

Transmission

Michael Lund, ph.d. student



Overview

- Network hardware: LAN, MAN, WAN, Internet
- Network software: Layered protocols, OSI, TCP/IP
- Multimedia requirements: multicasting, bandwidths



Two types of transmission technology

- Broadcast networks
 - A single communication channel is shared by all the computers on the network.
 - Small messages, called packets are sent by any computer and received by all the others.
 - All computers look in the address fields: Is this packet for me?
 - When broadcasting the packet is sent and processed by all computers on the network.
 - When multicasting the packet is sent and processed by a subset of computers.
- Point-to-point networks:
 - Many connections between individual computers. A packet is only received by one computer
 - Often multiple routes of different lengths are possible (so routing is important) in point-to-point networks



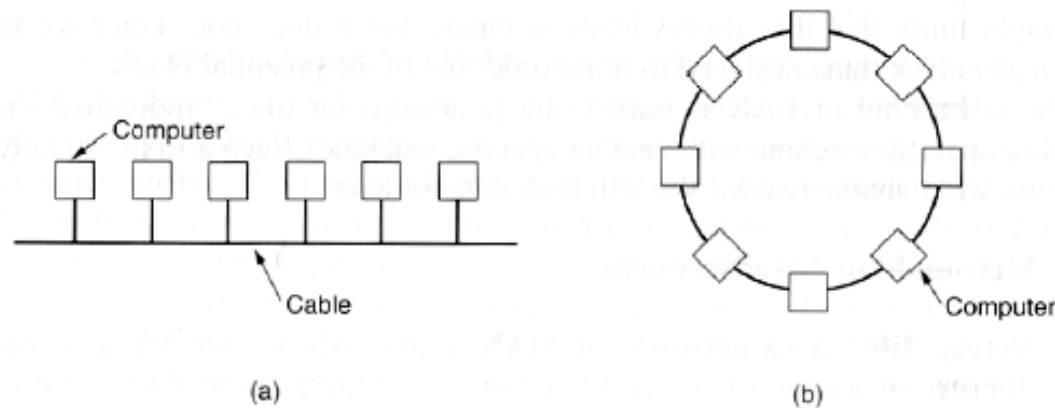
Classification by scale

Interprocessor distance	Processors located in same	Example
0.1 m	Circuit board	Data flow machine
1 m	System	Multicomputer
10 m	Room	Local area network
100 m	Building	
1 km	Campus	
10 km	City	Metropolitan area network
100 km	Country	Wide area network
1,000 km	Continent	
10,000 km	Planet	The Internet



Local Area Network

- Restricted in size (worst-case transmission time is bound and known)
- Typical LAN are broadcast networks.
- Speed is



Two broadcast networks (a) Bus, (b) Ring



Ethernet IEEE 802.3

- Bus-based Local Area Network of broadcast type.
- Decentralized control operating at 10 or 100 Mbit/sec
- Any computer may send whenever it wants.
- If two packets collide, the computer wait a random time and send again



IBM Token Ring IEEE 802.5

- Ring based LAN
- Each Bit propagates around its own, not waiting for the rest of the packet.
- Operating at 4 and 16 Mbps



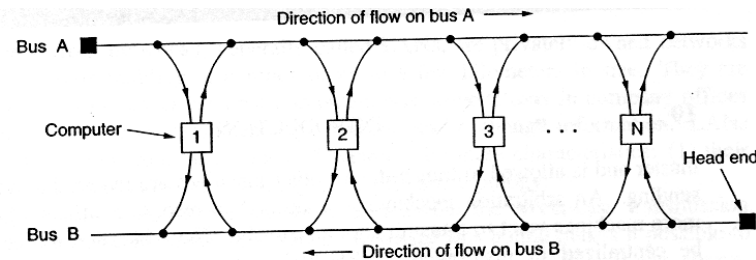
Dynamic and Static Broadcast Networks

- Static allocation
 - Time is divided into discrete intervals and a computer is only allowed to broadcast when its time slot comes up.
 - Time is wasted when computers have nothing to say.
- Dynamic allocation
 - Can either be centralized or decentralized
 - In Centralized allocation a specialized unit determines who goes next.
 - In decentralized allocation there is not central regulation. All computers must decide to transmit or not



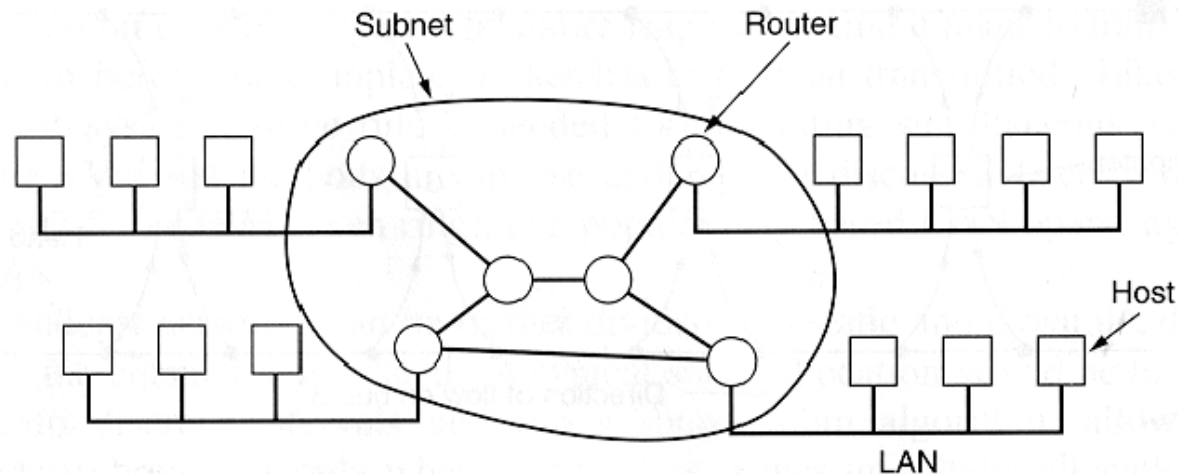
Metropolitan Area Networks

- Man is like LAN but bigger. Uses similar technology.
- Might cover a group of nearby corporate offices
- The IEEE 802.6 standard
 - Distributed Queue Dual Bus (DQDB)
 - Simplified design



Wide Area Networks

- WANs span large geographical areas (e.g. country)
- WANs are point-to-point networks
- Subnets consists of transmission lines and switching elements
- Each end node is called a Host
- Most WANs use Point-to-point, store-and-forward, or packet-switched subnets



Store-and-forward subnets

- Nearly all wide area networks have store-and-forward subnets
 - When a packet is sent from one router to another via one or more intermediate routers, the packet is received at each intermediate router in its entirety, stored there until the required output line is free, and then forwarded.



