PIE Tech

POLLACHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(Approved by **AICTE** and Affiliated to **Anna University**)

sky is the limit

Department of Computer Science and Engineering

Regulation 2021

III Year – V Semester

CS3591- Computer Networks

UNIT 1

FUNDAMENTALS & LINK LAYER

Building a Network

A computer network consists of two or more computers that are linked in order to share resources, exchange data files or to allow electronic communication.

Applications

- World Wide Web
- Email
- Streaming audio and video
- Chat rooms
- music (file) sharing

Requirements

The first step is to identify the set of constraints and requirements that influence network design. Before getting started, however, it is important to understand that the expectations you have of a network depend on your perspective:

■ An *application programmer* would list the services that his application needs, for example, a guarantee that each message the application sends will be delivered without error within a certain amount of time.

■ A *network designer* would list the properties of a cost-effective design, for example, that network resources are efficiently utilized and fairly allocated to different users.

■ A *network provider* would list the characteristics of a system that is easy to administer and manage, for example, in which faults can be easily isolated and where it is easy to account for usage.

(1) Connectivity

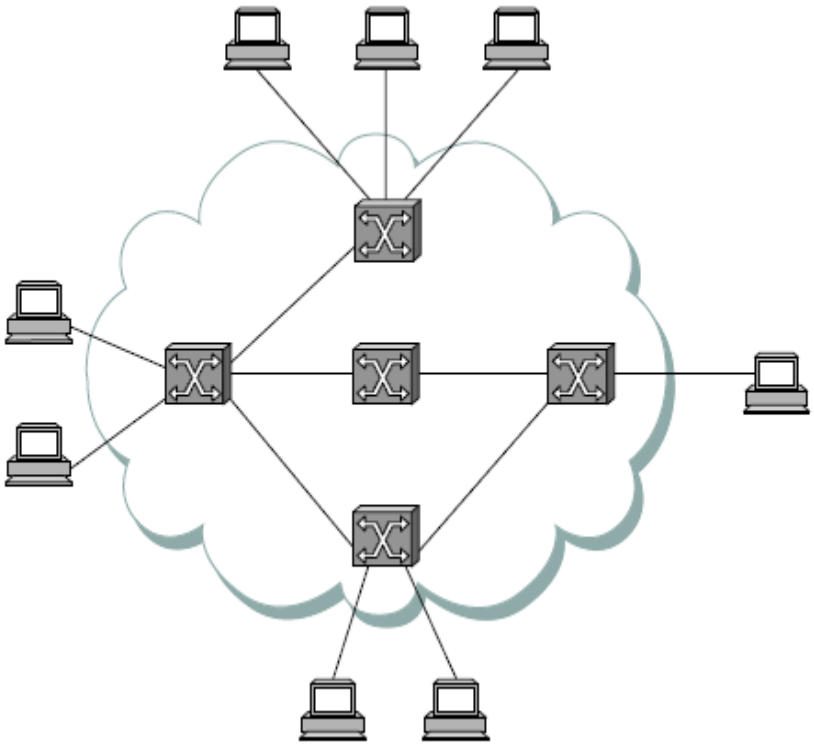
A network must provide connectivity among a set of computers. Sometimes it is enough to build a limited network that connects only a few select machines.

Links, Nodes, and Clouds

Network connectivity occurs at many different levels. At the lowest level, a network can consist of two or more computers directly connected by some physical medium, such as a coaxial

cable or an optical fiber. We call such a physical medium a *link*, and we often refer to the computers it connects as *nodes*.

Those nodes that are attached to at least two links run software that forwards data received on one link out on another. If organized in a systematic way, these forwarding nodes form a *switched network*. There are numerous types of switched networks, of which the two most common are *circuit-switched* and *packet-switched*.

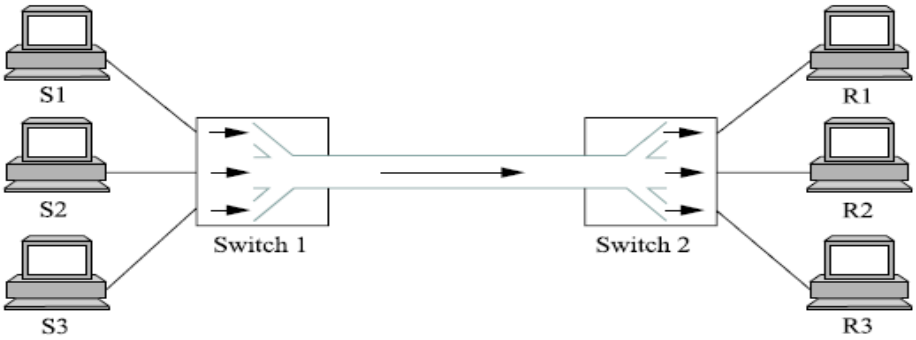


Switched Network

(2) Cost Effective Resource Sharing

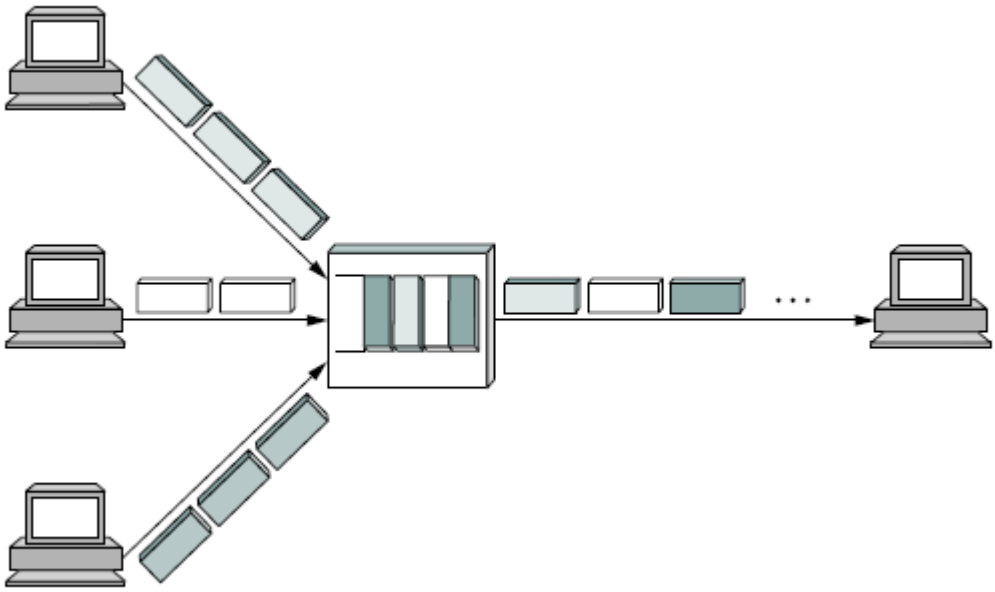
This section explains the key requirement of computer networks—efficiency—that lead us to packet switching as the strategy of choice.

To understand how hosts share a network, we need to introduce a fundamental concept, *multiplexing*, which means that a system resource is shared among multiple users.



There are several different methods for multiplexing multiple flows onto one physical link. One common method is *synchronous time-division multiplexing (STDM).*

In other words, during time quantum 1, data from S1 to R1 is transmitted; during time quantum 2, data from S2 to R2 is transmitted; in quantum 3, S3 sends data to R3. At this point, the first flow (S1 to R1) gets to go again, and the process repeats. Another method is *frequency-division multiplexing (FDM).*



(3) Support for Common Services

One way to characterize networks is according to their size. Two well known examples are local area networks (LANs) and wide area networks (WANs); the former typically extend less than 1 km, while the latter can be worldwide. Other networks are classified as metropolitan area networks

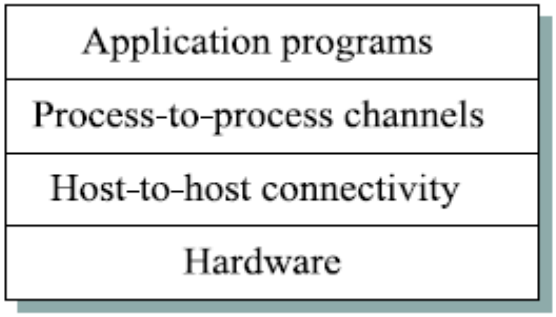
(MANs)

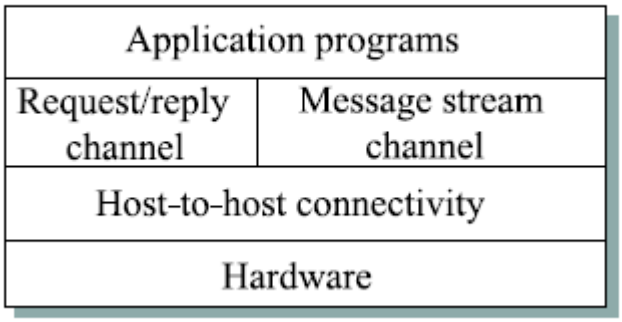
Layering and Protocols

When a system gets complex, the system designer introduces another level of abstraction. The idea of an abstraction is to define a unifying model that can capture some important aspect of the system, encapsulate this model in an object that provides an interface that can be manipulated by other components of the system

Abstractions naturally lead to layering, especially in network systems. The general idea is that you start with the services offered by the underlying hardware, and then add a sequence of layers, each providing a higher (more abstract) level of service. The services provided at the high layers are implemented in terms of the services provided by the low layers.

Layering provides two nice features. First, it decomposes the problem of building a network into more manageable components. Rather than implementing a monolithic piece of software that does everything you will ever want, you can implement several layers, each of which solves one part of the problem. Second, it provides a more modular design. If you decide that you want to add some new service, you may only need to modify the functionality at one layer, reusing the functions provided at all the other layers.



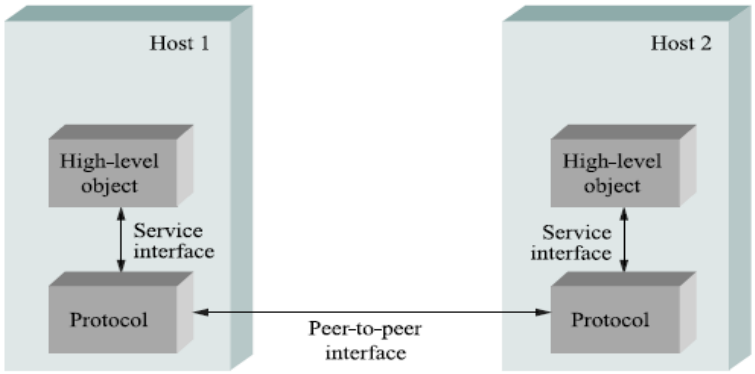


The abstract objects that make up the layers of a network system are called *protocols*. Each pr5otocol defines two different interfaces.

- Service interfaces
- Peer interfaces

First, it defines a *service interface* to the other objects on the same computer that want to use its communication services.

Second, a protocol defines a *peer interface* to its counterpart (peer) on another machine. This second interface defines the form and meaning of messages exchanged between protocol peers to implement the communication service.



Service and Peer Interface

To summarize, a protocol defines a communication service that it exports locally, along with a set of rules governing the messages that the protocol exchanges with its peer(s) to implement this service (the peer interface). This situation is illustrated below.

In addition, there are potentially multiple protocols at any given level, each providing a different communication service. We therefore represent the suite of protocols that make up a network system with a *protocol graph*. The nodes of the graph correspond to protocols, and the edges represent a *depends on* relation. In this example, suppose that the file access program on host 1 wants to send a message to its peer on host 2 using the communication service offered by protocol RRP.

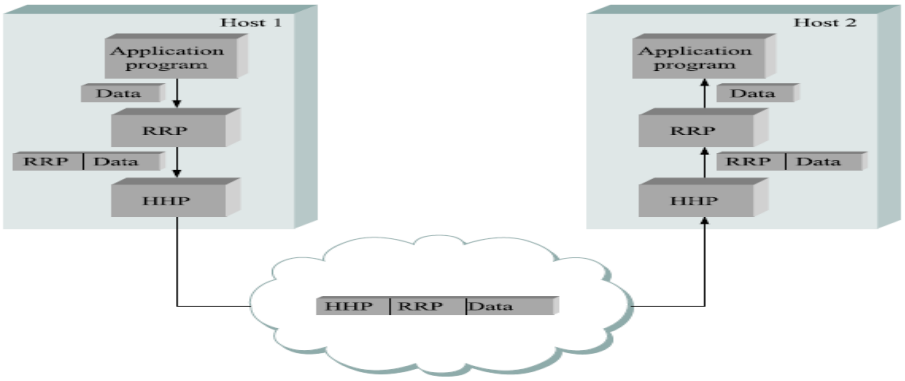
Encapsulation

When one of the application programs sends a message to its peer by passing the message to protocol RRP. From RRP's perspective, the message it is given by the application is an uninterrupted string of bytes.

RRP does this by attaching a *header* to the message. Generally speaking, a header is a small data structure—from a few bytes to a few dozen bytes—that is used among peers to

communicate with each other. In some cases, however, this peer-to-peer control information is sent at the end of the message, in which case it is called a *trailer*.

The rest of the message—that is, the data being transmitted on behalf of the application—is called the message's *body* or *payload*.



High-level messages are encapsulated inside of low-level messages

This process of encapsulation is then repeated at each level of the protocol graph; for example, HHP encapsulates RRP's message by attaching a header of its own. If we now assume that HHP sends the message to its peer over some network, then when the message arrives at the destination host, it is processed in the opposite order.

Internet Architecture

The Internet architecture, which is also sometimes called the TCP/IP architecture after its two main protocols, is depicted. An alternative representation is given in Figure 1.15. The Internet architecture evolved out of experiences with an earlier packet-switched network called the ARPANET. Both the Internet and the ARPANET were funded by the Advanced Research Projects Agency (ARPA), one of the R&D funding agencies of the U.S. Department of Defense.

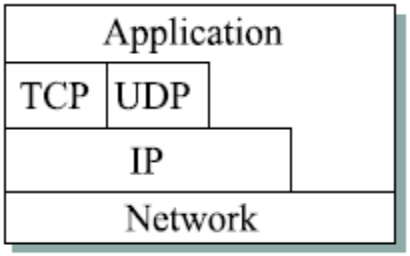
While the seven-layer OSI model can, with some imagination, be applied to the Internet, a four-layer model is often used instead. At the lowest level are a wide variety of network protocols, denoted NET1, NET2, and so on. In practice, these protocols are implemented by a combination of hardware (e.g., a network adaptor) and software.

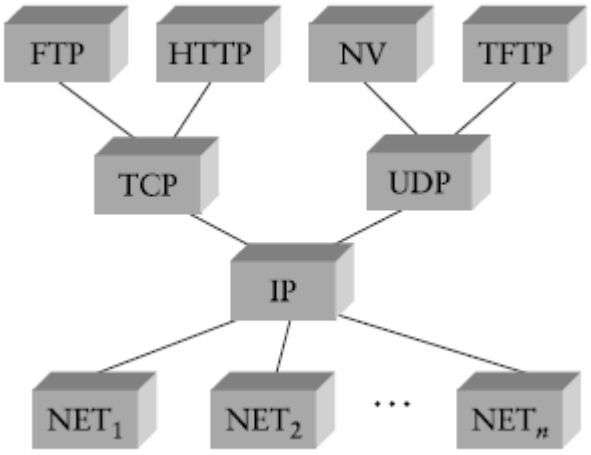
The second layer consists of a single protocol—the *Internet Protocol (IP)*. This is the protocol that supports the interconnection of multiple networking technologies into a single, logical internetwork. The third layer contains two main protocols—the *Transmission Control Protocol (TCP)* and the *User Datagram Protocol (UDP)*. TCP and UDP provide alternative logical channels to application programs: TCP provides a reliable byte-stream channel, and UDP

provides an unreliable datagram delivery channel. In the language of the Internet, TCP and UDP are sometimes called *end-to-end* protocols.

