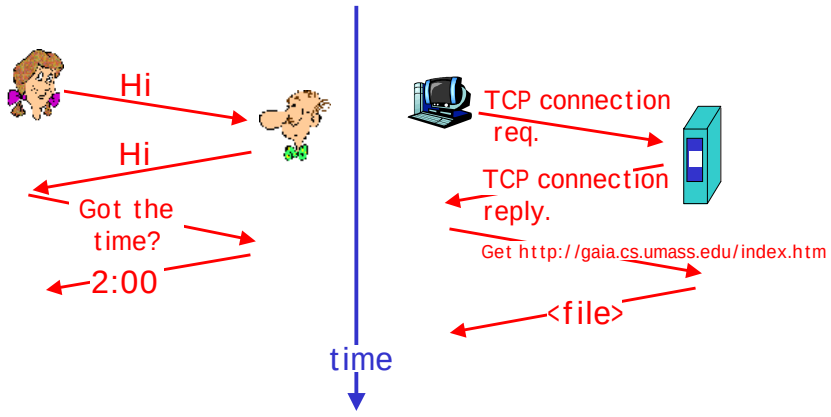
# What is a Network?

- Computer network
  - a set of computers using common protocols to communicate over connecting transmission media.
- Protocol
  - a formal description of message formats and the rules two or more machines follow to exchange messages.



# Protocols



# Classifying Networks

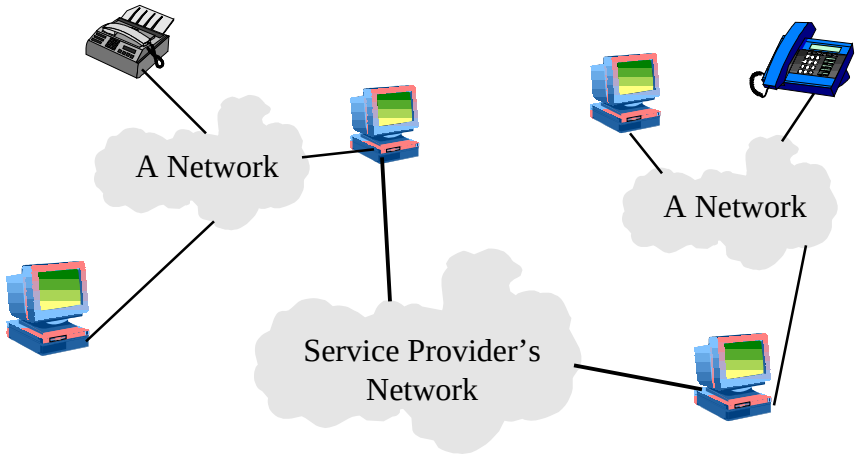
- Networks can be classified by size
  - Local Area Networks (small)
    - privately-owned
    - cover a small area
    - high data rates
  - Wide Area Networks (large)
    - owned/operated by a network provider
    - large capacity
    - often have an irregular topology



# Internetworks

- An internetwork, or *internet*, is formed when two networks are connected together.
- Two networks are joined using a computer that is directly connected to both networks
- A computer that joins two networks is called a gateway