Router interconnection topology

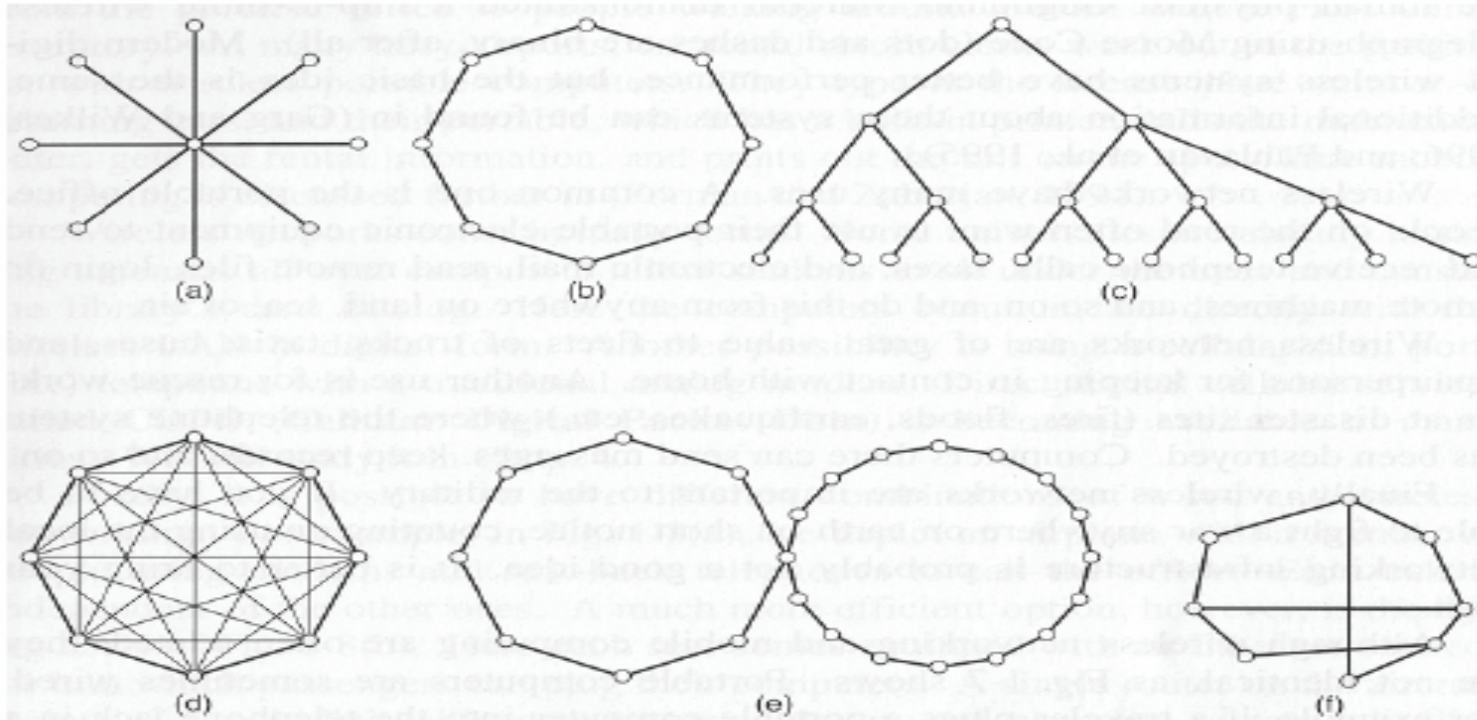


Fig. 1-6. Some possible topologies for a point-to-point subnet. (a) Star. (b) Ring. (c) Tree. (d) Complete. (e) Intersecting rings. (f) Irregular.