Running above the transport layer are a range of application protocols, such as FTP, TFTP (Trivial File Transport Protocol), Telnet (remote login), and SMTP (Simple Mail Transfer Protocol, or electronic mail), that enable the interoperation of popular applications. To understand the difference between an application layer protocol and an application, think of all the different World Wide Web browsers. There is a similarly large number of different implementations of web servers. The reason that you can use any one of these application programs to access a particular site on the Web is because they all conform to the same application layer protocol.

The Internet architecture has three features that are worth highlighting. First, the Internet architecture does not imply strict layering. Second, if you look closely at the protocol graph in Figure 1.14, you will notice an hourglass shape—wide at the top, narrow in the middle, and wide at the bottom. A final attribute of the Internet architecture (or more accurately, of the IETF culture) is that in order for a new protocol to be officially included in the architecture.





Network Software

Network architectures and protocol specifications are essential things, but a good blueprint is not enough to explain the phenomenal success of the Internet

- **Application Programming Interface (Sockets)**
- **Application(client/server)**
- **Protocol implementation issues**

The place to start when implementing a network application is the interface exported by the network. Since most network protocols are implemented in software (especially those high in the protocol stack), and nearly all computer systems implement their network protocols as part of the operating system, when we refer to the interface "exported by the network," we are generally referring to the interface that the OS provides to its networking subsystem. This interface is often called the network *application programming interface* (API).

Although each operating system is free to define its own network API (and most have), over time certain of these APIs have become widely supported; that is, they have been ported to operating systems other than their native system. This is what has happened with the *socket interface* originally provided by the Berkeley distribution of UNIX, which is now supported in virtually all popular operating systems.

Application Programming Interface (Sockets)

The first step is to create a socket, which is done with the following operation:

```
int socket(int domain, int type, int protocol)
```

The next step depends on whether you are a client or a server. On a server machine, the application process performs a *passive* open—the server says that it is prepared to accept connections, but it does not actually establish a connection. The server does this by invoking the following three operations:

```
int bind(int socket, struct sockaddr *address, int addr_len)
int listen(int socket, int backlog)
int accept(int socket, struct sockaddr *address, int *addr_len)
```

Example Application

The implementation of a simple client/server program that uses the socket interface to send messages over a TCP connection.

Clie

```
#include <stdio.h>
```

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define SERVER_PORT 5432
#define MAX_LINE 256
int
main(int argc, char * argv[])
{
FILE *fp;
struct hostent *hp;
struct sockaddr_in sin;
char *host;
char buf[MAX_LINE];
int s;
int len;
if (argc==2) {
host = argv[1];
}
else {
fprintf(stderr, "usage: simplex-talk host\n");
exit(1);
}
while (fgets(buf, sizeof(buf), stdin)) {
buf[MAX_LINE-1] = '\0';
len = strlen(buf) + 1;
send(s, buf, len, 0);
}
}
```

Server

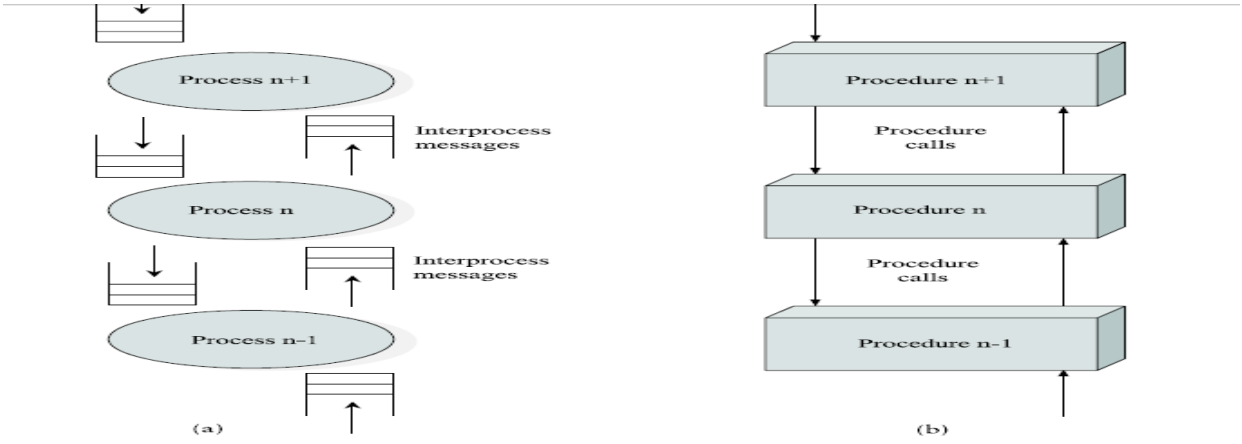
The server is equally simple. It first constructs the address data structure by filling in its own port number (SERVER_PORT).

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define SERVER_PORT 5432
#define MAX_PENDING 5
#define MAX_LINE 256
int
```

```
main()
{
struct sockaddr_in sin;
char buf[MAX_LINE];
int len;
int s, new_s;
/* build address data structure */
bzero((char *)&sin, sizeof(sin));
sin.sin_family = AF_INET;
sin.sin_addr.s_addr = INADDR_ANY;
sin.sin_port = htons(SERVER_PORT);
/* setup passive open */
if ((s = socket(PF_INET, SOCK_STREAM, 0)) < 0) {
perror("simplex-talk: socket");
exit(1);
}
while (len = recv(new_s, buf, sizeof(buf), 0))
fputs(buf, stdout);
close(new_s);
}
}
```

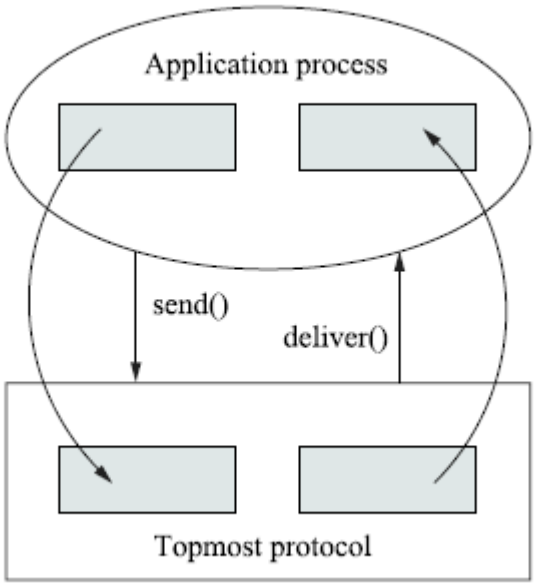
Process Model

Most operating systems provide an abstraction called a *process*, or alternatively, a *thread*. When the OS stops one process from executing on the CPU and starts up another one, we call the change a *context switch*.



Message Buffers

A second inefficiency of the socket interface is that the application process provides the buffer that contains the outbound message when calling send, and similarly it provides the buffer into which an incoming message is copied when invoking the receive operation. This forces the topmost protocol to copy the message from the application's buffer into a network buffer, and vice versa.



Copying incoming/outgoing messages between application buffer and network buffer

Performance

First get it right and then make it fast is valid in many settings, in networking it is usually necessary to "design for performance." It is, therefore, important to understand the various factors that impact network performance.

Bandwidth and Latency

Bandwidth and *throughput* are two of the most confusing terms used in networking. While we could try to give you a precise definition of each term, it is important that you know how other people might use them and for you to be aware that they are often used interchangeably. First of all, bandwidth is literally a measure of the width of a frequency band. For example, a voice-grade telephone line supports a frequency band ranging from 300 to 3,300 Hz;

A useful distinction might be made, however, between the bandwidth that is available on the link and the number of bits per second that we can actually transmit over the link in practice. We tend to use the word "throughput" to refer to the *measured performance* of a system.

Latency = Propagation + Transmit + Queue
Propagation = Distance/SpeedOfLight
Transmit = Size/Bandwidth

Delay × Bandwidth Product

It is also useful to talk about the product of these two metrics, often called the *delay × bandwidth product*. Intuitively, if we think of a channel between a pair of processes as a hollow pipe, where the latency corresponds to the length of the pipe and the bandwidth gives the diameter of the pipe, then the delay × bandwidth product gives the volume of the pipe—the maximum number of bits that could be in transit through the pipe at any given instant. For example, a transcontinental channel with a one-way latency of 50 ms and a bandwidth of 45 Mbps is able to hold.

$$50\times10-3 \text{ sec}\times45\times106 \text{ bits/sec}$$
$$= 2.25 \times106 \text{ bits}$$

High-Speed Networks

The bandwidths available on today's networks are increasing at a dramatic rate, and there is eternal optimism that network bandwidth will continue to improve. This causes network designers to start thinking about what happens in the limit, or stated another way, what is the impact on network design of having infinite bandwidth available.

Throughput = Transfer Size/Transfer Time

Where Transfer Time includes not only the elements of one-way Latency identified earlier in this section, but also any additional time spent requesting or setting up the transfer.

Transfer Time = RTT + 1/Bandwidth x Transfer Size

Application Performance Needs

This section has taken a network-centric view of performance; that is, we have talked in terms of what a given link or channel will support. The unstated assumption has been that application programs have simple needs—they want as much bandwidth as the network can provide.

However, some applications are able to state an upper limit on how much bandwidth they need. Video applications are a prime example. Suppose one wants to stream a video image; that is one-quarter the size of a standard TV image; that is, it has a resolution of 352 by 240 pixels. If

each pixel is represented by 24 bits of information, as would be the case for 24-bit color, then the size of each frame would be

$$(352 \times 240 \times 24)/8 = 247.5 \text{ KB}$$

Link Layer Services

A link layer protocol is used to move a datagram over an individual link layer protocol defines the format of the packets exchanged between the nodes at the ends of the link, as well as the action taken by these nodes when the packets are sent and received.

Services that can be offered by a link layer protocol include:

- Framing
- Flow control
- Error detection

Framing

Almost all link layer protocols encapsulate each network layer datagram within a link layer frame before transmission over the link. A frame consists of a data field in which the network layer datagram and inserted and a number of header fields. Frame may also include header and trailer fields.

Flow Control

A nodes receiver can incorrectly decide that a bit in a frame is zero when it was transmitted as a one, and vice versa. Such bit errors are introduced by signal attenuation and electromagnetic noise. Because there is no need to forward a datagram that has an error, many link layer protocols provide a mechanism to detect the presence of one or more errors

Error detection

The nodes on each side of a link have a limited amount of frame buffering capacity. This is a potential problem, as a receiving node may receive frame at a rate faster than it can process them. Without flow control, the receiver's buffer can overflow and frames can get lost. Similar to the transport layer, a link layer protocol can provide flow control in order to prevent the sending node on one side of a link from overwhelming on the other side of the link.

UNIT 2

MEDIA ACCESS & INTERNETWORKING

MEDIA ACCESS CONTROL

ETHERNET 802.3

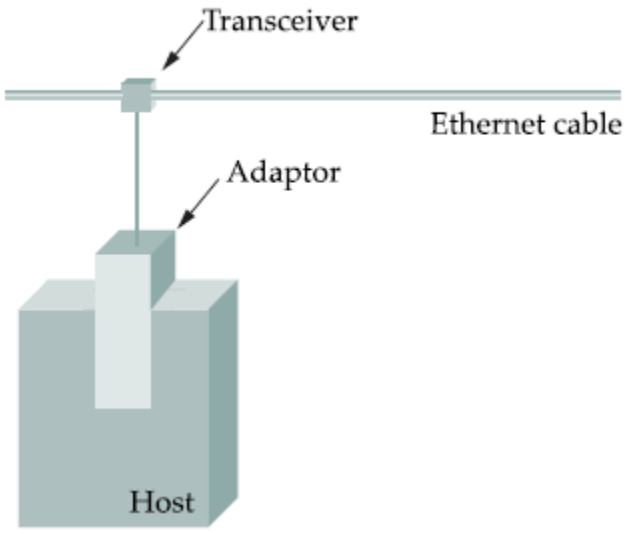
The Ethernet is easily the most successful local area networking technology of the last 20 years. Developed in the mid-1970s by researchers at the Xerox Palo Alto Research Center (PARC), the Ethernet is a working example of the more general carrier sense, multiple access with collision detect (CSMA/CD) local area network technology.

As indicated by the CSMA name, the Ethernet is a multiple-access network, meaning that a set of nodes send and receive frames over a shared link.

The "carrier sense" in CSMA/CD means that all the nodes can distinguish between an idle and a busy link, and "collision detect" means that a node listens as it transmits and can therefore detect when a frame it is transmitting has interfered (collided) with a frame transmitted by another node.

Physical Properties:

An Ethernet segment is implemented on a coaxial cable of up to 500 m. This cable is similar to the type used for cable TV, except that it typically has an impedance of 50 ohms instead of cable TV's 75 ohms. Hosts connect to an Ethernet segment by tapping into it; taps must be at least 2.5 m apart. A *transceiver* a small device directly attached to the tap—detects when the line is idle and drives the signal when the host is transmitting.



Ethernet transceiver and adaptor

Multiple Ethernet segments can be joined together by *repeaters*. A repeater is a device that forwards digital signals, much like an amplifier forwards analog signals. However, no more than four

repeaters may be positioned between any pair of hosts, meaning that an Ethernet has a total reach of only 2,500 m.

Any signal placed on the Ethernet by a host is broadcast over the entire network, that is, the signal is propagated in both directions, and repeaters forward the signal on all outgoing segments.

In addition to the system of segments and repeaters just described, alternative technologies have been introduced over the years. For example, rather than using a 50-ohm coax cable, an Ethernet can be constructed from a thinner cable known as 10Base2;

Today, a third cable technology is predominantly used, called 10BaseT, where the "T" stands for twisted pair. Typically, Category 5 twisted pair wiring is used. A 10BaseT segment is usually limited to under 100 m in length.

Access point

We now turn our attention to the algorithm that controls access to the shared Ethernet link. This algorithm is commonly called the Ethernet's *media access control (MAC)*.

Frame format

The 64-bit preamble allows the receiver to synchronize with the signal; it is a sequence of alternating 0s and 1s. Both the source and destination hosts are identified with a 48-bit address. The packet type field serves as the demultiplexing key, that is, it identifies to which of possibly many higher-level protocols this frame should be delivered. Each frame contains up to 1,500 bytes of data. Minimally, a frame must contain at least 46 bytes of data, even if this means the host has to pad the frame before transmitting it.

Address

Each host on an Ethernet—in fact, every Ethernet host in the world—has a unique Ethernet address. For example, 8:0:2b:e4:b1:2 is the human-readable representation of Ethernet address
00001000 00000000 00101011 11100100 10110001 00000010

In addition to these *unicast* addresses, an Ethernet address consisting of all 1s is treated as a *broadcast* address; all adaptors pass frames addressed to the broadcast address up to the host. Similarly, an address that has the first bit set to 1 but is not the broadcast address is called a *multicast* address. A given host can program its adaptor to accept some set of multicast addresses. Multicast addresses are used to send messages to some subset of the hosts on an Ethernet (e.g., all file servers). To summarize, an Ethernet adaptor receives all frames and accepts

- Frames addressed to its own address;
- Frames addressed to the broadcast address;

Transmitting algorithm

The receiver side of the Ethernet protocol is simple; the real smarts are implemented at the sender's side. The transmitter algorithm is defined as follows. When the adaptor has a frame to send and the line is idle, it transmits the frame immediately; there is no negotiation with the other adaptors.

