 1, and then sending more packets in increasingly wider rings, starting locally and then globally.
- **Route Maintenance :** The topology of a network can change spontaneously due to nodes moving or being switched off. The algorithm must handle this by broadcasting a Hello message to its neighbors, which is expected to respond. If no response is received, the

broadcaster knows that the neighbor has moved out of range or failed, and if a packet is sent to a non-responsive neighbor, it learns that the neighbor is no longer available.

- This information is used to purge routes that no longer work. Each node, N, tracks its active neighbors who have sent a packet for a possible destination during the last ΔT seconds. When any of N's neighbors becomes unreachable, it checks its routing table to see which routes have routes using the now-gone neighbor. The active neighbors inform each other recursively until all routes depending on the now-gone node are purged from all routing tables.
- Invalid routes are removed from the network, and senders can find new valid ones using the discovery mechanism. However, distance vector protocols can suffer from slow convergence or count-to-infinity problems after topology changes. To ensure rapid convergence, routes include a sequence number controlled by the destination, which increments every time a fresh ROUTE REPLY is sent. Senders request a fresh route by including the destination sequence number of the last used route in the ROUTE REQUEST.
- Intermediate nodes store routes with a higher sequence number or the few hops for the current sequence number. This on-demand protocol saves bandwidth and battery life compared to a standard distance vector protocol that broadcasts updates periodically.
- The text discusses ad hoc routing schemes, such as DSR (Dynamic Source Routing) and GPSR (Greedy Perimeter Stateless Routing). These schemes share route discovery and maintenance when routes overlap, allowing for resource savings. For example, if B wants to send packets to I, it will perform route discovery first, then reach D, which already has a route to I. The choice of protocol depends on the types of ad hoc networks that prove useful in practice.

## 5.3 Congestion Control Algorithms

- Too many packets present in (a part of) the network causes packet delay and loss that degrades performance. This situation is called **congestion**.
- The network and transport layers share responsibility for handling congestion. The network layer directly experiences congestion and must decide on handling excess packets.
- The most effective way to control congestion is to reduce the load placed by the transport layer on the network, requiring collaboration between the two layers.

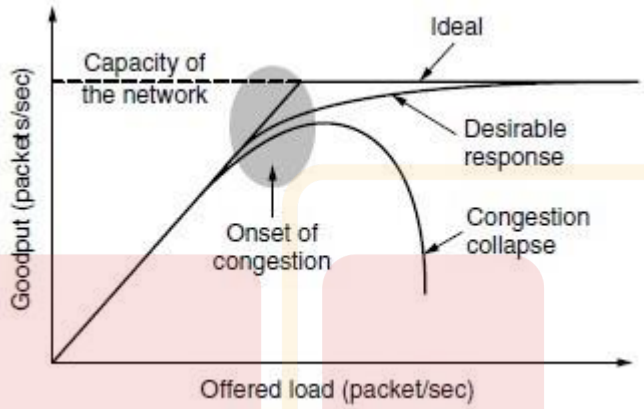



**Figure 3.1.** With too much traffic, performance drops sharply.

- The above figure depicts the onset of congestion, The network's carrying capacity is maintained when the number of packets sent is within its limits. If twice as many packets are sent, twice as many are delivered.
- However, as traffic loads approach the capacity, some packets are lost, consuming some of the capacity, causing the number of delivered packets to fall below the ideal curve, resulting in congested networks.
- A poorly designed network can experience congestion collapse, where performance decreases as load increases beyond capacity. This occurs when packets are delayed, making them no longer useful when they leave the network.
- In the early internet, packets spent waiting for backlogs could reach their maximum time. Senders re-transmitted delayed packets, wasting capacity. To capture these factors, the y-axis of Fig. 3.1 is given as **goodput**, which is the rate at which useful packets are delivered by the network.

- Congestion is a common issue in networks, causing queues and packet loss. Adding more memory can help, but it can worsen congestion as packets time out and duplicates are sent. Low-bandwidth links or routers that process packets slowly can also become congested.
- To improve congestion, traffic can be directed away from the bottleneck, but eventually, all regions will be congested, necessitating the need for faster network design.

- **Congestion control and flow contro**l are two distinct concepts in network management. Congestion control ensures the network can handle the offered traffic, involving all hosts and routers' behavior. Flow control focuses on the traffic between a sender and a receiver, ensuring a fast sender cannot transmit data faster than the receiver can absorb it.
- Congestion control is often confused with flow control, as the best way to handle both problems is to get the host to slow down. Understanding congestion control involves examining approaches at

different time scales, preventing congestion, and coping with it once it has set in.

### 5.3.1 Approaches to Congestion Control

1. The presence of congestion means that the load is (temporarily) greater than the resources (in a part of the network) can handle. Two solutions come to mind: **increase the resources or decrease the load.** As shown in Fig. 5-22, these solutions are usually applied on different time scales to either prevent congestion or react to it once it has occurred.

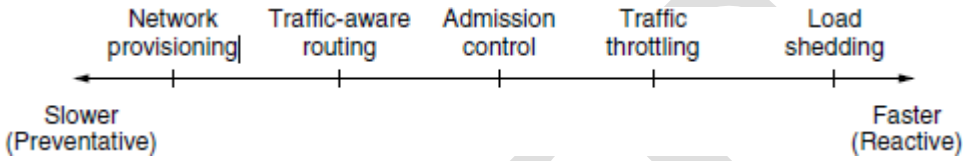



**Figure 3.2.** Timescales of approaches to congestion control.

2. To avoid congestion, build a network that matches the traffic it carries. Resources can be added dynamically during congestion, such as using spare routers or enabling lines.

3. Provisioning involves upgrading heavily utilized links and routers at the earliest opportunity. Routes can be tailored to changing traffic patterns, such as shifting traffic away from heavily used paths.

4. Traffic-aware routing allows mobile listeners to route packets around hotspots, Some local radio stations have helicopters flying around their cities to report on road congestion to make it possible for their mobile listeners to route their packets (cars) around hotspots. This is called **traffic-aware routing**. and splitting traffic across multiple paths is helpful.

5. However, sometimes it is not possible to increase capacity. The only way then to beat back the congestion is to decrease the load. In a virtual-circuit network, new connections can be refused if they would cause the network to become congested. This is called **admission control**.

6. At a fine level, when congestion is imminent the network can request traffic sources causing the problem to throttle their traffic to reduce congestion. To identify onset of congestion, routers can monitor average load, queuing delay or packet loss. Rising numbers indicate growing congestion.

7. To control sources, routers must participate in a feedback loop. The timescale needs careful adjustment:
   (a) If feedback is too frequent, the system will oscillate wildly.
   (b) If feedback is too slow, congestion control will react sluggishly.

8. Sending feedback messages during congestion adds more network load. When all else fails, network has to discard packets it cannot deliver, which is called load shedding. Good packet discard policies can help prevent congestion collapse.

### 5.3.2 Traffic-Aware Routing

1. The first approach we will examine is traffic-aware routing. The goal in taking load into account when computing routes is to shift traffic away from hotspots that will be the first places in the network to experience congestion.

2. The most direct way to do this is to set the link weight to be a function of the fixed link bandwidth and propagation delay plus the variable measured load or average queuing delay. Least-weight paths will then favor paths that are more lightly loaded, all else being equal.

3. Traffic-aware routing was used in the early Internet according to this model by Khanna and Zinky in 1989. However, there is a risk. Consider the network of Fig. 3.3, which is divided into two parts, East and West, connected by two links, CF and EI. Suppose that most of the traffic between East and West is using link CF, and, as a result, this link is heavily loaded with long delays. Including queueing delay in the weight used for the shortest path calculation will make EI more attractive. After the new routing tables have been installed, most of the East-West traffic will now go over EI, loading this link. Consequently, in the next update, CF will appear to be the shortest path. As a result, the routing tables may oscillate wildly, leading to erratic routing and many potential problems.

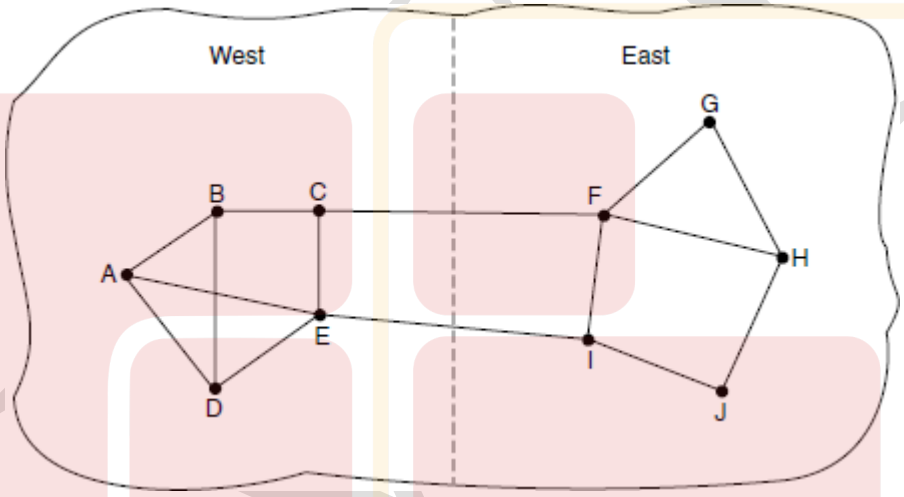



**Figure 3.3** A network in which the East and West parts are connected by two links.

4. If load is ignored and only bandwidth and propagation delay are considered, this problem does not occur. Attempts to include load but change weights within a narrow range only slow down routing oscillations. Two techniques can contribute to a successful solution. The first is multi path routing, in which there can be multiple paths from a source to a destination. In our example this means that the traffic can be spread across both of the East to West links.

5. The second one is for the routing scheme to shift traffic across routes slowly enough that it is able to converge, as in the scheme of Gallagher (1977). Given these difficulties, in the Internet routing protocols do not generally adjust their routes depending on the load. Instead, adjustments are made outside the routing protocol by slowly changing its inputs. This is called **Traffic engineering**.

### 5.3.3 Admission Control

1. Admission control is a technique used to prevent congestion in virtual circuit networks. The idea is to not establish a new virtual circuit if it would make the network congested. This is better than letting more circuits in when the network is already busy.

2. Determining when a new circuit will lead to congestion is the challenge. In telephone networks, it is straightforward since calls have fixed bandwidth. But in computer networks, virtual circuits have varying bandwidths.

3. So for admission control to work, the new circuit needs to provide some characterization of its traffic. Traffic can be described in terms of its rate and shape. But a simple description is difficult due to its bursty nature.

4. The average rate only tells half the story since traffic is often bursty. Bursty traffic like web traffic is harder to handle than streaming traffic with the same throughput since web traffic bursts are more likely to congest routers.

5. A commonly used method to capture this effect is the leaky bucket or token bucket model.
   The leaky bucket has two parameters:
   (a) The average rate which bounds the long term traffic rate
   (b) The burst size which bounds the maximum burst of traffic

6. The network can use traffic descriptions for admission control when deciding whether to admit new virtual circuits. One option is for the network to reserve enough capacity along circuit paths to prevent congestion. This guarantees quality of service for users.

7. Even without guarantees, the network can estimate how many circuits will fit within capacity without congestion using traffic descriptions. For example, 10 circuits at up to 10 Mbps passing through a 100 Mbps link can be admitted to avoid congestion. But this is wasteful since circuits may not transmit at full speed at the same time.

8. Measurements of past behavior and statistics can be used to estimate the right number of circuits to admit, trading better performance for acceptable risk.

9. Admission control can also be combined with traffic-aware routing by considering routes around traffic hotspots as part of the setup procedure. For example, consider the network illustrated in Fig. 3.4(a), in which two routers are congested, as indicated.

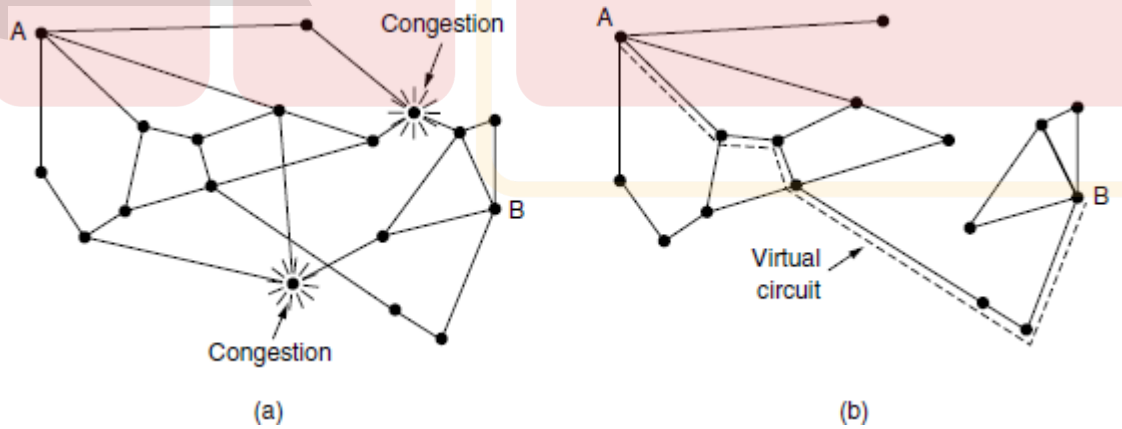



**Figure 3.4.** (a) A congested network. (b) The portion of the network that is not congested. A virtual circuit from *A* to *B* is also shown.

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

### 5.3.4 Traffic Throttling

Traffic Throttling is an approach used to avoid congestion. In networks and the internet, the senders try to send as much traffic as possible as the network can readily deliver. In a network when congestion is approaching it should tell the senders of packets to slow down them. Traffic Throttling can be used in virtual circuit networks and datagram networks.

Let us now look at some approaches to throttling traffic that can be used in both datagram networks and virtual-circuit networks. Each approach must solve two problems.

**Problem 1-** The router must be able to determine when the congestion is approaching. It must identify the congestion before it has arrived. For this, each router used in the network must continuously check for all the resources and their activities in the network. Router can continuously monitor using three possibilities. They are:

- Using output links
- To buffer the most useful and priority-based packets inside the router
- The total number of packets that are lost because of insufficient buffering

**Problem 2-** The second problem is that the router must send the feedback on time to the senders that are creating congestion. To deliver this feedback, the router must identify the senders properly. The router needs to send them warning efficiently without sending more packets in an already **congested** network. Different feedback mechanisms are used for solving this problem.

**Feedback Mechanisms**

**1. Choke packets**

Choke packets are a mechanism where the router directly sends the choked packet back to its sender or host. The header bit of the original packet is turned on so that it will not be able to generate any choke packet. At the time of congestion to decrease the load router will send back only the choked packets at a lower rate. In the case of datagram networks randomly, the packets are selected therefore it leads to more choked packets. The below diagram describes the choke packets approach.

**2. Explicit Congestion Notification**

In the explicit congestion notification approach the router does not send extra packets to the host but sets a bit of any one of the packet headers to inform that the network has approached with

congestion. When any packet is delivered in the network the destination sends a reply packet to the sender informing that congestion has occurred. In the case of choke packets, the sender then throttles its transmission. The below diagram describes the explicit congestion notification.

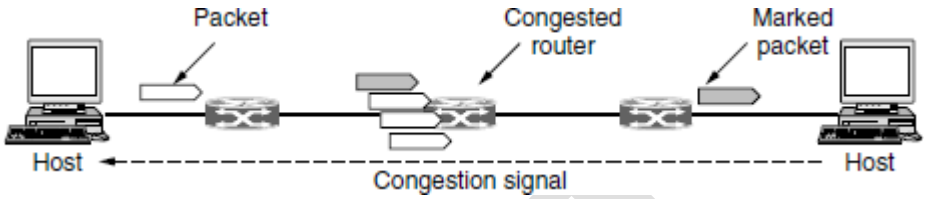



**Fig 3.5:** Explicit congestion notification

### 3. Hop-by-Hop Backpressure

After the congestion has been signaled still due to a slow signal many packets are received from the long distances. The choke packets have an effect at every step and each router requires more buffers. The main aim of this Hop-by-Hop Backpressure technique is to provide faster relief at the point of congestion in the network. This technique propagates in the opposite direction of the data and is majorly used in virtual circuits. The below diagram describes Hop-by-Hop Backpressure in detail.

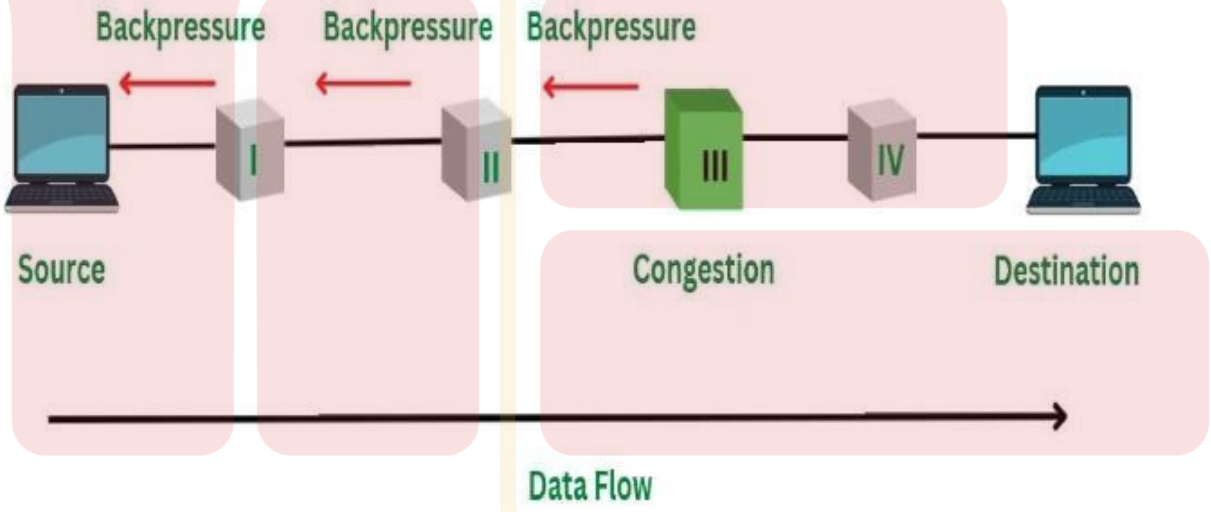



Fig 3.6: Hop by Hop Backpressure

## 5.3.5 Load Shedding

1. Load shedding is one of the techniques used for congestion control. A network router consists of a buffer. This buffer is used to store the packets and then route them to their destination. Load shedding is defined as an approach of discarding the packets when the buffer is full according to the strategy implemented in the data link layer. The selection of packets to discard is an important task. Many times packets with less importance and old packets are discarded.

2. The term comes from the world of electrical power generation, where it refers to the practice of utilities intentionally blacking out certain areas to save the entire grid from collapsing on hot summer days when the demand for electricity greatly exceeds the supply.

3. **Selection of Packets to be Discarded:** In the process of load shedding the packets need to be discarded in order to avoid congestion. Therefore which packet needs to be discarded is a question. Random early detection is one of the approach used to discard the packets.

**Random early detection**

4. This situation can be exploited to help reduce congestion. By having routers drop packets early, before the situation has become hopeless, there is time for the source to take action before it is too late. A popular algorithm for doing this is called **RED** (**Random Early Detection**)

5. To determine when to start discarding, routers maintain a running average of their queue lengths. When the average queue length on some link exceeds a threshold, the link is said to be congested and a small fraction of the packets are dropped at random.

6. Picking packets at random makes it more likely that the fastest senders will see a packet drop; this is the best option since the router cannot tell which source is causing the most trouble in a datagram network. The affected sender will notice the loss when there is no acknowledgement, and then the transport protocol will slow down. The lost packet is thus delivering the same message as a choke packet, but implicitly, without the router sending any explicit signal.

7. RED routers improve performance compared to routers that drop packets only when their buffers are full, though they may require tuning to work well. For example, the ideal number of packets to drop depends on how many senders need to be notified of congestion.

8. However, ECN is the preferred option if it is available. It works in exactly the same manner, but delivers a congestion signal explicitly rather than as a loss; RED is used when hosts cannot receive explicit signals.

## 5.4 QUALITY OF SERVICE

- The techniques we looked at in the previous sections are designed to reduce congestion and improve network performance.
- Some applications and customers require stronger performance guarantees than what can be achieved under typical circumstances.

### Overprovisioning

- An easy solution for ensuring good quality of service is to build a network with sufficient capacity to handle any anticipated traffic.
- This approach, known as overprovisioning, allows the network to carry application traffic with minimal loss and low latency, assuming a decent routing scheme.
- The result is optimal performance, and the telephone system is an example of a somewhat overprovisioned system, evident in the near-instant dial tone availability.

### Drawbacks of Overprovisioning:

- The primary drawback of overprovisioning is its high cost, essentially solving the problem by investing more resources.
- Quality of service mechanisms offer an alternative by enabling a network with less capacity to meet application requirements at a lower cost.
- Overprovisioning relies on expected traffic, and significant issues may arise if traffic patterns change unexpectedly.

### Benefits of Quality-of-Service Mechanisms:

- Quality of service mechanisms allow a network with less capacity to effectively meet application requirements at a lower cost compared to overprovisioning.
- These mechanisms enable the network to uphold performance guarantees even during traffic spikes, although this may involve rejecting some requests.
- Four issues must be addressed to ensure quality of service:
    1. What applications need from the network.
    2. How to regulate the traffic that enters the network.
    3. How to reserve resources at routers to guarantee performance.
    4. Whether the network can safely accept more traffic.

## 5.4.1 Application Requirements

- **Definition of Flow:**
- A stream of packets moving from a source to a destination is termed a flow (Clark, 1988).
- Flows can encompass all packets in a connection in a connection-oriented network or all packets sent from one process to another in a connectionless network.
- **Characterization of Flow Needs:**
    - Each flow's requirements can be described by four primary parameters: bandwidth, delay, jitter, and loss.
    - These parameters collectively determine the Quality of Service (QoS) necessary for the flow.
- **Common Applications and Network Requirements:**
    - Fig. 5-27 lists various common applications and the level of stringency in their network requirements.

- Notably, network requirements are less demanding when an application can enhance the service provided by the network.

- **Example Contrasts:**
  - Networks do not necessarily need to be lossless for reliable file transfer, as some loss can be rectified with retransmissions.
  - Delivery of packets with identical delays is not crucial for audio and video playout, as some jitter can be smoothed by buffering packets at the receiver.

- **Bandwidth Requirements:**
  - Applications exhibit varying bandwidth needs.
  - Email, audio in all forms, and remote login have relatively low bandwidth requirements.
  - File sharing and video in all forms demand a significant amount of bandwidth.
- **Delay Sensitivity:**
  - File transfer applications, including email and video, are not highly sensitive to delay.
  - Interactive applications like Web surfing and remote login are more delay-sensitive.
  - Real-time applications such as telephony and videoconferencing have strict delay requirements.
- **Jitter and its Impact:**
  - Jitter, the variation in delay or packet arrival times, is crucial.
  - Email, audio, and file transfer are generally not sensitive to jitter.
  - Remote login may be affected by jitter, causing bursts of updates on the screen.
  - Video and audio are extremely sensitive to jitter; even small variations can have a significant impact.
- **Loss Tolerance:**
  - The first four applications (email, audio, file transfer, and video) have more stringent loss requirements, requiring all bits to be delivered correctly.
  - Retransmissions are typically used to achieve this goal.
  - Audio and video applications can tolerate some lost packets without retransmission, as users may not notice short pauses or occasional skipped frames.
- **Network Support for QoS:**
  - Networks may support different categories of Quality of Service (QoS) to accommodate various applications.
  - ATM networks, once part of a grand networking vision, provide an example of supporting different QoS categories but have become a niche technology.
  - Networks may support different categories of QoS.
    1. Constant bit rate (e.g., telephony).
    2. Real-time variable bit rate (e.g., compressed videoconferencing).
    3. Non-real-time variable bit rate (e.g., watching a movie on demand).
    4. Available bit rate (e.g., file transfer).

## 5.4.2 Traffic Shaping

- **Traffic Characterization:**
  - Before a network can make Quality of Service (QoS) guarantees, it needs to know the nature of the traffic being guaranteed.

- In the telephone network, the characterization is straightforward, such as a voice call needing 64 kbps with specific sampling intervals.

- **Bursty Nature of Data Networks:**
  - Traffic in data networks is often bursty due to variations in traffic rates (e.g., compressed videoconferencing), user interactions (e.g., browsing new web pages), and task switching by computers.
- **Traffic Shaping:**
  - Traffic shaping is a technique that regulates the average rate and burstiness of data flow entering the network.
  - It allows applications to transmit diverse traffic patterns while providing a simple and useful way to describe these patterns to the network.
  - The customer and provider agree on a traffic pattern (shape) for a flow, often outlined in a Service Level Agreement (SLA) for long-term commitments.
- **SLA and Traffic Monitoring:**
  - The Service Level Agreement (SLA) outlines the agreed-upon traffic patterns between the customer and provider.
  - Traffic shaping, when adhered to by the customer, helps reduce congestion and ensures the network can fulfill its promises.
  - Traffic policing involves monitoring a traffic flow to verify adherence to the agreement, potentially dropping excess packets or marking them with lower priority.
- **Importance of Shaping and Policing:**
  - Shaping and policing are critical for real-time data, such as audio and video connections, which have stringent quality-of-service requirements.
  - While less important for peer-to-peer transfers that consume all available bandwidth, they are crucial for maintaining QoS commitments.

## Leaky and Token Buckets

- Now we will look at a more general way to characterize traffic, with the leaky bucket and token bucket algorithms. The formulations are slightly different but give an equivalent result.
- Try to imagine a bucket with a small hole in the bottom, as illustrated in Fig. 5-28(b). No matter the rate at which water enters the bucket, the outflow is at a constant rate, R, when there is any water in the bucket and zero when the bucket is empty. Also, once the bucket is full to capacity B, any additional water entering it spills over the sides and is lost.