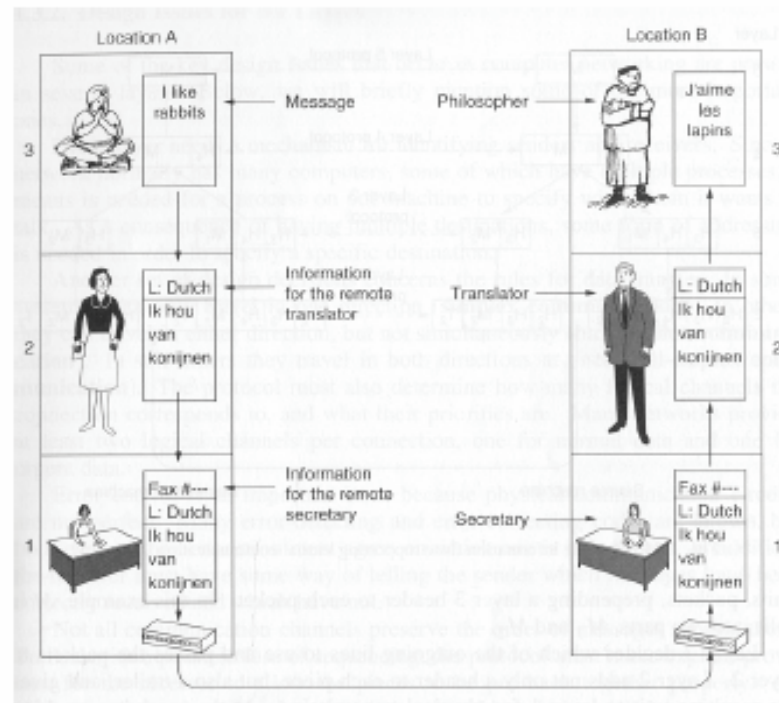
Wireless networks

- WLAN
- FWA
- GRPS
- Bluetooth
- IR
- Radio
- Satellites

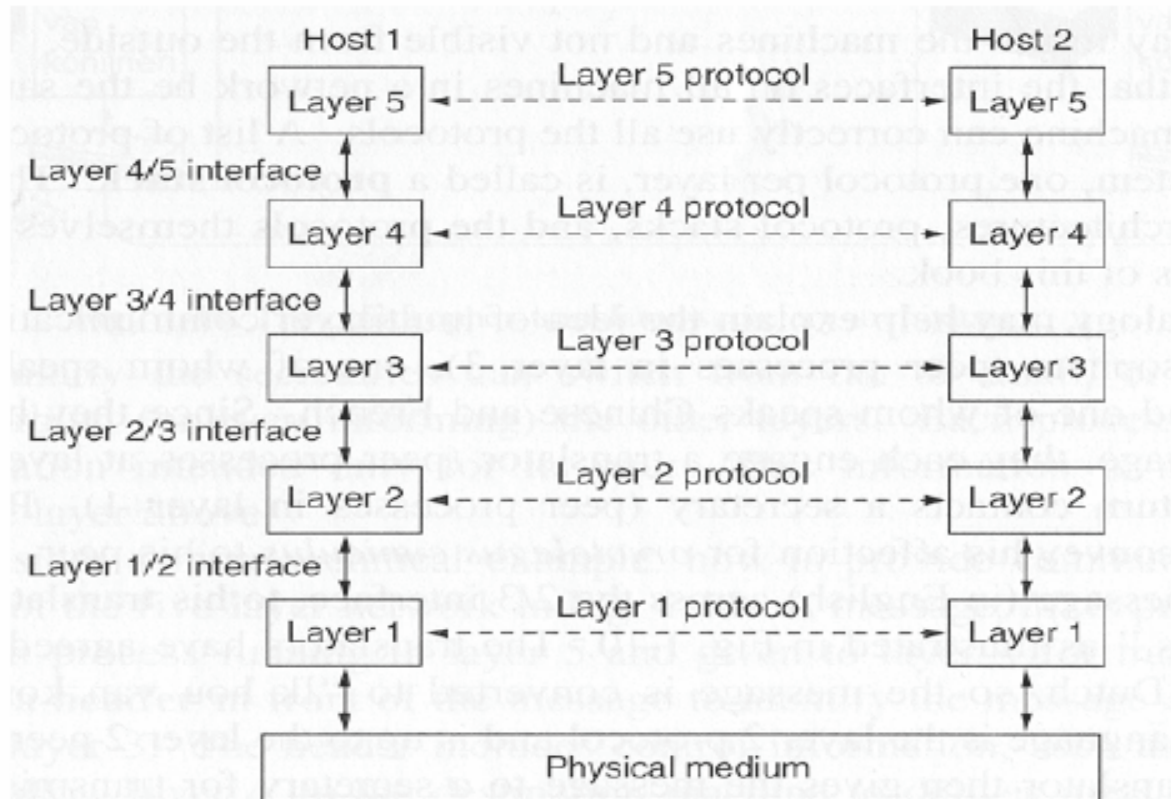


Protocols

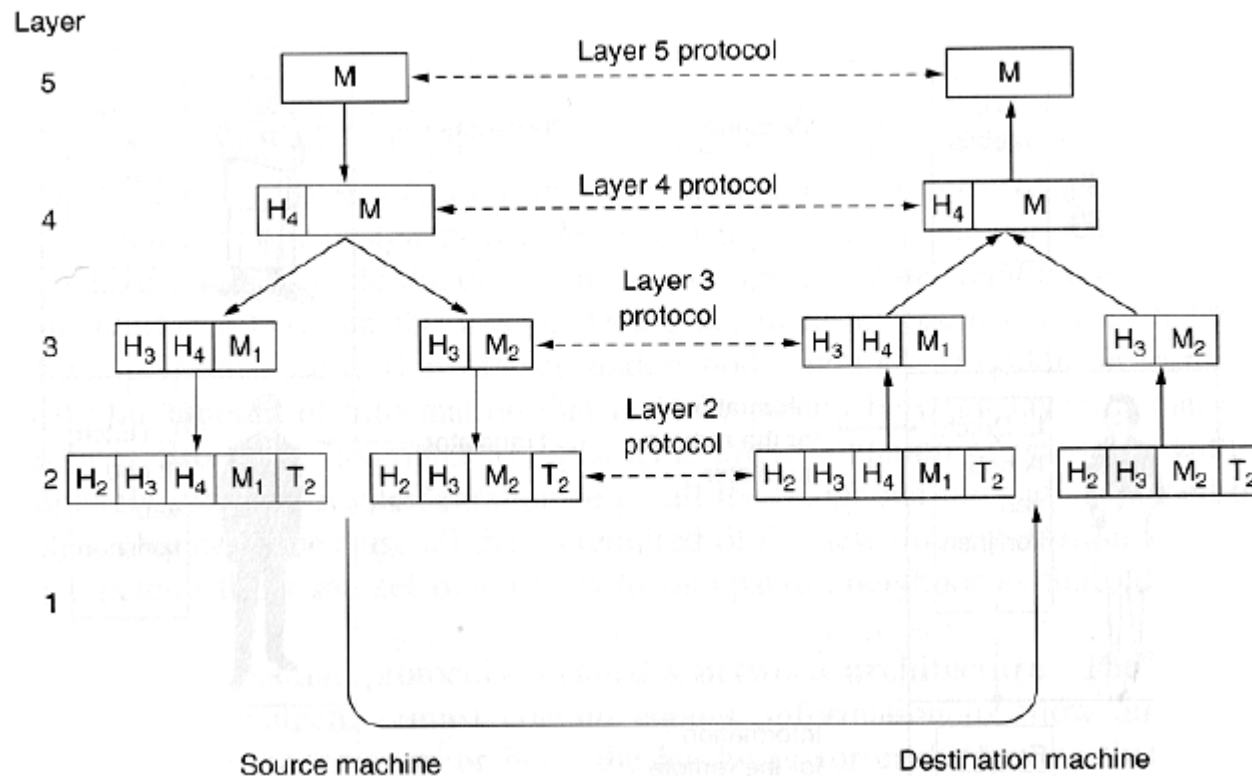
- A Protocol is a mutual agreement on language and how to communicate