WIRELESS LANS

BLUETOOTH (802.15.1)

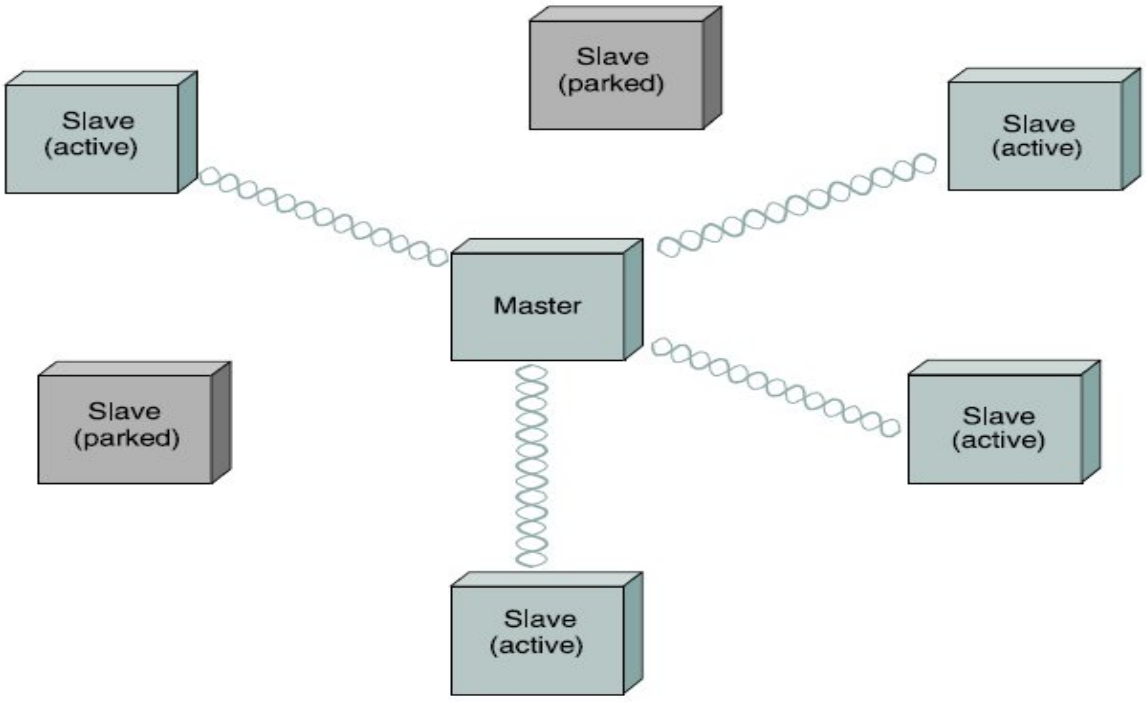
Bluetooth fills the niche of very short-range communication between mobile phones, PDAs, notebook computers, and other personal or peripheral devices. For example, Bluetooth can be used to connect a mobile phone to a headset, or a notebook computer to a printer.

Roughly speaking, Bluetooth is a more convenient alternative to connecting two devices with a wire. In such applications, it is not necessary to provide much range or bandwidth.

Bluetooth operates in the license-exempt band at 2.45 GHz. It has a range of only about 10 m. For this reason, and because the communicating devices typically belong to one individual or group, Bluetooth is sometimes categorized as a personal area network (PAN). Version 2.0 provides speeds up to 2.1 Mbps. Power consumption is low.

The basic Bluetooth network configuration, called a *piconet*, consists of a master device and up to seven slave devices.

Since Bluetooth operates in an license-exempt band, it is required to use a spread spectrum technique to deal with possible interference in the band. It uses frequency hopping with 79 *channels* (frequencies), using each for 625 μm at a time.



A Bluetooth piconet.

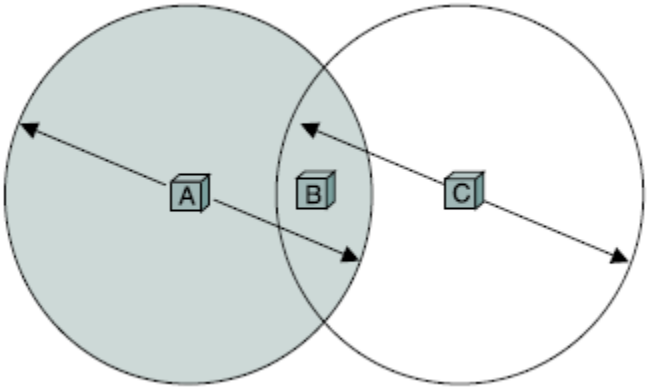
A slave device can be *parked*: set to an inactive, low-power state. A parked device cannot communicate on the piconet; it can only be reactivated by the master. A piconet can have up to 255 parked devices in addition to its active slave devices.

Wi-Fi(802.11)

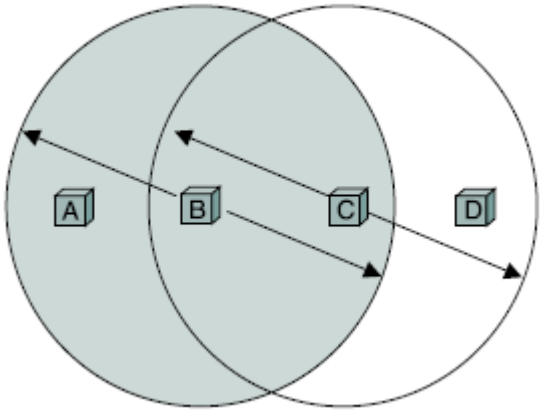
This section takes a closer look at a specific technology centered around the emerging IEEE 802.11 standard, also known as *Wi-Fi*.

Collision Avoidance:

Where A and C are both within range of B but not each other. Suppose both A and C want to communicate with B and so they each send it a frame. A and C are unaware of each other since their signals do not carry that far. These two frames collide with each other at B, but unlike an Ethernet, neither A nor C is aware of this collision. A and C are said to be *hidden nodes* with respect to each other.



The hidden node problem

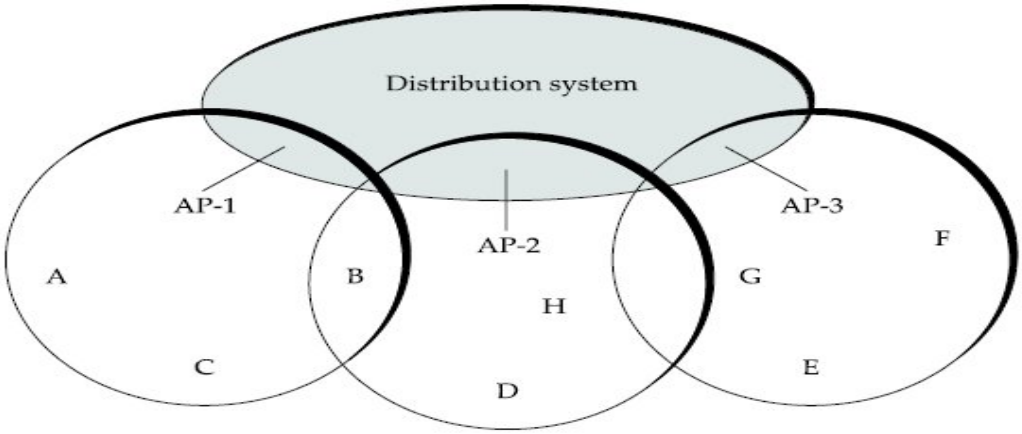


The exposed node problem

A related problem, called the *exposed node problem*, occurs under the circumstance, where each of the four nodes is able to send and receive signals that reach just the nodes to its immediate left and right. For example, B can exchange frames with A and C but it cannot reach D, while C can reach B and D but not A. Suppose B is sending to A. Node C is aware of this communication because it hears B's transmission. It would be a mistake, however, for C to conclude that it cannot transmit to anyone just because it can hear B's transmission.

Distributed System

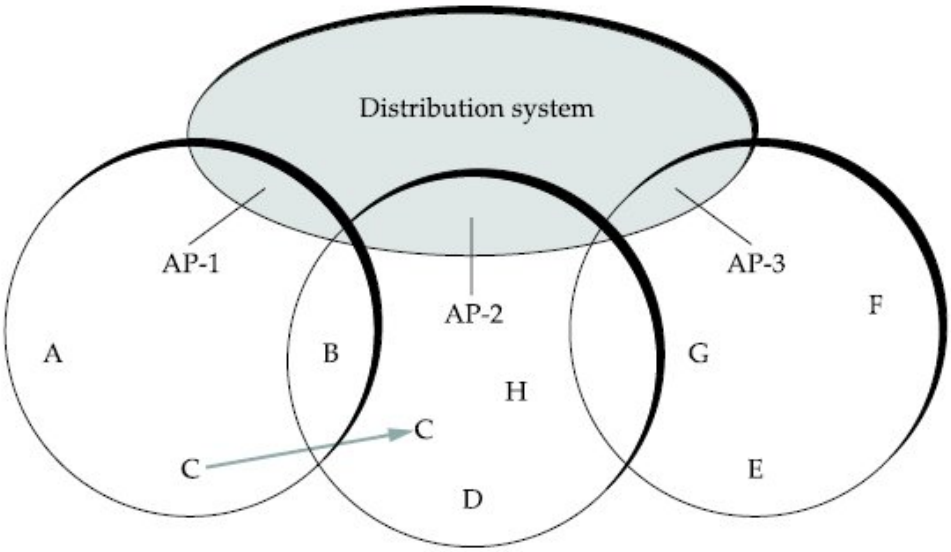
802.11 would be suitable for a network with a mesh (ad hoc) topology, and development of an 802.11s standard for mesh networks is nearing completion. At the current time, however, nearly all 802.11 networks use a base-station-oriented topology.



Access points connected to a distribution network.

The technique for selecting an AP is called *scanning* and involves the following four steps:

1 The node sends a Probe frame;
2 All APs within reach reply with a Probe Response frame;
3 The node selects one of the access points, and sends that AP an Association Request frame;
4 The AP replies with an Association Response frame.



Node mobility.

Frame format

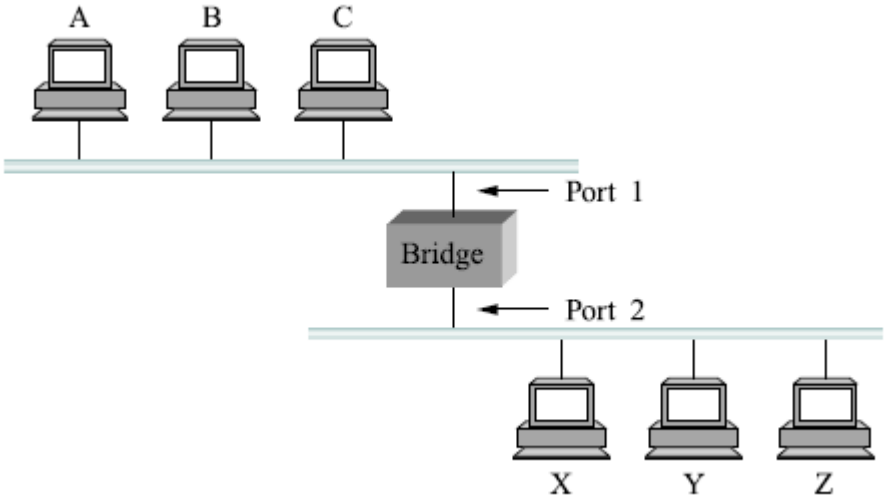
The frame contains the source and destination node addresses, each of which are 48 bits long, up to 2,312 bytes of data, and a 32-bit CRC. The Control field contains three subfields of interest (not shown): a 6-bit Type field that indicates whether the frame carries data, is an RTS or CTS frame, or is being used by the scanning algorithm; and a pair of 1-bit fields—called ToDS and FromDS—that are described below.

Bridges and LAN Switches

We begin by considering a class of switch that is used to forward packets between shared media LANs such as Ethernets. Such switches are sometimes known by the obvious name of LAN switches; historically they have also been referred to as bridges.

Learning Bridges:

The first optimization we can make to a bridge is to observe that it need not forward all frames that it receives. Then, whenever the bridge receives a frame on port 1 that is addressed to host A, it would not forward the frame out on port 2; there would be no need because host A would have already directly received the frame on the LAN connected to port 1. Anytime a frame addressed to host A was received on port 2, the bridge would forward the frame out on port 1.



Note that a bridge using such a table would be using the datagram (or connectionless) model of forwarding described in Section 3.1.1. Each packet carries a global address, and the bridge decides on which output to send a packet by looking up that address in a table.

Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	2

Implementation

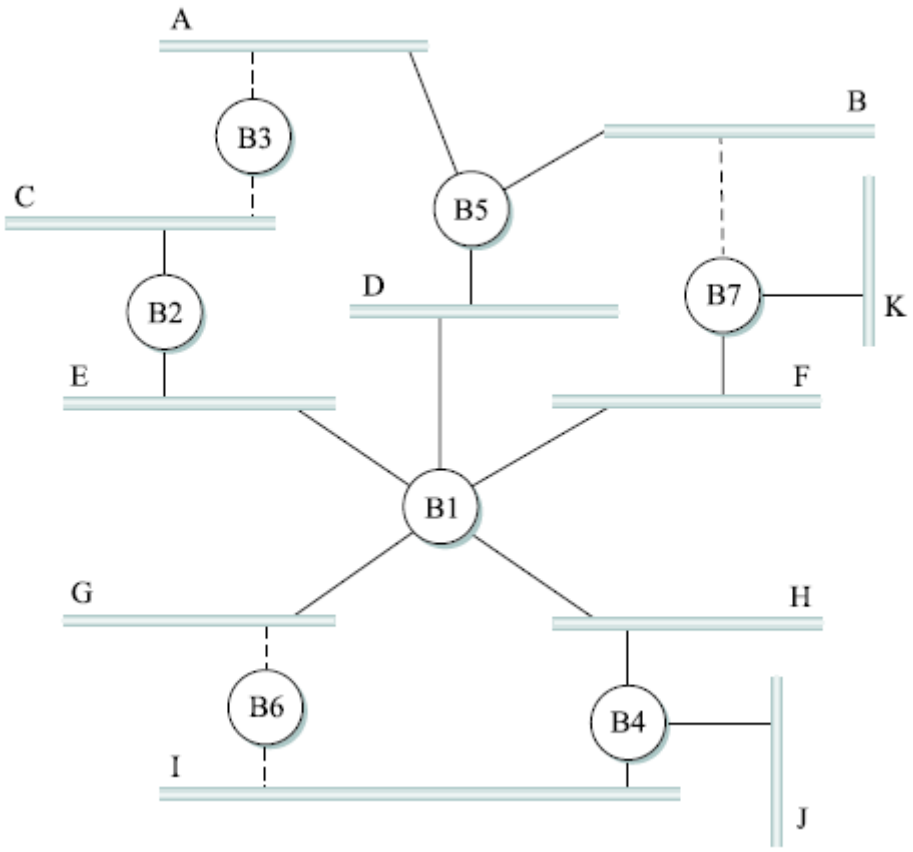
The code that implements the learning bridge algorithm is quite simple, and we sketch it here. Structure BridgeEntry defines a single entry in the bridge's forwarding table; these are stored in a Map structure (which supports mapCreate, mapBind, and MapResolve operations) to enable entries to be efficiently located when packets arrive from sources already in the table.

```
#define BRIDGE_TAB_SIZE 1024 /* max. size of bridging
table */
#define MAX_TTL 120 /* time (in seconds) before
an entry is flushed */
typedef struct {
MacAddr destination; /* MAC address of a node */
int ifnumber; /* interface to reach it */
u_short TTL; /* time to live */
Binding binding; /* binding in the Map */
} BridgeEntry;
int numEntries = 0;
Map bridgeMap = mapCreate(BRIDGE_TAB_SIZE,
sizeof(BridgeEntry));
```

The main idea of the spanning tree is for the bridges to select the ports over which they will forward frames. The algorithm selects ports as follows. Each bridge has a unique identifier; for our purposes, we use the labels B1, B2, B3, and so on. The algorithm first elects the bridge with the smallest ID as the root of the spanning tree;

Specifically, the configuration messages contain three pieces of information:

1. The ID for the bridge that is sending the message;
2. The ID for what the sending bridge believes to be the root bridge;
3. The distance, measured in hops, from the sending bridge to the root bridge.



- It identifies a root with a smaller ID or
- It identifies a root with an equal ID but with a shorter distance or
- The root ID and distance are equal, but the sending bridge has a smaller ID.