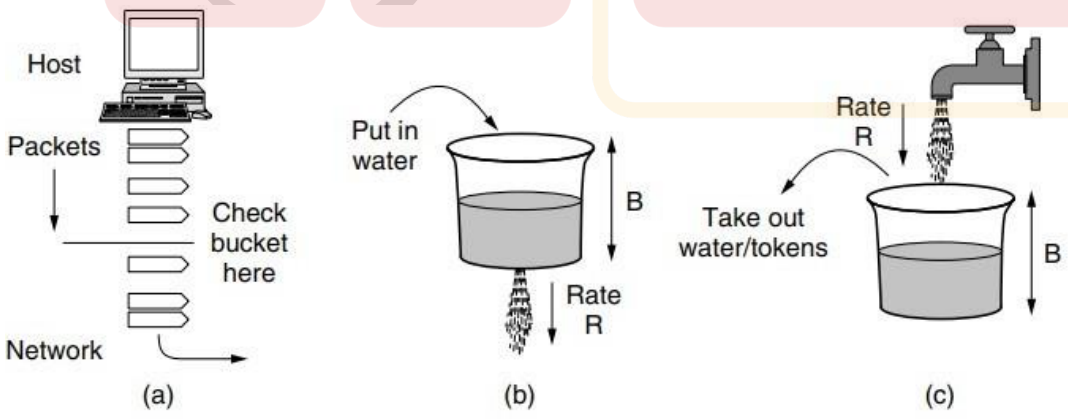



**Figure 5-28.** (a) Shaping packets. (b) A leaky bucket. (c) A token bucket.

- This bucket can be used to shape or police packets entering the network, as shown in Fig. 5-28(a). Conceptually, each host is connected to the network by an interface containing a leaky bucket. To send a packet into the network, it must be possible to put more water into the bucket. If a packet arrives when the bucket is full, the packet must either be queued until enough water leaks out to hold it or be discarded.
- The former might happen at a host shaping its traffic for the network as part of the operating system. The latter might happen in hardware at a provider network interface that is policing traffic entering the network. This technique was proposed by Turner (1986) and is called the leaky bucket algorithm.
- A different but equivalent formulation is to imagine the network interface as a bucket that is being filled, as shown in Fig. 5-28(c). The tap is running at rate R and the bucket has a capacity of B, as before. Now, to send a packet we must be able to take water, or tokens, as the contents are commonly called, out of the bucket (rather than putting water into the bucket).
- No more than a fixed number of tokens, B, can accumulate in the bucket, and if the bucket is empty, we must wait until more tokens arrive before we can send another packet. This algorithm is called the token bucket algorithm. Leaky and token buckets limit the long-term rate of a flow but allow short term bursts up to a maximum regulated length to pass through unaltered and without suffering any artificial delays.
- Large bursts will be smoothed by a leaky bucket traffic shaper to reduce congestion in the network. As an example, imagine that a computer can produce data at up to 1000 Mbps (125 million bytes/sec) and that the first link of the network also runs at this speed.
- The pattern of traffic the host generates is shown in Fig. 5-29(a). This pattern is bursty. The average rate over one second is 200 Mbps, even though the host sends a burst of 16,000 KB at the top speed of 1000 Mbps (for 1/8 of the second).

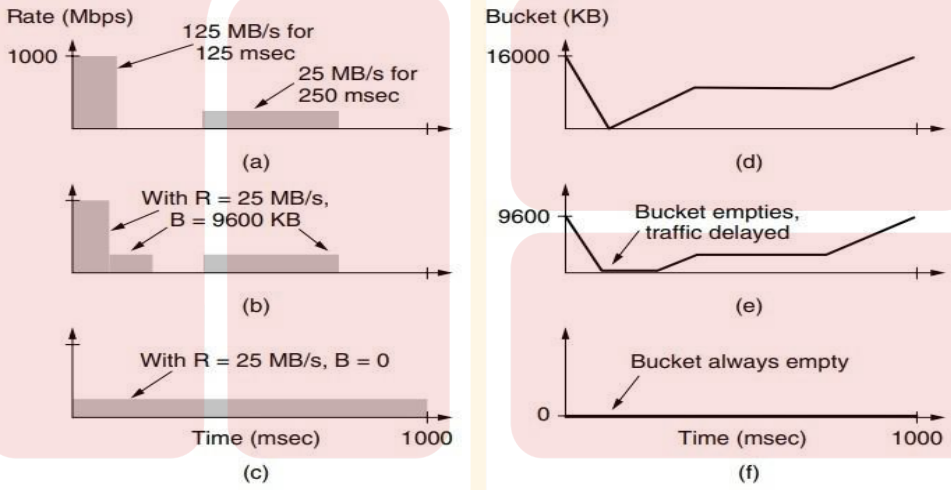



**Figure 5-29.** (a) Traffic from a host. Output shaped by a token bucket of rate 200 Mbps and capacity (b) 9600 KB and (c) 0 KB. Token bucket level for shaping with rate 200 Mbps and capacity (d) 16,000 KB, (e) 9600 KB, and (f) 0 KB.

- Now suppose that the routers can accept data at the top speed only for short intervals, until their buffers fill up. The buffer size is 9600 KB, smaller than the traffic burst. For long intervals, the routers work best at rates not exceeding 200 Mbps (say, because this is all the bandwidth given to the customer). The implication is that if traffic is sent in this pattern, some of it will be dropped in the network because it does not fit into the buffers at routers.

- To avoid this packet loss, we can shape the traffic at the host with a token bucket. If we use a rate, R, of 200 Mbps and a capacity, B, of 9600 KB, the traffic will fall within what the network can handle. The output of this token bucket is shown in Fig. 5-29(b).

- The host can send full throttle at 1000 Mbps for a short while until it has drained the bucket. Then it has to cut back to 200 Mbps until the burst has been sent. The effect is to spread out the burst over time because it was too large to handle all at once. The level of the token bucket is shown in Fig. 5- 29(e).
- Fig. 5-29(c) shows the output of a token bucket with R = 200 Mbps and a capacity of 0. This is the extreme case in which the traffic has been completely smoothed. No bursts are allowed, and the traffic enters the network at a steady rate. The corresponding bucket level, shown in Fig. 5-29(f), is always empty. Traffic is being queued on the host for release into the network and there is always a packet waiting to be sent when it is allowed
- Calculating the length of the maximum burst (until the bucket empties) is slightly tricky. It is longer than just 9600 KB divided by 125 MB/sec because while the burst is being output, more tokens arrive. If we call the burst length S sec., the maximum output rate M bytes/sec, the token bucket capacity B bytes, and the token arrival rate R bytes/sec, we can see that an output burst contains a maximum of B + RS bytes. We also know that the number of bytes in a maximum speed burst of length S seconds is MS. Hence, we have

$$B + RS = MS$$

- We can solve this equation to get $S = B/(M - R)$. For our parameters of B = 9600 KB, M = 125 MB/sec, and R = 25 MB/sec, we get a burst time of about 94 msec. A potential problem with the token bucket algorithm is that it reduces large bursts down to the long-term rate R. Using all of these buckets can be a bit tricky. When token buckets are used for traffic shaping at hosts, packets are queued and delayed until the buckets permit them to be sent. When token buckets are used for traffic policing at routers in the network, the algorithm is simulated to make sure that no more packets are sent than permitted. Nevertheless, these tools provide ways to shape the network traffic into more manageable forms to assist in meeting quality-of-service requirements.

### 5.4.3 Packet Scheduling

- **Regulating Traffic Shape:**
  - Regulating the shape of offered traffic is a crucial step.
  - However, providing a performance guarantee requires reserving sufficient resources along the route that packets take through the network.
- **Consistent Packet Routing:**
  - Assumption: For performance guarantees, packets of a flow must follow the same route.
  - Randomly spraying packets over routers makes it challenging to ensure any guarantees.
- **Virtual Circuit Setup:**
  - To address routing consistency, something akin to a virtual circuit must be established from the source to the destination.
  - All packets belonging to a specific flow must follow this established route.
- **Packet Scheduling Algorithms:**

- Algorithms that allocate router resources among packets of a flow and manage resources between competing flows are known as packet scheduling algorithms.

- These algorithms play a crucial role in ensuring that resources are appropriately distributed to meet performance guarantees.

**Three different kinds of resources can potentially be reserved for different flows:**

- **Bandwidth Reservation:**
  - Bandwidth is a critical resource.
  - If a flow requires 1 Mbps and the outgoing line has a capacity of 2 Mbps, reserving bandwidth means not oversubscribing the output line.
- **Buffer Space Allocation:**
  - Buffer space is another crucial resource.
  - Packets arriving at a router are buffered until they can be transmitted on the chosen outgoing line.
  - Buffers absorb small bursts of traffic as flows contend with each other.
  - For quality of service, some buffers might be reserved for specific flows, preventing competition with other flows for buffer space.
  - Maintaining a reserve ensures that a buffer is available when a flow needs it, up to a specified maximum value.
- **CPU Cycle Management:**
  - CPU cycles in routers are a potential scarce resource.
  - Processing a packet consumes router CPU time, limiting the number of packets the router can handle per second.
  - Certain packets, such as ICMP packets, may require more CPU processing.
  - Avoiding CPU overload is essential to ensure timely processing of packets, particularly those that demand additional computational resources.

## Packet Scheduling Algorithms:

- Packet scheduling algorithms allocate bandwidth and other router resources by determining which buffered packets to send on the output line next.

## FIFO (First-In First-Out) Scheduling:

- FIFO is the most straightforward scheduler.
- Each router maintains queues for each output line, buffering packets until they can be sent.
- Packets are sent in the same order they arrived, following the First-In First-Out (FIFO) or First-Come First-Serve (FCFS) principle.

## Tail Drop in FIFO Routers:

- FIFO routers often drop newly arriving packets when the queue is full.
- This behaviour is known as tail drop, where the newly arrived packet is dropped since it would have been placed at the end of the full queue.

- Various scheduling algorithms create different opportunities for deciding which packet to drop when buffers are full.

## Challenges of FIFO for Quality of Service:

- FIFO scheduling is simple but not ideal for providing good quality of service.
- In scenarios with multiple flows, an aggressive sender in the first flow can adversely impact the performance of other flows.
- Aggressive senders can hog router capacity, starving other flows and reducing their quality of service.
- The order-of-arrival processing in FIFO can lead to delays for packets of other flows, exacerbating the impact of aggressive senders.

- Many packet scheduling algorithms have been devised that provide stronger isolation between flows and thwart attempts at interference (Bhatti and Crowcroft, 2000). One of the first ones was the fair queueing algorithm devised by Nagle (1987). When the line becomes idle, the router scans the queues round-robin, as shown in Fig. 5-30.
- It then takes the first packet on the next queue. In this way, with n hosts competing for the output line, each host gets to send one out of every n packets. It is fair in the sense that all flows get to send packets at the same rate. Sending more packets will not improve this rate. Although a start, the algorithm has a flaw: it gives more bandwidth to hosts that use large packets than to hosts that use small packets. Demers et al suggested an improvement in which the round-robin is done in such a way as to simulate a byte-by-byte round-robin, instead of a packet-by-packet round-robin.
- The trick is to compute a virtual time that is the number of the round at which each packet would finish being sent. Each round drains a byte from all of the queues that have data to send. The packets are then sorted in order of their finishing times and sent in that order. This algorithm and an example of finish times for packets arriving in three flows are illustrated in Fig. 5-31. If a packet has length L, the round at which it will finish is simply L rounds after the start time. The start time is either the finish time of the previous packet, or the arrival time of the packet, if the queue is empty when it arrives

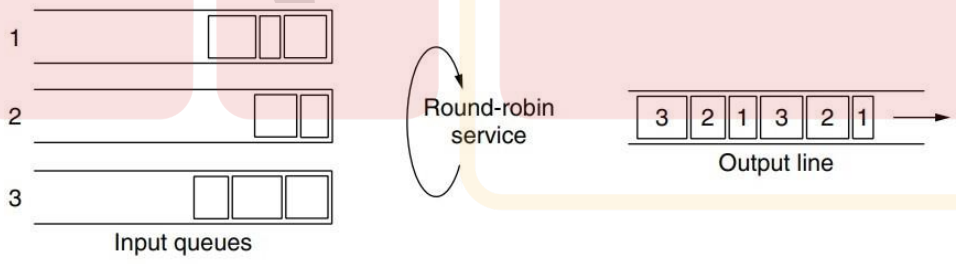
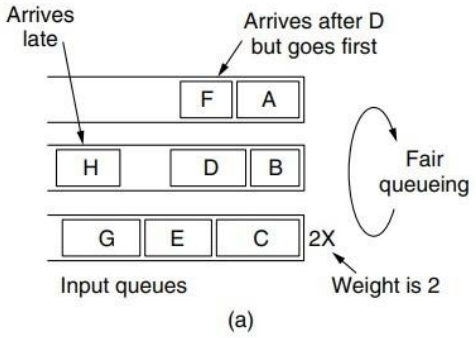



**Figure 5-30.** Round-robin fair queueing.




| Packet | Arrival time | Length | Finish time | Output order |
|---|---|---|---|---|
| A | 0 | 8 | 8 | 1 |
| B | 5 | 6 | 11 | 3 |
| C | 5 | 10 | 10 | 2 |
| D | 8 | 9 | 20 | 7 |
| E | 8 | 8 | 14 | 4 |
| F | 10 | 6 | 16 | 5 |
| G | 11 | 10 | 19 | 6 |
| H | 20 | 8 | 28 | 8 |

(b)

**Figure 5-31.** (a) Weighted Fair Queueing. (b) Finishing times for the packets.

- From the table in Fig. 5-32(b), and looking only at the first two packets in the top two queues, packets arrive in the order A, B, D, and F. Packet A arrives at round 0 and is 8 bytes long, so its finish time is round 8. Similarly, the finish time for packet B is 11. Packet D arrives while B is being sent. Its finish time is 9 byte-rounds after it starts when B finishes, or 20. Similarly, the finish time for F is 16. In the absence of new arrivals, the relative sending order is A, B, F, D, even though F arrived after D. It is possible that another small packet will arrive on the top flow and obtain a finish time before D. It will only jump ahead of D if the transmission of that packet has not started
- One shortcoming of this algorithm in practice is that it gives all hosts the same priority. In many situations, it is desirable to give, for example, video servers more bandwidth than, say, file servers. This is easily possible by giving the video server two or more bytes per round. This modified algorithm is called WFQ (Weighted Fair Queueing). Letting the number of bytes per round be the weight of a flow, W, we can now give the formula for computing the finish time:

$$Fi = \max(Ai, Fi - 1) + Li / W$$

- where Ai is the arrival time, Fi is the finish time, and Li is the length of packet i. The bottom queue of Fig. 5-31(a) has a weight of 2, so its packets are sent more quickly as you can see in the finish times given in Fig. 5-31(b).
- As a final example of a scheduler, packets might carry timestamps and be sent in timestamp order. Clark et al. (1992) describes a design in which the timestamp records how far the packet is behind or ahead of schedule as it is sent through a sequence of routers on the path. Packets that have been queued behind other packets at a router will tend to be behind schedule, and the packets that have been serviced first will tend to be ahead of schedule. Sending packets in order of their timestamps has the beneficial effect of speeding up slow packets while at the same time slowing down fast packets. The result is that all packets are delivered by the network with a more consistent delay.

### 5.4.4 Admission Control

- **QoS Elements and Admission Control:**
  - QoS guarantees are established through the process of admission control.
  - Admission control was initially used to control congestion, providing a weak

performance guarantee.

- The network decides to accept or reject the flow based on its capacity and commitments to other flows.

- **Reservations and Path Considerations:**
  - Reservations must be made at all routers along the route to prevent congestion.
  - Some routing algorithms send all traffic over the best path, potentially causing rejection of flows if there's insufficient spare capacity.
  - QoS routing and splitting traffic over multiple paths are techniques to accommodate QoS guarantees for new flows.
- **Complexity in Acceptance or Rejection Decision:**
  - The decision to accept or reject a flow involves more than comparing requested resources with router excess capacity.
  - Applications may not have knowledge of buffers or CPU cycles, necessitating a different way to describe flows and translate descriptions to router resources.
- **Tolerance Levels and Guarantees:**
  - Some applications are more tolerant of occasional missed deadlines than others.
  - Applications must choose between hard guarantees or behavior that holds most of the time.
  - Hard guarantees are expensive due to constraining worst-case behavior, so many applications are satisfied with guarantees for most packets.
- **Negotiation of Flow Parameters:**
  - Some applications may be willing to adjust flow parameters, while others may not..
  - Properties like the number of pixels per frame and audio bandwidth may be negotiable for certain applications.
- **Importance of Accurate Flow Description:**
  - Due to the involvement of multiple parties in flow negotiation (sender, receiver, routers along the path), accurate flow description is crucial.
  - Flows need to be described in terms of specific negotiable parameters.
- **Flow Specification:**
  - A set of parameters that describe a flow is termed a flow specification.
  - The sender, such as a video server, typically generates a flow specification proposing desired parameters.
  - As the specification travels along the route, each router may modify parameters as necessary, but modifications can only reduce the flow, not increase it.
- **Parameter Modifications:**
  - Each router along the path examines and modifies the flow specification.
  - Modifications are allowed to decrease the flow, such as lowering the data rate, but not to increase it.

- **Establishment of Parameters:**
  - When the flow specification reaches the destination, the negotiated parameters can be established.
- **Queueing Delay and Burst Size:**
  - The largest queueing delay experienced by a flow is a function of the burst size of the token bucket.
  - In cases of smooth traffic without bursts, packets are drained from the router as quickly as they arrive, resulting in no queueing delay (ignoring packetization effects).
  - Conversely, if traffic is saved up in bursts, the maximum queueing delay occurs when a maximum-size burst arrives at the router all at once. The maximum delay is the time taken to drain this burst at the guaranteed bandwidth, or $B/R$ (ignoring packetization effects).
- **Guarantees and Token Buckets:**
  - Guarantees provided by token buckets are hard guarantees.
  - Token buckets limit the burstiness of the traffic source.
  - Fair queueing isolates the bandwidth allocated to different flows, ensuring that each flow meets its bandwidth and delay guarantees regardless of the behaviour of other competing flows at the router.
- **Isolation of Flows:**
  - Competing flows at the router cannot break the guarantee even if they save up traffic and send it all at once.
  - The combination of token buckets and fair queueing ensures that each flow's guarantees remain intact.
- **Path Through Multiple Routers:**
  - The guarantees hold for a path through multiple routers in any network topology.
  - Each flow receives a minimum bandwidth at each router, as guaranteed.
  - Maximum delay for each flow is ensured, even in the worst-case scenario where a burst of traffic hits the first router and competes with other flows. The burst's delay is at most $D$, and this delay smooths the burst, preventing further queueing delays at later routers.

### 5.4.5 Integrated Services

- **Timeframe and Efforts (1995-1997):**
  - Between 1995 and 1997, the Internet Engineering Task Force (IETF) dedicated substantial effort to creating an architecture for streaming multimedia.
  - This initiative resulted in over two dozen RFCs, starting with RFCs 2205–2212.
  - The overarching name for this work is "integrated services," addressing both unicast and multicast applications.
- **Integrated Services Aimed at Unicast and Multicast:**
  - The integrated services architecture was designed to cater to both unicast and multicast applications.
  - Unicast example: a single user streaming a video clip from a news site.
  - Multicast example: digital television stations broadcasting their programs as IP packet streams to many receivers at various locations.
- **Focus on Multicast:**
  - The discussion will concentrate on multicast, considering unicast as a special case of multicast.
- **Dynamic Group Membership in Multicast:**
  - In many multicast applications, groups can dynamically change membership.
  - Examples include people joining and leaving a video conference or switching between different channels (e.g., a soap opera or the croquet channel).
- **Challenges with Bandwidth Reservation for Dynamic Groups:**
  - Traditional approaches involving senders reserving bandwidth in advance do not work well in dynamic multicast scenarios.
  - Tracking entries and exits of the audience for each sender becomes impractical, especially for systems transmitting television to a large number of subscribers

**RSVP—The Resource reservation Protocol**

- **Visible Component of Integrated Services Architecture:**
  - The primary component of the integrated services architecture that is visible to network users is RSVP (Resource Reservation Protocol).
  - RSVP is detailed in RFCs 2205–2210.
  - Other protocols are employed for the actual transmission of data.
  -
- **Functionality of RSVP:**

- RSVP is responsible for making reservations in the network.

- It supports multiple senders transmitting to multiple groups of receivers.
- Allows individual receivers to switch channels freely.
- Optimizes bandwidth utilization and simultaneously eliminates congestion issues.

**Simplest Form of RSVP:**

- In its simplest form, RSVP utilizes multicast routing with spanning trees.
- Each multicast group is assigned a group address.
- To send to a group, a sender includes the group's address in its packets.
- The standard multicast routing algorithm constructs a spanning tree that covers all members of the group.
- RSVP doesn't define the multicast routing algorithm but adds a little extra information periodically multicast to the group. This information instructs routers along the tree to maintain specific data structures in their memories.

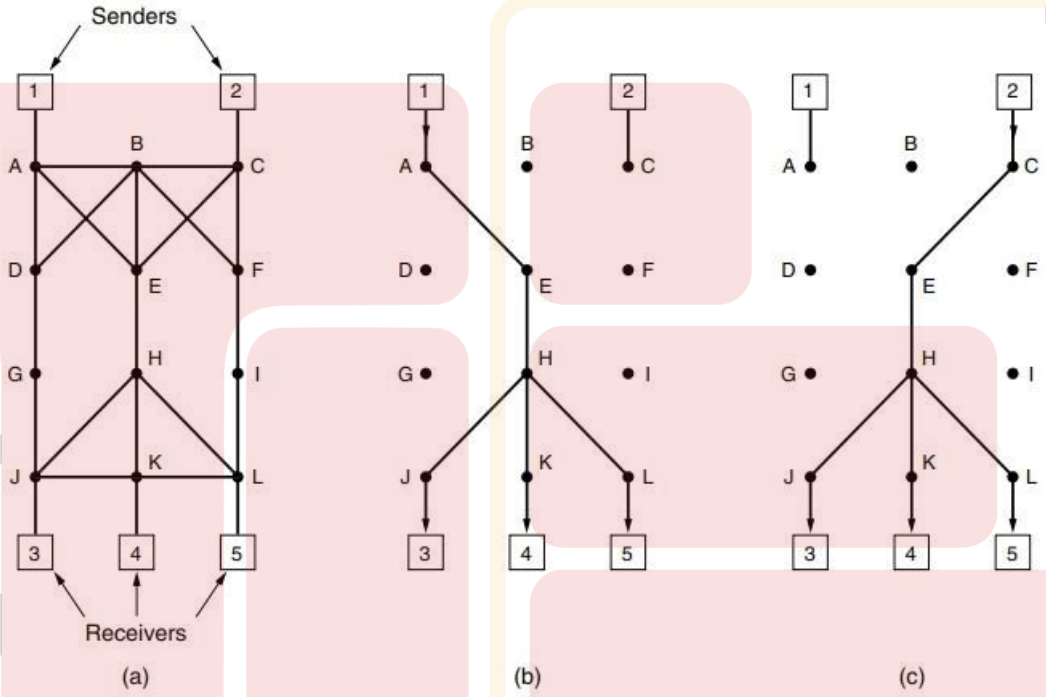



**Figure 5-34.** (a) A network. (b) The multicast spanning tree for host 1. (c) The multicast spanning tree for host 2.

As an example, consider the network of Fig. 5-34(a). Hosts 1 and 2 are multicast senders, and hosts 3, 4, and 5 are multicast receivers. In this example, the senders and receivers are disjoint, but in general, the two sets may overlap. The multicast trees for hosts 1 and 2 are shown in Fig. 5-34(b) and Fig. 5-34(c), respectively.

- **Improving Reception and Avoiding Congestion:**
  - To enhance reception quality and prevent congestion, any receiver within a group has the capability to send a reservation message up the tree to the sender.
- **Reservation Message Propagation:**

- The reservation message is propagated using the reverse path forwarding algorithm.
- The reverse path forwarding algorithm is a mechanism previously discussed.

- **Failure Reporting for Insufficient Bandwidth:**
  - If there is insufficient bandwidth available at any point along the path, the router reports back a failure in the reservation process.
- **Reservation Completion:**
  - By the time the reservation message reaches the source (sender), bandwidth has been successfully reserved all the way from the sender to the receiver who initiated the reservation request.
- **Spanning Tree Usage:**
  - The spanning tree, as established by the multicast routing algorithm, serves as the path for the reservation message, ensuring that all routers along the way are involved in the reservation process.

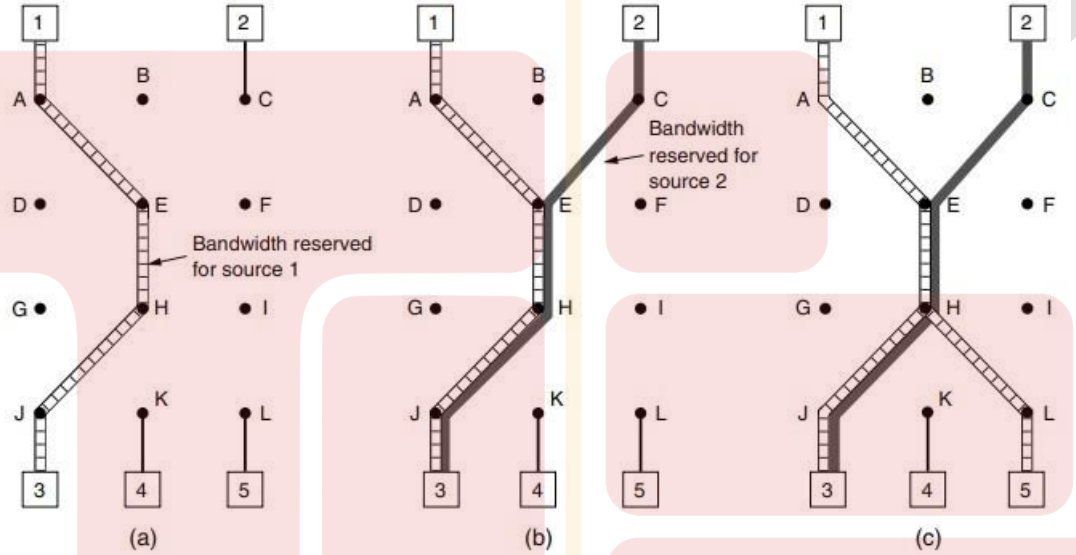



**Figure 5-35.** (a) Host 3 requests a channel to host 1. (b) Host 3 then requests a second channel, to host 2. (c) Host 5 requests a channel to host 1.

An example of such a reservation is shown in Fig. 5-35(a). Here host 3 has requested a channel to host 1. Once it has been established, packets can flow from 1 to 3 without congestion. Now consider what happens if host 3 next reserves a channel to the other sender, host 2, so the user can watch two television programs at once. A second path is reserved, as illustrated in Fig. 5-35(b). Note that two separate channels are needed from host 3 to router E because two independent streams are being transmitted.