Layered protocols



Protocol structure



Protocol design issues

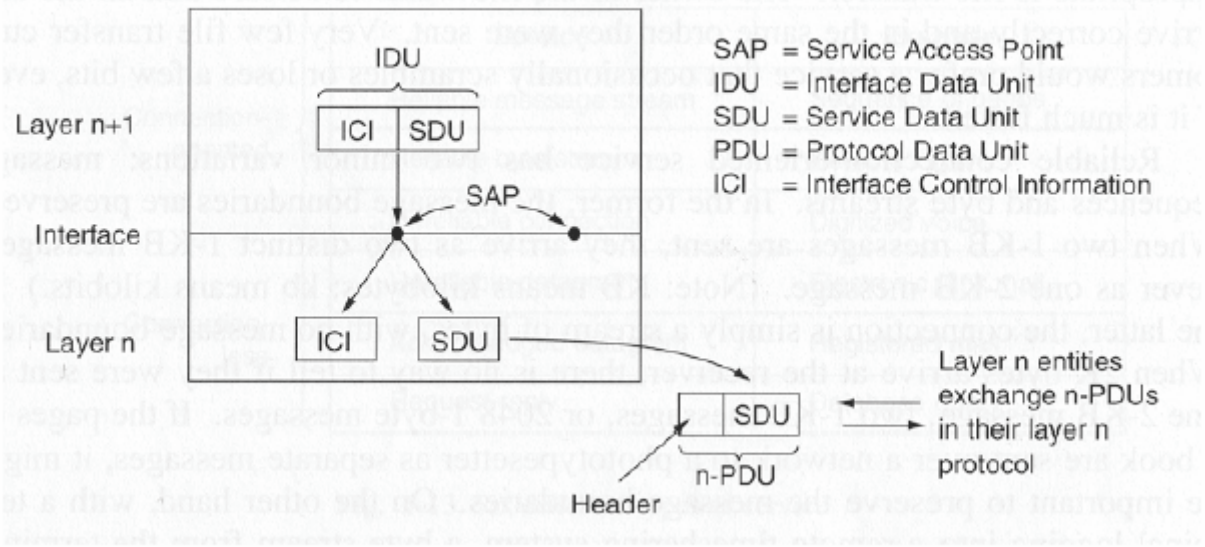
- Every layer need a way to identify senders and receivers.
- Data transfer rule
 - Simplex one-way transmission
 - Half-duplex two-way transmission but not simultaneously
 - Duplex two-way transmission
- Error control: error-detection or error-correcting
- Packet out of order.
 - Packet numbering
 - Handling packets out of order
- Transfer synchronization: agreed speed, handshake
- Multiplexing (combine multiple unrelated messages into one)



Protocol interfaces

Service user

Service provider



Service structure

- Connectionless service: each packets has the full destination address and is routed independent of the other packets.
- Connection-oriented service: like telephone system. Establish a connection, use the connection, and then release the connection. (e.g. file transfer)

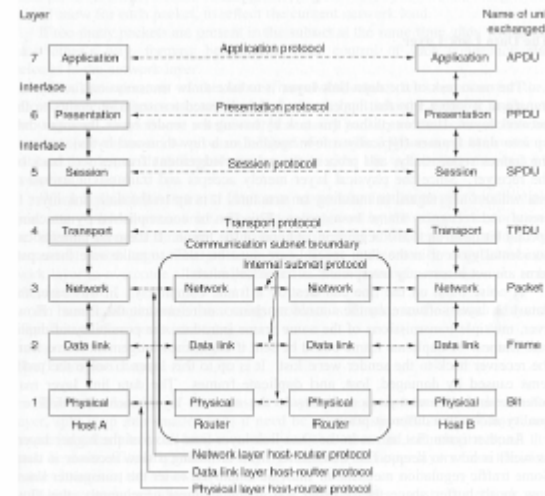
	Service	Example
Connection-oriented	Reliable message stream	Sequence of pages
	Reliable byte stream	Remote login
	Unreliable connection	Digitized voice
Connection-less	Unreliable datagram	Electronic junk mail
	Acknowledged datagram	Registered mail
	Request-reply	Database query