# An internet

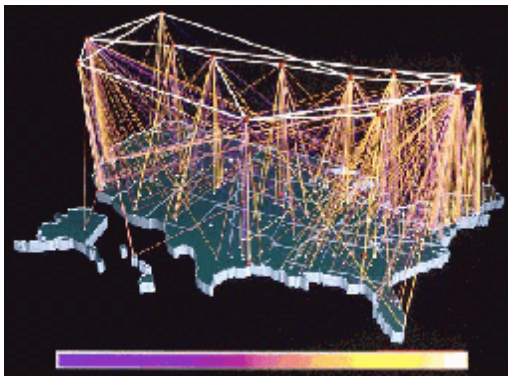





# What Is The Internet?

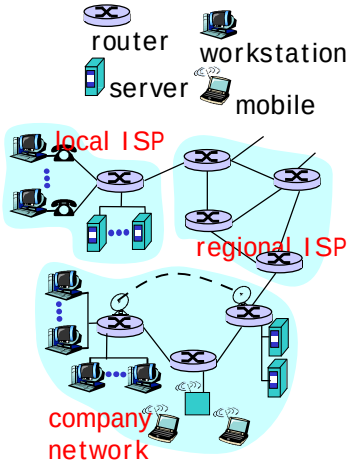
- The Internet
  - *"Internet (noun) - A sprawling collection of computer networks that spans the globe, connecting government, military, educational and commercial institutions, as well as private citizens to a wide range of computer services, resources, and information. A set of network conventions and common tools are employed to give the appearance of a single large network, even though the computers that are linked together use many different hardware and software platforms."*
- An Intranet
  - *"Intranet (noun) - A contained collection of computers and networks within an organization (it may span the globe), connecting the organization's members and/or employees to a range of computer services, resources, and information. A set of network conventions and common tools are employed to give the appearance of a single large network, even though the computers that are linked together use many different hardware and software platforms. It's more than a fancy name for the corporate LAN/WAN"*

# The Internet in the USA





# Nuts and Bolts



## How Did It Get Started?

- The Internet started as the ARPAnet
  - Started in the mid 60s, working in early 70s
  - Designed for the military
  - Could only be used by the military
- Applications of the ARPAnet included
  - Electronic Mail
  - Remote Access
  - File Transfer



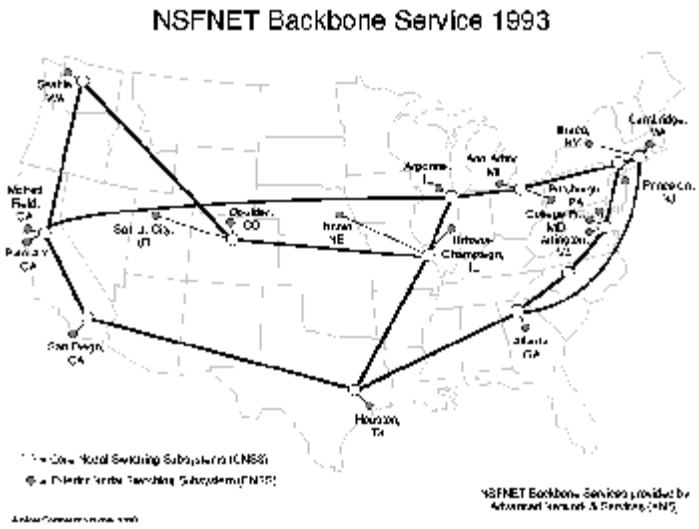
## Consequences

- The ARPAnet provided services to its users and served as a test bed for network research.
- To connect to the ARPAnet an organization had to have a contract with the DoD.
- As a result many small, special interest, networks were created.

# NSFnet

- In the late 80s NSF supported the creation of 5 supercomputer centers.
- NSF Decided to use ARPAnet technology to provide remote access, but could not use the ARPAnet to do this.
- In 1985 NSF announced its decision to build the NSFnet.





## Commercialization

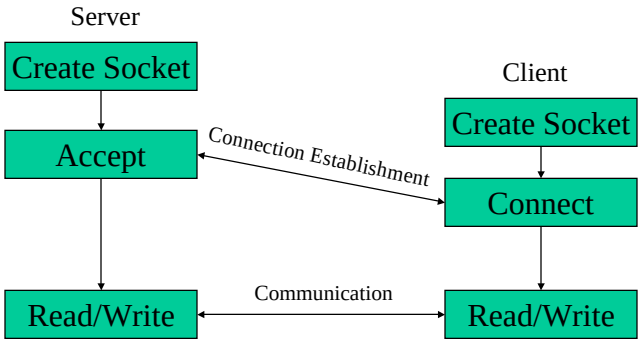
- During NSF's support of the Internet commercial use was forbidden by law.
- On April 30th, 1995 NSF pulled the plug on the NSFnet and turned it over to the private sector.
- Since that time commercial use of the Internet has grown dramatically.