Broadcast and Multicast

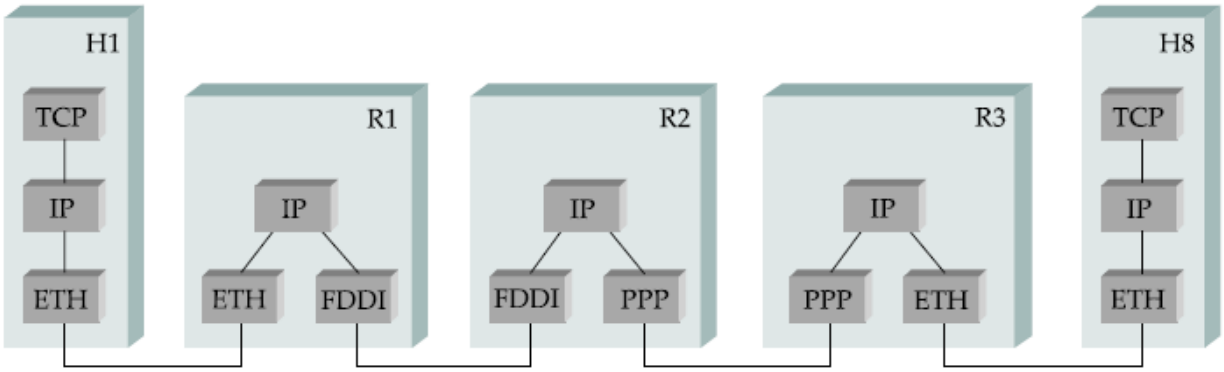
LANs support both broadcast and multicast, then bridges must also support these two features.

Broadcast is simple—each bridge forwards a frame with a destination broadcast address out on each active (selected) port other than the one on which the frame was received.

Multicast can be implemented in exactly the same way, with each host deciding for itself whether or not to accept the message.

What Is an Internetwork?

An *internetwork* is an interconnected collection of such networks. Sometimes, to avoid ambiguity, we refer to the underlying networks that we are interconnecting as *physical* networks.



Service Model

A good place to start when you build an internetwork is to define its *service model*, that is, the host-to-host services you want to provide.

Datagram Delivery

The “best-effort” part means that if something goes wrong and the packet gets lost, corrupted, misdelivered, or in any way fails to reach its intended destination, the network does nothing—it made its best effort, and that is all it has to do. It does not make any attempt to recover from the failure. This is sometimes called an *unreliable* service.

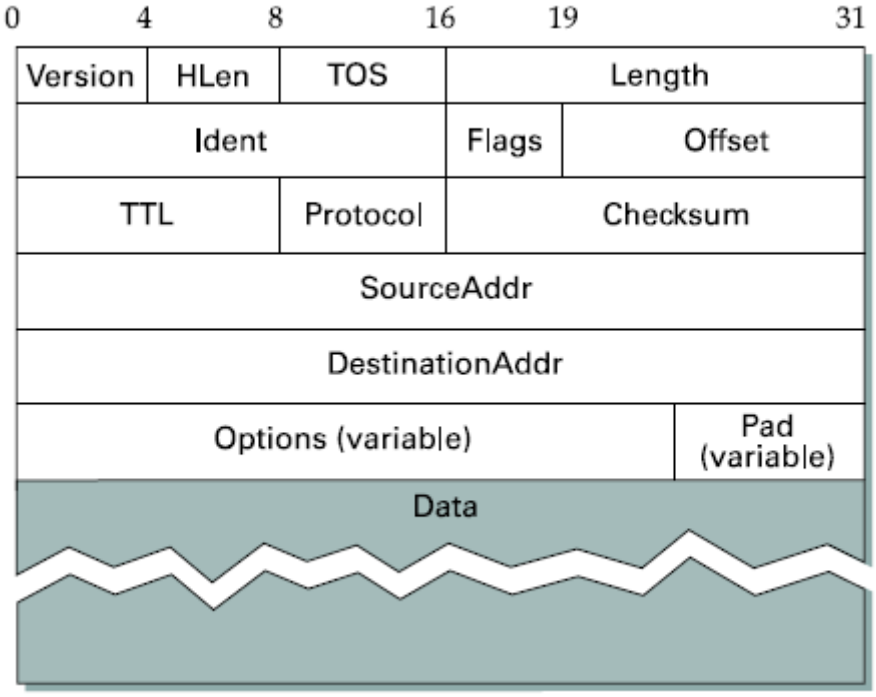
Packet Format

A key part of the IP service model is the type of packets that can be carried. The IP datagram, like most packets, consists of a header followed by a number of bytes of data.

Fragmentation

Two important points are discussed under fragmentation

1. Each fragment is itself a self-contained IP datagram that is transmitted over a sequence of physical networks, independent of the other fragments;
2. Each IP datagram is reencapsulated for each physical network over which it travels.

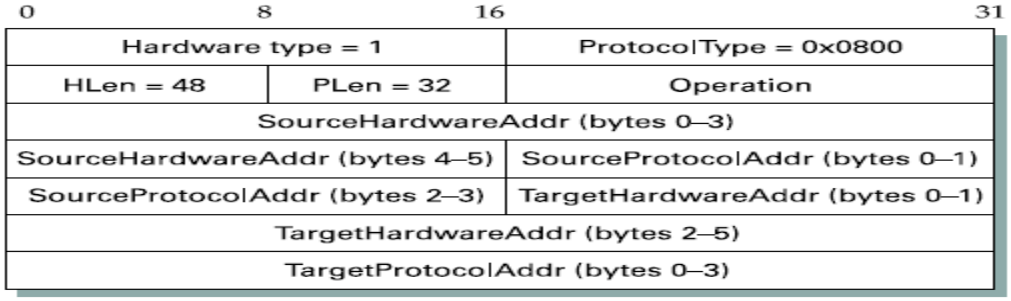


Address Translation (ARP)

The main issue is that IP datagrams contain IP addresses, but the physical interface hardware on the host or router to which you want to send the datagram only understands the addressing scheme of that particular network. Thus, we need to translate the IP address to a link-level address that makes sense on this network

Mappings—the major differences are in the address sizes. In addition to the IP and link-layer addresses of both sender and target, the packet contains

- A HardwareType field, which specifies the type of physical network (e.g., Ethernet);
- A ProtocolType field, which specifies the higher-layer protocol (e.g., IP);
- HLen ("hardware" address length) and PLen ("protocol" address length) fields, which specify the length of the link-layer address and higher-layer protocol address, respectively;
- An Operation field, which specifies whether this is a request or a response;
- The source and target hardware (Ethernet) and protocol (IP) addresses.



ARP packet format for mapping IP addresses into Ethernet addresses.

Host Configuration (DHCP)

DHCP relies on the existence of a DHCP server that is responsible for providing configuration information to hosts. There is at least one DHCP server for an administrative domain. At the simplest level, the DHCP server can function just as a centralized repository for host configuration information. Consider, for example, the problem of administering addresses in the internetwork of a large company. DHCP saves the network administrators from having to walk around to every host in the company with a list of addresses and network map in hand and configuring each host manually.

DHCP uses the concept of a *relay agent*. There is at least one relay agent on each network, and it is configured with just one piece of information: the IP address of the DHCP server. When a relay agent receives a DHCPDISCOVER message, it uni casts it to the DHCP server and awaits the response

The message is actually sent using a protocol called the User Datagram Protocol (UDP) that runs over IP.

DHCP is derived from an earlier protocol called BOOTP, and some of the packet fields are thus not strictly relevant to host configuration.

ERROR REPORTING

The next issue is how the Internet treats errors. While IP is perfectly willing to drop datagrams when the going gets tough—for example, when a router does not know how to forward the datagram or when one fragment of a datagram fails to arrive at the destination—it does not necessarily fail silently.

IP is always configured with a companion protocol, known as the Internet Control Message Protocol (ICMP), that defines a collection of error messages that are sent back to the source host whenever a router or host is unable to process an IP datagram successfully.

ICMP also defines a handful of control messages that a router can send back to a source host. One of the most useful control messages, called an ICMP-Redirect, tells the source host that there is a better route to the destination.

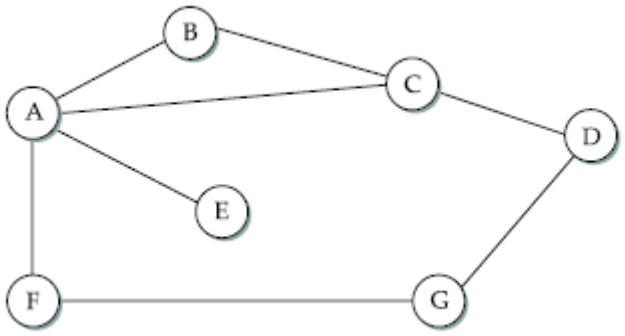
UNIT 3

ROUTING

DISTANCE VECTOR (RIP)

The distance-vector each node constructs a one-dimensional array (a vector) containing the "distances" (costs) to all other nodes and distributes that vector to its immediate neighbors. The starting assumption for distance-vector routing is that each node knows the cost of the link to each of its directly connected neighbors.

To see how a distance-vector routing algorithm works, it is easiest to consider an like the one depicted in the cost of each link is set to 1, so that a least-cost path is simply the one with the fewest hops.



Information Stored at Node	Distance to Reach Node						
	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Destination	Cost	Next Hop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

The next step in distance-vector routing is that every node sends a message to its directly connected neighbors containing its personal list of distances. For example, node F tells node A that it can reach node G at a cost of 1; A also knows it can reach F at a cost of 1, so it adds these costs to get the cost of reaching G by means of F. This total cost of 2 is less than the current cost of infinity, so A records that it can reach G at a cost of 2 by going through F.

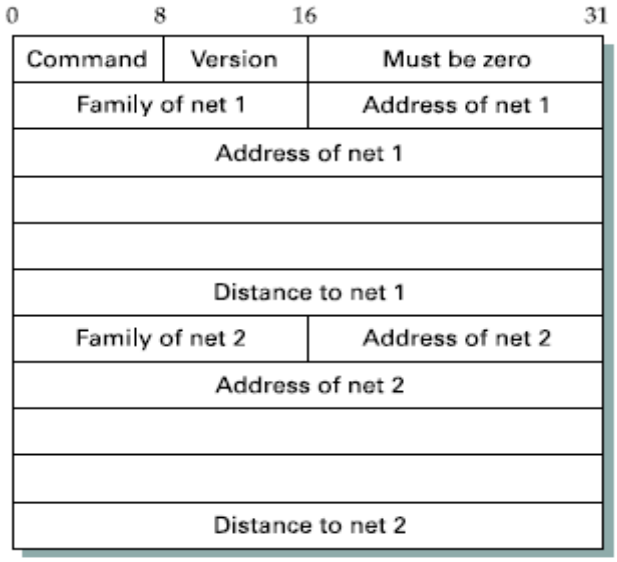
Similarly, A learns from C that D can be reached from C at a cost of 1; it adds this to the cost of reaching C (1) and decides that D can be reached via C at a cost of 2, which is better than the old cost of infinity. At the same time, A learns from C that B can be reached from C at a cost of 1, so it concludes that the cost of reaching B via C is 2. Since this is worse than the current cost of reaching B (1), this new information is ignored.

In the absence of any topology changes, it only takes a few exchanges of information between neighbors before each node has a complete routing table. The process of getting consistent routing information to all the nodes is called *convergence*.

There are a few details to fill in before our discussion of distance-vector routing is complete. First, we note that there are two different circumstances under which a given node decides to send a routing update to its neighbors. One of these circumstances is the *periodic* update.

Routing information Protocol (RIP)

One of the most widely used routing protocols in IP networks is the Routing Information Protocol (RIP). Its widespread use is due in no small part to the fact that it was distributed along with the popular Berkeley Software Distribution (BSD) version of Unix, from which many commercial versions of Unix were derived. It is also extremely simple. RIP is the canonical example of a routing protocol built on the distance-vector algorithm just described.



RIP packet format

LINK STATE (OSPF)

Link-state routing is the second major class of intra domain routing protocol. The starting assumptions for link-state routing are rather similar to those for distance-vector routing. Each node is assumed to be capable of finding out the state of the link to its neighbors (up or down) and the cost of each link.

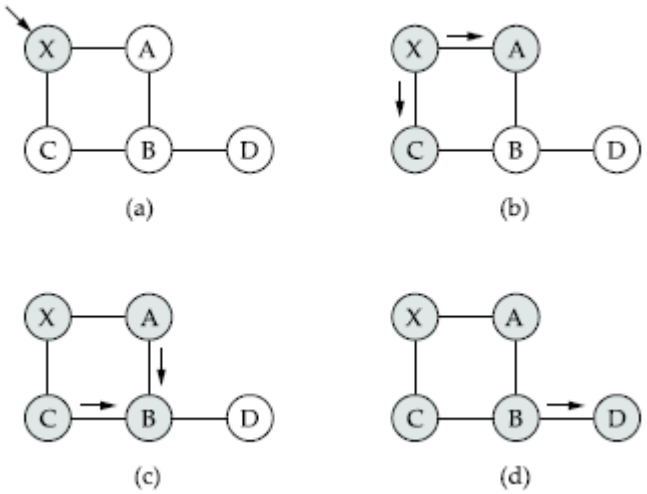
Reliable Flooding

Reliable flooding is the process of making sure that all the nodes participating in the routing protocol get a copy of the link-state information from all the other nodes. As the term "flooding" suggests, the basic idea is for a node to send its link-state information out on all of its

directly connected links, with each node that receives this information forwarding it out on all of *its* links.

■The ID of the node that created the LSP;
■A list of directly connected neighbors of that node, with the cost of the link to each one;
■A sequence number;
■A time to live for this packet.

The first two items are needed to enable route calculation; the last two are used to make the process of flooding the packet to all nodes reliable. Reliability includes making sure that you have the most recent copy of the information, since there may be multiple, contradictory LSPs from one node traversing the network.



Route Calculation

1 Initialize the Confirmed list with an entry for myself; this entry has a cost of 0.
2 For the node just added to the Confirmed list in the previous step, call it node Next, select its LSP.
3 For each neighbor (Neighbor) of Next, calculate the cost (Cost) to reach this Neighbor as the sum of the cost from myself to Next and from Next to Neighbor.

(a) If Neighbor is currently not on either the Confirmed or the Tentative list, then add (Neighbor, Cost, NextHop) to the Tentative list, where NextHop is the direction I go to reach Next.

(b) If Neighbor is currently on the Tentative list, and the Cost is less than the currently listed cost for Neighbor, then replace the current entry with (Neighbor, Cost, NextHop), where NextHop is the direction I go to reach Next.

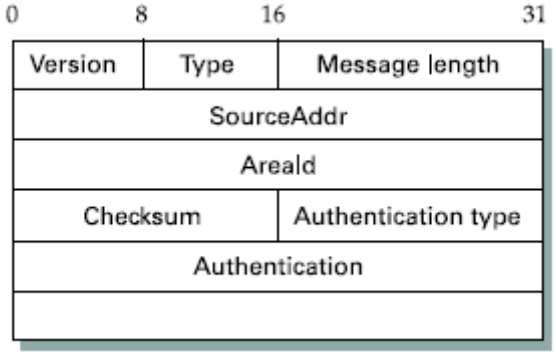
4 If the Tentative list is empty, stop. Otherwise, pick the entry from the Tentative list with the lowest cost, move it to the Confirmed list, and return to step 2.

OSPF adds quite a number of features to the basic link-state algorithm described above, including the following:

■Authentication of routing messages: This is a nice feature, since it is all too common for some misconfigured host to decide that it can reach every host in the universe at a cost of 0.

■Additional hierarchy: Hierarchy is one of the fundamental tools used to make systems more scalable. OSPF introduces another layer of hierarchy into routing by allowing a domain to be partitioned into *areas*.

■Load balancing: OSPF allows multiple routes to the same place to be assigned the same cost and will cause traffic to be distributed evenly over those routes.



METRICS

Metrics

The preceding discussion assumes that link costs, or metrics, are known when we execute the routing algorithm. In this section, we look at some ways to calculate link costs that have

proven effective in practice. One example that we have seen already, which is quite reasonable and very simple, is to assign a cost of 1 to all links—the least-cost route will then be the one with the fewest hops.