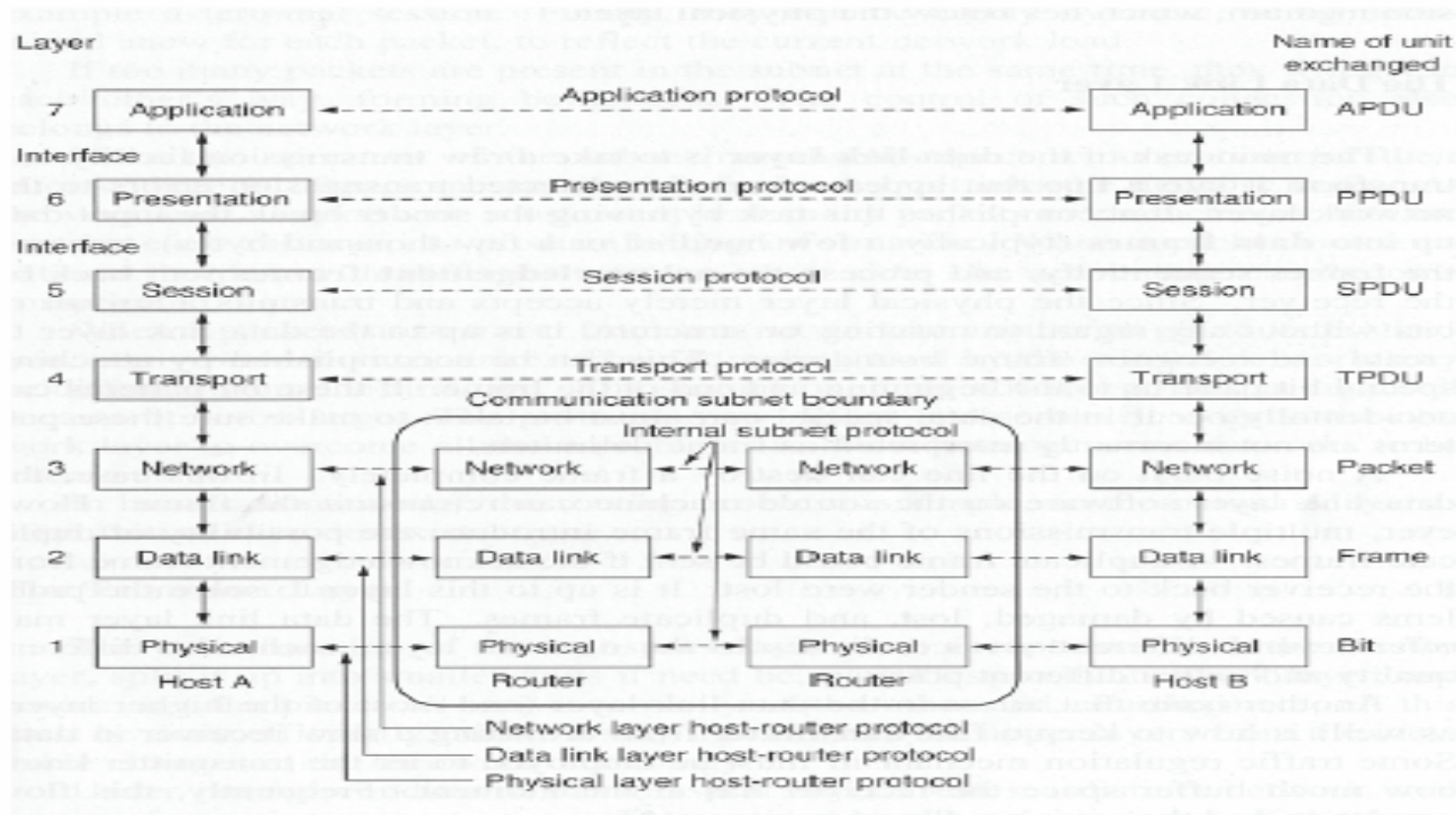
The OSI ISO reference model

Principles:

Layer corresponds to abstraction. Each layer has a well defined function. Layers must aim towards standards. Minimal information flow across interfaces. Each Layer should have ONE function



The OSI ISO reference model



Physical layer

- Function: transmit bits over the communication channel.
- A 1 bit send must be received as a bit, etc.
- Design issues are in terms of physical units:
 - How many volts should be use to represent 1
 - How many milliseconds last a bit
 - Half-Duplex or Duplex
 - Establishing connections
 - number of pins on connector, etc.



Data link layer

- Function: make communication channel seem error free
- Method: break data into frames.
- Each frame has parities to check for consistency
- If one frame is obscured it must be retransmitted
- Traffic regulation mechanism



The network Layer

- Function: make networks have virtual addresses, make the communication flow
- Provide a number of services: routing of messages, etc.
- Also accounting may be part of network layer (produce billing information).
- The network layer may handle protocol differences (to allow heterogeneous networks to be connected).
- Addresses may be static or dynamic
- In Broadcasting networks the network layer is often thin (no routing problem)



The transport layer

- Function: interface between the network layer and the session layer.
- Basic property is to split data into smaller packets that may be handled by the network layer.
- Create a distinct network connection for each transport connection.
- It determines which of the network layer services to use.
- If high throughput is required it may double a channel using multiplexing.
- If creating a network connection is expensive it may share a channel using multiplexing
- It hides the physical structure of the network.
- Most popular connection is error-free point-to-point channel



The session layer

- Function: to provide session between machines
- A session is a communication sequence.
- Functions are:
 - Log into remote timesharing system
 - File transfer
- Dialogue control: one-way, two-way
- Tokens management: Handle tokens between operators (critical operations)
- Synchronization: checkpoints in data streams



The presentation layer

- Function: handle syntax and semantics of the information transmitted, represent abstract data types.
- It codes in standard way:
 - Integers
 - Characters strings
 - Floating point numbers