## Types of Transfer

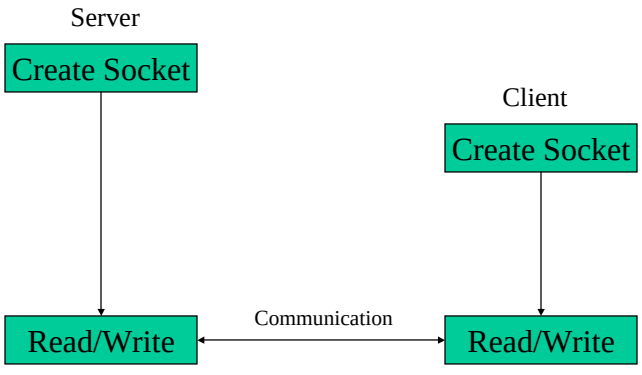
- Networks typically provide two types of transfer
  - Connection-oriented
    - often reliable
    - stream based
  - Connectionless
    - often unreliable
    - datagram based

# Connection-oriented Transfer






# Connectionless Transfer



# The TCP/IP Protocol Suite

- TCP/IP is a set of protocols that were created specifically to allow development of network and internetwork communications on a global scale
- TCP/IP is the most commonly used protocol within the internet.
- TCP/IP is normally considered to be a four-layer system.



# The TCP/IP Protocol Suite

| | |
|---|---|
| Application | Telnet, FTP, e-mail, etc. |
| Transport | TCP, UDP |
| Network | IP |
| Link | device driver and interface card |

# RFCs

- All official standards in the internet community are published as a *Request for Comments*, or *RFC*.
- All RFCs are available at no charge through electronic mail, FTP, or the Web.
- A nice place to get RFCs is at
  - http://www.rfc-editor.org/



# IP: Internet Protocol

- IP is the workhorse protocol of the TCP/IP protocol suite
- IP provides an unreliable, connectionless, datagram delivery service
- RFC791 is the official specification of IP

# Addressing

- A distinction is made between names, addresses, and routes
  - A name indicates what we seek
  - An address indicates where it is
  - A route indicates how to get there
- IP deals primarily with addresses. It is the task of higher level protocols to make the mapping from names to addresses.



# IP Addresses

- Every host on the internet must have a unique *Internet Address* (an IP address)
- IP addresses are 32-bit numbers and are divided into two components: the host address and the network address
  - The number of bits assigned to the host and network varies depending on the class of the address

# Dotted Decimal Notation

- IP addresses are normally written as four numbers, one for each byte of the address.
  - 129.21.38.169
- The easiest way to differentiate between the classes is to look at the first number

| *Class* | *Range* |
|---|---|
| ***A*** | 0.0.0.0 to 127.255.255.255 |
| ***B*** | 128.0.0.0 to 191.255.255.255 |
| ***C*** | 192.0.0.0 to 223.255.255.255 |
| ***D*** | 224.0.0.0 to 239.255.255.255 |
| ***E*** | 240.0.0.0 to 247.255.255.255 |



# Assigning IP Addresses

- Since every interface must have a unique IP address, there must be a central authority for assigning numbers
- That authority is the *Internet Network Information Center*, called the InterNIC.
- The InterNIC assigns only network ids, the assignment of host ids is up to the system administrator

## Transmission Control Protocol

- TCP provides a connection-oriented, reliable, byte stream service (RFC793)
- TCP is an independent, general purpose protocol that can be adapted for use with delivery systems other than IP.



## TCP Streams

- A stream of 8-bit bytes is exchanged across a TCP connection.
- The treatment of the byte stream by TCP is similar to the treatment of a file by the UNIX operating system.
- Connections provided by TCP allow concurrent transfer in both directions. Such connections are called *full duplex*.