The ARPANET was the testing ground for a number of different approaches to link-cost calculation.

$$\text{Delay} = (\text{DepartTime} - \text{ArrivalTime}) + \text{TransmissionTime} + \text{Latency}$$

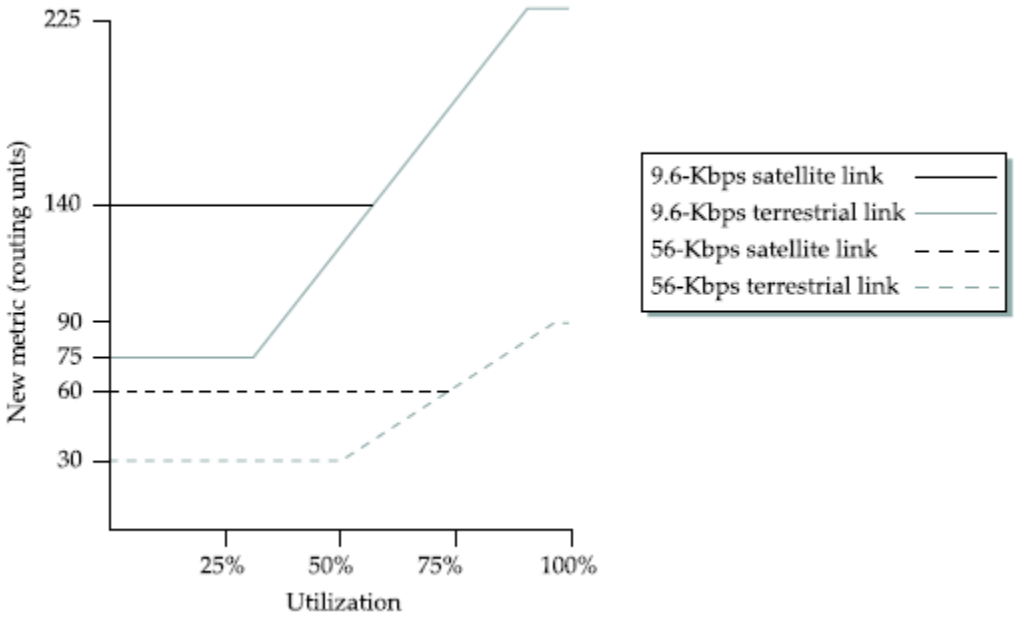
The original ARPANET routing metric measured the number of packets that were queued waiting to be transmitted on each link, meaning that a link with 10 packets queued waiting to be transmitted was assigned a larger cost weight than a link with 5 packets queued for transmission.

A second version of the ARPANET routing algorithm, sometimes called the "new routing mechanism," took both link bandwidth and latency into consideration and used delay, rather than just queue length, as a measure of load. This was done as follows. First, each incoming packet was time stamped with its time of arrival at the router (ArrivalTime);

Although an improvement over the original mechanism, this approach also had a lot of problems. Under light load, it worked reasonably well, since the two static factors of delay dominated the cost. Under heavy load, however, a congested link would start to advertise a very high cost. This caused all the traffic to move off that link, leaving it idle, so then it would advertise a low cost, thereby attracting back all the traffic, and so on.

The compression of the dynamic range was achieved by feeding the measured utilization, Observe the following:

- A highly loaded link never shows a cost of more than three times its cost when idle;
- The most expensive link is only seven times the cost of the least expensive;
- A high-speed satellite link is more attractive than a low-speed terrestrial link;
- Cost is a function of link utilization only at moderate to high loads.



GLOBEL INTERNET AREAS

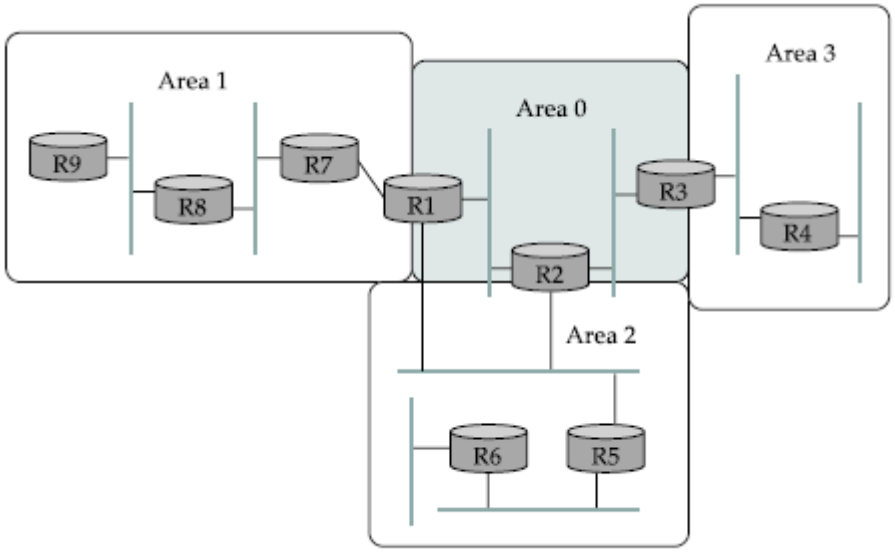
As if we didn't already have enough hierarchy, link-state intra domain routing protocols provide a means to partition a routing domain into sub domains called *areas*. (The terminology varies somewhat among protocols—we use the OSPF terminology here.) By adding this extra level of hierarchy, we enable single domains to grow larger without overburdening the intra domain routing protocols.

An area is a set of routers that are administratively configured to exchange link-state information with each other. There is one special area—the backbone area, also known as area 0. An example of a routing domain divided into areas. Routers R1, R2, and R3 are members of the backbone area. They are also members of at least one non backbone area; R1 is actually a member of both area 1 and area 2. A router

that is a member of both the backbone area and a non backbone area is an area border router (ABR).

All the routers in the area send link-state advertisements to each other, and thus develop a complete, consistent map of the area. However, the link-state advertisements of routers that are not area border routers do not leave the area in which they originated. This has the effect of making the flooding and route calculation processes considerably more scalable.

For example, router R4 in area 3 will never see a link-state advertisement from router R8 in area 1. As a consequence, it will know nothing about the detailed topology of areas other than its own. A virtual link between routers. Such a virtual link is obtained by configuring a router that is not directly connected to area 0.



BGP(Inter domain Routing)

At the beginning of this section we introduced the notion that the Internet is organized as autonomous systems, each of which is under the control of a single administrative entity.

The basic idea behind autonomous systems is to provide an additional way to hierarchically aggregate routing information in a large internet, thus improving scalability.

We now divide the routing problem into two parts: routing within a single autonomous system and routing between autonomous systems. Since another name for autonomous systems in the Internet is routing *domains*, we refer to the two parts of the routing problem as interdomain routing and intradomain routing.

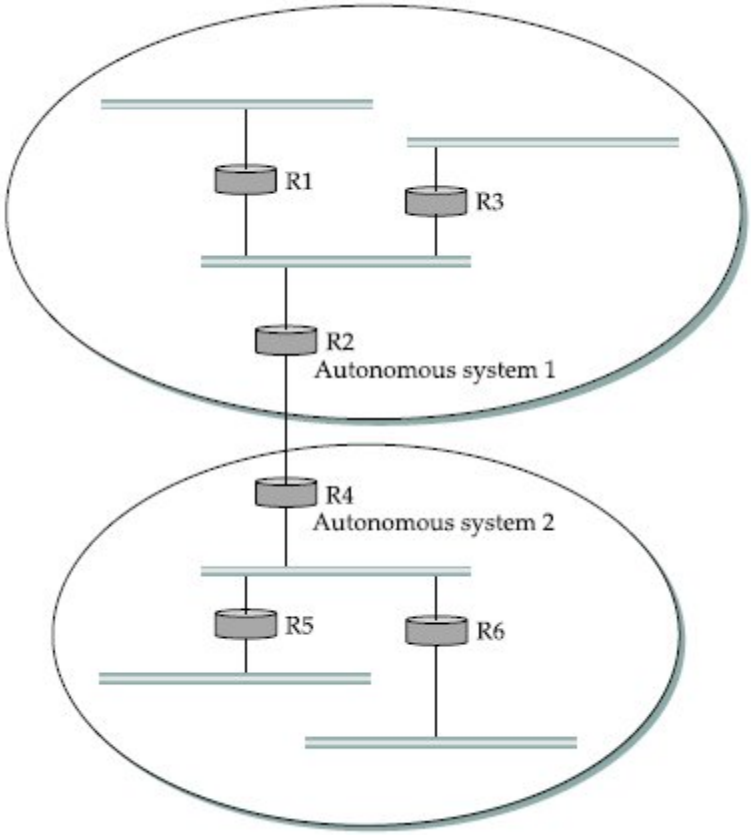
Perhaps the most important challenge of interdomain routing today is the need for each AS to determine its own routing *policies*. A simple example routing policy implemented at a particular AS might look like this Whenever possible, I prefer to send traffic via AS X than via AS Y, but I'll use AS Y if it is the only path, and I never want to carry traffic from AS X to AS Y or vice versa.

A key design goal of interdomain routing is that policies like the example above, and much more complex ones, should be supported by the interdomain routing system. To make the problem harder, I need to be able to implement such a policy without any help from other ASs, and in the face of possible misconfiguration or malicious behavior by other ASs.

There have been two major interdomain routing protocols in the recent history of the Internet.

(1) Exterior Gateway Protocol (EGP)
(2) Border Gateway Protocol (BGP)

This rough sketch of the Internet, if we define *local traffic* as traffic that originates at or terminates on nodes within an AS, and *transit traffic* as traffic that passes through an AS, we can classify ASs into three types:



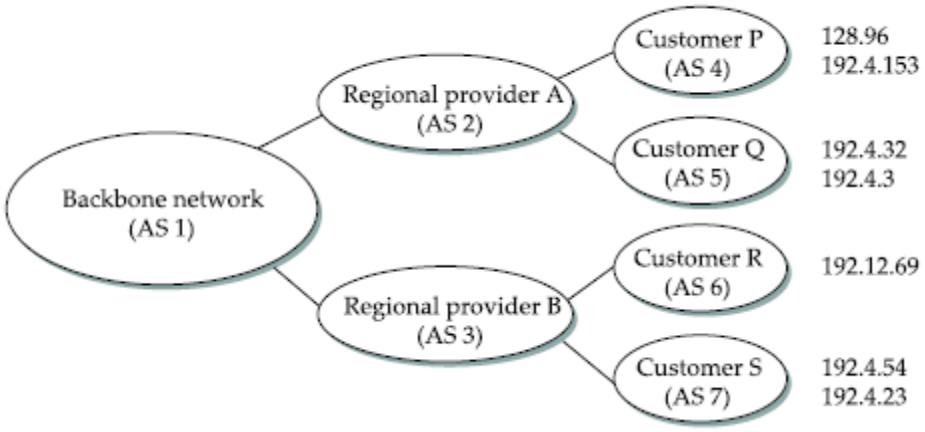
A network with two autonomous systems.

■ *Stub AS:* an AS that has only a single connection to one other AS; such an AS will only carry local traffic. The small corporation is an example of a stub AS.
■ *Multihomed AS:* an AS that has connections to more than one other AS but that refuses to carry transit traffic; for example, the large corporation at the top
■ *Transit AS:* an AS that has connections to more than one other AS and that is designed to carry both transit and local traffic, such as the backbone providers

There are additional factors that make interdomain routing hard. The first is simply a matter of scale. An Internet backbone router must be able to forward any packet destined anywhere in the Internet. A second challenge in interdomain routing arises from the autonomous nature of the domains. The third challenge involves the issue of trust.

Integrating Interdomain and Intradomain Routing

While the preceding discussion illustrates how a BGP speaker learns interdomain routing information, the question still remains as to how all the other routers in a domain get this information.



Example of a network running BGP.

IP Version 6 (IPV6)

The motivation for a new version of IP is the same as the motivation for the techniques described so far in this section:

Historical Perspective

■ Support for real-time services;
■ Security support;

■ Auto configuration (i.e., the ability of hosts to automatically configure themselves with such information as their own IP address and domain name);
■ Enhanced routing functionality, including support for mobile hosts.

Addressing and Routing

IPv6 provides a 128-bit address space, as opposed to the 32 bits of version 4. Thus, while version 4 can potentially address 4 billion nodes if address assignment efficiency reaches 100%.

Address Space Allocation

Drawing on the effectiveness of CIDR in IPv4, IPv6 addresses are also classless, but the address space is still subdivided in various ways based on the leading bits.

Address Notation

There is some special notation for writing down IPv6 addresses. The standard representation is x:x:x:x:x:x:x:x where each "x" is a hexadecimal representation of a 16-bit piece of the address. An example would be

47CD:1234:4422:ACO2:0022:1234:A456:0124

Any IPv6 address can be written using this notation. Since there are a few special types of IPv6 addresses, there are some special notations that may be helpful in certain circumstances.

47CD:0000:0000:0000:0000:0000:A456:0124
could be written
47CD::A456:0124

Globel Unicast Addresses

The most important sort of addressing that IPv6 must provide is plain old unicast addressing. It must do this in a way that supports the rapid rate of addition of new hosts to the

Internet and that allows routing to be done in a scalable way as the number of physical networks in the Internet grows.

Prefix	Use
00...0 (128 bits)	Unspecified
00...1 (128 bits)	Loopback
1111 1111	Multicast addresses
1111 1110 10	Link local unicast
1111 1110 11	Site local unicast
Everything else	Global unicast

Address prefix assignments for IPv6.

Packet Format

As with many headers, this one starts with a Version field, which is set to 6 for IPv6. The Version field is in the same place relative to the start of the header as IPv4's Version field so that header-processing software can immediately decide which header format to look for.

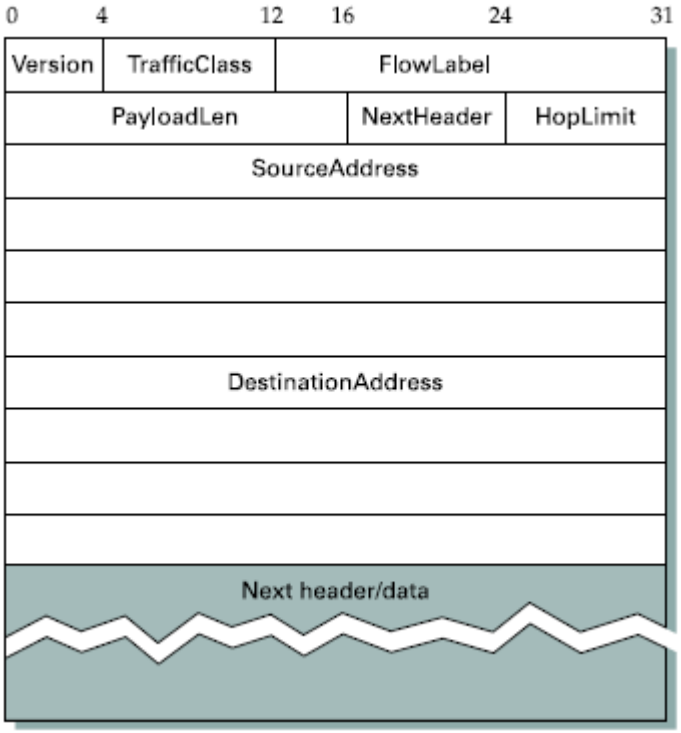
The PayloadLen field gives the length of the packet, excluding the IPv6 header, measured in bytes. The NextHeader field cleverly replaces both the IP options and the Protocol field of IPv4. If options are required, then they are carried in one or more special headers following the IP header, and this is indicated by the value of the NextHeader field.

Multicast

Multiaccess networks like Ethernet and token rings implement multicast in hardware. IP's original many-to-many multicast has been supplemented with support for a form of one-to-many multicast. In this model of one-to-many multicast, called *sourcespecific multicast (SSM).*

To contrast it with SSM, IP's original many-to-many model is sometimes referred to as *any source multicast (ASM)*.

A host signals its desire to join or leave a multicast group by communicating with its local router using a special protocol for just that purpose. In IPv4, that protocol is *Internet Group Management Protocol (IGMP)*;



IPv6 packet header.