- **Receiver's Reservation Options:**
  - When making a reservation, a receiver has the option to specify one or more sources from which it wants to receive data.
  - The receiver can indicate whether these source choices are fixed for the entire duration of the reservation or if it wants the flexibility to change sources later.

- **Optimizing Bandwidth Planning:**
  - Routers utilize the information provided by the receiver to optimize bandwidth planning.
  - Specifically, if two receivers agree not to change sources later on, routers can set them up to share a path efficiently.
- **Decoupling Reserved Bandwidth and Source Choice:**
  - In the fully dynamic case, reserved bandwidth is decoupled from the choice of the data source.
  - Once a receiver has reserved bandwidth, it can switch to another source while retaining the portion of the existing path that is valid for the new source.

### 5.4.6 Differentiated Services

**Flow-Based Quality of Service (QoS):**

- **Advantages:**
  - Offers good QoS for one or more flows by reserving necessary resources along the route.
- **Disadvantages:**
  - Requires advance setup for establishing each flow, which doesn't scale well with a large number of flows.
  - Maintains internal per-flow state in routers, making them vulnerable to crashes.
  - Involves substantial changes to router code and complex router-to-router exchanges for flow setup.

**Class-Based Quality of Service (Differentiated Services - DiffServ):**

- **Approach:**
  - Simpler QoS approach, largely implemented locally in each router without advance setup for the entire path.
  - Utilizes an architecture called Differentiated Services (DiffServ), standardized by IETF.
- **Administrative Domain:**
  - Implemented by a set of routers forming an administrative domain (e.g., ISP or telco).
  - Defines service classes with corresponding forwarding rules.
- **Packet Marking:**
  - If a customer subscribes to DiffServ, incoming packets are marked with the corresponding class.
  - Class information is carried in the Differentiated Services field of IPv4 and IPv6 packets.

- **Traffic Conformance:**
  - Traffic within a class may be required to conform to specific shapes (e.g., leaky bucket) or limitations.
- **Ease of Implementation:**
  - Requires no advance setup, resource reservation, or time-consuming end-to-end negotiation for each flow.
  - Relatively easy to implement compared to flow-based schemes.

**Comparison Example - Internet Telephony:**

- **Flow-Based Scheme:**
  - Each telephone call gets dedicated resources and guarantees.
- **Class-Based Scheme:**
  - All telephone calls share resources reserved for the telephony class, preventing interference from other classes.
  - No individual telephone call receives private resources reserved exclusively for it.

## Expedited Forwarding

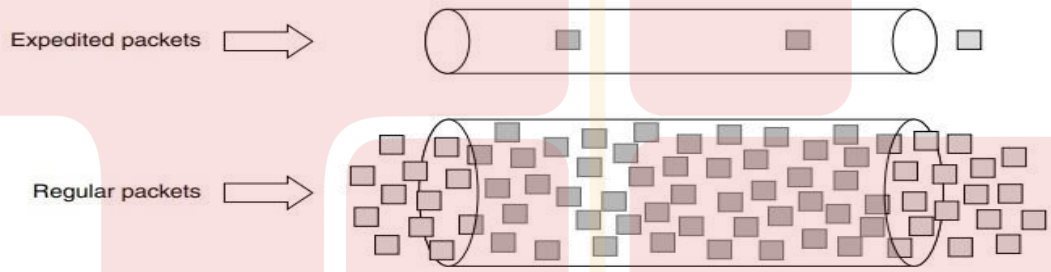



**Figure 5-36.** Expedited packets experience a traffic-free network.

**Expedited Forwarding (EF) Service Class:**

- **Purpose:**
  - Designed to provide low-loss, low-delay, and low-jitter service.
  - Suited for applications with stringent requirements, such as Voice over IP (VoIP).
- **Two-Class System:**
  - Consists of two classes of service: regular and expedited.
  - Majority of traffic is expected to be regular, and a limited fraction is designated as expedited.
- **Symbolic Representation:**
  - Represented as a "two-tube" system, emphasizing the separation of regular and expedited traffic.
  - Note: There is one physical line; logical pipes illustrate reserved bandwidth for

different service classes.

- **Implementation Strategy:**
  - Packets are classified as expedited or regular and marked accordingly.
  - Classification may occur at the sending host or the ingress (first) router.
  - Advantage of host-based classification: More information about packet flows is available.
  - Example: VoIP packets may be marked for expedited service by hosts.
- **Traffic Policing:**
  - In cases where marking is done by the host, the ingress router may police the traffic to ensure customers send only the allocated amount of expedited traffic.
- **Queueing Mechanism:**
  - Routers within the network may have two output queues for each outgoing line: one for expedited packets and one for regular packets.
  - Arrival packets are queued accordingly, and the expedited queue is given priority over the regular queue.
- **Network Behaviour:**
  - Expedited packets, despite the actual presence of heavy regular traffic, experience the network as unloaded due to priority treatment.
- **Marking by Hosts:**
  - Facilitates the application of expedited service without requiring modifications to existing applications.
  - Common practice for VoIP packets, ensuring preferential treatment in supporting networks.

**Assured Forwarding**

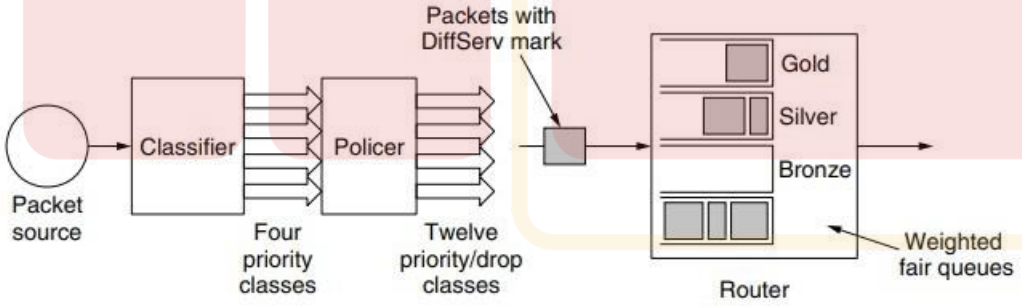



**Figure 5-37.** A possible implementation of assured forwarding.

**Assured Forwarding Service Class:**

- **Objective:**
  - Aimed at managing service classes with greater complexity than expedited

forwarding.

- Described in RFC 2597.

- **Priority Classes:**
  - Specifies four priority classes, each allocated specific resources.
  - Top three classes often referred to as gold, silver, and bronze.
- **Discard Classes:**
  - Defines three discard classes for packets experiencing congestion: low, medium, and high.
  - Combining priority and discard classes results in 12 service classes.
- **Packet Processing Steps:**
  1. **Packet Classification:**
     - Classify packets into one of the four priority classes.
     - Classification can occur on the sending host or the ingress router.
     - Rate of higher-priority packets may be limited as part of the service offering.
  2. **Discard Class Determination:**
     - Pass packets through a traffic policer (e.g., token bucket) to identify discard class.
     - Low discard for packets within small bursts, medium discard for those exceeding small bursts, and high discard for packets exceeding large bursts.
     - Encode priority and discard class in each packet.
  3. **Router Processing:**
     - Routers in the network use a packet scheduler to handle different classes.
     - Weighted fair queueing commonly employed for the four priority classes.
     - Higher-weighted classes receive more bandwidth, preventing starvation of lower classes.
- **Preferential Dropping:**
  - Within a priority class, packets with a higher discard class can be preferentially dropped.
  - Algorithms like RED (Random Early Detection) may be used, starting to drop packets before buffer space is exhausted during congestion.
  - Enables acceptance of low discard packets while dropping high discard packets.
- **Packet Scheduler:**
  - Implementation may involve weighted fair queueing, giving higher weights to higher-priority classes.

- Ensures that higher-priority classes receive a larger share of bandwidth.

- **Complexity and Flexibility:**
  - Assured forwarding introduces a more elaborate scheme, allowing for nuanced management of service classes.
  - Offers flexibility in handling different classes and responding to congestion scenarios.

# THE TRANSPORT LAYER

- The network layer provides end-to-end packet delivery using data- grams or virtual circuits.
- The transport layer builds on the network layer to pro- vide data transport from a process on a source machine to a process on a destination machine with a desired level of reliability that is independent of the physical networks currently in use.
- It provides the abstractions that applications need to use the network. Without the transport layer, the whole concept of layered proto- cols would make little sense.

## 6.1 THE TRANSPORT SERVICE

- **we will examine two sets of transport layer primitives:**
  - First comes a simple (but hypothetical) one to show the basic ideas.
  - Then comes the interface commonly used in the Internet.

### 6.1.1 Services Provided to the Upper Layers

- The ultimate goal of the transport layer is to provide efficient, reliable, and cost-effective data transmission service to its users, normally processes in the application layer.
- The software and/or hardware within the transport layer that does the work is called the **transport entity**.
- The transport entity can be located in the operating system kernel, in a library package bound into network applications, in a separate user process, or even on the network interface card.
- The (logical) relationship of the network, transport, and application layers is illustrated in Fig. 6-1.

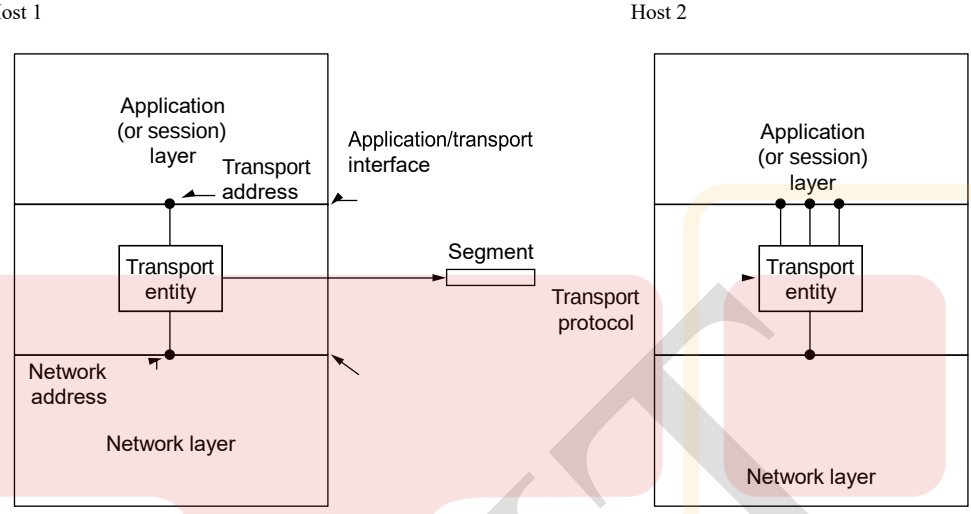



**Figure 6-1.** The network, transport, and application layers

- Connection-oriented and connectionless, there are also two types of transport service.
- The connection-oriented transport service is similar to the connection-oriented network service in many ways. In both cases, connections have three phases: establishment, data transfer, and release. Addressing and flow control are also similar in both layers.
- The connectionless transport service is also very similar to the connectionless network service. However, note that it can be difficult to provide a connectionless transport service on top of a connection-oriented network service, since it is inefficient to set up a connection to send a single packet and then tear it down immediately afterwards.

### 6.1.2 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.
- 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.
- 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.

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

**Figure 6-2.** The primitives for a simple transport service.

- To see how these primitives might be used, consider an application with a ser- ver 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 that blocks 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.

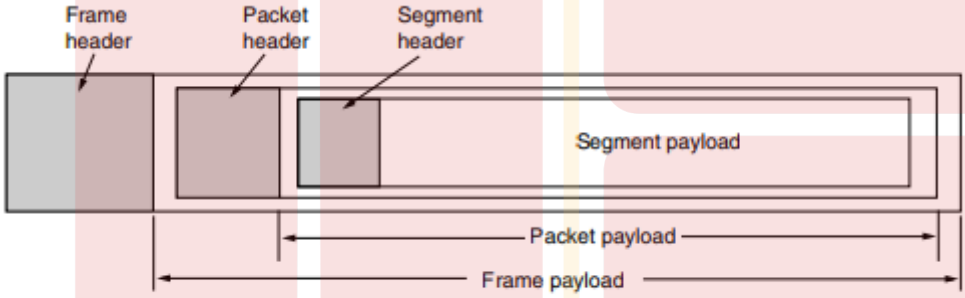



**Figure 6-3.** Nesting of segments, packets, and frames.

- The term **segment** for messages sent from transport entity to transport entity. TCP, UDP and other Internet protocols use this term.
- Some older protocols used the ungainly name **TPDU** (**Transport Protocol Data Unit**). That term isnot used much any more now but you may see it in older papers and books.
- When a frame arrives, the data link layer processes the frame header and, if the destination address matches for local deliv- ery, passes the contents of the frame payload field up to the network entity.

### 6.1.3 Berkeley Sockets

- Inspect another set of transport primitives, the socket prim- itives as they are used for TCP.

- Sockets were first released as part of the Berke- ley UNIX 4.2BSD software distribution in 1983.
- They quickly became popular. The primitives are now widely used for Internet programming on many operating systems, especially UNIX-based systems, and there is a socket-style API for Windows called ‘‘winsock.’

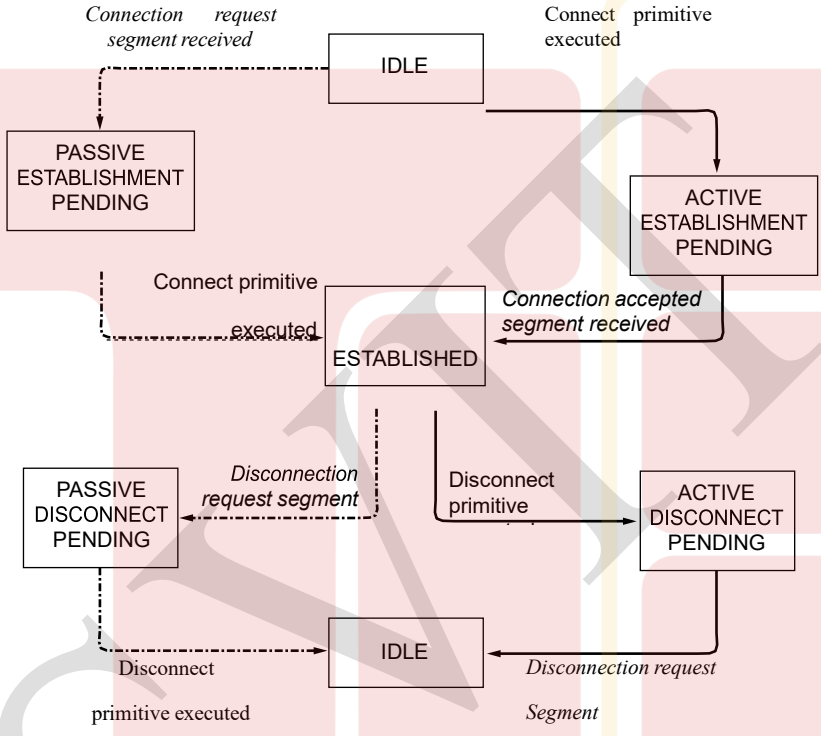


Figure 6-4. A state diagram for a simple connection management scheme. Transitions labeled in italics are caused by packet arrivals. The solid lines show the client's state sequence. The dashed lines show the server's state sequence

| Primitive | Meaning |
|---|---|
| SOCKET | Create a new communication endpoint |
| BIND | Associate a local address with a socket |
| LISTEN | Announce willingness to accept connections; give queue size |
| ACCEPT | Passively establish an incoming connection |
| CONNECT | Actively attempt to establish a connection |
| SEND | Send some data over the connection |
| RECEIVE | Receive some data from the connection |
| CLOSE | Release the connection |

**Figure 6-5.** The socket primitives for TCP.

The primitives are listed in Fig. 6-5. Roughly speaking, they follow the model of our first example but offer more features and flexibility.

- The first four primitives in the list are executed in that order by servers. The SOCKET primitive creates a new endpoint and allocates table space for it within the transport entity.
- The parameters of the call specify the addressing format tobe used, the type of service desired (e.g., reliable byte stream), and the protocol.
- The socket API is often used with the TCP protocol to provide a connection-oriented service called a **reliable byte stream**, which is simply the reliable bit pipe.
- **DCCP** (**Datagram Congestion Controlled Protocol**) is a version of UDP with congestion control (Kohler et al., 2006). It is up to the tran- sport users to understand what service they are getting.
- Newer protocols and interfaces have been devised that support groups of related streams more effectively and simply for the application.
- Two examples are **SCTP** (**Stream Control Transmission Protocol**) defined in RFC 4960 and **SST** (**Structured Stream Transport**) (Ford, 2007).
- These protocols must change the socket API slightly to get the benefits of groups of relatedstreams, and they also support features such as a mix of connection-oriented and connectionless traffic and even multiple network paths.

## 6.1.4 An Example of Socket Programming: An Internet File Server

- As an example of the nitty-gritty of how real socket calls are made, consider the client and server code of Fig. 6-6. Here we have a very primitive Internet file server along with an example client that uses it.
- The code has many limitations (discussed below), but in principle the server code can be compiled and run on any UNIX system connected to the Internet.
- It starts out by including some standard headers, the last three of which contain the main Internet-related definitions and data structures.
- Next comes a definition of *SERVER PORT* as 12345. This num- ber was chosen arbitrarily. Any number between 1024 and 65535 will work justas well, as long as it is not in use by some other process; ports below 1023 are re- served for privileged users.
- The next two lines in the server define two constants needed. The first one determines the chunk size in bytes used for the file transfer.

```
/* This page contains a client program that can request a file from the server program
 * on the next page. The server responds by sending the whole file.
 */

#include      <sys/types.h>
#include      <sys/socket.h>
#include      <netinet/in.h>
#include <netdb.h>

#define SERVER_PORT 12345                    /* arbitrary, but client & server must agree */
#define BUF_SIZE 4096                        /* block transfer size */

int main(int argc, char **argv)

{

  int c, s, bytes;

  char buf[BUF_SIZE];                        /* buffer for incoming file */
  struct hostent *h;                         /* info about server */

  struct sockaddr_in channel;                /* holds IP address */

  if (argc != 3) fatal("Usage: client server-name file-name");

  h = gethostbyname(argv[1]);                /* look up host's IP address */if
  (!h) fatal("gethostbyname failed");

  s = socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
```

```
if (s <0) fatal("socket"); memset(&channel,
0, sizeof(channel)); channel.sin family=
AF INET; _ _

memcpy(&channel.sin addr.s addr, h->h addr, h->h_ length);
channel.sin port= htons(SERVER PORT);


c = connect(s, (struct sockaddr *) &channel, sizeof(channel));if (c <
0) fatal("connect failed");
/* Connection is now established. Send file name including 0 byte at end. */write(s,
argv[2], strlen(argv[2])+1);
/* Go get the file and write it to standard output. */while
(1) {

    bytes = read(s, buf, BUF SIZE);          /* read from socket */
    if (bytes <= 0) exit(0);                 /* check for end of file */

    write(1, buf, bytes);                    /* write to standard output */

}

}


fatal(char *string)

{

 printf("%s\n", string);
 exit(1);

}
```

**Figure 6-6.** Client code using sockets. The server code is on the next page.

```
#include <sys/types.h>                       /* This is the server code */
#include <sys/fcntl.h>

#include    <sys/socket.h>
#include     <netinet/in.h>
#include <netdb.h>

#define SERVER PORT 12345                    /* arbitrary, but client & server must agree */
#define BUF SIZE 4096                        /* block transfer size */
_
#define QUEUE SIZE 10

int main(int argc, char *argv[])

{

 int s, b, l, fd, sa, bytes, on = 1;

 char buf[BUF SIZE];                         /* buffer for outgoing file */
```

```
struct sockaddr_in channel;                    /* holds IP address */

/* Build address structure to bind to socket. */

memset(&channel, 0, sizeof(channel));          /* zero channel */
channel.sin_family = AF_INET;

channel.sin_addr.s_addr = htonl(INADDR_ANY);
channel.sin_port = htons(SERVER_PORT);

/* Passive open. Wait for connection. */

s = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);     /* create socket */if
(s < 0) fatal("socket failed");

setsockopt(s, SOL_SOCKET, SO_REUSEADDR, (char *) &on, sizeof(on));

b = bind(s, (struct sockaddr *) &channel, sizeof(channel));if (b
< 0) fatal("bind failed");

l = listen(s, QUEUE_SIZE);                     /* specify queue size */if
(l < 0) fatal("listen failed");

/* Socket is now set up and bound. Wait for connection and process it. */while (1)
{
    sa = accept(s, 0, 0);                      /* block for connection request */if
    (sa < 0) fatal("accept failed");

    read(sa, buf, BUF_SIZE);                   /* read file name from socket */

    /* Get and return the file. */

    fd = open(buf, O_RDONLY);                  /* open the file to be sent back */if
    (fd < 0) fatal("open failed");

    while (1) {

        bytes = read(fd, buf, BUF_SIZE); /* read from file */

        if (bytes <= 0) break;                 /* check for end of file */

        write(sa, buf, bytes);                 /* write bytes to socket */

    }

    close(fd);                                 /* close file */

    close(sa);                                 /* close connection */

  }

}
```

## 6.2 ELEMENTS OF TRANSPORT PROTOCOLS

- The transport service is implemented by a transport protocol used between the two transport entities.
- However, significant differences between the two also exist. These dif- ferences are due to major dissimilarities between the environments in which the two protocols operate, as shown in Fig. 6-7.

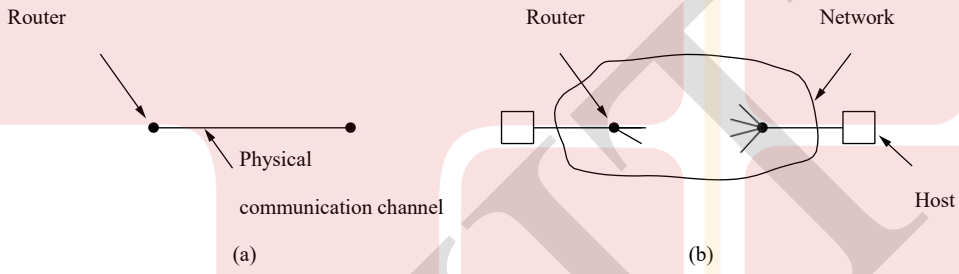



**Figure 6-7.** (a) Environment of the data link layer. (b) Environment of the transport layer.

- At the data link layer, two routers communicate directly via a physical channel, whether wired or wireless, whereas at the transport layer, this physical channel is replaced by the entire network. This difference has many important implications for the protocols.
- For one thing, over point-to-point links such as wires or optical fiber, it is usually not necessary for a router to specify which router it wants to talk to—each outgoing line leads directly to a particular router.
- Even on wireless links, the process is not much different. Just sending a message is sufficient to have it reach all other destinations.
- When a router sends a packet over a link, it may arrive or be lost, but it cannot bounce around for a while, go into hiding in a far corner of the world, and sudden- ly emerge after other packets that were sent much later.

### 6.2.1 Addressing

- When an application (e.g., a user) process wishes to set up a connection to a remote application process, it must specify which one to connect to.

- The method normally used is to define transport addresses to which processes can listen for connection requests. In the Internet, these endpoints are called **ports**.
- We will use the generic term **TSAP** (**Transport Service Access Point**) to mean a specific endpoint in the transport layer.
- The analogous endpoints in the network layer (i.e., network layer addresses) are not-surprisingly called **NSAPs** (**Network Service Access Points**).
- Application processes, both clients and servers, can attach themselves to a local TSAP to establish a connection to a remote TSAP.

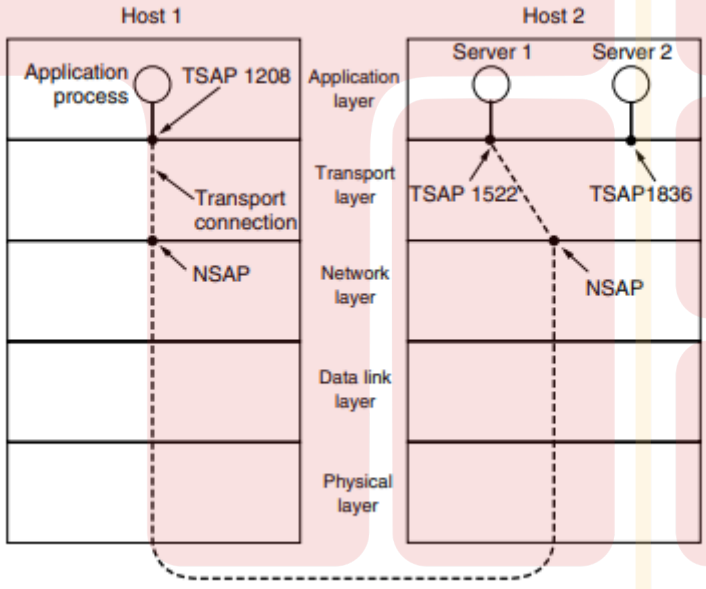



**Figure 6-8.** TSAPs, NSAPs, and transport connections.

➢ A possible scenario for a transport connection is as follows:

1. A mail server process attaches itself to TSAP 1522 on host 2 to wait for an incoming call. How a process attaches itself to a TSAP is outside the networking model and depends entirely on the local operating system. A call such as our LISTEN might be used, for example.
2. An application process on host 1 wants to send an email message, so it attaches itself to TSAP 1208 and issues a CONNECT request. The request specifies TSAP 1208 on host 1 as the source and TSAP 1522 on host 2 as the destination. This action ultimately results in a transport connection being established between the application process and the server.

3. The application process sends over the mail message.
4. The mail server responds to say that it will deliver the message.
5. The transport connection is released.
   - To handle this situation, an alternative scheme can be used. In this scheme, there exists a special process called a **portmapper**.
   - The user then sends a message specifying the service name, and the portmapper sends back the TSAP address.
   - The function of the portmapper is analogous to that of a directory assistance operator in the telephone system—it provides a mapping of names onto numbers.
   - It is wasteful to have each of them active and listening to a stable TSAP address all day long. An alternative scheme is shown in Fig. 6-9 in a simplified form. It is known as the **initial connection protocol**.
   - Instead of every conceivable server listening at a well-known TSAP, each machine that wishes to offer services to remote users has a special **process server** that acts as a proxy for less heavily used servers.
   - This server is called *inetd* on UNIX systems. It listens to a set of ports at the same time, waiting for a connection request. Potential users of a service begin by doing a CONNECT request, specifying the TSAP address of the service they want.