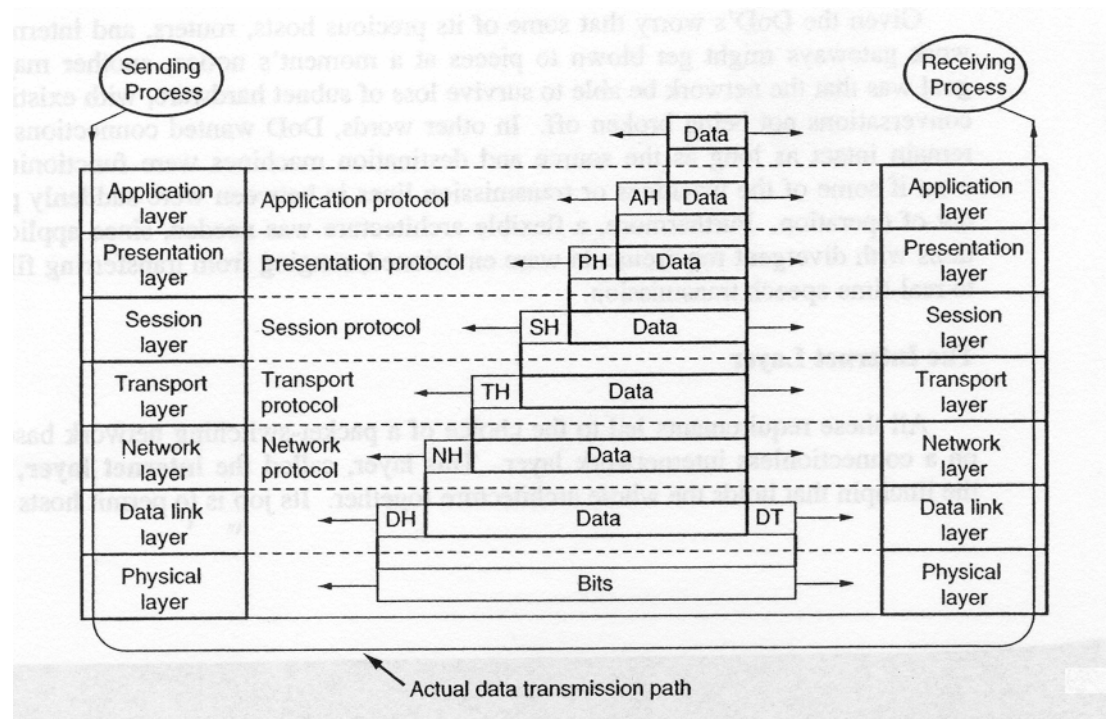


The application layer

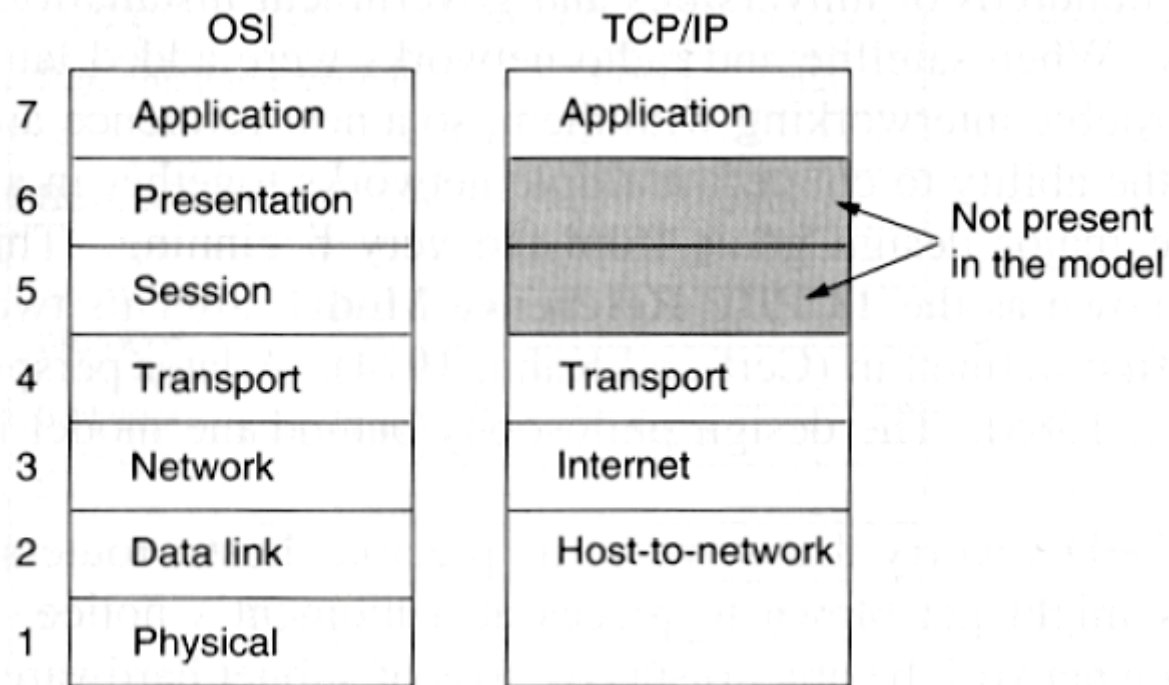
- Function: provide a number of standard protocols
- This may be terminal types: TelNet, ect.
- These are Network Virtual Terminals
- Or this may be file transfer protocols: ftp
- Or this may be handling emails, news, http



Data Transmission in the OSI Model



The TCP/IP reference model



The Internet Layer

- Permits hosts to inject packets into any network and have them travel independently to the destination
- The Internet Layer defines an official packet format and protocol called IP (Internet Protocol)
- Major issues
 - Packet routing
 - Congestion control

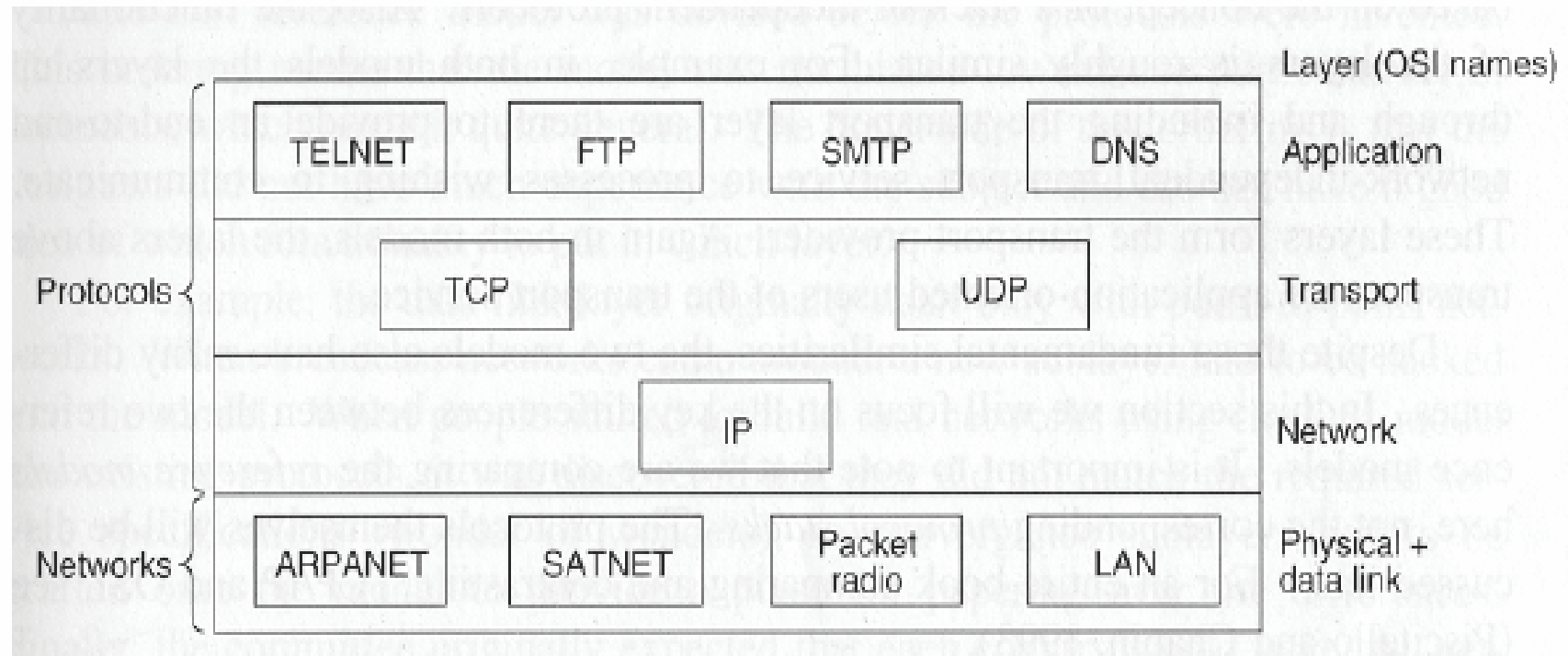


The transport layer

- TCP (Transmission Control Protocol) is a reliable error-free connection-oriented protocol
- UDP (User Datagram Protocol) is an unreliable connectionless protocol



TCP/IP structure



The Internet

- Traditionally four services: Email, News, Remote Login, FTP
- CERN physicist Tim Berners-Lee came up with http
- Together with the viewer Mosaic we had WWW