Multicast address

IP has a subrange of its address space reserved for multicast addresses. In IPv4, these addresses are assigned in the class D address space, and IPv6 also has a portion of its address space reserved for multicast group addresses.

Thus, there are 28 bits of possible multicast addresses in IPv4 when we ignore the prefix shared by all multicast addresses. This presents a problem when attempting to take advantage of hardware multicasting on a LAN. Let's take the case of Ethernet.

Ethernet multicast addresses have only 23 bits when we ignore their shared prefix. In other words, to take advantage of Ethernet multicasting, IP has to map 28-bit IP multicast addresses into 23-bit Ethernet multicast addresses. This is implemented by taking the low-order 23 bits of any IP multicast address to use as its Ethernet multicast address, and ignoring the high-order 5 bits.

When a host on an Ethernet joins an IP multicast group, it configures its Ethernet interface to receive any packets with the corresponding Ethernet multicast address. Unfortunately, this causes the receiving host to receive not only the multicast traffic it desired, but also traffic sent to any of the other 31 IP multicast groups that map to the same Ethernet address, if they are routed to that Ethernet.

DVMRP

Distance-vector routing, which we discussed for unicast, can be extended to support multicast. The resulting protocol is called *Distance Vector Multicast Routing Protocol*, or DVMRP.

Recall that, in the distance-vector algorithm, each router maintains a table of _Destination, Cost, NextHop_ tuples, and exchanges a list of _Destination, Cost_ pairs with its directly connected neighbors. Extending this algorithm to support multicast is a two-stage process. First, we create a broadcast mechanism that allows a packet to be forwarded to all the networks on the internet. Second, we need to refine this mechanism so that it prunes back networks that do not have hosts that belong to the multicast group.

There are two major shortcomings to this approach. The first is that it truly floods the network; it has no provision for avoiding LANs that have no members in the multicast group. We address this problem below. The second limitation is that a given packet will be forwarded over a LAN by each of the routers connected to that LAN. This is due to the forwarding strategy of flooding packets on all links other than the one on which the packet arrived

The solution to this second limitation is to eliminate the duplicate broadcast packets that are generated when more than one router is connected to a given LAN. One way to do this is to designate one router as the "parent" router for each link, relative to the source, where only the parent router is allowed to forward multicast packets from that source over the LAN. The router that has the shortest path to source S is selected as the parent; a tie between two routers would be broken according to which router has the
smallest address.

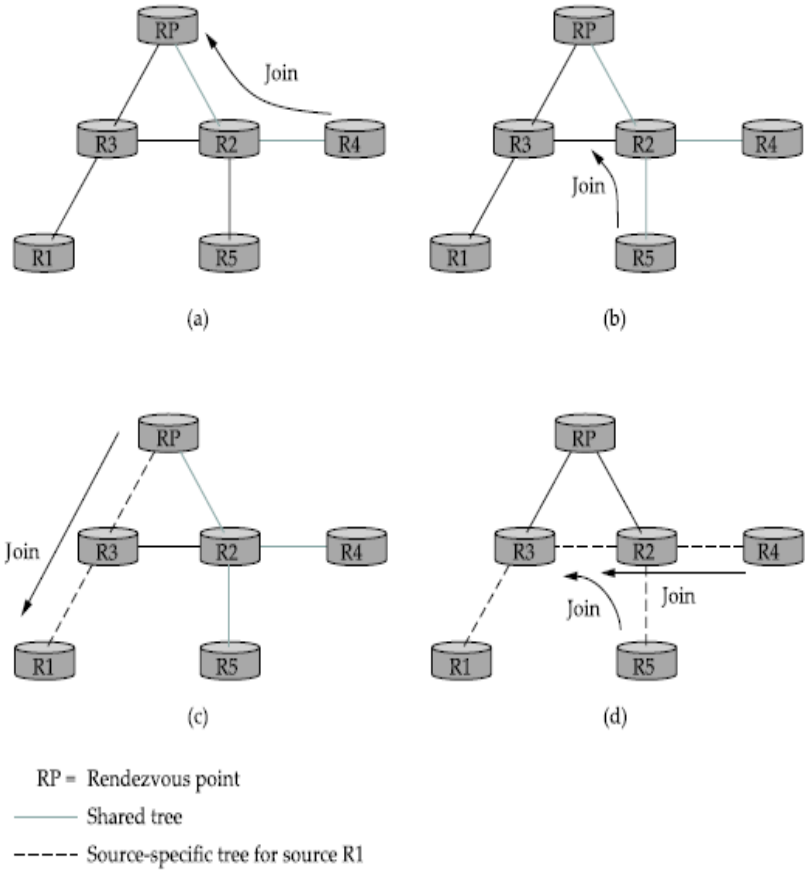
PIM

Protocol-independent multicast, or PIM, was developed in response to the scaling problems of earlier multicast routing protocols. In particular, it was recognized that the existing protocols did not scale well in environments where a relatively small proportion of routers want to receive traffic for a certain group.

In PIM-SM, routers explicitly join the multicast distribution tree using PIM protocol messages known as Join messages. To address this, PIM-SM assigns to each group a special router known as the *rendezvous point (RP)*.

A multicast forwarding tree is built as a result of routers sending Join messages to the RP.

PIM-SM allows two types of tree to be constructed: a *shared* tree, which may be used by all senders, and a *source-specific* tree, which may be used only by a specific sending host.



PIM operation. (a) R4 sends Join to RP and joins shared tree; (b) R5 joins shared tree; (c) RP builds source-specific tree to R1 by sending Join to R1; (d) R4 and R5 build source-specific tree to R1 by sending Joins to R1.

UNIT 4

TRANSPORT LAYER

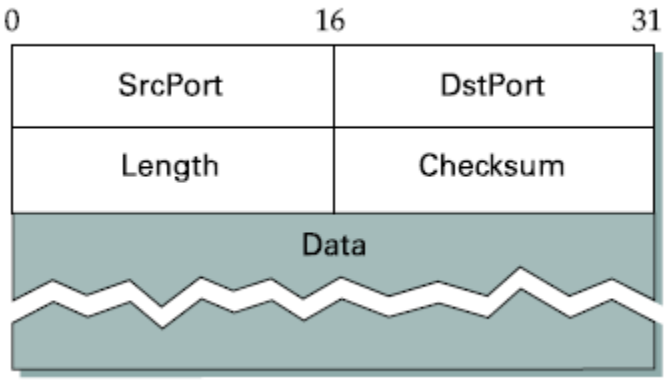
OVERVIEW OF TRANSPORT LAYER

UDP

The simplest possible transport protocol is one that extends the host-to-host delivery service of the underlying network into a process-to-process communication service. There are likely to be many processes running on any given host, so the protocol needs to add a level of demultiplexing, thereby allowing multiple application processes on each host to share the network. Aside from this requirement, the transport protocol adds no other functionality to the best-effort service provided by the underlying network. The Internet's User Datagram Protocol (UDP) is an example of such a transport protocol.

The only interesting issue in such a protocol is the form of the address used to identify the target process. Although it is possible for processes to *directly* identify each other with an OS-assigned process ID (pid), such an approach is only practical in a closed distributed system in which a single OS runs on all hosts and assigns each process a unique ID. A more common approach, and the one used by UDP, is for processes to *indirectly* identify each other using an abstract locater, often called a *port* or *mailbox*. The basic idea is for a source process to send a message to a port and for the destination process to receive the message from a port.

The header for an end-to-end protocol that implements this demultiplexing function typically contains an identifier (port) for both the sender (source) and the receiver (destination) of the message. For example, the UDP header. Notice that the UDP port field is only 16 bits long. This means that there are up to 64K possible ports, clearly not enough to identify all the processes on all the hosts in the Internet.



Format for UDP header

Reliable Byte Stream (TCP)

In terms of the properties of transport protocols given in the problem statement at the start of this chapter, TCP guarantees the reliable, in-order delivery of a stream of bytes. It is a full-duplex protocol, meaning that each TCP connection supports a pair of byte streams, one flowing in each direction.

(1) **End-to-End Issues**
(2) **Segment Format**
(3) **Connection Establishment and Termination**
(4) **Sliding Window Revisited**
(5) **Triggering Transmission**
(6) **Silly Window Syndrome**

End-to-End Issues

At the heart of TCP is the sliding window algorithm. TCP supports logical connections between processes that are running on any two computers in the Internet. This means that TCP needs an explicit connection establishment phase.

Second, whereas a single physical link that always connects the same two computers has a fixed RTT, TCP connections are likely to have widely different round-trip times.

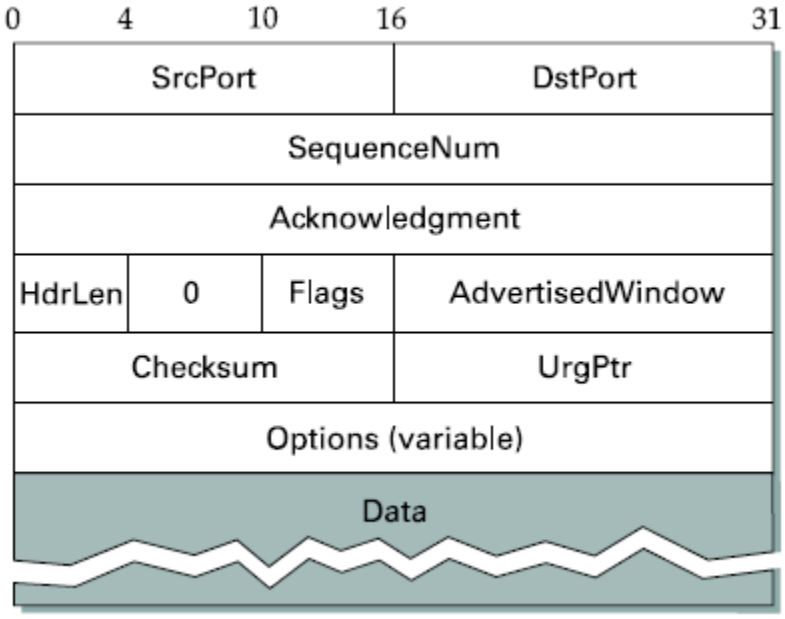
A third difference is that packets may be reordered as they cross the Internet, but this is not possible on a point-to-point link where the first packet put into one end of the link must be the first to appear at the other end.

Fourth, the computers connected to a point-to-point link are generally engineered to support the link.

Fifth, because the transmitting side of a directly connected link cannot send any faster than the bandwidth of the link allows, and only one host is pumping data into the link

Segment Format

TCP is a byte-oriented protocol, which means that the sender writes bytes into a TCP connection and the receiver reads bytes out of the TCP connection.



TCP header format

（SrcPort, SrcIPAddr, DstPort, DstIPAddr）

Connection Establishment and Termination

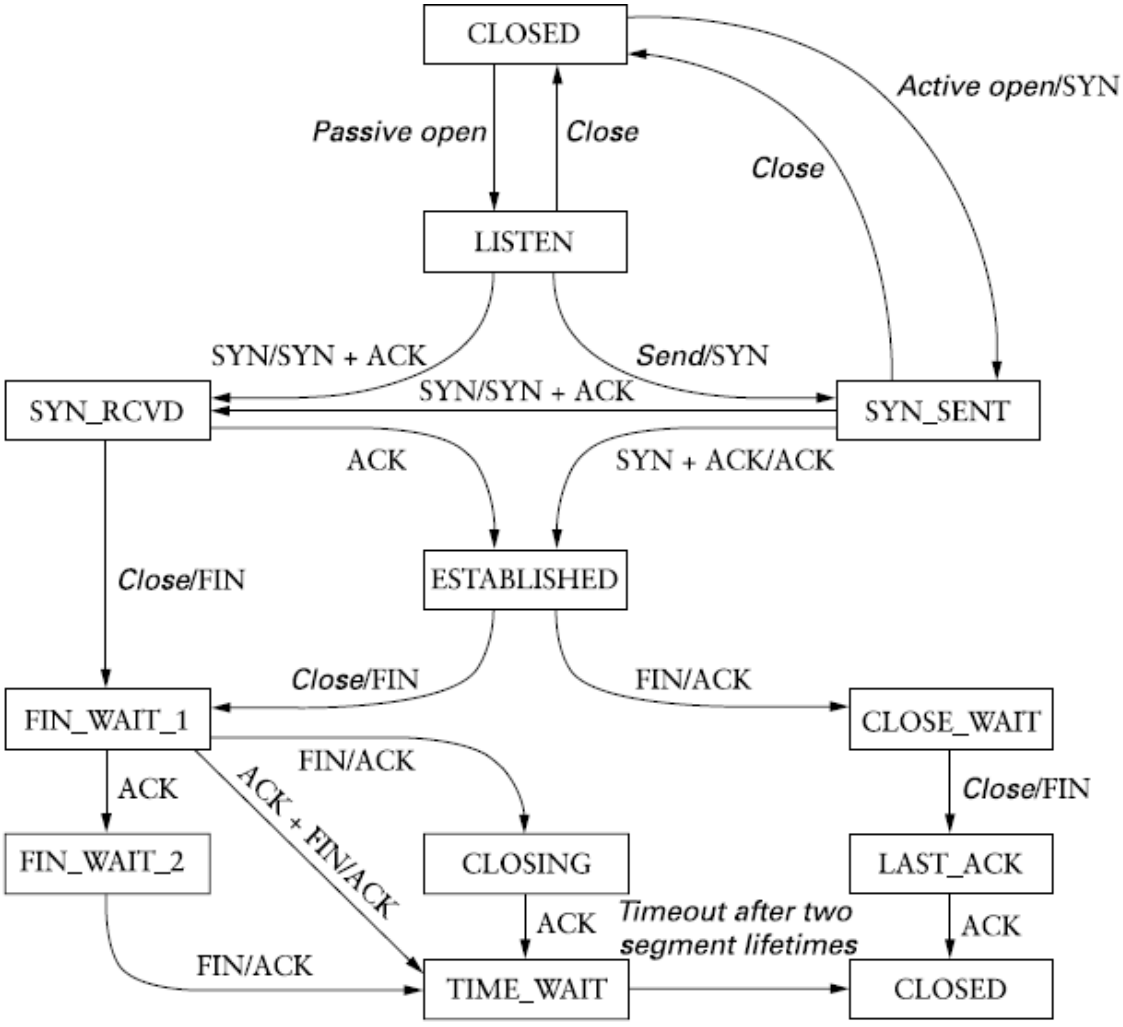
A TCP connection begins with a client (caller) doing an active open to a server (callee).

Three- way Handshake

The algorithm used by TCP to establish and terminate a connection is called a *three-wayhandshake*. We first describe the basic algorithm and then show how it is used by TCP. The three-way handshake involves the exchange of three messages between the client and the server

State-Transition Diagram

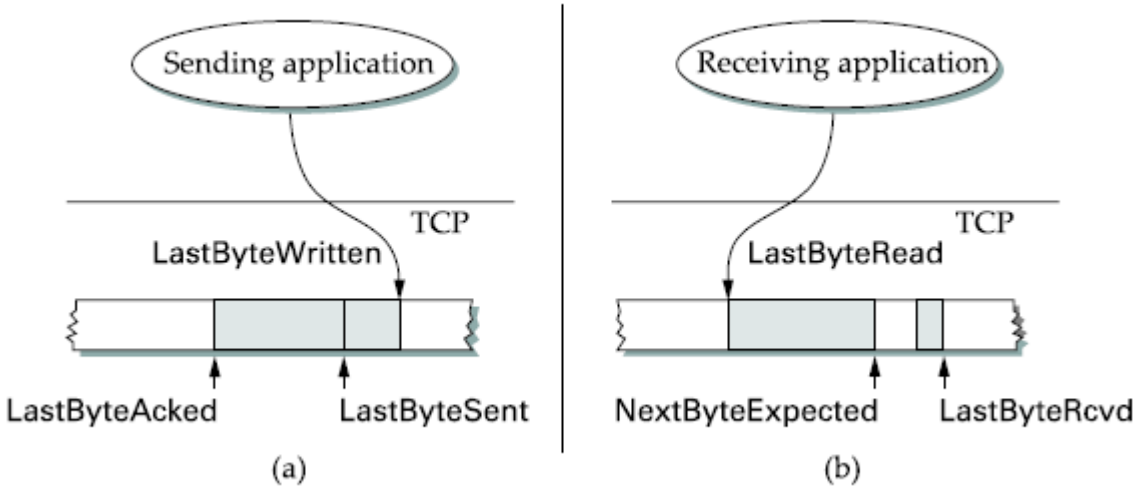
TCP is complex enough that its specification includes a state-transition diagram. This diagram shows only the states involved in opening a connection (everything above ESTABLISHED) and in closing a connection (everything below ESTABLISHED).