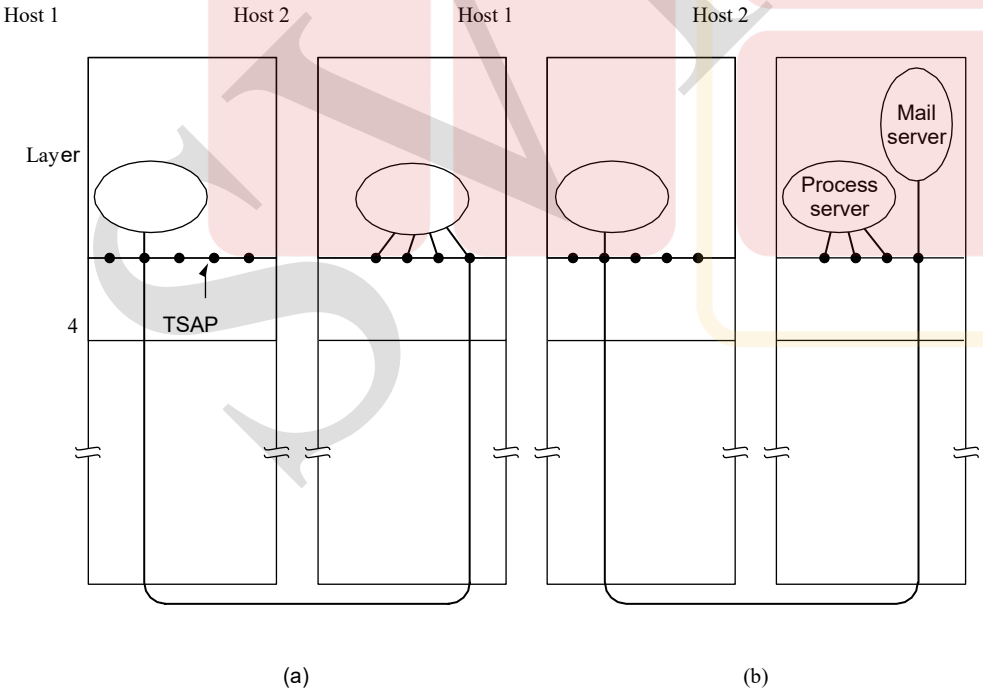



**Figure 6-9.** How a user process in host 1 establishes a connection with a mailserver in host 2 via a process server.

## 6.2.2 Connection Establishment

- At first glance, it would seem sufficient for one transport entity to just send a CONNECTION REQUEST segment to the destination and wait for a CONNECTION ACCEPTED reply.
- The worst possible nightmare is as follows. A user establishes a connection with a bank, sends messages telling the bank to transfer a large amount of money to the account of a not-entirely-trustworthy person.

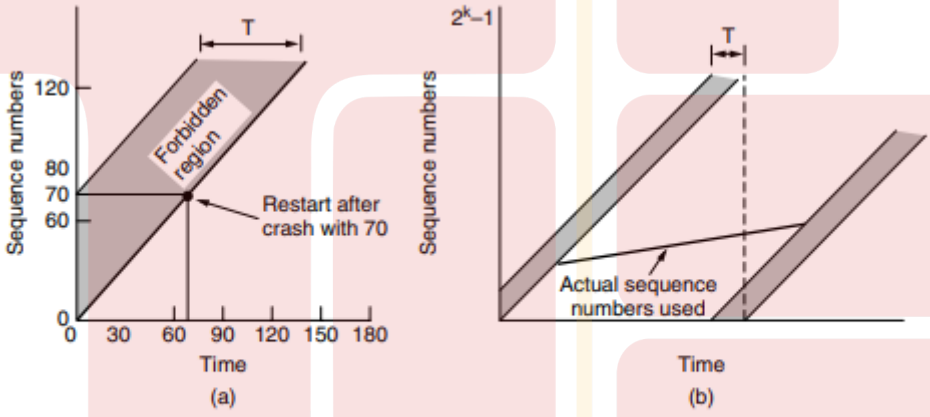



**Figure 6-10.** (a) Segments may not enter the forbidden region. (b) The resynchronization problem.

- Packet lifetime can be restricted to a known maximum using one (or more) of the following techniques:

1. Restricted network design.
2. Putting a hop counter in each packet.
3. Timestamping each packet.

- Once both transport entities have agreed on the initial sequence number, any sliding window protocol can be used for data flow control.
- . Within a connection, a timestamp is used to extend the 32-bit sequence number so that it will not wrap within the maximum packet lifetime, even for gigabit-per-second connections.

- This mechanism is a fix to TCP that was needed as it was used on faster and faster links. It is described in RFC 1323 and called **PAWS** (**ProtectionAgainst Wrapped Sequence numbers**).
- Across connections, for the initial se- quence numbers and before PAWS can come into play, TCP originally used the clock-based scheme just described.

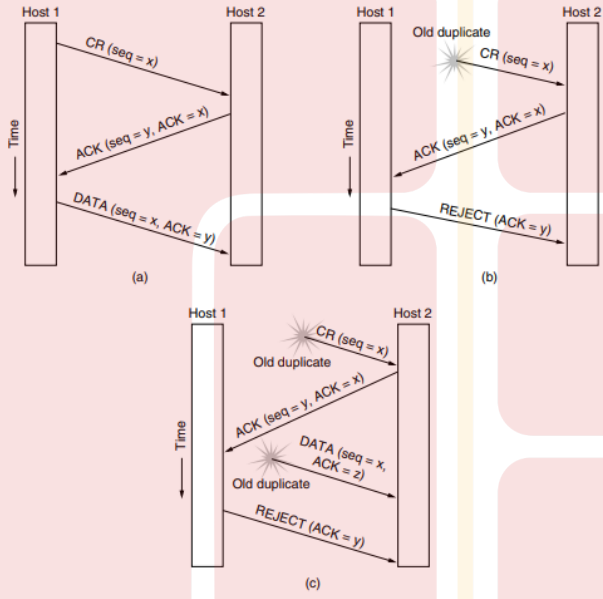



**Figure 6-11.** Three protocol scenarios for establishing a connection using a three-way handshake. CR denotes CONNECTION REQUEST. (a) Normal opera- tion. (b) Old duplicate CONNECTION REQUEST appearing out of nowhere.

(c) Duplicate CONNECTION REQUEST and duplicate ACK.

- However, it remains important that the initial sequence numbers not repeat for an interval even though they appearrandom to an observer.

## 6.2.3 Connection Release

- Releasing a connection is easier than establishing one. Nevertheless, there are more pitfalls than one might expect here. As we mentioned earlier, there are two styles of terminating a connection: asymmetric release and symmetric release.
- Asymmetric release is the way the telephone system works: when one

party hangs up, the connection is broken. Symmetric release treats the connection as two sep- arate unidirectional connections and requires each one to be released separately.

- Asymmetric release is abrupt and may result in data loss. Consider the scen- ario of Fig. 6-12. After the connection is established, host 1 sends a segment that arrives properly at host 2.

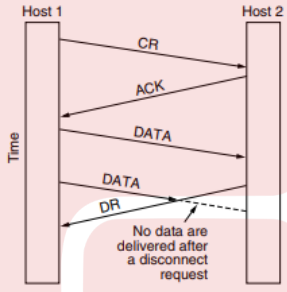



**Figure 6-12.** Abrupt disconnection with loss of data.

- Then host 1 sends another segment. Unfortunately, host 2 issues a DISCONNECT before the second segment arrives. The result is that the connection is released and data are lost.

- There is a famous prob- lem that illustrates this issue. It is called the **two-army problem**.
- Imagine that a white army is encamped in a valley, as shown in Fig. 6-13. On both of the sur- rounding hillsides are blue armies.
- The white army is larger than either of the blue armies alone, but together the blue armies are larger than the white army

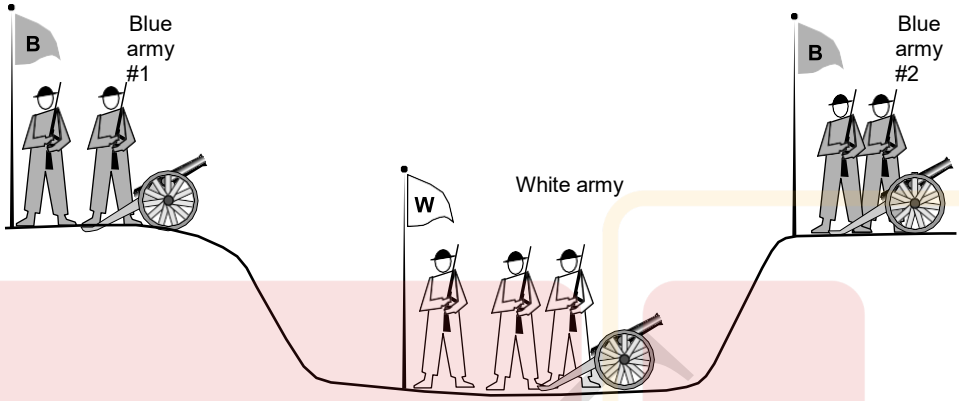



Figure 6-13. The two-army problem

- If either blue army attacks by itself, it will be defeated, but if the two blue armies at- tack simultaneously, they will be victorious.
- The blue armies want to synchronize their attacks. However, their only communication medium is to send messengers on foot down into the valley, where they might be captured and the message lost (i.e., they have to use an unreliable communication channel).

- In Fig. 6-14(a), we see the normal case in which one of the users sends a DR (DISCONNECTION REQUEST) segment to initiate the connection release. When it arrives, the recipient sends back a DR segment and starts a timer, just in case its DR is lost.
- When this DR arrives, the original sender sends back an ACK segment and releases the connection. Finally, when the ACK segment arrives, the receiver also releases the connection.
- . Releasing a connection means that the transport entity removes the information about the connection from its table of currently open connections and signals the connection's owner (the transport user) somehow.
- If the final ACK segment is lost, as shown in Fig. 6-14(b), the situation is saved by the timer. When the timer expires, the connection is released anyway.
- In Fig. 6-14(c), we see how this works, assuming that the second time no segments are lost and all segments are delivered correctly and on time.
- Our last scenario, Fig. 6-14(d), is the same as Fig. 6-14(c) except that now we assume all the repeated attempts to retransmit the DR also fail due to lost seg- ments.

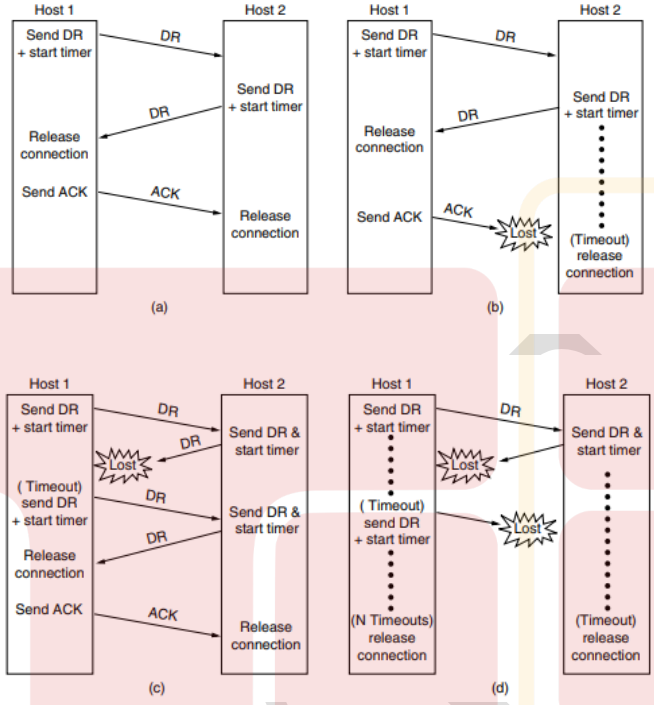



**Figure 6-14.** Four protocol scenarios for releasing a connection. (a) Normal case of three-way handshake. (b) Final ACK lost. (c) Response lost. (d) Re- sponse lost and subsequent DRs lost.

## 6.2.4 Error Control and Flow Control

- Error control is ensuring that the data is deliv- ered with the desired level of reliability, usually that all of the data is delivered without any errors.
- Flow control is keeping a fast transmitter from overrunning a slow receiver.

Given that these mechanisms are used on frames at the link layer, it is natural to wonder why they would be used on segments at the transport layer as well.

1 A frame carries an error-detecting code (e.g., a CRC or checksum) that is used to check if the information was correctly received.
2 A frame carries a sequence number to identify itself and is retrans- mitted by the sender until it receives an acknowledgement of suc- cessful receipt from the receiver. This is called **ARQ** (**Automatic Repeat reQuest**).
3 There is a maximum number of frames that the sender will allow to be outstanding at any time, pausing if the receiver is not acknowledg-ing frames quickly enough. If this maximum is one packet the proto- col

is called **stop-and-wait**. Larger windows enable pipelining and improve performance on long, fast links.

4 The **sliding window** protocol combines these features and is also used to support bidirectional data transfer.

- In Fig. 6-15(a). Howev- er, if there is wide variation in segment size, from short requests for Web pages to large packets in peer-to-peer file transfers, a pool of fixed-sized buffers presents problems.
- Another approach to the buffer size problem is to use variable-sized buffers,as in Fig. 6-15(b).
- A third possibility is to dedicate a single large circular buffer per connection, as in Fig. 6-15(c).

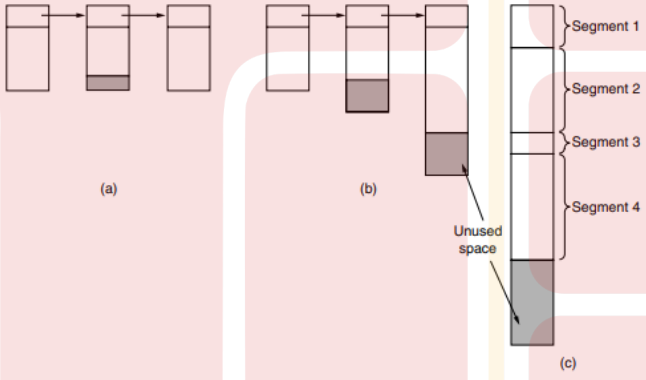



**Figure 6-15.** (a) Chained fixed-size buffers. (b) Chained variable-sized buffers. (c) One large circular buffer per connection.

- Figure 6-16 shows an example of how dynamic window management might work in a datagram network with 4-bit sequence numbers.
- Initially, *A* wants eight buffers, but it is granted only four of these. It then sends three segments, of which the third is lost.
- Segment 6 acknowledges receipt of all segments up to and including sequence number 1, thus allowing *A* to release those buffers, and furthermore informs *A* that it has permission to send three more segments starting beyond 1 (i.e., segments 2, 3, and 4).
- The next segment from *B* to *A* allocates another buffer and allows *A* to continue. This will happen when *B* has bufferspace, likely because the transport user has accepted more segment data.

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

**Figure 6-16.** Dynamic buffer allocation. The arrows show the direction oftransmission. An ellipsis (...) indicates a lost segment.

- Lookat line 16. *B* has now allocated more buffers to *A*, but the allocation segment was lost. Oops. Since control segments are not sequenced or timed out, *A* is now deadlocked.
- When buffer space no longer limits the maximum flow, another bottleneckwill appear: the carrying capacity of the network.

### 6.2.5 Multiplexing

- When a segment comes in, some way is needed to tell which process to give it to. This situation, called **multiplexing**, is shown in Fig. 6-17(a).
- If a user needs more bandwidth or more reliability than one of the network paths can pro- vide, a way out is to have a connection that distributes the traffic among multiple network paths on a round-robin basis, as indicated in Fig. 6-17(b). This modus operandi is called **inverse multiplexing**.
- **SCTP** (**Stream Control Transmission Protocol**), which can runa connection using multiple network interfaces.

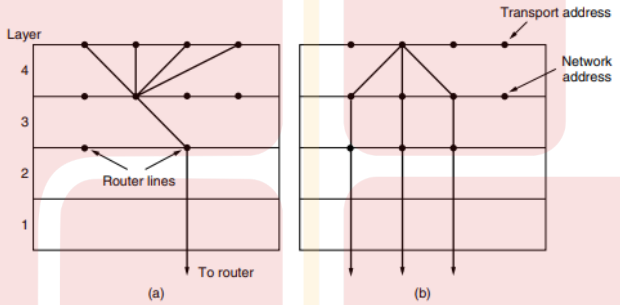



**Figure 6-17.** (a) Multiplexing. (b) Inverse multiplexing.

### 6.2.6 Crash Recovery

If hosts and routers are subject to crashes or connections are long-lived (e.g., large software or media downloads), recovery from these crashes becomes an issue. If the transport entity is entirely within the hosts, recovery from network and router crashes is straightforward. The transport entities expect lost segments all the time and know how to cope with them by using retransmissions.

| Strategy used by sending host | Strategy used by receiving host: First ACK, then write | | | Strategy used by receiving host: First write, then ACK | | |
|---|---|---|---|---|---|---|
| | AC(W) | AWC | C(AW) | C(WA) | W AC | WC(A) |
| Always retransmit | OK | DUP | OK | OK | DUP | DUP |
| Never retransmit | LOST | OK | LOST | LOST | OK | OK |
| Retransmit in S0 | OK | DUP | LOST | LOST | DUP | OK |
| Retransmit in S1 | LOST | OK | OK | OK | OK | DUP |

OK = Protocol functions correctly
DUP = Protocol generates a duplicate message
LOST = Protocol loses a message

**Figure 6-18.** Different combinations of client and server strategies.

## 6.3 CONGESTION CONTROL

- If the transport entities on many machines send too many packets into the net- work too quickly, the network will become congested, with performance degraded as packets are delayed and lost.
- Controlling congestion to avoid this problem is the combined responsibility of the network and transport layers.
- Congestion oc- curs at routers, so it is detected at the network layer. However, congestion is ulti- mately caused by traffic sent into the network by the transport layer.
- The Internet relies heavily on the transport layer for congestion control, and specific algorithms are built into TCP and other protocols.

### 6.3.1 Desirable Bandwidth Allocation

- we must specify the state in which a good congestion control algorithm will operate the network
- The goal is more than to simply avoid congestion. It is to find a good al- location of bandwidth to the transport entities that are using the network

## Efficiency and Power

- An efficient allocation of bandwidth across transport entities will use all of the network capacity that is available. However, it is not quite right to think that if there is a 100-Mbps link, five transport entities should get 20 Mbps each.
- The **goodput** (or rate of useful packets arriving at the receiver) as a function of the offered load. This curve and a matching curve for the delay as a function of the offered load are given in Fig. 6-19.

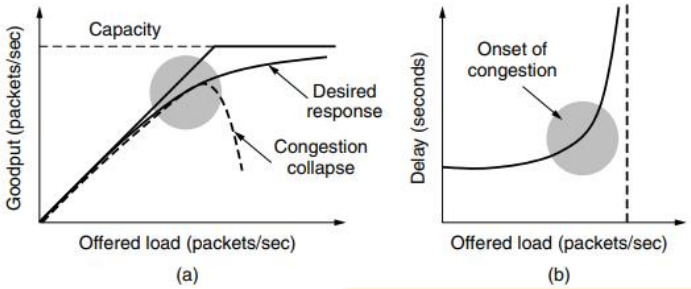
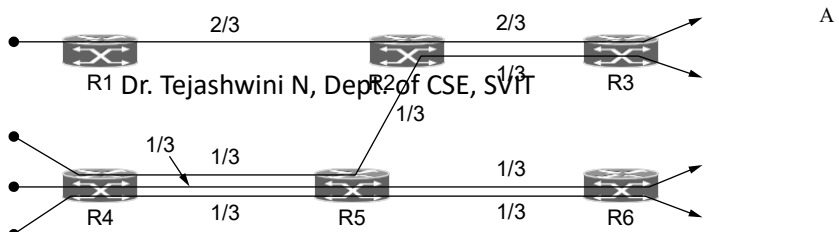



**Figure 6-19.** (a) Goodput and (b) delay as a function of offered load.

- As the load increases in Fig. 6-19(a) goodput initially increases at the same rate, but as the load approaches the capacity, goodput rises more gradually.
- This falloff is because bursts of traffic can occasionally mount up and cause some losses at buffers inside the network.
- In this state, senders are furiously sending packets, but in- creasingly little useful work is being accomplished.
- The corresponding delay is given in Fig. 6-19(b). Initially the delay is fixed, representing the propagation delay across the network.
- As the load approaches the capacity, the delay rises, slowly at first and then much more rapidly
- The delay cannot really go to infinity, except in a model in which the routers have infinite buffers. Instead, packets will be lost after experiencing the maximum buffering delay
- For both goodput and delay, performance begins to degrade at the onset of congestion.
- This point is be- low the capacity. To identify it, Kleinrock (1979) proposed the metric of **power**, where $power = \frac{load}{delay}$

## Max-Min Fairness

- Perhaps the first consideration is to ask what this problem has to do with con- gestion control. After all, if the network gives a sender some amount of bandwidth to use, the sender should just use that much bandwidth.
- They may for some flows if quality of service is supported, but many connections will seek to use whatever bandwidth is available or be lumped together by the network under a common allocation.
- A second consideration is what a fair portion means for flows in a network. Itis simple enough if $N$ flows use a single link, in which case they can all have $1/N$ of the bandwidth.
- An allocation is max-min fair if the bandwidth given to one flow cannot be increased without decreasing the bandwidth given to another flow with an alloca- tion that is no larger.
- A third consideration is the level over which to consider fairness. A network could be fair at the level of connections, connections between a pair of hosts, orall connections per host.



A

B

B

Figure 6-20. Max-min bandwidth allocation for four flows.

- Let us see an example. A max-min fair allocation is shown for a network with four flows, *A*, *B*, *C*, and *D*, in Fig. 6-20. Each of the links between routers has the same capacity, taken to be 1 unit, though in the general case the links will have different capacities.
- Notice that all of the other links have spare capacity. However, this capacity cannot be given to any of the flows without decreasing the capacity of another, lower flow.

Convergence

- A final criterion is that the congestion control algorithm converges quickly to a fair and efficient allocation of bandwidth.
- A good congestion control algorithm should rapidly converge to the ideal operating point, and it should track that point as it changes over time.

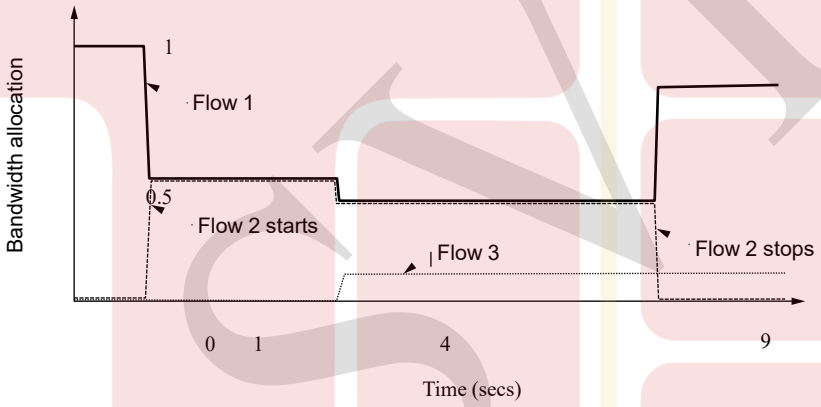



**Figure 6-21.** Changing bandwidth allocation over time.

- An example of a bandwidth allocation that changes over time and converges quickly is shown in Fig. 6-21,
- Initially, flow 1 has all of the bandwidth. One sec- ond later, flow 2 starts. It needs bandwidth as well.
- The first flow quickly captures 80% of the bandwidth. At all times, the total allocated bandwidth is approximately 100%, so that the network is fully used, and competing flows get equal treatment.

## 6.3.2 Regulating the Sending Rate

- The first is flow control, in the case that there is insufficient buffering at the receiver.

- The second is congestion, in the case that there is insufficient capaci- ty in the network

- In Fig. 6-22, we see this problem illustrated hydraulically. In Fig. 6-22(a), we see a thick pipe leading to a small-capacity receiver. This is a flow-control limited situation.
- . In Fig. 6-22(b), the limiting factor is not the bucket capacity, but the internal carrying capacity of the network.

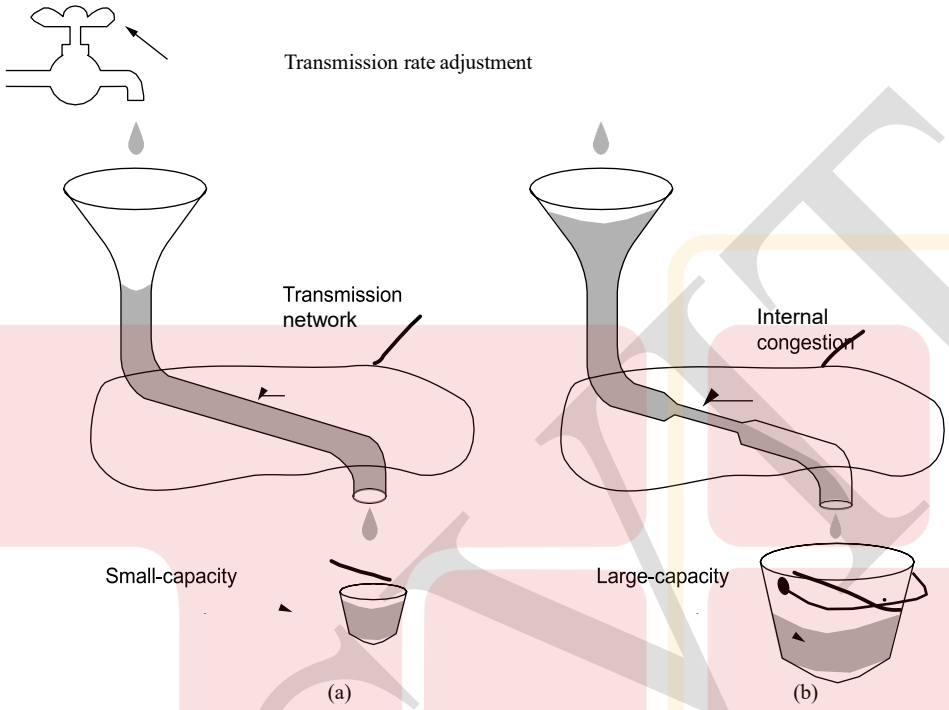



**Figure 6-22.** (a) A fast network feeding a low-capacity receiver. (b) A slownetwork feeding a high-capacity receiver.

- Combinationsare also possible. For example, Windows includes Compound TCP that uses both packet loss and delay as feedback signals (Tan et al., 2006). These designs are summarized in Fig. 6-23.
- . The way in which the rates are increased or decreased is given by a **control law**. These laws have a major effect on performance.

| Protocol | Signal | Explicit? | Precise? |
|---|---|---|---|
| XCP | Rate to use | Yes | Yes |
| TCP with ECN | Congestion warning | Yes | No |
| FAST TCP | End-to-end delay | No | Yes |
| Compound TCP | Packet loss & end-to-end delay | No | Yes |
| CUBIC TCP | Packet loss | No | No |
| TCP | Packet loss | No | No |

**Figure 6-23.** Signals of some congestion control protocols.

- Chiu and Jain (1989) studied the case of binary congestion feedback and con- cluded that **AIMD** (**Additive Increase Multiplicative Decrease**) is the appropr- iate control law to arrive at the efficient and fair operating point
  - The graph in Fig. 6-24 shows the bandwidth allocated to user 1 on the x-axis and to user 2 on the y-axis.
  - This is shown by the dotted efficiency line. A congestion signal is given by the network to both users when the sum of their allocations crosses this line.
  - The intersection of these lines is the de- sired operating point, when both users have the same bandwidth and all of the net- work bandwidth is used.