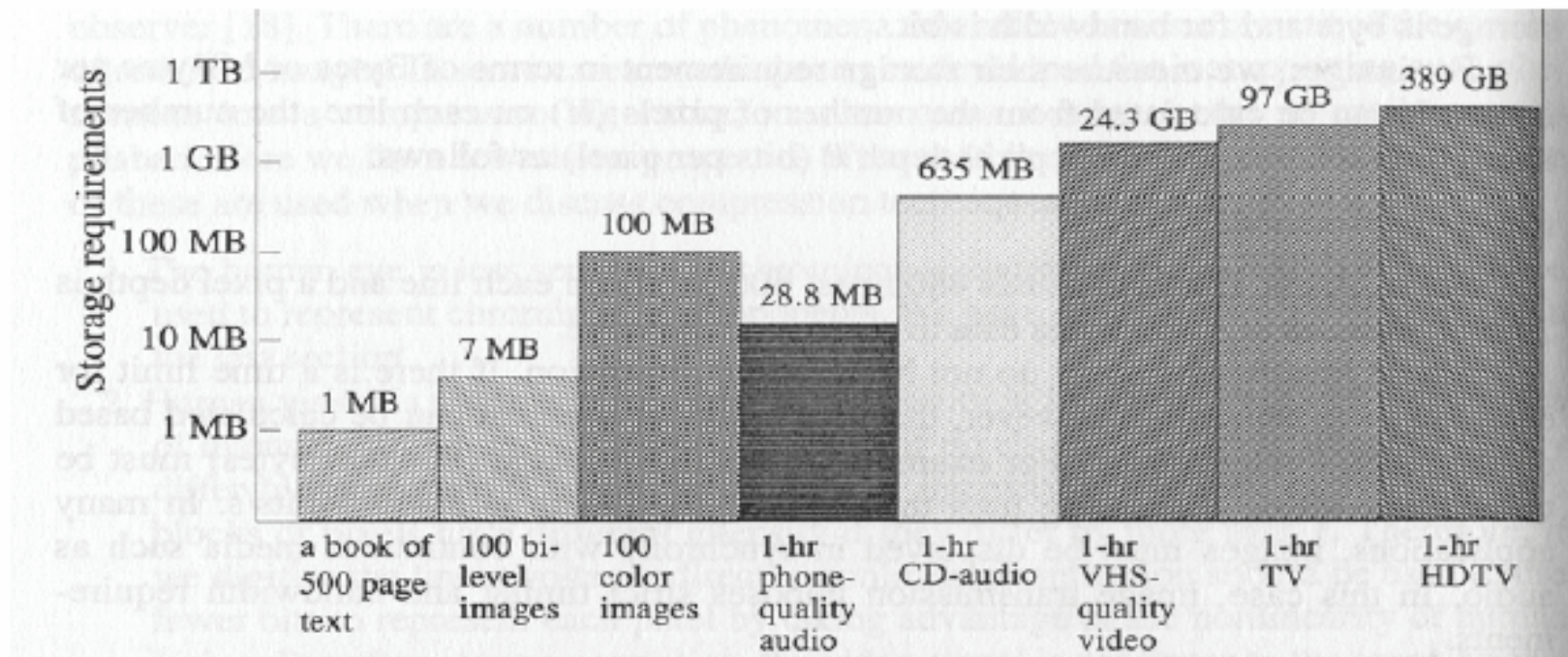
Multimedia requirements

- Quality Of Service (QOS) relates to:
- Bandwidth
- Average network delay
- Maximal network delay

<i>Applications</i>	<i>Data Rate (kbit/s)</i>
CD-Audio	1,411.2
DAT	1,536
Digital telephone	64
Digital radio, long-play DAT	1,024
Television-quality video	216,000
VHS-quality video	54,000
HDTV	864,000



Multimedia file sizes



Java Network programming

- One of Java's strengths is easy networking.
- Underlying details of networking have been abstracted away
- In Java, you create a socket to make a connection to another machine. The socket is wrapped with a stream object. Then you can use the same method calls as with all other streams.
- Networking in Java is quite similar to reading and writing files.



Sockets

- The socket is the software abstraction used to represent the “terminals” of a connection between two machines.
- For a given connection, there’s a socket on each machine (as an abstraction can the connection be viewed as a hypothetical “cable” running between the two machines with each end of the “cable” plugged into a socket). The whole idea of the abstraction is that we don’t have to know more than is necessary.
- From the socket you get an `InputStream` and `OutputStream` (or, with the appropriate converters, `Reader` and `Writer`).
- There are two stream-based socket classes:
 - A `ServerSocket` that a server uses to “listen” for incoming connections (the server socket must be given a port number)
 - A `Socket` that a client uses in order to initiate a connection (the socket must give both a IP address and a port number).
- Once a client makes a socket connection, the `ServerSocket` returns a corresponding `Socket` through which communications will take place on the server side (at this point you have a `Socket` to `Socket` connection). The methods `getInputStream()` and `getOutputStream()` produce the corresponding `InputStream` and `OutputStream` objects from each `Socket`. (These must be wrapped inside buffers and formatting classes just like any other stream object)



The IP Address

- The the IP (Internet Protocol) address is a unique address identifying machines on a network.
- The IP address can exist in two forms:
 - The DNS (Domain Name System) form, such as time-A.timefreq.blrdoc.gov or www.itu.dk
 - The dotted quad" form, which is four unsigned 8-bit numbers separated by dots, such as 132.163.4.104
- The IP address can either be static or dynamic.
 - A static IP address is a permanent address that is assigned to a computer by an Internet service provider. Static addresses is used for web pages, ftp services, ...
 - A dynamic IP address is a temporarily IP address assigned by a Dynamic Host Configuration Protocol ([DHCP](#)) server from a pool of IP addresses. Dynamic addresses is used by most dial-up Internet service provider (ISP). Each time you dial up, you are assigned a temporary IP address.