TCP state-transition diagram

Sliding Window Revisited

We are now ready to discuss TCP's variant of the sliding window algorithm, which serves several purposes: (1) it guarantees the reliable delivery of data, (2) it ensures that data is delivered in order, and (3) it enforces flow control between the sender and the receiver.



Reliable and Ordered Delivery

$$\text{LastByteAcked} \leq \text{LastByteSent}$$

$$\text{LastByteSent} \leq \text{LastByteWritten}$$

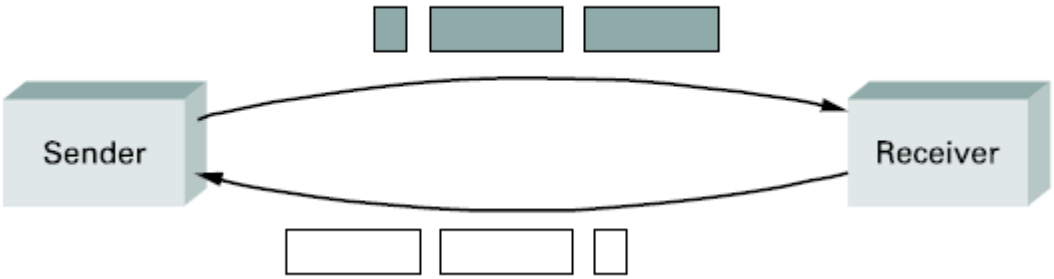
$$\text{LastByteRead} < \text{NextByteExpected}$$

$$\text{NextByteExpected} \leq \text{LastByteRcvd}+1$$

Triggering Transmission

We next consider a surprisingly subtle issue: how TCP decides to transmit a segment. As described earlier, TCP supports a byte-stream abstraction; that is, application programs write bytes into the stream, and it is up to TCP to decide that it has enough bytes to send a segment.

Silly Window Syndrome



Silly window syndrome

Nagles Algorithm

When the application produces data to send
if both the available data and the window ≥ MSS
send a full segment
else
if there is unACKed data in flight
buffer the new data until an ACK arrives
else
send all the new data now

Adaptive Retransmission

TCP guarantees the reliable delivery of data, it retransmits each segment if an ACK is not received in a certain period of time. TCP sets this timeout as a function of the RTT it expects between the two ends of the connection.

Jacobson/Karels Algorithm

$$\text{Difference} = \text{SampleRTT} - \text{EstimatedRTT}$$

$$\text{EstimatedRTT} = \text{EstimatedRTT} + (\delta \times \text{Difference})$$

$$\text{Deviation} = \text{Deviation} + \delta(|\text{Difference}| - \text{Deviation})$$

Record Boundaries

Since TCP is a byte-stream protocol, the number of bytes written by the sender are not necessarily the same as the number of bytes read by the receiver.

TCP Congestion Management

TCP congestion control is for each source to determine how much capacity is available in the network, so that it knows how many packets it can safely have in transit. Once a given source has this many packets in transit.

(1) Adaptive increase/multiplicative decrease

(2) Slow start

(3) Fast retransmit and fast recovery

Adaptive increase/multiplicative decrease

TCP maintains a new state variable for each connection, called CongestionWindow, which is used by the source to limit how much data it is allowed to have in transit at a given time. The congestion window is congestion control's counterpart to flow control's advertised window.

$$\text{MaxWindow} = \text{MIN}(\text{CongestionWindow}, \text{AdvertisedWindow})$$

$$\text{EffectiveWindow} = \text{MaxWindow} - (\text{LastByteSent} - \text{LastByteAcked})$$

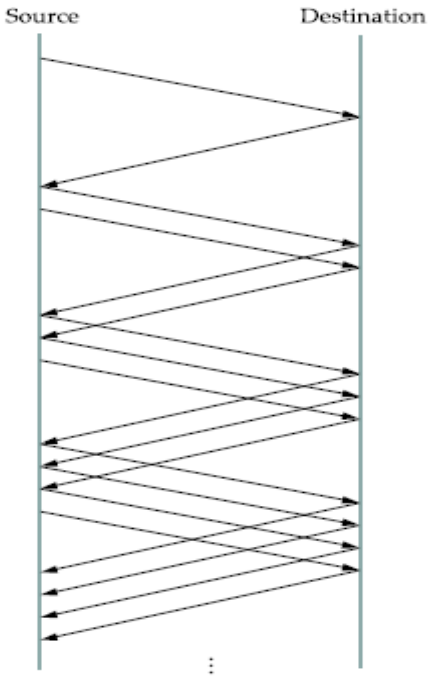
That is, MaxWindow replaces AdvertisedWindow in the calculation of EffectiveWindow. Thus, a TCP source is allowed to send no faster than the slowest component. Taken together, the mechanism is commonly called *additive increase/multiplicative decrease (AIMD)*;

$$\text{Increment} = \text{MSS} \times (\text{MSS}/\text{CongestionWindow})$$

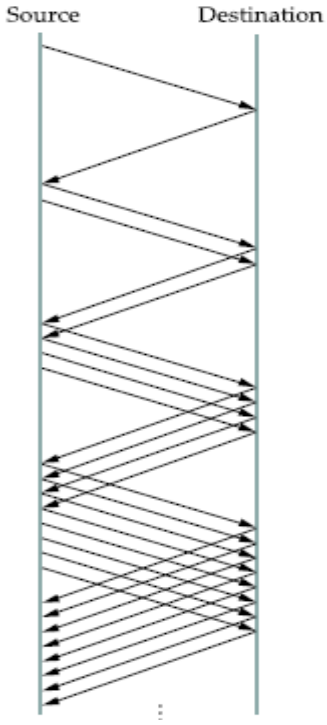
$$\text{CongestionWindow} += \text{Increment}$$

Slow Start

The additive increase mechanism just described is the right approach to use when the source is operating close to the available capacity of the network, but it takes too long to ramp up a connection when it is starting from scratch. TCP, therefore, provides a second mechanism, ironically called *slow start*



Packets in transit during additive increase, with one packet being added each RTT



Packets in transit during slow start

1 A timeout happens, causing the congestion window to be divided by 2, dropping it from approximately 22 to 11 KB, and CongestionThreshold is set to this amount;

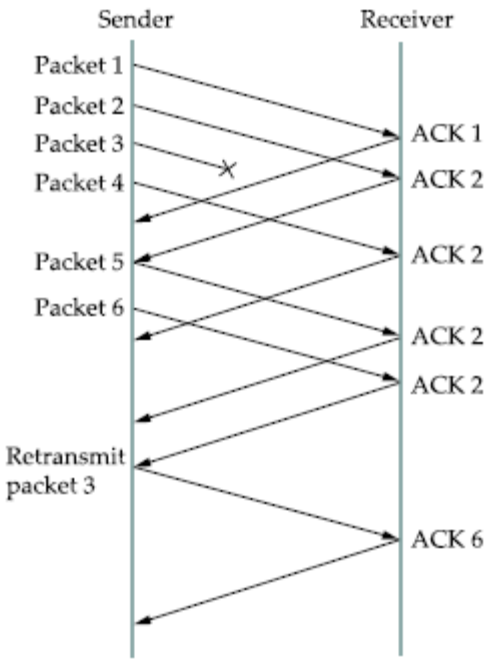
2 CongestionWindow is reset to one packet, as the sender enters slow start;

3 Slow start causes CongestionWindow to grow exponentially until it reaches CongestionThreshold;

4 CongestionWindow then grows linearly.

Fast Retransmit and Fast Recovery

The mechanisms described so far were part of the original proposal to add congestion control to TCP. It was soon discovered, however, that the coarse-grained implementation of TCP timeouts led to long periods of time during which the connection went dead while waiting for a timer to expire. Because of this, a new mechanism called *fast retransmit*



Fast retransmit based on duplicate ACKs

Flow Control

TCP cannot yet acknowledge the data the packet contains because earlier data has not yet arrived—TCP resends the same acknowledgment it sent the last time. This second transmission of the same acknowledgment is called a *duplicate ACK*.

Congestion Avoidance Mechanisms

It is important to understand that TCP's strategy is to control congestion once it happens, as opposed to trying to avoid congestion in the first place. In fact, TCP repeatedly increases the load it imposes on the network in an effort to find the point at which congestion occurs, and then it backs off from this point.

DECbit

The first mechanism was developed for use on the Digital Network Architecture (DNA), a connectionless network with a connection-oriented transport protocol. This mechanism could, therefore, also be applied to TCP and IP. As noted above, the idea here is to more evenly split the responsibility for congestion control between the routers and the end nodes. Each router monitors the load it is experiencing and explicitly notifies the end nodes when congestion is about to occur. This notification is implemented by setting a binary congestion bit in the packets that flow through the router; hence the name DECbit. The destination host then copies this congestion bit into the ACK it sends back to the source. Finally, the source adjusts its sending rate so as to avoid congestion. The following discussion describes the algorithm in more detail.

A single congestion bit is added to the packet header. A router sets this bit in a packet if its average queue length is greater than or equal to 1 at the time the packet arrives. This average queue length is measured over a time interval that spans the last busy+idle cycle, plus the current busy cycle.

Essentially, the router calculates the area under the curve and divides this value by the time interval to compute the average queue length. Using a queue length of 1 as the trigger for setting the congestion bit is a trade-off between significant queuing (and hence higher throughput) and increased idle time (and hence lower delay). In other words, a queue length of 1 seems to optimize the power function.

RED

RED Stands for Random Early Detection. The main idea is to provide congestion control at the router for TCP flows. RED is based on DECbit and was designed to work well with TCP.

RED implicitly notifies sender by dropping packets. Packet dropping probability is increased as the average queue length increases.

Properties of RED

(1) Drops packets before queue is full, in the hope of reducing the rates of some flows.
(2) Drops packets for each flow roughly in proportion to its rate.
(3) Drops are spaced out in time.
(4) Because it uses average queue length RED is tolerant of bursts
(5) Random drops hopefully desynchronize TCP sources.

RED calculates the average queue length using a weighted running average following formula is used for calculating average queue length.

Average length=(1-weight)* Average length + weight * Sample length

RED uses a packet drop profile to handle packet discarding. This profile defines a set of dropping probabilities according to the level of queue occupancy.

1. If the queue occupancy lies beneath the minimum threshold, then packet drop does not occur.
2. If the queue occupying is some where between minimum threshold and packets.
3. When the queue occupancy crosses maximum threshold, then all now packets attempting to gather the queue are discarded.

Quality of Services

In any multimedia application audio/video packets are delay sensitive but internet all packets are treated equally.

Analyzing varying network scenarios principles of quality of services(Qos) needed for multimedia application are delivered.

(1) Policing
(2) Integrated Services

Packet-switched networks have offered the promise of supporting multimedia applications, that is, those that combine audio, video, and data. After all, once digitized, audio and video information.

There is more to transmitting audio and video over a network than just providing sufficient bandwidth, however. Participants in a telephone conversation, for example, expect to be able to converse in such a way that one person can respond to something said by the other and be heard almost immediately. Thus, the timeliness of delivery can be very important. We refer to applications that are sensitive to the timeliness of data as *real time applications*.

The distinguishing characteristic of real-time applications is that they need some sort of assurance *from the network* that data is likely to arrive on time (for some definition of "on time"). Whereas a non-real-time application can use an end-to-end retransmission strategy to make sure that data arrives *correctly*, such a strategy cannot provide timeliness: Retransmission only adds to total latency if data arrives late.

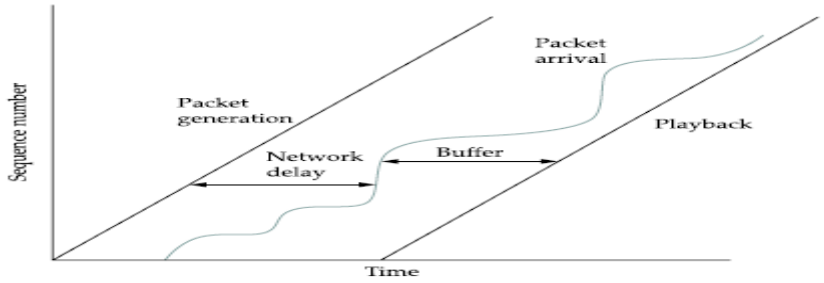
Application Requirements

Before looking at the various protocols and mechanisms that may be used to provide quality of service to applications, we should try to understand what the needs of those applications are. To begin, we can divide applications into two types: real-time and non real-time.

An audio application

REAL TIME AUDIO EXAMPLE

Data is generated by collecting samples from a microphone and digitizing them using an analog-to-digital (A→*D*) converter. The digital samples are placed in packets, which are transmitted across the network and received at the other end. At the receiving host, the data must be *played back* at some appropriate rate.



A playback buffer

If it gets delayed a long time, then it will not need to be stored for very long in the receiver's buffer before being played back. Thus, we have effectively added a constant offset to the playback time of all packets as a form of insurance. We call this offset the *playback point.* The only time we run into trouble is if packets get delayed in the network for such a long time that they arrive after their playback time, causing the playback buffer to be drained.

Approaches to QoS Support

Considering this rich space of application requirements, what we need is a richer service model that meets the needs of any application. This leads us to a service model with not just one class (best effort), but with several classes, each available to meet the needs of some set of applications. Toward this end, we are now ready to look at some of the approaches that have been developed to provide a range of qualities of service. These can be divided into two broad categories:

- *Fine-grained* approaches, which provide QoS to individual applications or flows;
- *Coarse-grained* approaches, which provide QoS to large classes of data or aggregated traffic.

UNIT 5

APPLICATION LAYER

Traditional Applications

- World wide web, electronic mail system and domain name system are the traditional application of the application layer network
- Application needs their own protocols. These applications are part network protocol and part traditional application program.
- This chapter explains some of the most popular network applications available today. Two of the most popular applications are world wide web and email system
- Both of these applications use the request / reply method. The uses send requests to servers, which then respond accordingly.
- Hypertext transport protocol (HTTP) is an application protocol. HTTP is used to retrieve web pages from remote servers.
- A web client uses application programs like internet explorer, chrome, firefox and mozila.
- Widely used standardized application protocol.

Electronic Mail

E-mail is an asynchronous communication medium. Electronic mail is used for sending a single message that includes text, voice, and video.

Electronic mail is fast, easy to distribute and in expensive.

Simple mail transfer protocol is the standard mechanisms for electronic mail in the internet. SMTP is the TCP/IP mail delivery protocol.