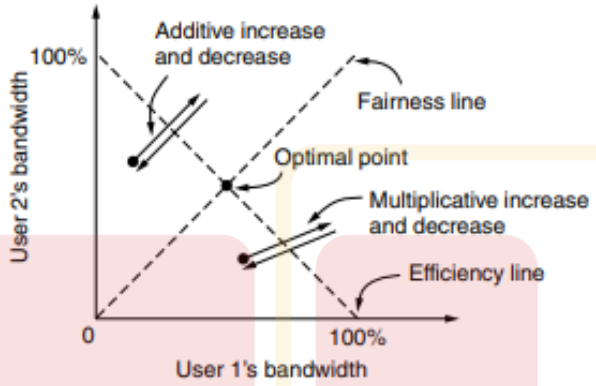



**Figure 6-24.** Additive and multiplicative bandwidth adjustments.

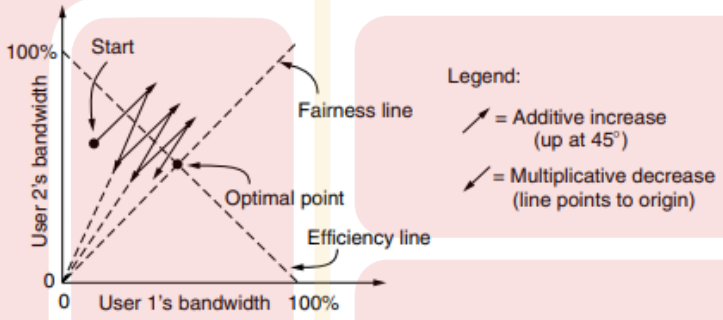



**Figure 6-25.** Additive Increase Multiplicative Decrease (AIMD) control law.

  - This behavior is the AIMD control law, and it is shown in Fig. 6-25. It can beseen that the path traced by this behavior does converge to the optimal point thatis both fair and efficient.
  - AIMD is the control law that is used by TCP, based on this argument and an- other stability argument.
  - In Sec. 6.5, we will describe in detail how TCP implements an AIMD control law to adjust the sending rate and provide congestion control.
  - .TCP uses this strategy. If the window size is $W$ and the round-trip time is $RTT$, the equivalent rate is $W/RTT$.

### 6.3.3 Wireless Issues

- The main issue is that packet loss is often used as a congestion signal, including by TCP as we have just discussed.

- Wireless networks lose packets all the time due to transmission er- rors.
- To function well, the only packet losses that the congestion control algorithm should observe are losses due to insufficient bandwidth, not losses due to trans- mission errors.
- One solution to this problem is to mask the wireless losses by using retransmissions over the wireless link.

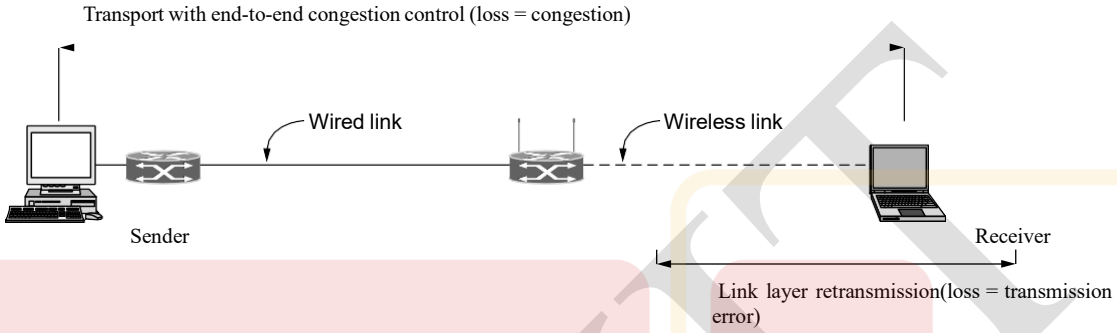



**Figure 6-26.** Congestion control over a path with a wireless link.

- Fig. 6-26 shows a path with a wired and wireless link for which the masking strategy is used.
- There are two aspects to note. First, the sender does not necessarily know that the path includes a wireless link, since all it sees is the wired link to which it is attached.
- Internet paths are heterogeneous and there is no general method for the sender to tell what kind of links comprise the path.
- This complicates the congestion control problem, as there is no easy way to use one protocol for wireless links and another protocol for wired links.

## 6.4 THE INTERNET TRANSPORT PROTOCOLS: UDP

- The Internet has two main protocols in the transport layer, a connectionless protocol and a connection-oriented one. The protocols complement each other.
- The connectionless protocol is UDP. It does almost nothing beyond sending packets between applications, letting applications build their own protocols on top as needed. The connection-oriented protocol is TCP.
- Since UDP isa transport layer protocol that typically runs in the operating system and protocols that use UDP typically run in user space, these uses might be considered applications.

## 6.4.1 Introduction to UDP

- The Internet protocol suite supports a connectionless transport protocol called UDP (User Datagram Protocol).
- UDP provides a way for applications to send encapsulated IP datagrams without having to establish a connection.
- UDP transmits **segments** consisting of an 8-byte header followed by the pay- load. The header is shown in Fig. 6-27.
- The two **ports** serve to identify the end- points within the source and destination machines.
- By copying the *Source port* field from the incoming segment into the *Destination port* field of the outgoing segment, the process sending the reply can specify which process on the sending machine is to get it.

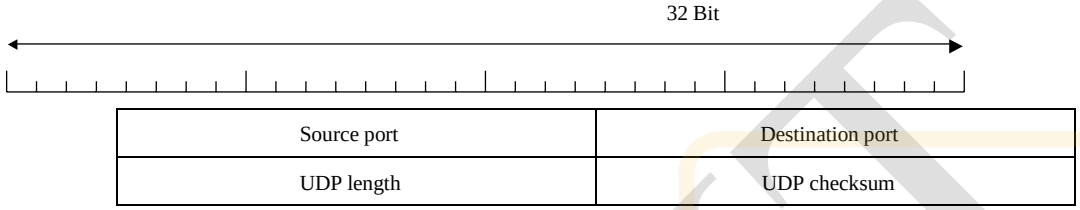



**Figure 6-27.** The UDP header.

- The pseudoheader for the case of IPv4 is shown in Fig. 6-28.
- It contains the 32-bit IPv4 addresses of the source and destination machines, the protocol number for UDP (17), and the byte count for the UDP segment (including the header).

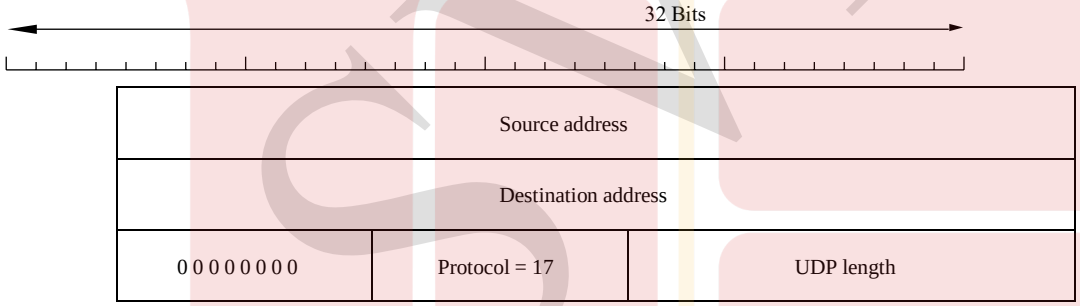



**Figure 6-28.** The IPv4 pseudoheader included in the UDP checksum.

- It does not do flow control, congestion control, or retransmission upon receipt of a bad segment.

## 6.4.2 Remote Procedure Call

- In a certain sense, sending a message to a remote host and getting a reply back is a lot like making a function call in a programming language.

- This observation has led people to try to arrange request-reply interactions on networks to be cast in the form of procedure calls.
- When a process on machine 1 calls a procedure on machine 2, the calling process on 1 is suspended and execution of the called pro- cedure takes place on 2.
- Information can be transported from the caller to the cal- lee in the parameters and can come back in the procedure result.

- No message pas- sing is visible to the application programmer. This technique is known as **RPC (Remote Procedure Call)**
- In the simplest form, to call a remote procedure, the client program must be bound with a small library procedure, called the **client stub.**
- The represent the server procedure in the client's address space. Similarly, the server is bound with a procedure called the **server stub**.

    - The actual steps in making an RPC are shown in Fig. 6-29. Step 1 is the cli- ent calling the client stub. This call is a local procedure call, with the parameters pushed onto the stack in the normal way.
    - Step 2 is the client stub packing the pa- rameters into a message and making a system call to send the message. Packing the parameters is called **marshaling.**
    - Step 3 is the operating system sending the message from the client machine to the server machine.
    - Step 4 is the operating system passing the incoming packet to the server stub.
    - Finally, step 5 is the server stub calling the server procedure with the unmarshaled parameters.

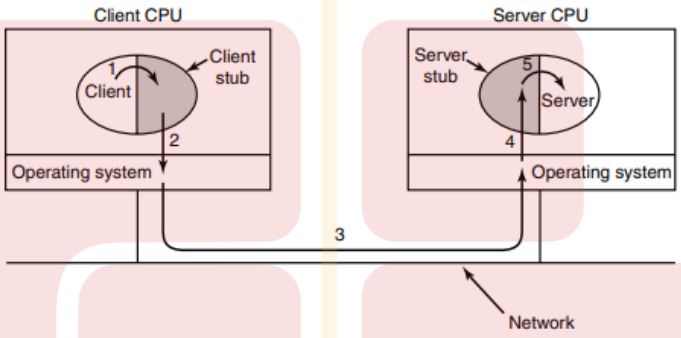



**Figure 6-29.** Steps in making a remote procedure call. The stubs are shaded.

## 6.4.3 Real-Time Transport Protocols

- Thus was **RTP (Real-time Transport Protocol)** born. It is described in RFC 3550 and is now in widespread use for multimedia applications.
- The first is the RTP protocol for transporting audio and video data in packets.
- The second is the processing that takes place, mostly at the receiver, to play out the audio and video at the right time.

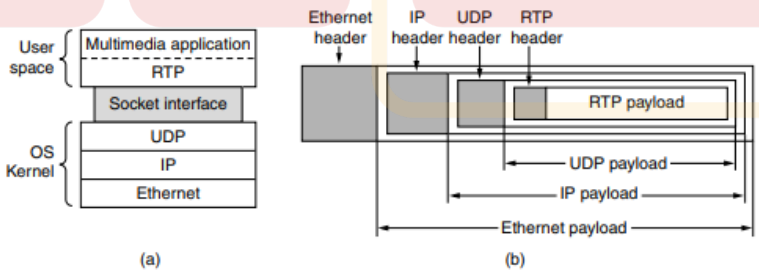



**Figure 6-30.** (a) The position of RTP in the protocol stack. (b) Packet nesting.

➢ RTP—The Real-time Transport Protocol

- The basic function of RTP is to multiplex several real-time data streams ontoa single stream of UDP packets.
- The UDP stream can be sent to a single destina- tion (unicasting) or to multiple destinations (multicasting).
- The RTP header is illustrated in Fig. 6-31. It consists of three 32-bit words and potentially some extensions.
- The first word contains the *Version* field, which is already at 2. Let us hope this version is very close to the ultimate version since there is only one code point left.

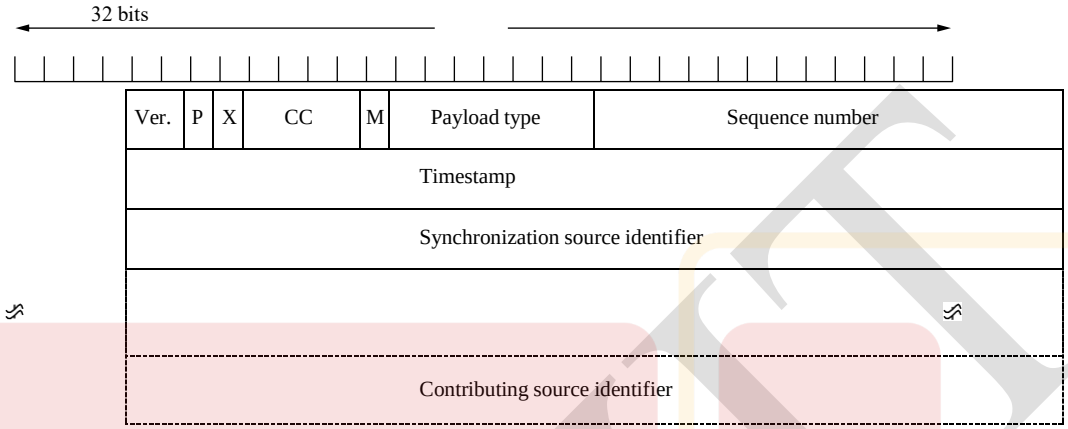



**Figure 6-31.** The RTP header.

- The *M* bit is an application-specific marker bit. It can be used to mark the start of a video frame, the start of a word in an audio channel, or some- thing else that the application understands.

➢ RTCP—The Real-time Transport Control Protocol

- RTP has a little sister protocol (little sibling protocol?) called **RTCP** (**Real- time Transport Control Protocol**).
- The first function can be used to provide feedback on delay, variation in delay or jitter, bandwidth, congestion, and other network properties to the sources
- The *Payload type* field is used to tell the destination what encoding algorithm is used for the current packet, making it possible to vary it on demand.
- RTCP also handles interstream synchronization. The problem is that different streams may use different clocks, with different granularities and different drift rates.

➢ Playout with Buffering and Jitter Control

- The packets are injected with exactly the right intervals be- tween them at the sender, they will reach the receiver with different relative times. This variation in delay is called **jitter**.
- The solution to this problem is to **buffer** packets at the receiver before they are played out to reduce the jitter.

  - As an example, in Fig. 6-32 we see a stream of packets being delivered with a substantial amount of jitter.
  - Packet 1 is sent from the server at $t$ = 0 sec and arrives at the client at $t$ = 1 sec.

- As the packets arrive, they are buffered on the client machine.

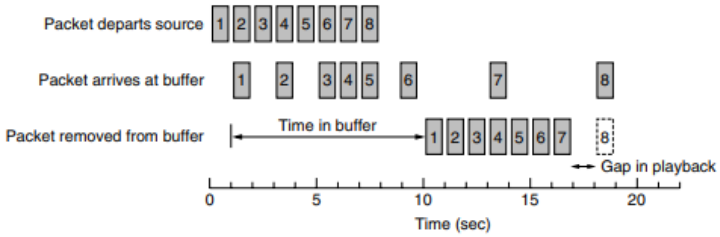



**Figure 6-32.** Smoothing the output stream by buffering packets.

- A key consideration for smooth playout is the **playback point**, or how long to wait at the receiver for media before playing it out. Deciding how long to wait depends on the jitter.
  - The difference between a low-jitter and high-jitter con- nection is shown in Fig. 6-33.
  - The average delay may not differ greatly between the two, but if there is high jitter the playback point may need to be much further out to capture 99% of the packets than if there is low jitter.
  - One way to avoid this problem for audio isto adapt the playback point between **talkspurts**, in the gaps in a conversation.

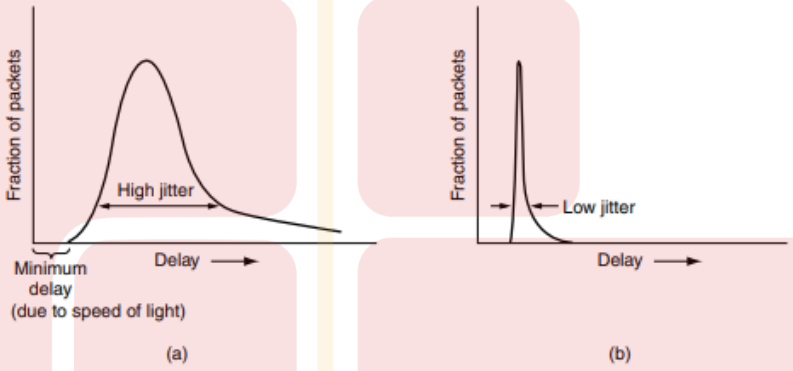



**Figure 6-33.** (a) High jitter. (b) Low jitter.

## 6.5 THE INTERNET TRANSPORT PROTOCOLS: TCP

- UDP is a simple protocol and it has some very important uses, such as client- server interactions and multimedia, but for most Internet applications, reliable, se- quenced delivery is needed.

- UDP cannot provide this, so another protocol is re- quired. It is called TCP and is the main workhorse of the Internet.

### 6.5.1 Introduction to TCP

- **TCP** (**Transmission Control Protocol**) was specifically designed to providea reliable end-to-end byte stream over an unreliable internetwork.
- An internet- work differs from a single network because different parts may have wildly different topologies, bandwidths, delays, packet sizes, and other parameters.

- TCP was designed to dynamically adapt to properties of the internetwork and to be robust

in the face of many kinds of failures.

## 6.5.2 The TCP Service Model

- TCP service is obtained by both the sender and the receiver creating end points, called **sockets.**
- Each socket has a socket num- ber (address) consisting of the IP address of the host and a 16-bit number local to that host, called a **port**.
- Port numbers below 1024 are reserved for standard services that can usually only be started by privileged users (e.g., root in UNIX systems). They are called **well-known ports**.

| Port | Protocol | Use |
|---|---|---|
| 20, 21 | FTP | File transfer |
| 22 | SSH | Remote login, replacement for Telnet |
| 25 | SMTP | Email |
| 80 | HTTP | World Wide Web |
| 110 | POP-3 | Remote email access |
| 143 | IMAP | Remote email access |
| 443 | HTTPS | Secure Web (HTTP over SSL/TLS) |
| 543 | RTSP | Media player control |
| 631 | IPP | Printer sharing |

**Figure 6-34.** Some assigned ports.

- The list of well-known ports is given at *www.iana.org*. Over 700 havebeen assigned. A few of the better-known ones are listed in Fig. 6-34.
  - Doing so would clutter up memory with daemons that were idle most of the time. Instead, what is commonly done is to have a single daemon, called **inetd** (**Internet daemon**) in UNIX, attach itself to multiple ports and wait for the first incoming connection.
  - All TCP connections are full duplex and point-to-point. Full duplex means that traffic can go in both directions at the same time. Point-to-point means that each connection has exactly two end points.
- if the sending process does four 512-byte writes to a TCP stream, these data may be delivered to the receiving process as four 512-byte chunks, two 1024-byte chunks, one 2048-byte chunk (seeFig. 6-35), or some other way.


**Figure 6-35.** (a) Four 512-byte segments sent as separate IP datagrams. (b) The 2048 bytes of data delivered to the application in a single READ call.

- The reader of a file cannot tell whether the file was written a block at a time, a byte at a time, or all in one blow.

- For Internet archaeologists, we will also mention one interesting feature of TCP service that remains in the protocol but is rarely used: **urgent data**.

### 6.5.3 The TCP Protocol

- 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 link has an **MTU** (**Maximum**
- The basic protocol used by TCP entities is the sliding window protocol with a dynamic window size.
- When a sender transmits a segment, it also starts a timer. When the segment arrives at the destination, the receiving TCP entity sends back a segment.

### 6.5.4 The TCP Segment Header

- Figure 6-36 shows the layout of a TCP segment. Every segment begins with a fixed-format, 20-byte header.
- . The fixed header may be followed by header options. After the options, if any, up to 65,535 □ 20 □ 20 □ 65, 495 data bytes may follow, where the first 20 refer to the IP header and the second to the TCP header.

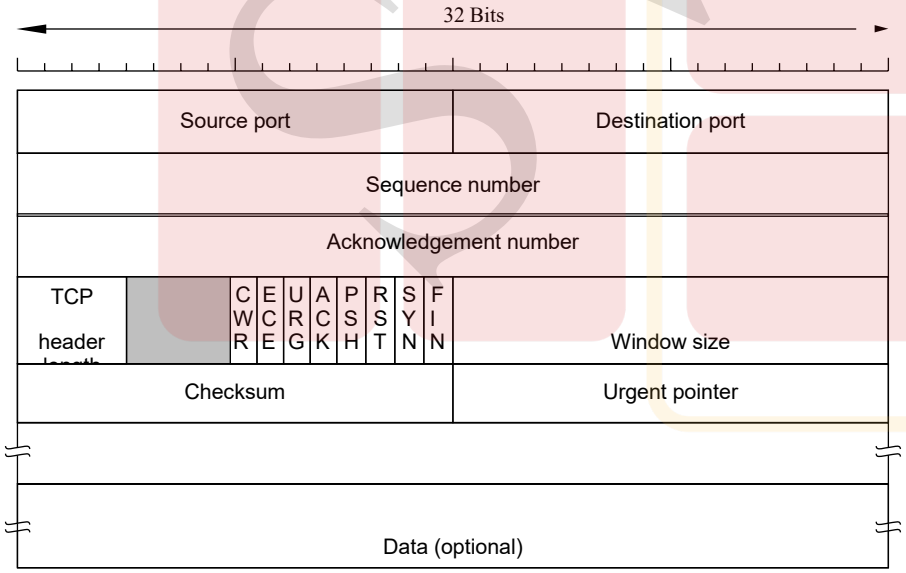



**Figure 6-36.** The TCP header.

- This connection identifier is called a **5 tuple** because it consists of five pieces of information: the protocol (TCP), source IP and source port, and destination IP and destination port.

- It is a **cumulative acknowledgement** because it summarizes the received data with a single number.
- The *TCP header length* tells how many 32-bit words are contained in the TCP header.
- The *Urgent pointer* is used to indicate a byte offset from the current sequence number at which urgent data are to be found.
- A widely used option is the one that allows each host to specify the **MSS** (**Maximum Segment Size**) it is willing to accept.
- The **window scale** option allows the sender and receiver to negotiate a window scale factor at the start of a connection.
- The **timestamp** option carries a timestamp sent by the sender and echoed by the receiver.
- The **PAWS** (**Protection Against Wrapped Sequence numbers**) scheme discards ar- riving segments with old timestamps to prevent this problem.
- Finally, the **SACK** (**Selective ACKnowledgement**) option lets a receiver tell a sender the ranges of sequence numbers that it has received.

### 6.5.5 TCP Connection Establishment

- When this segment arrives at the destination, the TCP entity there checks to see if there is a process that has done a LISTEN on the port given in the *Destination port* field.
  - The sequence of TCP segments sent in the normal case is shown in Fig. 6-37(a).

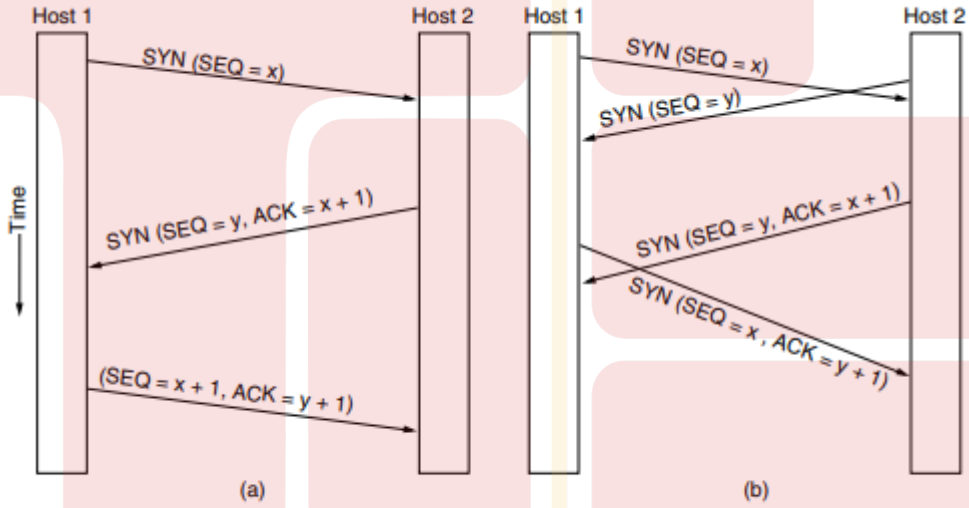



**Figure 6-37.** (a) TCP connection establishment in the normal case. (b) Simultaneous connection establishment on both sides.

  - In the event that two hosts simultaneously attempt to establish a connection between the same two sockets, the sequence of events is as illustrated in Fig. 6- 37(b).
- A vulnerability with implementing the three-way handshake is that the listening process must remember its sequence number as soon it responds with its own *SYN* segment.
- This means that a malicious sender can tie up resources ona host by sending a stream of *SYN* segments and never following through to com- plete the connection. This attack is called a **SYN flood.**
- One way to defend against this attack is to use **SYN cookies**. Instead of remembering the sequence number, a host chooses a cryptographically generated sequence number, puts it on the outgoing segment, and forgets it.

- There are some caveats, such as the inability to handle TCP options, so SYN cookies may be used only when the hostis subject to a SYN flood.

### 6.5.6 TCP Connection Release

- To release a connection, either party can send a TCP segment with the *FIN* bit set, which means that it has no more data to transmit.
- Data may continue to flow indefinitely in the other direction, how- ever. When both directions have been shut down, the connection is released.
- There is, in fact, no essential difference between the two hosts releasing sequentially or simultaneously.