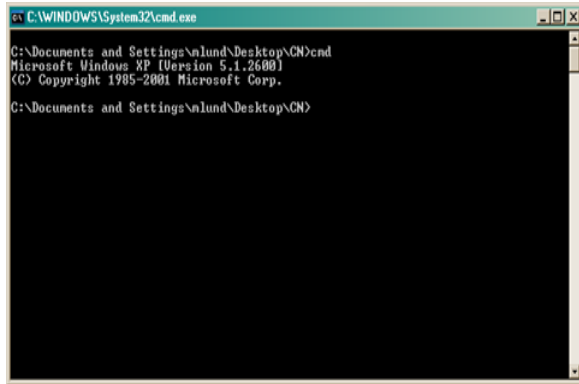


Ports

- An IP address isn't enough to identify a connection (many connections can exist on one machine).
- Each IP machine also contains ports
 - When you're setting up a connection you must choose a port where both client and server agree to connect.
 - The port is not a physical location in a machine, but a software abstraction.
 - The idea is that if you ask for a particular port, you're requesting the service that's associated with the port number. Typically, each service is associated with a unique port number on a given server machine. It's up to the client to know ahead of time which port number the desired service is running on.
- The system services reserve the use of ports 1 through 1024, so you shouldn't use those or any other port that you know to be in use.

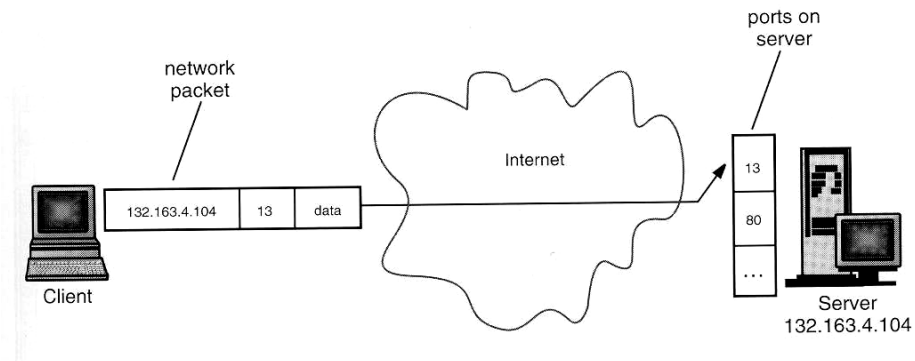


TelNet



Time of day service (National Institute of Standards and Technology in Boulder, Colorado)

```
telnet time-A.timefreq.bldrdoc.gov 13
```



Java SocketTest

```
1 /**
2  * @version 1.10 1997-06-27
3  * @author Cay Horstmann
4  */
5
6 import java.io.*;
7 import java.net.*;
8
9 /**
10  * This program makes a socket connection to the atomic clock
11  * in Boulder, Colorado, and prints the time that the
12  * server sends.
13  */
14 public class SocketTest
15 {
16     public static void main(String[] args)
17     {
18         try
19         {
20             Socket s = new Socket("time-A.timefreq.bldrdoc.gov",
21                                   13);
22
23             BufferedReader in = new BufferedReader
24                 (new InputStreamReader(s.getInputStream()));
25             boolean more = true;
26             while (more)
27             {
28                 String line = in.readLine();
29                 if (line == null)
30                     more = false;
31                 else
32                     System.out.println(line);
33             }
34         }
35         catch (IOException e)
36         {
37             e.printStackTrace();
38         }
39     }
40 }
41
42
```



Connection oriented TCP Based Application

- Client Program
 - Create a socket
 - Connect it to a server on a remote machine
 - Use it to send/receive data to/from remote machine
 - When done close socket
- Server Program
 - Create a socket
 - Bind it to a well known port on local machine
 - Wait for clients



A simple server

```
1 /**
2  * @version 1.10 1997-06-27
3  * @author Cay Horstmann
4  */
5
6 import java.io.*;
7 import java.net.*;
8
9 public class EchoServer
10 { public static void main(String[] args )
11   { try
12     { ServerSocket s = new ServerSocket(8189);
13       Socket incoming = s.accept( );
14       BufferedReader in = new BufferedReader
15         (new InputStreamReader(incoming.getInputStream()));
16       PrintWriter out = new PrintWriter
17         (incoming.getOutputStream(), true /* autoFlush */);
18
19       out.println( "Hello! Enter BYE to exit." );
20
21       boolean done = false;
22       while (!done)
23       { String line = in.readLine();
24         if (line == null) done = true;
25         else
26         { out.println("Echo: " + line);
27
28           if (line.trim().equals("BYE"))
29             done = true;
30         }
31       }
32       incoming.close();
33     }
34     catch (Exception e)
35     { System.out.println(e);
36     }
37   }
38 }
39
40
41
```



EchoServer

TelNet



Threads in the Server

```
13 public class ThreadedEchoServer
14 {
15     public static void main(String[] args )
16     {
17         try
18         {
19             int i = 1;
20             ServerSocket s = new ServerSocket(8189);
21
22             for (;;)
23             {
24                 Socket incoming = s.accept( );
25                 System.out.println("Spawning " + i);
26                 Thread t = new ThreadedEchoHandler(incoming, i);
27                 t.start();
28                 i++;
29             }
30         }
31         catch (Exception e)
32         {
33             e.printStackTrace();
34         }
35     }
36 }
37
38 /**
39  * This class handles the client input for one server socket
40  * connection.
41  */
42 class ThreadedEchoHandler extends Thread
43 {
44     /**
45     * Constructs a handler.
46     * @param i the incoming socket
47     * @param c the counter for the handlers (used in prompts)
48     */
49     public ThreadedEchoHandler(Socket i, int c)
50     {
51         incoming = i; counter = c;
52     }
53
54     public void run()
55     {
56         try
57         {
58             BufferedReader in = new BufferedReader
59             (new InputStreamReader(incoming.getInputStream()));
60             PrintWriter out = new PrintWriter
61             (incoming.getOutputStream(), true /* autoFlush */);
62
63             out.println( "Hello! Enter BYE to exit." );
64
65             boolean done = false;
66             while (!done)
67             {
68                 String str = in.readLine();
69                 if (str == null) done = true;
70                 else
71                 {
72                     out.println("Echo (" + counter + "): " + str);
73                     if (str.trim().equals("BYE"))
74                         done = true;
75                 }
76             }
77             incoming.close();
78         }
79         catch (Exception e)
80         {
81             e.printStackTrace();
82         }
83     }
84
85     private Socket incoming;
86     private int counter;
87 }
88 }
```



Other Java Methods

- Datagram
- Object Streams
- Remote Method Invocation (RMI)