Email is not a real time service in the fairly large delays can be tolerated

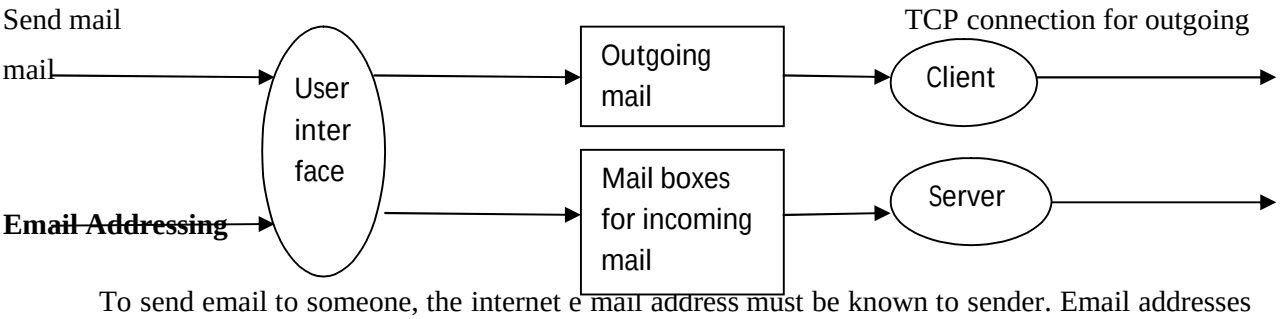
It is also not connection oriented in network connection does not need to be setup expressly for each individual message

Mail server handles incoming and outgoing mails

Following are the ways to access the e-mail

1) Web based email services
2) E-mail through a LAN

3) Unix shell account
4) Using mail client



To send email to someone, the internet e mail address must be known to sender. Email addresses look like this vilas@hotmail.com

The email address has two main parts, joined by @. In this example, vilas is the username. User name can contain numbers, under scores, periods and some other special characters. Commas, spaces and parenthesis are not allowed

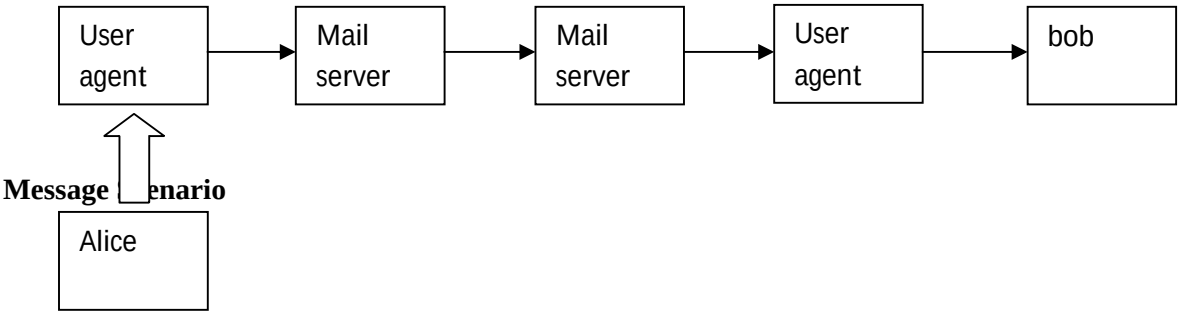
SIMPLE NETWORK TANSFER PROTOCOL (SMTP)

- SMTP is application layer protocol of TCP/IP model
- SMTP transfers message from senders mail servers to the recipients mail servers
- SMTP interacts with the local mail system and not the user
- SMTP uses a TCP socket on port 25 to transfer email reliability from client to server
- Email is temporarily stored on the local and eventually transferred directly to receiving server.
- Client/Server interaction follows and command/response algorithm
  - ✓ Commands are a status code and an optional phase
  - ✓ Response are a status code and an optional phase
- Mail client application interacts with a local SMTP server to initiate the delivery of an email message
- There is a input queue and an output queue at the interface between the local mail system and the client and the server parts of the SMTP
- The client is concerned with initiating the transfer of mail to another system while server is concerned with receiving mail. Before the email message can be transferred, the application process must be set up a TCP connection to the local SMTP server.

SCENARIO: ALICE SENDS MESSAGE TO BOB

- Alice uses user agent (UA) to compose message and bob@sinhgad.edu

- Alice UA sends message to her mail server, message placed in message queue.
- Client side of SMTP opens TCP Connection with Bobs mail Server
- SMTP clients sends alices message over the TCP connection
- Bobs mail server places the message in bobs mailbox
- Bob invokes his user agent to resd message.



Message enario

SMTP Commands are

HELLO: initiate a mail transaction, identifying the sender to the recipient

MAIL FROM: tells the remote SMTP that a new mail transaction is beginning

RCPT TO: The sending SMTP sends a RCPT command for each intended receiver

DATA: If accepted, the sender transfers the actual message. End of message is indicated by sending "." On a line by itself

QUIT: Terminate the connection

MULTIPURPOSE INTERNET MAIL EXTENSIONS

- MIME is a supplementary protocol that allows non ASCII data to be sent through SMTP
- MIME defined by IETF to allow transmission
- It allows arbitrary data to be encoded in ASCII for normal transmission
- All media types that are sent or received over the www
- Message sent using MIME encoding include information that describes the type of data and the encoding that was used.
- RFC822 specifies the exact for mail header lines as well as their semantic interpretations
- MIME define five headers

  MIME-version

Content-type

Content-transfer-encoding

Content-id

Content-description

Mail Message Header

- From : iresh@email.com
- To: rupali@sinhged.edu
- MIME version
- Content type: image gif
- Content- transfer-encoding

Mail Types and Subtypes

- Each MIME content type must contain two identifiers
  - ✓ Content type
  - ✓ Content Subtypes
- There are seven standard content types that can appear in a MIME content type declaration

Type	**Subtype**	**Description**
Text Multipart	Plain Mixed	Unformatted Text Body contain ordered part of different data types
	Parallel	Same as above
	Digest	Similar to mixed, but the default is message
	Alternative	Parts are different version of the same message
Video	MPEG	Video is in MPEG format
Audio	Basic	Single channel encoding of voice at 8 Khz
Image	JPEG	Image is in JPEG format

POP 3(Post office protocol 3)

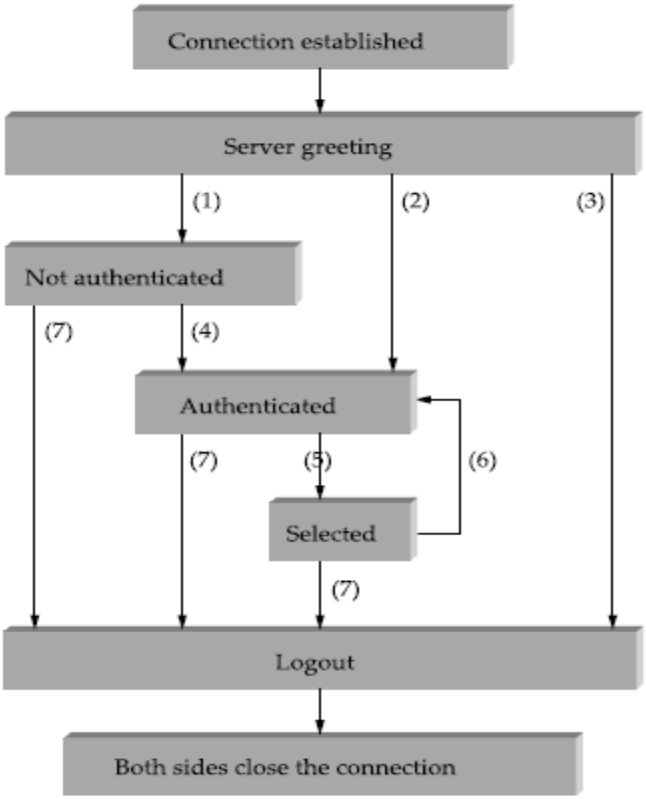
- Post office protocol 3(POP3) is used to transfer email messages from a mail server to mail client software
- POP3 begins when the user agent opens a TCP connection to the mail server on port 110
- After TCP connection established POP3 progresses three phases
    - ✓ Authorization
    - ✓ Transaction
    - ✓ Update
- In authorization phase, user agent sends a user name and a password to authenticate the user downloading the mail.
- In transaction phase, the user agent retrieves massages. In this phase, user agent can also mark message for deletion, remove marks
- In update phase, it occurs after the client has issued the quit command, ending the POP3 session.
- POP3 has two modes: Delete mode
  Keep mode
- In the keep mode, the mail remains in the mail box after retrieval

Limitations of POP3

1. POP3 does not allow, the mail remains in the mailbox after retrieval
2. In the keep mode, the mail remains in the mailbox after retrieval

IMAP

- IMAP is the internet mail access protocol. IMAP is more powerful to help the user who uses multiple computers
- IMAP does not copy email to the users personal machine because the user may have several.
- An IMAP client connects to a server by using TCP.
- IMAP supports the following modes for accessing email messages
    - ✓ Offline mode
    - ✓ Online mode
    - ✓ Disconnected mode



IMAP state transition diagram

Offline mode : A client periodically connects to the server to download email messages. After downloading, messages are deleted from the server

Online mode: Client process email messages on the server. The email messages are stored on the server itself but are processed by an application on the clients end

Disconnected mode: in this mode, both offline and online modes are supported.

IMAP4 provides the following extra functions

1) User can check the email header prior to downloading
2) User can partially download email
3) A user can create, delete or rename mailboxes on the mail server
4) A user can create a hierarchy of mail boxes in a folder for email storage
5) User can search the contents of the email for a specific string of characters.

Not authenticate: client provides authentication information to the server.

Authenticated: Server verify the information and client is now allowed to perform operations on a mailbox

Selected: Client is allowed to access of manipulate individual message within the mailbox

Logout: Client sends logout command for closing IMAP session

HYPERTEXT TRANSFER PROTOCOL

- The standard web transfer protocol is hyper text transfer protocol
- The HTTP protocol consists of two fairly distinct items, the set of request from browsers to server and the set of responses going back the other way
- All the newer versions of HTTP support two kinds of request : simple request and full request. A simple request is just a single GET line naming the page desired, without the protocol version
- Full request are indicated by the presence of the protocol version on the GET request line. Requests may consist of multiple lines, followed by a blank line to indicate the end of the request
- Although HTTP was designed for use in the web, it has been intentionally made more general than necessary with an eye to future object oriented applications
- When accessing general objects, additional object specific methods may also be available. The names are case sensitive, so GET is a legal method.

HTTP TRANSACTION

- HTTP uses the services of TCP. HTTP is a stateless protocol
- The client initializes the transaction by sending a request message
- The server replies by sending a response

Message

- HTTP message are two types
  - ✓ **Request**
  - ✓ **Response**
- Both message type used same format
- Request message consists of a request line, headers and a body

Request line

- ✓ **Request type**
- ✓ **Resources**

- ✓ **HTTP version**
- ❖ URL is a standard for specifying any kind of information on the internet. The URL define four things
    1) Method
    2) Host computer
    3) Port
    4) Path

Example http://www.technicalpublication.org/home.html

- ❖ The method is the protocol used to retrieve the document. Several different protocols can retrieve a document
- ❖ The URL can optionally contain the port number of the server.
- ❖ The request type field in a request message defines several kinds of messages referred to as methods
    1. **GET**
    2. **HEAD**
    3. **POST**
    4. **PUT**
    5. **PATCH**
    6. **DELETE**
    7. **COPY**
    8. **MOVE**
    9. **LINK**
    10. **OPTION**

Web services

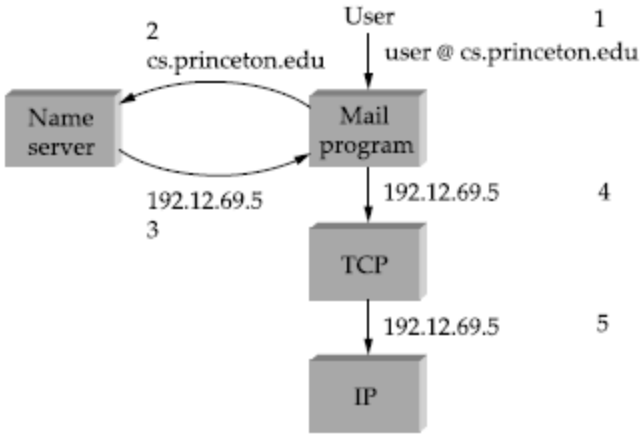
- ❖ An applications framework includes the presentation services, server side processing, session management application data caching, application logic, persistence, transaction
- ❖ Web services are self contained modular applications that can be described, published, located
- ❖ Operations in a web services architecture are:

(1) Publish
(2) Find
(3) Bind

NAME SERVICES (DNS)

We have been using addresses to identify hosts. While perfectly suited for processing by routers, addresses are not exactly user friendly. It is for this reason that a unique *name* is also typically assigned to each host in a network. Already in this section we have seen application protocols like HTTP using names such as www.princeton.edu. We now describe how a naming service can be developed to map user-friendly names into router-friendly addresses. Name services are sometimes called *middleware*

Host names differ from host addresses in two important ways. First, they are usually of variable length and mnemonic, thereby making them easier for humans to remember. (In contrast, fixed-length numeric addresses are easier for routers to process.) Second, names typically contain no information that helps the network locate (route packets toward) the host. Addresses, in contrast, sometimes have routing information embedded in them; *flat* addresses (those not divisible into component parts) are the exception. Before getting into the details of how hosts are named in a network, we first introduce some basic terminology. First, a *namespace* defines the set of possible names. A namespace can be either *flat* (names are not divisible into components), or it can be *hierarchical* (Unix file names are an obvious example). Second, the naming system maintains a collection of *bindings* of names to values.



Names translated into addresses, where the numbers 1–5 show the sequence of steps in the process.

Domain Hierarchy

DNS implements a hierarchical namespace for Internet objects. Unlike Unix file names, which are processed from left to right with the naming components separated with slashes, DNS names are processed from right to left and use periods as the separator. (Although they are processed from right to

left, humans still “read” domain names from left to right.) An example domain name for a host is cicada.cs.princeton.edu.

Name Servers

The complete domain name hierarchy exists only in the abstract. We now turn our attention to the question of how this hierarchy is actually implemented. The first step is to partition the hierarchy into sub trees called *zones*.

Each name server implements the zone information as a collection of *resource records*. In essence, a resource record is a name-to-value binding, or more specifically, a 5-tuple that contains the following fields:

(Name, Value, Type, Class, TTL)

The Name and Value fields are exactly what you would expect, while the Type field specifies how the Value should be interpreted. For example, Type = A indicates that the Value is an IP address. Thus, records implement the name-to-address mapping

We have been assuming. Other record types include

- NS: The Value field gives the domain name for a host that is running a name server that knows how to resolve names within the specified domain.
- CNAME: The Value field gives the canonical name for a particular host; it is used to define aliases.
- MX: The Value field gives the domain name for a host that is running a mail server that accepts messages for the specified domain.

Name Resolution

Given a hierarchy of name servers, we now consider the issue of how a client engages these servers to resolve a domain name. To illustrate the basic idea, suppose the client wants to resolve the name penguins.cs.princeton.edu relative to the set of servers given in the previous subsection.

Network Management (SNMP)

A network is a complex system, both in terms of the number of nodes that are involved and in terms of the suite of protocols that can be running on any one node. Even if you restrict yourself to worrying about the nodes within a single administrative domain, such as a campus, there might be dozens of routers and hundreds—or even thousands—of hosts to keep track of. If you think about all the state that is maintained and manipulated on any one of those nodes—for example, address translation tables, routing tables, TCP connection state, and so on—then it is easy to become depressed about the prospect of having to manage all of this information.

SNMP is essentially a specialized request/reply protocol that supports two kinds of request messages: GET and SET.

SNMP depends on a companion specification called the *management information base (MIB).* The MIB defines the specific pieces of information—the MIB *variables*— that you can retrieve from a network node.

The current version of MIB, called MIB-II, organizes variables into 10 different *groups*. You will recognize that most of the groups correspond to one of the protocols described in this book, and nearly all of the variables defined for each group should look familiar. For example:

- ✓ System: general parameters of the system (node) as a whole, including where the node is located, how long it has been up, and the system's name.
- ✓ Interfaces: information about all the network interfaces (adaptors) attached to this node, such as the physical address of each interface, or how many packets have been sent and received on each interface.
- ✓ Address translation: information about the Address Resolution Protocol (ARP), and in particular, the contents of its address translation table.
- ✓ IP: variables related to IP, including its routing table, how many datagrams it has successfully forwarded, and statistics about datagram reassembly. Includes counts of how many times IP drops a datagram for one reason or another.
- ✓ TCP: information about TCP connections, such as the number of passive and active opens, the number of resets, the number of timeouts, default timeout settings, and so on. Per-connection information persists only as long as the connection exists.
- ✓ UDP: information about UDP traffic, including the total number of UDP datagrams that have been sent and received.