### 6.5.7 TCP Connection Management Modeling

- The steps required to establish and release connections can be represented in a finite state machine with the 11 states listed in Fig. 6-38.
- In each state, certain events are legal. When a legal event happens, some action may be taken.
- Each connection starts in the *CLOSED* state. It leaves that state when it does either a passive open (LISTEN) or an active open (CONNECT).
- If the other side does the opposite one, a connection is established and the state becomes *ESTA-BLISHED.*
- Connection release can be initiated by either side. When it is com- plete, the state returns to *CLOSED.*

| State | Description |
|---|---|
| CLOSED | No connection is active or pending |
| LISTEN | The server is waiting for an incoming call |
| SYN RCVD | A connection request has arrived; wait for ACK |
| SYN SENT | The application has started to open a connection |
| ESTABLISHED | The normal data transfer state |
| FIN WAIT 1 | The application has said it is finished |
| FIN WAIT 2 | The other side has agreed to release |
| TIME WAIT | Wait for all packets to die off |
| CLOSING | Both sides have tried to close simultaneously |
| CLOSE WAIT | The other side has initiated a release |
| LAST ACK | Wait for all packets to die off |

**Figure 6-38.** The states used in the TCP connection management finite state machine.

- The finite state machine itself is shown in Fig. 6-39. The common case of a client actively connecting to a passive server is shown with heavy lines—solid for the client, dotted for the server.

- Each line in Fig. 6-39 is marked by an *event/action* pair. The event can either bea user-initiated system call (CONNECT, LISTEN, SEND, or CLOSE), a segment arrival (*SYN*, *FIN*, *ACK*, or *RST*), or, in one case, a timeout of twice the maximum packet lifetime.

- When an application program on the client machine issues a CONNECT re- quest, the local TCP entity creates a connection record, marks it as being in the *SYN SENT* state, and shoots off a *SYN* segment.

- When the *ACK* arrives, a transition is made to the state *FIN WAIT 2* and one direction of the connection is closed. When the other side closes, too, a *FIN* comes in, which is acknowledged.

- When a *SYN* comes in, it is acknowledged and the server goes to the *SYN RCVD* state. When the server's *SYN* is itself acknowledged, the three-way handshake is complete and the server goes to the *ESTABLISHED* state. Data transfer can now occur.

- When the client is done transmitting its data, it does a CLOSE, which causes a *FIN* to arrive at the server (dashed box marked "passive close").

- The server is then signaled. When it, too, does a CLOSE, a *FIN* is sent to the client.

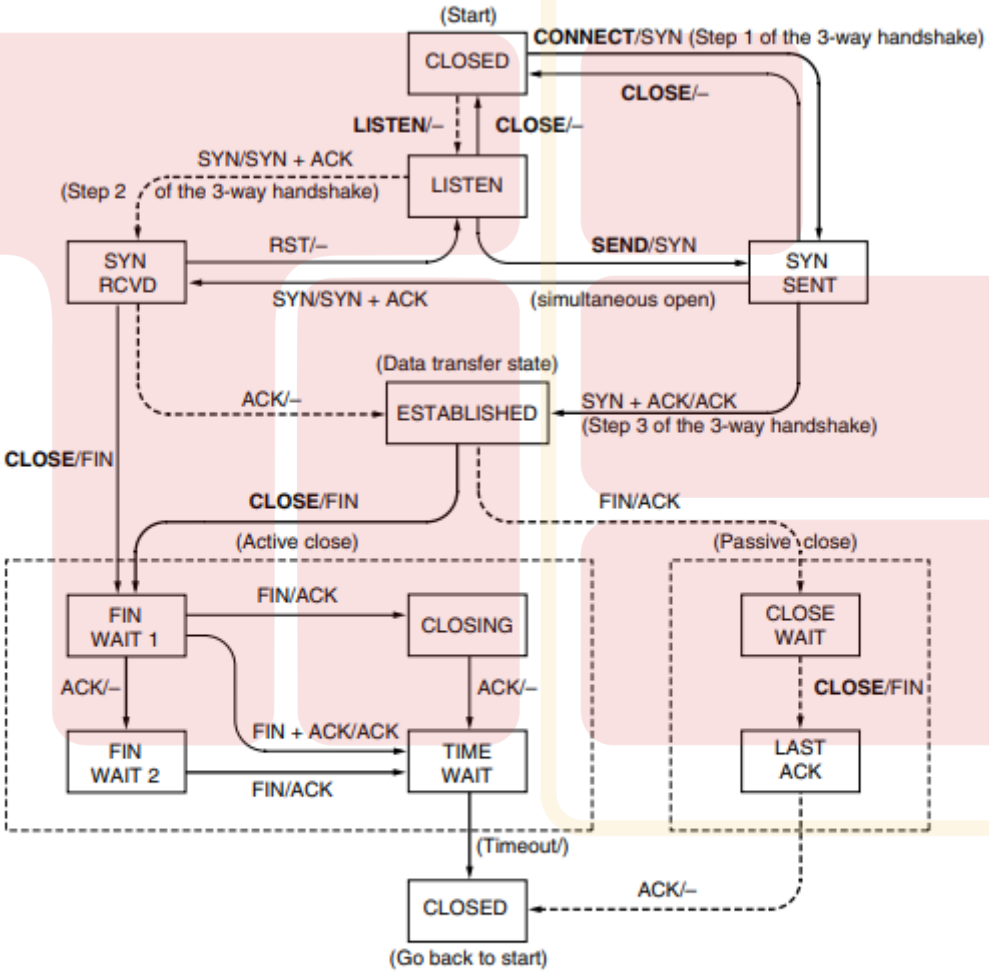



**Figure 6-39.** TCP connection management finite state machine. The heavy solid line is the normal path for a client. The heavy dashed line is the normal path for a server. The light lines are unusual events. Each transition is labeled with the event causing it and the action resulting from it, separated by a slash.

## ❖ Application Layer

- Network applications are the raisons d'être of a computer network—if we couldn't conceive of any useful applications, there wouldn't be any need for networking infrastructure and protocols to support them.
- These applications have been the driving force behind the Internet's success, motivating people in homes, schools, governments, and businesses to make the Internet an integral part of their daily activities.
- They include the killer application of the mid-1990s, the World Wide Web, encompassing Web surfing, search, and electronic commerce.
- They include instant messaging and P2P file sharing, the two killer applications introduced at the end of the millennium.
- we have seen the emergence of a new generation of social networking applications—such as Facebook, Instagram, Twitter, and WeChat—which have created engaging human networks on top of the Internet's network or routers and communication links.

## ❖ Principles of Network Applications

- At the core of network application development is writing programs that run on different end systems and communicate with each other over the network.
- For example, in the Web application there are two distinct programs that communicate with each other: the browser program running in the user's host (desktop, laptop, tablet, smartphone, and so on).
- when developing your new application, you need to write software that will run on multiple end systems.
- This software could be written, for example, in C, Java, or Python. Importantly, you do not need to write software that runs on network-core devices, such as routers or link-layer switches.
- Network-core devices do not function at the application layer but instead function at lower layers—specifically at the network layer and below.
- his basic design—namely, confining application software to the end systems—as shown in Figure 5.1, has facilitated the rapid development and deployment of a vast array of network applications.

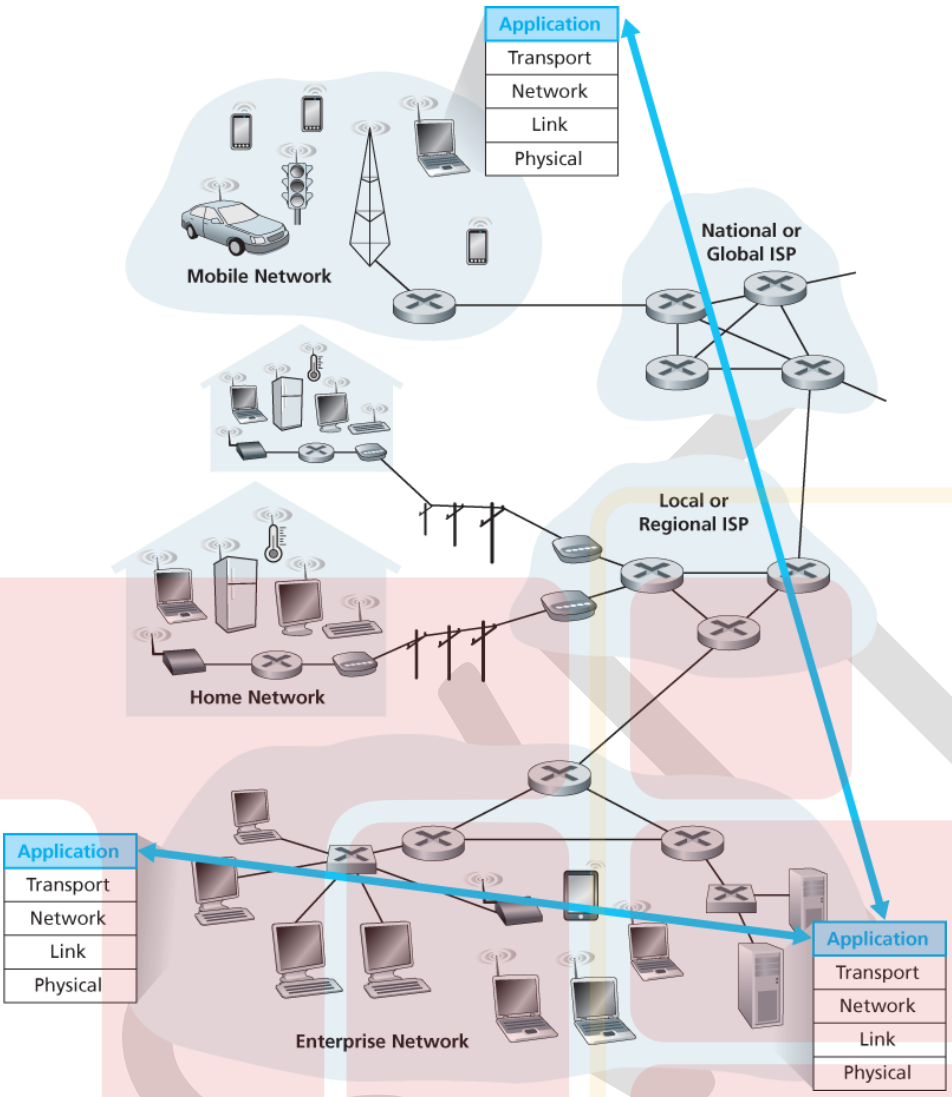



Figure 5.1 Communication for a network application takes place between end systems at the application layer

## ➢ Network Application Architectures

- Before diving into software coding, you should have a broad architectural plan for your application.
- From the application developer's perspective, the network architecture is fixed and provides a specific set of services to applications.
- In choosing the application architecture, an application developer will likely draw on one of the two predominant architectural paradigms used in modern network applications: the client-server architecture or the peer-to-peer (P2P) architecture.
- In a client-server architecture, there is an always-on host, called the server, which services requests from many other hosts, called clients.
- When a Web server receives a request for an object from a client host, it responds by sending the requested object to the client host. Note that with the client-server architecture, clients do not directly communicate with each other.
- Some of the better-known applications with a client-server architecture include the Web, FTP, Telnet, and e-mail. The client-server architecture is shown in Figure 5.2(a).
- In a P2P architecture, there is minimal (or no) reliance on dedicated servers in data centers. Instead the application exploits direct communication between pairs of intermittently connected hosts, called peers.

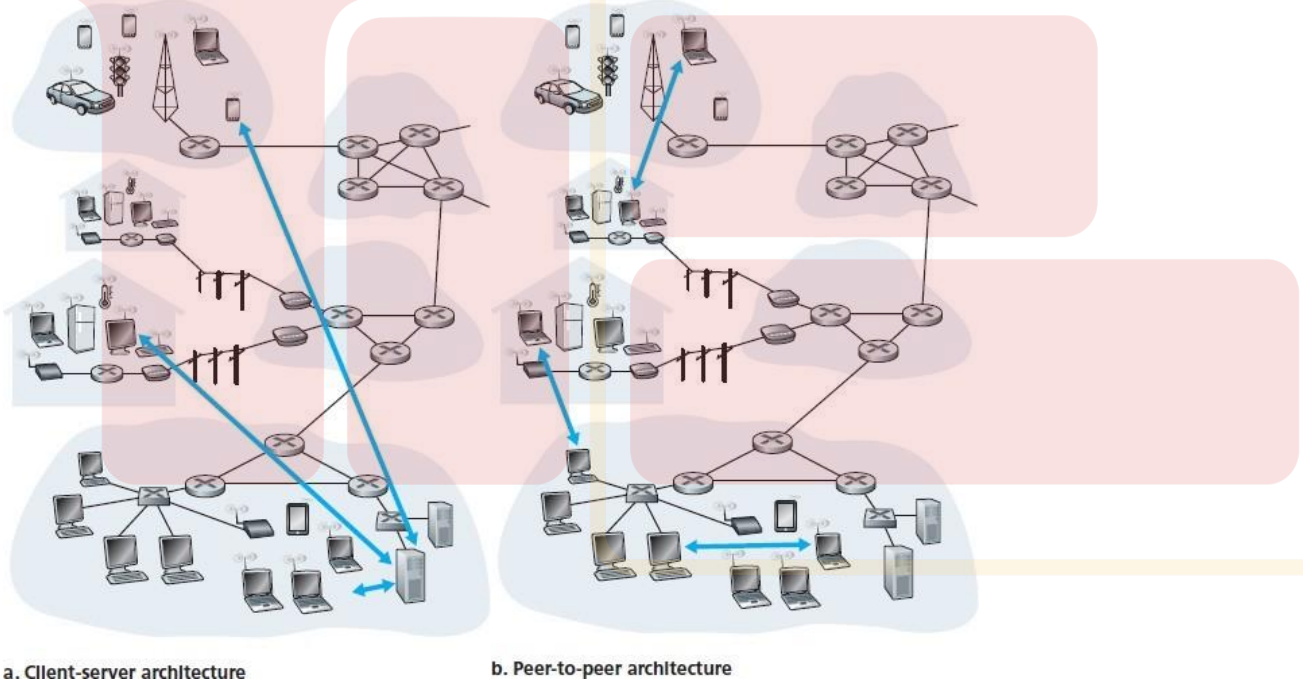



Figure 5.2 (a) Client-server architecture; (b) P2P architecture

- P2P applications face challenges of security, performance, and reliability due to their highly decentralized structure.

## ❖ Processes Communicating

- Before building your network application, you also need a basic understanding of how the programs, running in multiple end systems, communicate with each other. In the jargon of operating systems, it is not actually programs but processes that communicate.
- A process can be thought of as a program that is running within an end system. When processes are running on the same end system, they can communicate with each other with interprocess communication, using rules that are governed by the end system's operating system.
- Processes on two different end systems communicate with each other by exchanging messages across the computer network. A sending process creates and sends messages into the network; a receiving process receives these messages and possibly responds by sending messages back.

### ➢ *Client and Server Processes*

- A network application consists of pairs of processes that send messages to each other over a network.
- The Web application a client browser process exchanges messages with a Web server process.
- With the Web, a browser is a client process and a Web server is a server process.
- *In the context of a communication session between a pair of processes, the process that initiates the communication is labeled as the client. The process that waits to be contacted to begin the session is the server.*
- In the Web, a browser process initializes contact with a Web server process; hence the browser process is the client and the Web server process is the server.

### ➢ *The Interface Between the Process and the Computer Network*

- Any message sent from one process to another must go through the underlying network.
- A process sends messages into, and receives messages from, the network through a software interface called a socket.
- When a process wants to send a message to another process on another host, it shoves the message out its door (socket).
- Once the message arrives at the destination host, the message passes through the receiving process's door (socket), and the receiving process then acts on the message.
- Figure 5.3 illustrates socket communication between two processes that communicate over the Internet.
- It is also referred to as the Application Programming Interface (API) between the application and the network, since the socket is the programming interface with which

network applications are built.

### ➢ *Addressing Processes*

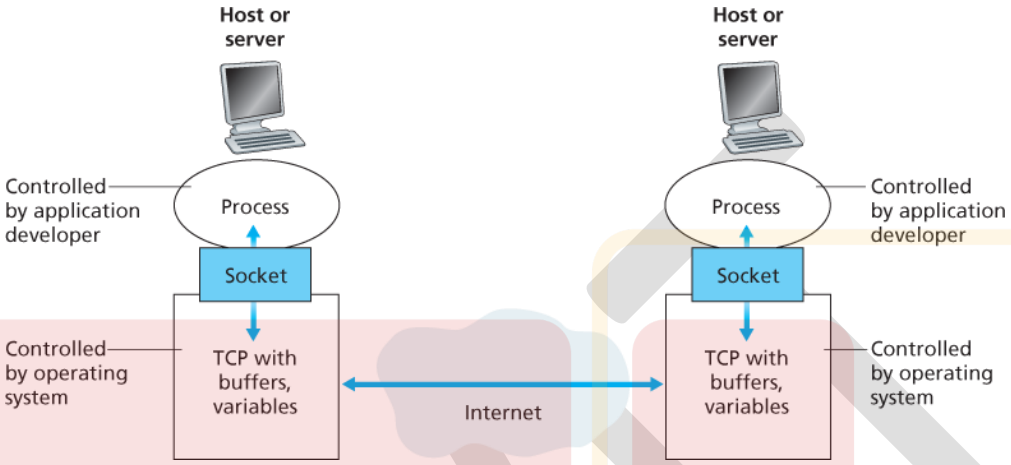



Figure 5.3 Application processes, sockets, and underlying transport protocol

- The only control that the application developer has on the transport-layer side is (1) the choice of transport protocol and (2) perhaps the ability to fix a few transport-layer parameters such as maximum buffer and maximum segment sizes.
- in order for a process running on one host to send packets to a process running on another host, the receiving process needs to have an address.
- Once the application developer chooses a transport protocol (if a choice is available), the application is built using the transport-layer services provided by that protocol.
- Two pieces of information need to be specified: (1) the address of the host and (2) an identifier that specifies the receiving process in the destination host.
- This information is needed because in general a host could be running many network applications.
- A destination port number serves this purpose.
- Popular applications have been assigned specific port numbers.

## ❖ Transport Services Available to Applications

- The application at the sending side pushes messages through the socket. At the other side of the socket, the transport-layer protocol has the responsibility of getting the messages to the socket of the receiving process.
- Many networks, including the Internet, provide more than one transport-layer protocol. When you develop an application, you must choose one of the available transport-layer protocols.
- We can broadly classify the possible services along four dimensions: reliable data

transfer, throughput, timing, and security.

### ➢ *Reliable Data Transfer*

- For many applications—such as electronic mail, file transfer, remote host access, Web document transfers, and financial applications—data loss can have devastating consequences.
- To support these applications, something has to be done to guarantee that the data sent by one end of the application is delivered correctly and completely to the other end of the application.
- One important service that a transport-layer protocol can potentially provide to an application is process-to-process reliable data transfer.
- When a transport protocol provides this service, the sending process can just pass its data into the socket and know with complete confidence that the data will arrive without errors at the receiving process.
- When a transport-layer protocol doesn't provide reliable data transfer, some of the data sent by the sending process may never arrive at the receiving process.
- In these multimedia applications, lost data might result in a small glitch in the audio/video—not a crucial impairment.

### ➢ *Throughput*

- Communication session between two processes along a network path, is the rate at which the sending process can deliver bits to the receiving process.
- Because other sessions will be sharing the bandwidth along the network path, and because these other sessions will be coming and going, the available throughput can fluctuate with time
- These observations lead to another natural service that a transport- layer protocol could provide, namely, guaranteed available throughput at some specified rate.
- The application could request a guaranteed throughput of r bits/sec, and the transport protocol would then ensure that the available throughput is always at least r bits/sec.
- Many current multimedia applications are bandwidth sensitive, although some multimedia applications may use adaptive coding techniques to encode digitized voice or video at a rate that matches the currently available throughput.
- . Electronic mail, file transfer, and Web transfers are all elastic applications.

### ➢ *Timing*

- A transport-layer protocol can also provide timing guarantees. As with throughput guarantees, timing guarantees can come in many shapes and forms.
- Guarantee might be that every bit that the sender pumps into the socket arrives at the receiver's socket no more than 100 msec later.

- Long delays in Internet telephony, for example, tend to result in unnatural pauses in the conversation; in a multiplayer game or virtual interactive environment.
- For non-real-time applications, lower delay is always preferable to higher delay, but no tight constraint is placed on the end-to-end delays.

### ➢ *Security*

- Finally, a transport protocol can provide an application with one or more security services.
- The sending host, a transport protocol can encrypt all data transmitted by the sending process, and in the receiving host, the transport-layer protocol can decrypt the data before delivering the data to the receiving process.
- A transport protocol can also provide other security services in addition to confidentiality, including data integrity and end-point authentication.

## ❖ Transport Services Provided by the Internet

- The Internet (and, more generally, TCP/IP networks) makes two transport protocols available to applications, UDP and TCP.
- When you (as an application developer) create a new network application for the Internet, one of the first decisions you have to make is whether to use UDP or TCP.
- Each of these protocols offers a different set of services to the invoking applications

### ➢ *TCP Services*

- The TCP service model includes a connection-oriented service and a reliable data transfer service.
- When an application invokes TCP as its transport protocol, the application receives both of these services from TCP.

→ <u>Connection-oriented service:</u>

- CP has the client and server exchange transport-layer control information with each other before the application-level messages begin to flow.

| Application | Data Loss | Throughput | Time-Sensitive |
|---|---|---|---|
| File transfer/download | No loss | Elastic | No |
| E-mail | No loss | Elastic | No |
| Web documents | No loss | Elastic (few kbps) | No |
| Internet telephony/ Video conferencing | Loss-tolerant | Audio: few kbps–1Mbps Video: 10 kbps–5 Mbps | Yes: 100s of msec |
| Streaming stored audio/video | Loss-tolerant | Same as above | Yes: few seconds |
| Interactive games | Loss-tolerant | Few kbps–10 kbps | Yes: 100s of msec |
| Smartphone messaging | No loss | Elastic | Yes and no |

Figure 5.4 Requirements of selected network applications

- The connection is a full-duplex connection in that the two processes can send messages to each other over the connection at the same time.
- When the application finishes sending messages, it must tear down the connection.

→ Reliable data transfer service:

- The communicating processes can rely on TCP to deliver all data sent without error and in the proper order.
- TCP also includes a congestion-control mechanism, a service for the general welfare of the Internet rather than for the direct benefit of the communicating processes.
- The TCP congestion-control mechanism throttles a sending process (client or server) when the network is congested between sender and receiver.

### ❖ FOCUS ON SECURITY SECURING TCP

- Neither TCP nor UDP provides any encryption—the data that the sending process passes into its socket is the same data that travels over the network to the destination process.
- Privacy and other security issues have become critical for many applications, the Internet community has developed an enhancement for TCP, called Secure Sockets Layer (SSL).
- The receiving socket passes the encrypted data to SSL, which decrypts the data. Finally, SSL passes the cleartext data through its SSL socket to the receiving process.
- TCP congestion control also attempts to limit each TCP connection to its fair share of network bandwidth.

### ➢ *UDP Services*

- UDP is a no-frills, lightweight transport protocol, providing minimal services.
- UDP is connectionless, so there is no handshaking before the two processes start to

communicate.

- UDP does not include a congestion-control mechanism, so the sending side of UDP can pump data into the layer below (the network layer) at any rate it pleases.

➢ *Services Not Provided by Internet Transport Protocols*

- We have organized transport protocol services along four dimensions: reliable data transfer, throughput, timing, and security.
- TCP can be easily enhanced at the application layer with SSL to provide security services.
- These applications often work fairly well because they have been designed to cope, to the greatest extent possible, with this lack of guarantee.
- many firewalls are configured to block (most types of) UDP traffic, Internet telephony applications often are designed to use TCP as a backup if UDP communication fails.
- Figure 5.5 Popular Internet applications, their application-layer protocols, and their underlying transport protocols

| Application | Application-Layer Protocol | Underlying Transport Protocol |
|---|---|---|
| Electronic mail | SMTP [RFC 5321] | TCP |
| Remote terminal access | Telnet [RFC 854] | TCP |
| Web | HTTP [RFC 2616] | TCP |
| File transfer | FTP [RFC 959] | TCP |
| Streaming multimedia | HTTP (e.g., YouTube) | TCP |
| Internet telephony | SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype) | UDP or TCP |

Figure 5.5 Popular Internet applications, their application-layer protocols, and their underlying transport protocols

### ❖ Application-Layer Protocols

- An application-layer protocol defines how an application's processes, running on different end systems, pass messages to each other

→ In particular, an application-layer protocol defines:

- The types of messages exchanged, for example, request messages and response messages
- The syntax of the various message types, such as the fields in the message and how the fields are delineated
- The semantics of the fields, that is, the meaning of the information in the fields.

- Some application-layer protocols are specified in RFCs and are therefore in the public domain.
- Many other application-layer protocols are proprietary and intentionally not available in

the public domain. For example, Skype uses proprietary application-layer protocols.

- Protocols that define how messages are passed between servers, how messages are passed between servers and mail clients, and how the contents of message headers are to be interpreted.

### ❖ Network Applications Covered in This Book

- we discuss five important applications: the Web, electronic mail, directory service video streaming, and P2P applications.
- We first discuss the Web, not only because it is an enormously popular application, but also because its application-layer protocol, HTTP, is straightforward and easy to understand.
- E-mail is more complex than the Web in the sense that it makes use of not one but several application-layer protocols.
- Most users do not interact with DNS directly; instead, users invoke DNS indirectly through other applications.

## ❖ The Web and HTTP

- The Web was the first Internet application that caught the general public's eye. It dramatically changed, and continues to change, how people interact inside and outside their work environments.
- This is unlike traditional broadcast radio and television, which force users to tune in when the content provider makes the content available.
- It is enormously easy for any individual to make information available over the Web—everyone can become a publisher at extremely low cost.
- Hyperlinks and search engines help us navigate through an ocean of information.

### ➢ Overview of HTTP

- The HyperText Transfer Protocol (HTTP), the Web's application-layer protocol, is at the heart of the Web.
- HTTP is implemented in two programs: a client program and a server program.
- TTP defines the structure of these messages and how the client and server exchange the messages.
- . The base HTML file references the other objects in the page with the objects' URLs.
- Each URL has two components: the hostname of the server that houses the object and the object's path name. For example, the URL

```
http://www.someSchool.edu/someDepartment/picture.gif
```

- Web servers, which implement the server side of HTTP, house Web objects, each addressable by a URL.
- HTTP defines how Web clients request Web pages from Web servers and how servers

transfer Web pages to clients

- The server receives the requests and responds with HTTP response messages that contain the objects.

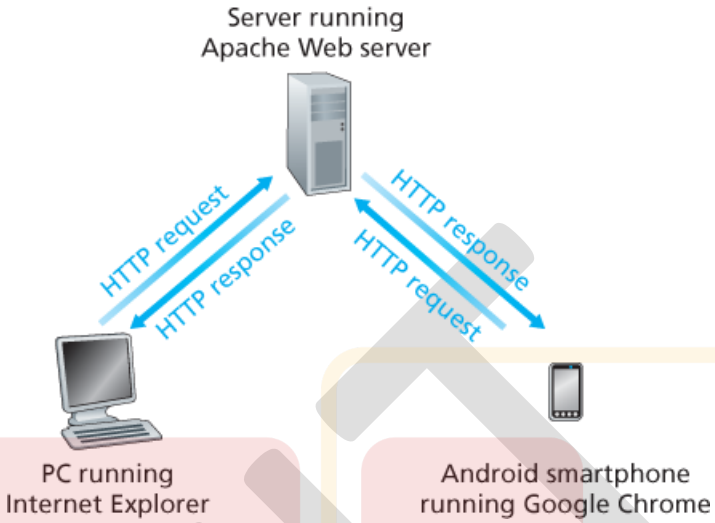



Figure 5.6 HTTP request-response behavior

- It is important to note that the server sends requested files to clients without storing any state information about the client.
- HTTP server maintains no information about the clients, HTTP is said to be a stateless protocol.
- A Web server is always on, with a fixed IP address, and it services requests from potentially millions of different browsers.

### ❖ Non-Persistent and Persistent Connections

- In many Internet applications, the client and server communicate for an extended period of time, with the client making a series of requests and the server responding to each of the requests.
- Depending on the application and on how the application is being used, the series of requests may be made back-to-back, periodically at regular intervals, or intermittently.
- Although HTTP uses persistent connections in its default mode, HTTP clients and servers can be configured to use non-persistent connections instead.

#### ➢ *HTTP with Non-Persistent Connections*

- Let's walk through the steps of transferring a Web page from server to client for the case of non- persistent connections.
- Further suppose the URL for the base HTML file is

```
http://www.someSchool.edu/someDepartment/home.index
```

→ Here is what happens:

1. The HTTP client process initiates a TCP connection to the server www.someSchool.edu on port number 80, which is the default port number for HTTP. Associated with the TCP connection, there will be a socket at the client and a socket at the server.
2. The HTTP client sends an HTTP request message to the server via its socket. The request message includes the path name /someDepartment/home .index . (We will discuss HTTP messages in some detail below.)
3. The HTTP server process receives the request message via its socket, retrieves the object /someDepartment/home.index from its storage (RAM or disk), encapsulates the object in an HTTP response message, and sends the response message to the client via its socket.
4. The HTTP server process tells TCP to close the TCP connection. (But TCP doesn't actually terminate the connection until it knows for sure that the client has received the response message intact.)
5. The HTTP client receives the response message. The TCP connection terminates. The message indicates that the encapsulated object is an HTML file. The client extracts the file from the response message, examines the HTML file, and finds references to the 10 JPEG objects.
6. The first four steps are then repeated for each of the referenced JPEG objects.

- The steps above illustrate the use of non-persistent connections, where each TCP connection is closed after the server sends the object—the connection does not persist for other objects.
- In the steps described above, we were intentionally vague about whether the client obtains the 10 JPEGs over 10 serial TCP connections, or whether some of the JPEGs are obtained over parallel TCP connections.
- Before continuing, let's do a back-of-the-envelope calculation to estimate the amount of time that elapses from when a client requests the base HTML file until the entire file is received by the client.

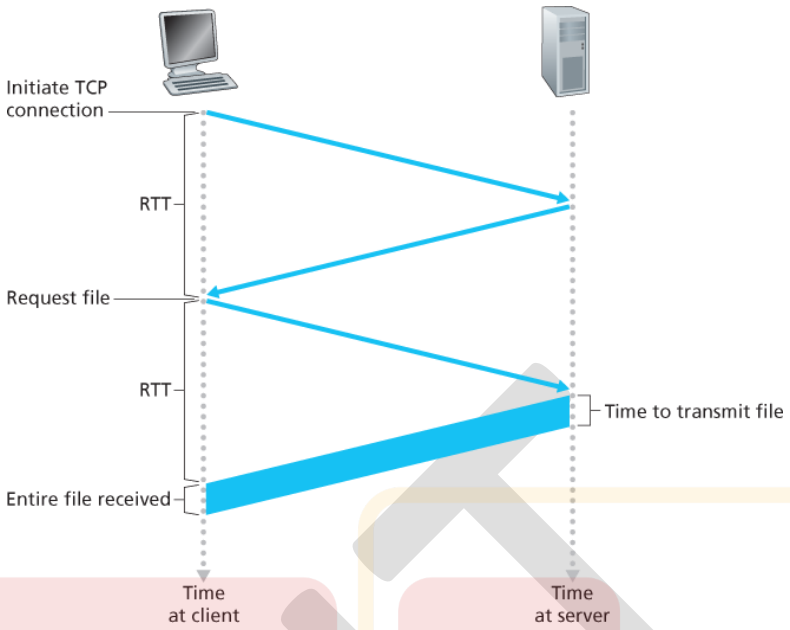



Figure 5.7 Back-of-the-envelope calculation for the time needed to request and receive an HTML file

- TCP connection between the browser and the Web server; this involves a "three-way handshake"—the client sends a small TCP segment to the server, the server acknowledges and responds with a small TCP segment,
- This HTTP request/response eats up another RTT. Thus, roughly, the total response time is two RTTs plus the transmission time at the server of the HTML file.

➢ *HTTP with Persistent Connections*

- Non-persistent connections have some shortcomings. First, a brand-new connection must be established and maintained for each requested object.
- This can place a significant burden on the Web server, which may be serving requests from hundreds of different clients simultaneously.
- Subsequent requests and responses between the same client and server can be sent over the same connection.
- Multiple Web pages residing on the same server can be sent from the server to the same client over a single persistent TCP connection.

➢ **HTTP Message Format**

- The HTTP specifications [RFC 1945; RFC 2616; RFC 7540] include the definitions of the HTTP message formats.
- There are two types of HTTP messages, request messages and response messages, both of which are discussed below.

→ *HTTP Request Message*

- Below we provide a typical HTTP request message:

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0Accept-
language: fr
```

- The first line of an HTTP request message is called the request line; the subsequent lines are called the header lines.
- The great majority of HTTP request messages use the GET method. The GET method is used when the browser requests an object, with the requested object identified in the URL field.
- By including the Connection: close header line, the browser is telling the server that it doesn't want to bother with persistent connections; it wants the server to close the connection after sending the requested object.
- This header line is useful because the server can actually send different versions of the same object to different types of user agents.
- the Accept-language: header indicates that the user prefers to receive a French version of the object, if such an object exists on the server; otherwise, the server should send its default version.
- POST message, the user is still requesting a Web page from the server, but the specific contents of the Web page depend on what the user entered into the form fields.
- The HEAD method is similar to the GET method. When a server receives a request with the HEAD method, it responds with an HTTP message but it leaves out the requested object.
- The PUT method is often used in conjunction with Web publishing tools.
- It allows a user to upload an object to a specific path (directory) on a specific Web server.
- The PUT method is also used by applications that need to upload objects to Web servers.
- The DELETE method allows a user, or an application, to delete an object on a Web server.

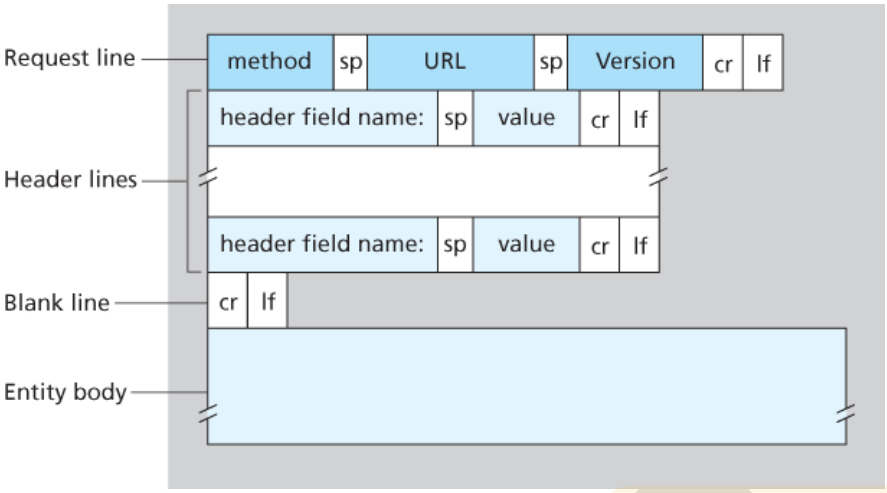



Figure 5.8 General format of an HTTP request message

➢ *HTTP Response Message*

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821 Content-Type: text/html

(data data data data data ...)
```

- The entity body is the meat of the message—it contains the requested object itself (represented by data data data data data ... ).
- The *Date:* header line indicates the time and date when the HTTP response was created and sent by the server.
- *The Server:* header line indicates that the message was generated by an Apache Web serve.
- The *Last- Modified:* header line indicates the time and date when the object was created or last modified.
- *The Content-Type:* header line indicates that the object in the entity body is HTML text.
- *200 OK:* Request succeeded and the information is returned in the response.
- *301 Moved Permanently*: Requested object has been permanently moved; *the new*
- *URL is specified in Location* : header of the response message. The client software will automatically retrieve the new URL.
- *400 Bad Request:* This is a generic error code indicating that the request could not be understood by the server

- *505 HTTP Version Not Supported:* The requested HTTP protocol version is not supported by the server.

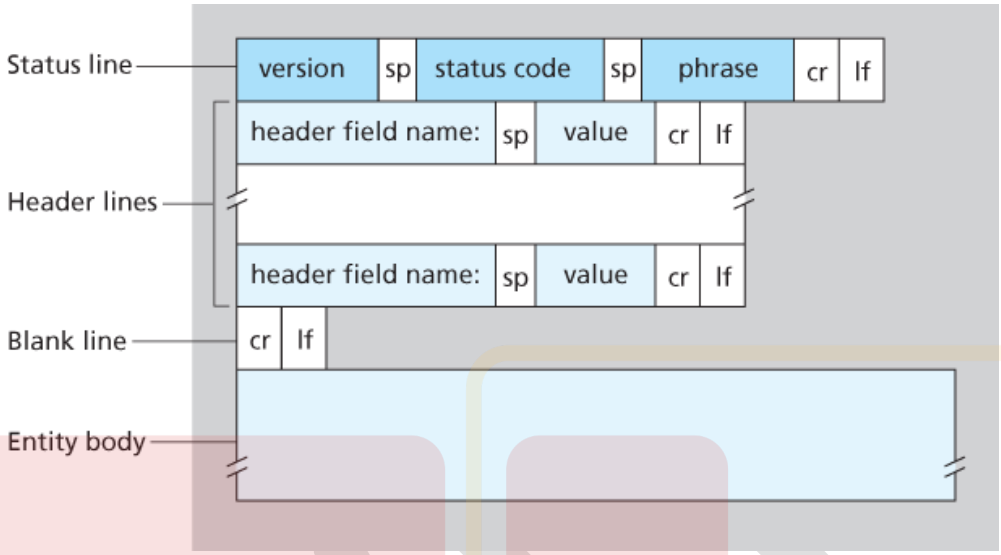



Figure 5.9 General format of an HTTP response message

→ Then type in a one-line request message for some object that is housed on the server. For example, if you have access to a command prompt, type:


```
telnet gaia.cs.umass.edu 80

GET /kurose_ross/interactive/index.php HTTP/1.1

Host: gaia.cs.umass.edu
```

- This opens a TCP connection to port 80 of the host gaia.cs.umass.edu and then sends the HTTP request message.
- A number of header lines that can be used within HTTP request and response messages.

The HTTP specification defines many, many more header lines that can be inserted by browsers, Web servers, and network cache servers.

- Web servers behave similarly: There are different products, versions, and configurations, all of which influence which header lines are included in response messages.

➢ *User-Server Interaction: Cookies*

- This simplifies server design and has permitted engineers to develop high-performance Web servers that can handle thousands of simultaneous TCP connections.
- Susan's browser, including in the HTTP response a Set-cookie: header, which contains the identification number. For example, the header line might be:

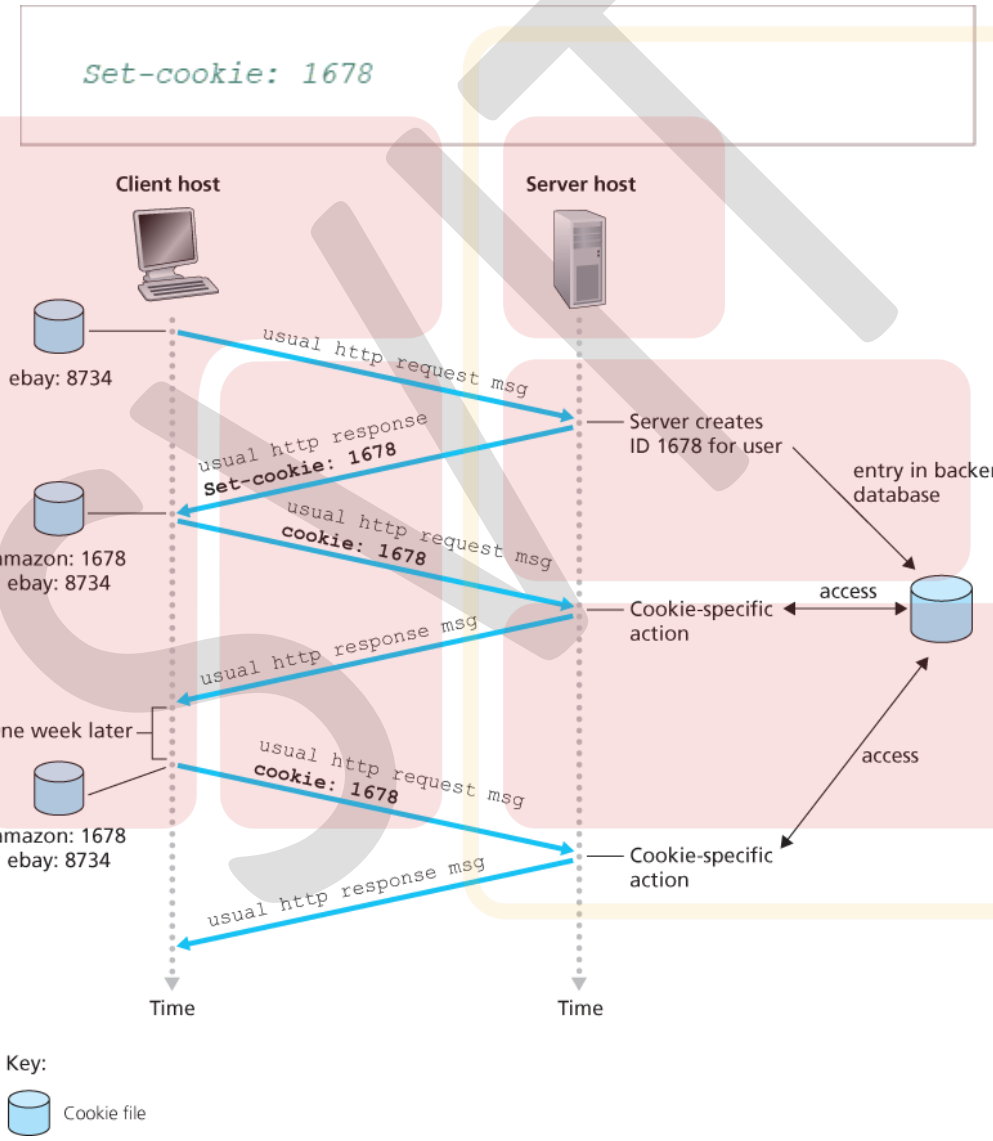



Figure 5.10 Keeping user state with cookies

```
Cookie: 1678
```

- *Cookie: 1678* in the request messages. Amazon also recommends products to Susan based on Web pages she has visited at Amazon in the past.
- During the subsequent sessions, the browser passes a cookie header to the server, thereby identifying the user to the server.
- Although cookies often simplify the Internet shopping experience for the user, they are controversial because they can also be considered as an invasion of privacy.
- Cookie Central [Cookie Central 2016] includes extensive information on the cookie controversy.

## ➢ Web Caching

- A Web cache—also called a proxy server—is a network entity that satisfies HTTP requests on the behalf of an origin Web server.
- The Web cache has its own disk storage and keeps copies of recently requested objects in this storage.
- Once a browser is configured, each browser request for an object is first directed to the Web cache.

→ Here is what happens:

1. The browser establishes a TCP connection to the Web cache and sends an HTTP request for the object to the Web cache.
2. The Web cache checks to see if it has a copy of the object stored locally. If it does, the Web cache returns the object within an HTTP response message to the client browser.
3. If the Web cache does not have the object, the Web cache opens a TCP connection to the origin server, that is, to www.someschool.edu . The Web cache then sends an HTTP request for the object into the cache-to-server TCP connection. After receiving this request, the origin server sends the object within an HTTP response to the Web cache.
4. When the Web cache receives the object, it stores a copy in its local storage and sends a copy, within an HTTP response message, to the client browser (over the existing TCP connection between the client browser and the Web cache).

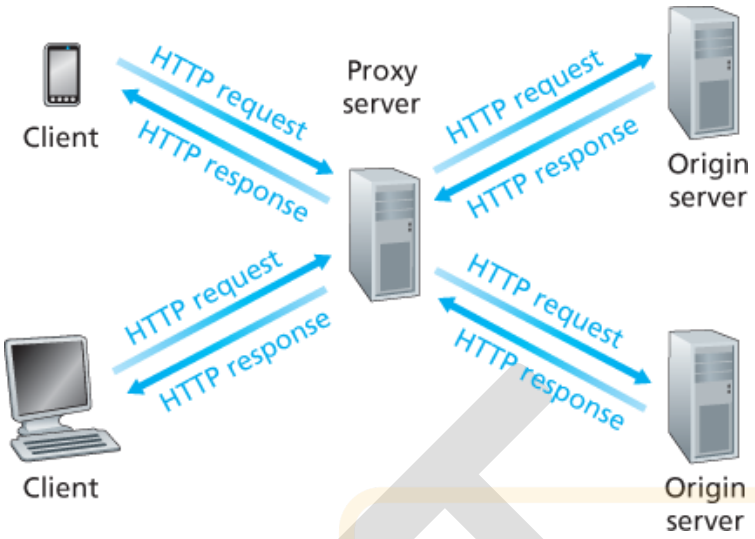



Figure 5.11 Clients requesting objects through a Web cache

- Note that a cache is both a server and a client at the same time. When it receives requests from and sends responses to a browser, it is a server.
- When it sends requests to and receives responses from an origin server, it is a client.
- Typically a Web cache is purchased and installed by an ISP.
- Web caching has seen deployment in the Internet for two reasons. First, a Web cache can substantially reduce the response time for a client request, particularly if the bottleneck bandwidth between the client and the origin server is much less than the bottleneck bandwidth between the client and the cache.
- By reducing traffic, the institution (for example, a company or a university) does not have to upgrade bandwidth as quickly, thereby reducing costs.
- The total response time—that is, the time from the browser's request of an object until its receipt of the object—is the sum of the LAN delay, the access delay (that is, the delay between the two routers), and the Internet delay.
- A traffic intensity of 0.15 on a LAN typically results in, at most, tens of milliseconds of delay; hence, we can neglect the LAN delay.
- Thus, the average response time to satisfy requests is going to be on the order of minutes, if not more, which is unacceptable for the institution's users. Clearly something must be done.
- Through the use of Content Distribution Networks (CDNs), Web caches are increasingly playing an important role in the Internet.

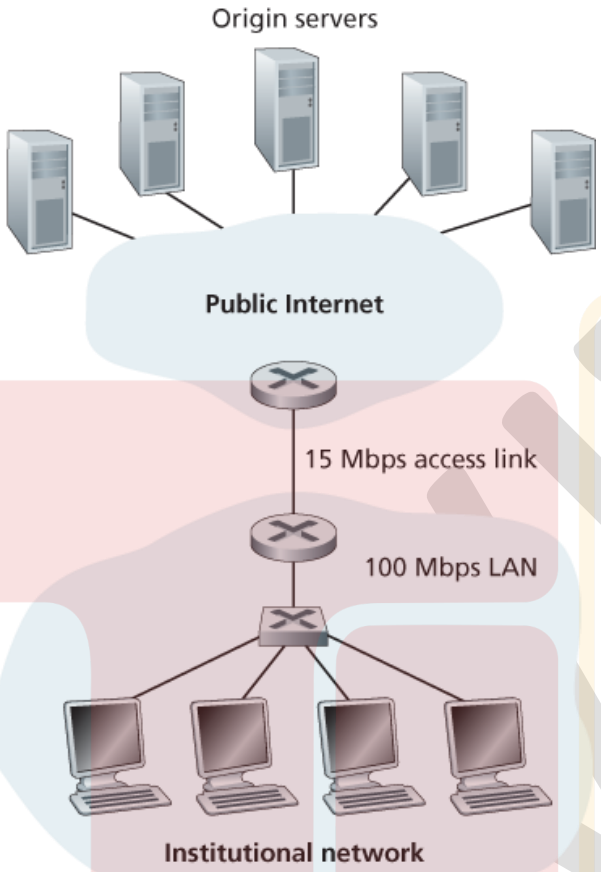



Figure 5.12: Bottleneck between an institutional network and the Internet

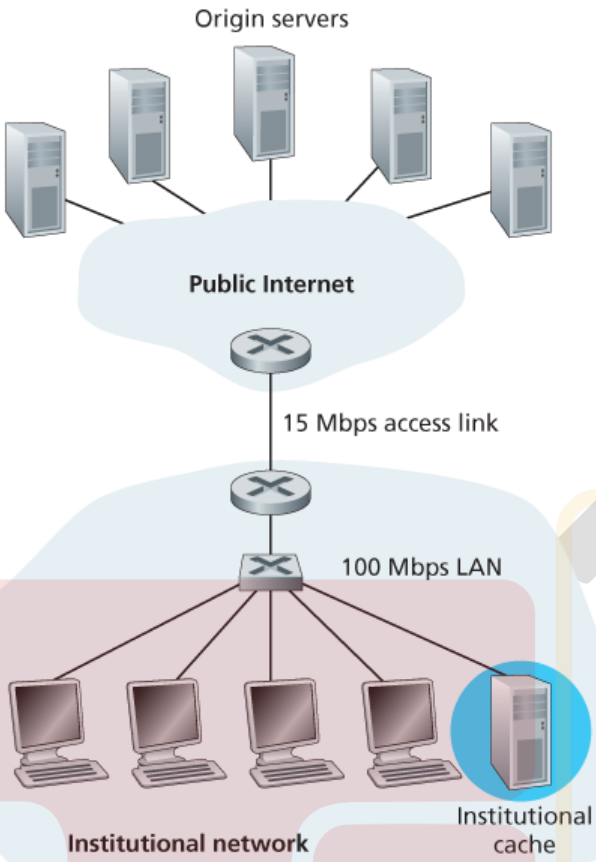



Figure 5.13: Adding a cache to the institutional network

➢ *The Conditional GET*

- Although caching can reduce user-perceived response times, it introduces a new problem—the copy of an object residing in the cache may be stale.
- In other words, the object housed in the Web server may have been modified since the copy was cached at the client.
- To illustrate how the conditional GET operates, let's walk through an example. First, on the behalf of a requesting browser, a proxy cache sends a request message to a Web server:

```
GET /fruit/kiwi.gif HTTP/1.1 Host:
www.exotiquecuisine.com
```

- The cache forwards the object to the requesting browser but also caches the object locally.
- Third, one week later, another browser requests the same object via the cache, and the object is still in the cache.
- Second, the Web server sends a response message with the requested object to the cache.

```
HTTP/1.1 200 OK
Date: Sat, 3 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 9 Sep 2015 09:23:24
Content-Type: image/gif
(data data data data data ...)
```

- If-modified-since: header line is exactly equal to the value of the Last-Modified: header line that was sent by the server one week ago.
- his conditional GET is telling the server to send the object only if the object has been modified since the specified date.

```
HTTP/1.1 304 Not Modified
Date: Sat, 10 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)

(empty entity body)
```

- The Web server still sends a response message but does not include the requested object in the response message.
- Including the requested object would only waste bandwidth and increase user-perceived response time, particularly if the object is large.

## ❖ Electronic Mail in the Internet

- Electronic mail has been around since the beginning of the Internet. It was the most popular application when the Internet was in its infancy [Segaller 1998], and has become more elaborate and powerful over the years.
- In contrast with postal mail, electronic mail is fast, easy to distribute, and inexpensive.
- These components in the context of a sender, Alice, sending an e-mail message to a recipient, Bob. User agents allow users to read, reply to, forward, save, and compose messages.
- Mail servers form the core of the e-mail infrastructure. Each recipient, such as Bob, has a mailbox located in one of the mail servers.

- This diagram that it has three major components: user agents, mail servers, and the Simple Mail Transfer Protocol (SMTP).

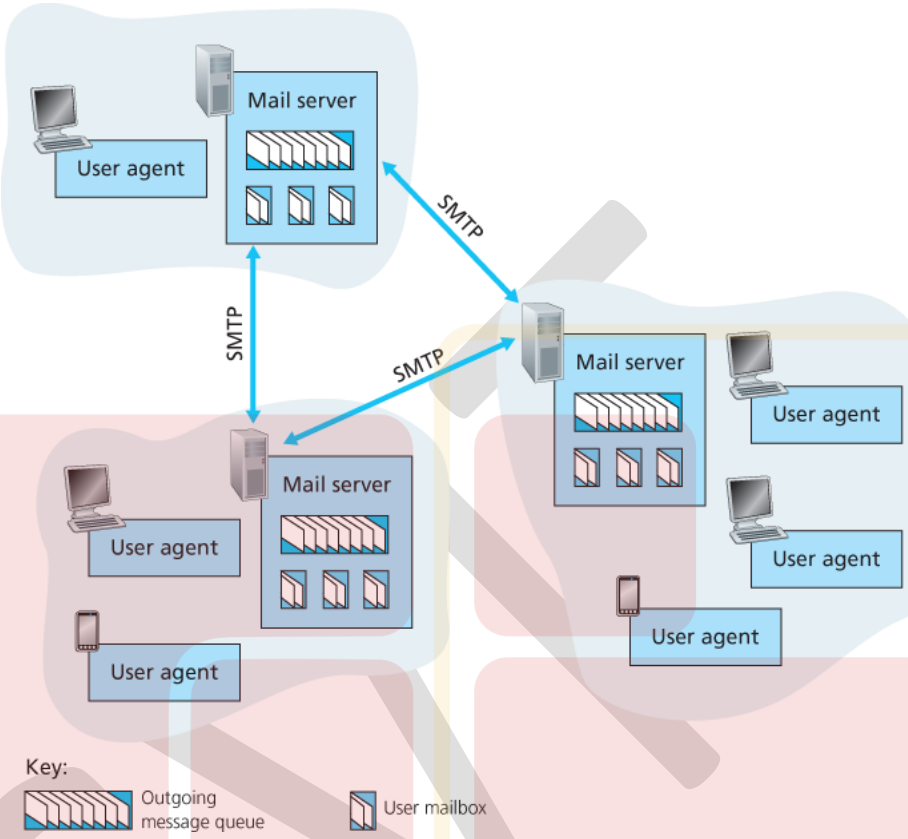



Figure 5.14 A high-level view of the Internet e-mail system

- typical message starts its journey in the sender's user agent, travels to the sender's mail server, and travels to the recipient's mail server, where it is deposited in the recipient's mailbox.
- Alice's mail server must also deal with failures in Bob's mail server.
- If Alice's server cannot deliver mail to Bob's server, Alice's server holds the message in a message queue and attempts to transfer the message later.
- SMTP is the principal application-layer protocol for Internet electronic mail. It uses the reliable data transfer service of TCP to transfer mail from the sender's mail server to the recipient's mail server.
- When a mail server sends mail to other mail servers, it acts as an SMTP client.
- When a mail server receives mail from other mail servers, it acts as an SMTP server.

→ **SMTP**

- SMTP, defined in RFC 5321, is at the heart of Internet electronic mail. As mentioned above, SMTP transfers messages from senders' mail servers to the recipients' mail servers.
- SMTP is much older than HTTP

➢ To illustrate the basic operation of SMTP, let's walk through a common scenario. Suppose Alice wants to send Bob a simple ASCII message.

1. Alice invokes her user agent for e-mail, provides Bob's e-mail address (for example, bob@someschool.edu ), composes a message, and instructs the user agent to send the message.
2. Alice's user agent sends the message to her mail server, where it is placed in a message queue.
3. The client side of SMTP, running on Alice's mail server, sees the message in the message queue. It opens a TCP connection to an SMTP server, running on Bob's mail server.
4. After some initial SMTP handshaking, the SMTP client sends Alice's message into the TCP connection.
5. At Bob's mail server, the server side of SMTP receives the message. Bob's mail server then places the message in Bob's mailbox.
6. Bob invokes his user agent to read the message at his convenience.

→ The scenario is summarized in Figure 5.15.

- It is important to observe that SMTP does not normally use intermediate mail servers for sending mail, even when the two mail servers are located at opposite ends of the world.

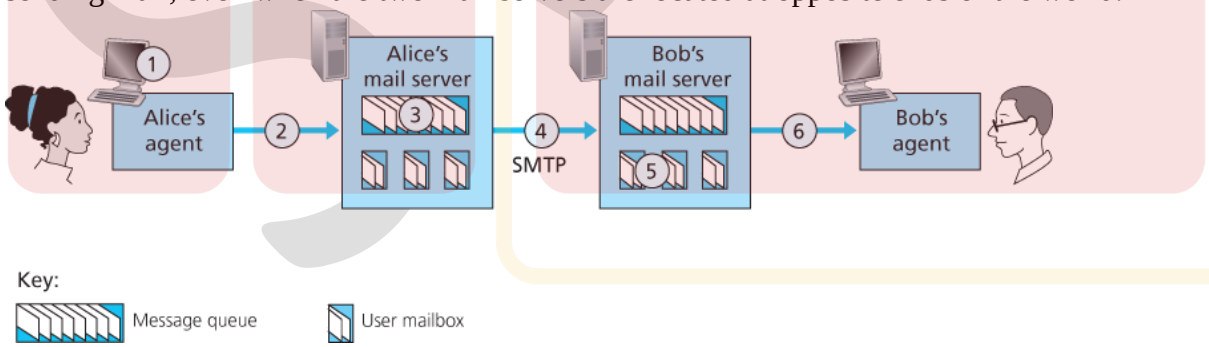



Figure 5.15 Alice sends a message to Bob

- In particular, if Bob's mail server is down, the message remains in Alice's mail server and waits for a new attempt—the message does not get placed in some intermediate mail server

- The client then repeats this process over the same TCP connection if it has other messages to send to the server; otherwise, it instructs TCP to close the connection

```
S: 220 hamburger.edu
C: HELO crepes.fr
S:  250 Hello crepes.fr, pleased to meet you
```

- The hostname of the client is crepes.fr and the hostname of the server is hamburger.edu
- The following transcript begins as soon as the TCP connection is established.

```
C:  MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr ... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickels?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

- As part of the dialogue, the client issued five commands: HELO (an abbreviation for HELLO), MAIL FROM , RCPT TO , DATA , and QUIT .
- For each message, the client begins the process with a new MAIL FROM: crepes.fr , designates the end of message with an isolated period, and issues QUIT only after all messages have been sent.

```
telnet serverName 25
```

- It is highly recommended that you use Telnet to carry out a direct dialogue with an SMTP server. To do this, issue where serverName is the name of a local mail server.

## ❖ Comparison with HTTP

- SMTP with HTTP. Both protocols are used to transfer files from one host to another: HTTP transfers files (also called objects) from a Web server to a Web client (typically a browser); SMTP transfers files (that is, e-mail messages) from one mail server to another

mail server.

- SMTP is primarily a push protocol—the sending mail server pushes the file to the receiving mail server.
- A second difference, which we alluded to earlier, is that SMTP requires each message, including the body of each message, to be in 7-bit ASCII format.
- A third important difference concerns how a document consisting of text and images (along with possibly other media types) is handled

## ❖ Mail Message Formats

- Throughout this discussion we have tacitly assumed that Bob reads his mail by logging onto the server host and then executing a mail reader that runs on that host.
- Consider the path an e-mail message takes when it is sent from Alice to Bob. We just learned that at some point along the path the e-mail message needs to be deposited in Bob's mail server.
- This could be done simply by having Alice's user agent send the message directly to Bob's mail server.
- SMTP has been designed for pushing e-mail from one host to another.
- Alice's user agent doesn't have any recourse to an unreachable destination mail server.

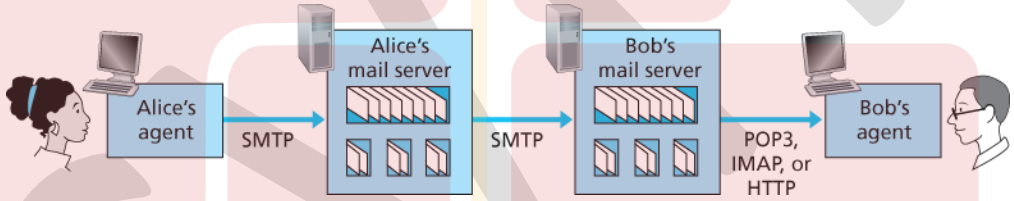



Figure 5.16 E-mail protocols and their communicating entities

- By having Alice first deposit the e-mail in her own mail server, Alice's mail server can repeatedly try to send the message to Bob's mail server, say every 30 minutes, until Bob's mail server becomes operational.
- There are currently a number of popular mail access protocols, including Post Office Protocol—Version 3 (POP3), Internet Mail Access Protocol (IMAP), and HTTP.
- A mail access protocol, such as POP3, is used to transfer mail from the recipient's mail server to the recipient's user agent.

→ *POP3*

- POP3 is an extremely simple mail access protocol. It is defined in [RFC 1939], which is short and quite readable. Because the protocol is so simple, its functionality is rather limited.
- During the first phase, authorization, the user agent sends a username and a password (in the clear) to authenticate the user.

- In a POP3 transaction, the user agent issues commands, and the server responds to each command with a reply.
- There are two possible responses: +OK (sometimes followed by server-to-client data), used by the server to indicate that the previous command was fine; and -ERR .
- To illustrate these two commands, we suggest that you Telnet directly into a POP3 server, using port 110, and issue these commands.

```
telnet mailServer 110
+OK POP3 server ready
user bob
+OK
pass hungry
+OK user successfully logged on
```

The authorization phase has two principal commands: user <username> and pass <password> .

- The user agent first asks the mail server to list the size of each of the stored messages. The user agent then retrieves and deletes each message from the server.

```
S: .
C: retr 1
S: (blah blah ...
S: .................
S: ..... blah)
S: .
C: dele 1
C: retr 2
S: (blah blah ...
S: .................
S: ..... blah)
S: .
C: dele 2C:
quit
S: +OK POP3 server signing off
```

- list , retr , dele , and quit . The syntax for these commands is defined in RFC 1939.

```
C: listS:
1498
S: 2 912
```

- During a POP3 session between a user agent and the mail server, the POP3 server maintains some state information; in particular, it keeps track of which user messages have been marked deleted.
- This lack of state information across sessions greatly simplifies the implementation of a POP3 server.

→ *IMAP*

- with POP3 access, once Bob has downloaded his messages to the local machine, he can create mail folders and move the downloaded messages into the folders.

- the IMAP protocol, defined in [RFC 3501], was invented. Like POP3, IMAP is a mail access protocol. I
- An IMAP server will associate each message with a folder; when a message first arrives at the server, it is associated with the recipient's INBOX folder.
- The IMAP protocol provides commands to allow users to create folders and move messages from one folder to another.
- IMAP also provides commands that allow users to search remote folders for messages matching specific criteria.
- Another important feature of IMAP is that it has commands that permit a user agent to obtain components of messages.

→ *Web-Based E-Mail*

- More and more users today are sending and accessing their e-mail through their Web browsers.
- When a recipient, such as Bob, wants to access a message in his mailbox, the e-mail message is sent from Bob's mail server to Bob's browser using the HTTP protocol rather than the POP3 or IMAP protocol.
- When a sender, such as Alice, wants to send an e-mail message, the e-mail message is sent from her browser to her mail server over HTTP rather than over SMTP.
- Alice's mail server, however, still sends messages to, and receives messages from, other mail servers using SMTP.

## ❖ DNS—The Internet's Directory Service

- An IP address consists of four bytes and has a rigid hierarchical structure.
- where each period separates one of the bytes expressed in decimal notation from 0 to 255.
- An IP address is hierarchical because as we scan the address from left to right, we obtain more and more specific information about where the host is located in the Internet.
- when we scan a postal address from bottom to top, we obtain more and more specific information about where the addressee is located.

→ **Services Provided by DNS**

- We have just seen that there are two ways to identify a host—by a hostname and by an IP address.
- People prefer the more mnemonic hostname identifier, while routers prefer fixed-length, hierarchically structured IP addresses.
- DNS is commonly employed by other application-layer protocols—including HTTP and SMTP to translate user-supplied hostnames to IP addresses.

✓ This is done as follows.

1. The same user machine runs the client side of the DNS application.

2. The browser extracts the hostname, www.someschool.edu , from the URL and passes the hostname to the client side of the DNS application.
3. The DNS client sends a query containing the hostname to a DNS server.
4. The DNS client eventually receives a reply, which includes the IP address for the hostname.
5. Once the browser receives the IP address from DNS, it can initiate a TCP connection to the HTTP server process located at port 80 at that IP address.

- the desired IP address is often cached in a "nearby" DNS server, which helps to reduce DNS network traffic as well as the average DNS delay.

DNS provides a few other important services in addition to translating hostnames to IP addresses:

*Host aliasing:* A host with a complicated hostname can have one or more alias names. For example, a hostname such as relay1.west-coast.enterprise.com could have, say, two aliases such as enterprise.com and www.enterprise.com .

*Mail server aliasing:* For obvious reasons, it is highly desirable that e-mail addresses be mnemonic. For example, if Bob has an account with Yahoo Mail, Bob's e-mail address might be as simple as bob@yahoo.mail.

*Load distribution:* DNS is also used to perform load distribution among replicated servers, such as replicated Web servers. Busy sites, such as cnn.com , are replicated over multiple servers, with each server running on a different end system and each having a different IP address.

PRINCIPLES IN PRACTICE

DNS: CRITICAL NETWORK FUNCTIONS VIA THE CLIENT-SERVER PARADIGM

- The role of the DNS is quite different from Web, file transfer, and e-mail applications.
- Unlike these applications, the DNS is not an application with which a user directly interacts. Instead, the DNS provides a core Internet function—namely, translating hostnames to their underlying IP addresses, for user applications and other software in the Internet.
- The DNS, which implements the critical name-to- address translation process using clients and servers located at the edge of the network, is yet another example of that design philosophy.

➢ **Overview of How DNS Works**

- The application will invoke the client side of DNS, specifying the hostname that needs to be translated.
- DNS in the user's host then takes over, sending a query message into the network.

- This mapping is then passed to the invoking application.
- A simple design for DNS would have one DNS server that contains all the mappings.
- Although the simplicity of this design is attractive, it is inappropriate for today's Internet, with its vast (and growing) number of hosts.

→ The problems with a centralized design include:

- A single point of failure. If the DNS server crashes, so does the entire Internet!
- Traffic volume. A single DNS server would have to handle all DNS queries (for all the HTTP requests and e-mail messages generated from hundreds of millions of hosts).
- Distant centralized database. A single DNS server cannot be "close to" all the querying clients. If we put the single DNS server in New York City, then all queries from Australia must travel to the other side of the globe, perhaps over slow and congested links. This can lead to significant delays.
- Maintenance. The single DNS server would have to keep records for all Internet hosts. Not only would this centralized database be huge, but it would have to be updated frequently to account for every new host.

➢ *A Distributed, Hierarchical Database*

- DNS server has all of the mappings for all of the hosts in the Internet. Instead, the mappings are distributed across the DNS servers.

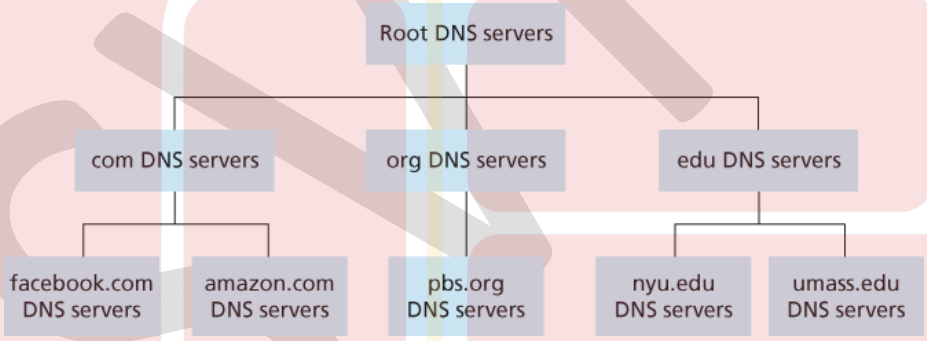



Figure 5.17 Portion of the hierarchy of DNS servers

- The client first contacts one of the root servers, which returns IP addresses for TLD servers for the top-level domain com .

*Root DNS servers:* There are over 400 root name servers scattered all over the world. Figure 5.18 shows the countries that have root names servers, with countries having more than ten darkly shaded.

*Top-level domain (TLD) servers:* For each of the top-level domains — top-level domains such as com, org, net, edu, and gov, and all of the country top-level domains such as uk, fr, ca, and jp — there is TLD server (or server cluster).

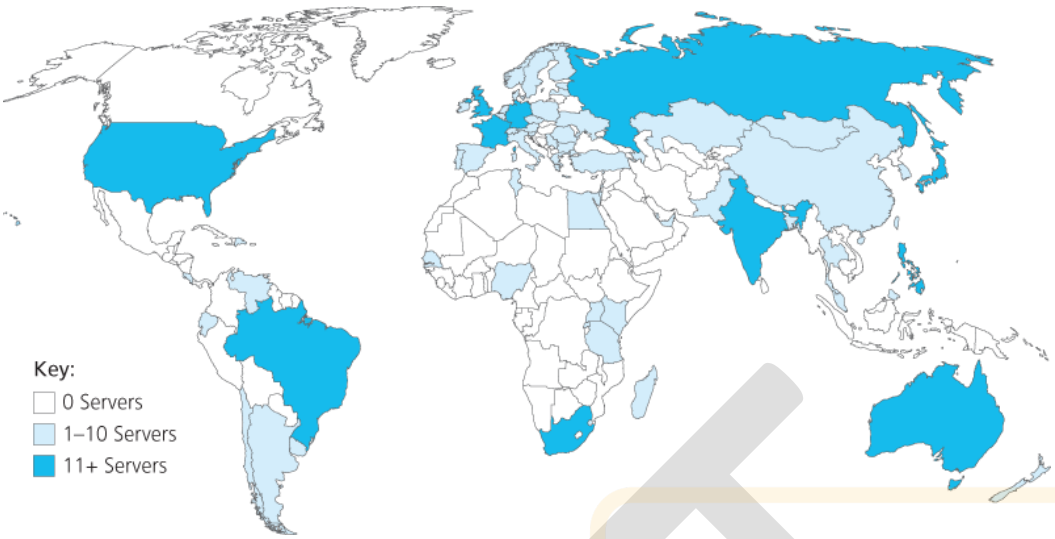



Figure 5.18 DNS root servers in 2016

*Authoritative DNS servers:* Every organization with publicly accessible hosts (such as Web servers and mail servers) on the Internet must provide publicly accessible DNS records that map the names of those hosts to IP addresses.

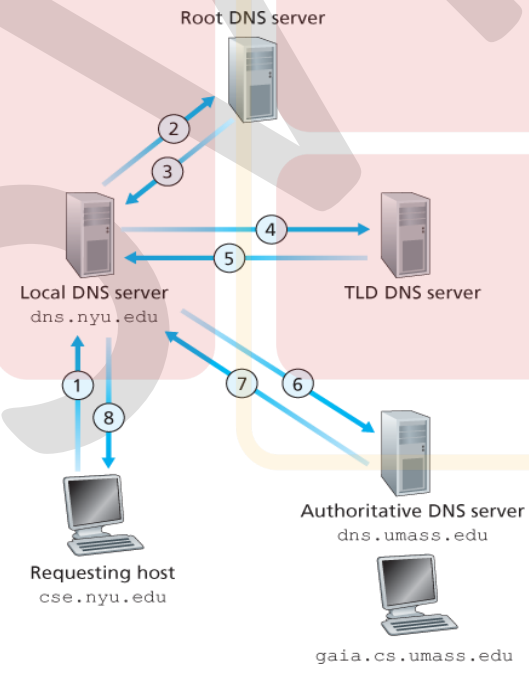



Figure 5.19 Interaction of the various DNS servers

- The TLD server knows the authoritative DNS server for the hostname. In general this not always true. Instead, the TLD server may know only of an intermediate DNS server, which in turn knows the authoritative DNS server for the hostname.

➢ *DNS Caching*

- DNS extensively exploits DNS caching in order to improve the delay performance and to reduce the number of DNS messages ricocheting around the Internet.
- when a DNS server receives a DNS reply (containing, for example, a mapping from a hostname to an IP address), it can cache the mapping in its local memory.
- DNS servers discard cached information after a period of time.
- A local DNS server can also cache the IP addresses of TLD servers, thereby allowing the local DNS server to bypass the root DNS servers in a query chain.

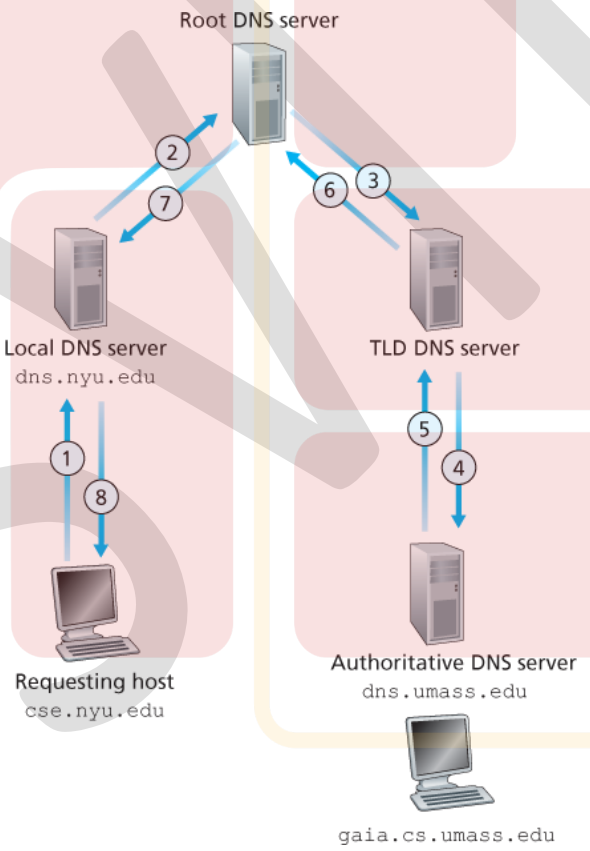



Figure 5.20 Recursive queries in DNS

Recursive queries in DNS

- **DNS Records and Messages**
  - The DNS servers that together implement the DNS distributed database store resource records (RRs), including RRs that provide hostname-to-IP address mappings.

```
(Name, Value, Type, TTL)
```

→ four-tuple that contains the following fields:

- TTL is the time to live of the resource record; it determines when a resource should be removed from a cache. In the example records given below, we ignore the TTL field. The meaning of Name and Value depend on Type : If Type=A , then Name is a hostname and Value is the IP address for the hostname. Thus, a Type A record provides the standard hostname-to-IP address mapping. As an example, (relay1.bar.foo.com, 145.37.93.126, A) is a Type A record.
- If Type=NS , then Name is a domain (such as foo.com ) and Value is the hostname of an authoritative DNS server that knows how to obtain the IP addresses for hosts in the domain. This record is used to route DNS queries further along in the query chain. As an example, (foo.com,dns.foo.com, NS) is a Type NS record.
- If Type=CNAME , then Value is a canonical hostname for the alias hostname Name . This record can provide querying hosts the canonical name for a hostname. As an example, (foo.com, relay1.bar.foo.com, CNAME) is a CNAME record.

- If Type=MX , then Value is the canonical name of a mail server that has an alias hostname Name . As an example, (foo.com, mail.bar.foo.com, MX) is an MX record. MX records allow the hostnames of mail servers to have simple aliases. Note that by using the MX record, a company can have the same aliased name for its mail server and for one of its other servers (such as its Web server). To obtain the canonical name for the mail server, a DNS client would query for an MX record; to obtain the canonical name for the other server, the DNS client would query for the CNAME record.

→ *DNS Messages*

- These are the only two kinds of DNS messages. Furthermore, both query and reply messages have the same format, as shown in Figure 5.21.The semantics of the various fields in a DNS message are as follows:

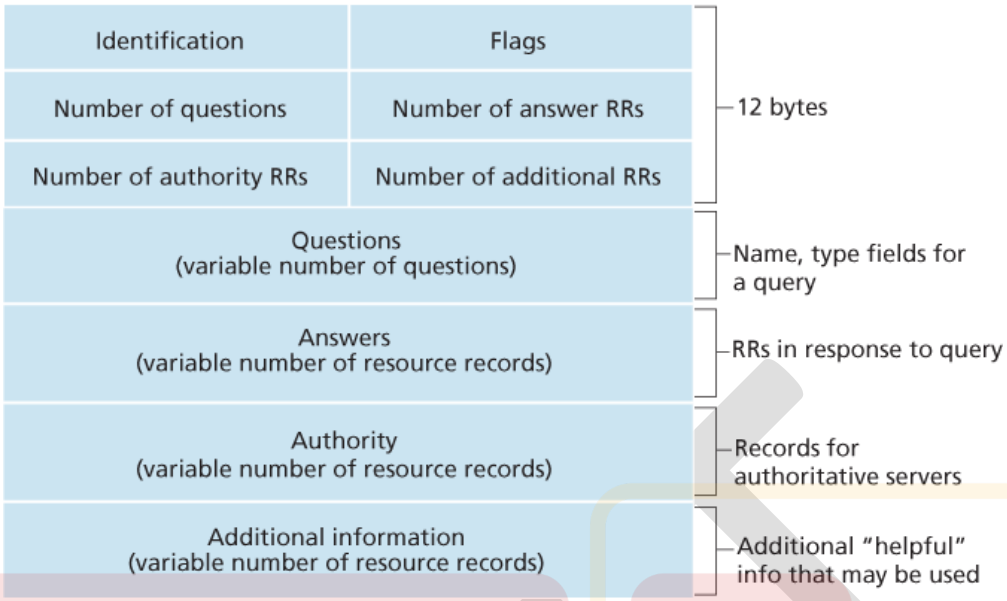



Figure 5.21 DNS message format

→ *Inserting Records into the DNS Database*

- A registrar is a commercial entity that verifies the uniqueness of the domain name, enters the domain name into the DNS database (as discussed below), and collects a small fee from you for its services.

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

- When you register the domain name networkutopia.com with some registrar, you also need to provide the registrar with the names and IP addresses of your primary and secondary authoritative DNS servers.

➢ FOCUS ON SECURITY DNS VULNERABILITIES

- The first type of attack that comes to mind is a DDoS bandwidth-flooding attack (see Section 1.6) against DNS servers.
- large-scale attack caused minimal damage, having little or no impact on users' Internet experience.
- The attackers did succeed at directing a deluge of packets at the root servers.
- A potentially more effective DDoS attack against DNS would be send a deluge of DNS queries to top-level-domain servers, for example, to all the top-level-domain servers that handle the .com domain.
- DNS could potentially be attacked in other ways. In a man-in-the-middle attack, the attacker intercepts queries from hosts and returns bogus replies.
- The TLD com server sends a reply to Alice's local DNS server, with the reply containing the two resource records.