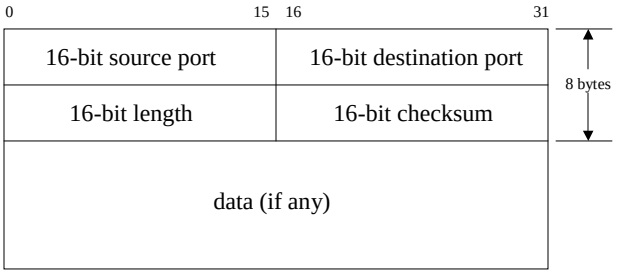
# TCP Ports

- TCP uses protocol port numbers to identify the ultimate destination within a machine.
- How does one determine the port to communicate with?
  - Well-known Ports
  - Randomly Assigned Ports



# User Datagram Protocol

- UDP is a simple, unreliable, datagram-oriented, transport layer protocol (RFC768).



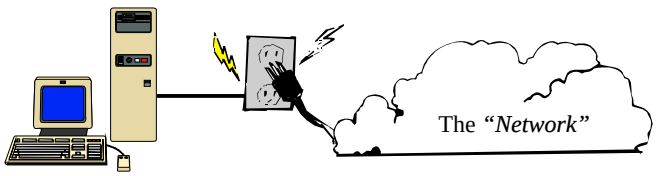
# UNIX Networking

- In the early 80s UNIX was being developed by several organizations
- One of the most influential development groups was UC Berkeley
  - 4BSD provided support for the DARPA Internet networking protocols, TCP/IP
- Some consider 4BSD responsible for the popularity of the TCP/IP protocols



# Sockets

- Berkeley sockets are one of the most widely used communication APIs
- A socket is an object from which messages are sent and received



# java.net

The `java.net` package provides networking support in java.

- Classes
  - DatagramPacket
  - DatagramSocket
  - InetAddress
  - ServerSocket
  - Socket
- Exceptions
  - BindException
  - ConnectException
  - MalformedURLException
  - NoRouteToHostException
  - ProtocolException
  - SocketException
  - UnknownHostException
  - UnknownServiceException



# Class InetAddress

```
public boolean equals(Object obj);

public byte[] getAddress();
public static InetAddress[] getAllByName(String host);
public static InetAddress getByName(String host);
public String getHostName();
public static InetAddress getLocalHost();

public int hashCode();
public String toString();
```

This class represents an Internet Protocol (IP) address.

Applications should use the methods `getLocalHost()`, `getByName()`, or `getAllByName()` to create a new `InetAddress` instance.

# HostInfo.java

```
import java.net.*;
import java.io.*;
import java.util.*;

public class HostInfo {
  public static void main(String argv[]) {
    InetAddress ipAddr;

    try {
      ipAddr = InetAddress.getLocalHost();
      System.out.println("This is "+ipAddr);
    }
    catch (UnknownHostException e) {
      System.out.println("Unknown host");
    }
}
```



# Resolver.java

```
import java.net.*;
import java.io.*;
import java.util.*;

public class Resolver {
  public static void main(String argv[]) {
    InetAddress ipAddr;

    try {
      ipAddr = InetAddress.getByName(argv[0]);
      System.out.print("IP address = "+ipAddr+"\n");
    }
    catch (UnknownHostException e){
      System.out.println("Unknown host");
    }
  }
}
```

# Daytime Service

Most UNIX servers run the daytime service on TCP port 13.

```
cobalt> telnet kiev.cs.rit.edu 13
Trying 129.21.38.145...
Connected to kiev.
Escape character is '^]'.
Fri Feb  6 08:33:44 1998
Connection closed by foreign host.
```

It is easy to write a Java daytime client. All the program needs to do is to establish a TCP connection on port 13 of a remote host.

A TCP style connection is made using the Socket class.



# Class Socket

```
// Constructors (partial list)
public Socket()
public Socket(InetAddress address, int port);
public Socket(String host, int port);

// Methods (partial list)
public void close();

public InetAddress getInetAddress();
public int getLocalPort();

public InputStream getInputStream();
public OutputStream getOutputStream();

public int getPort();
public String toString();
```

# DayTimeClient.java

```
import java.net.*; import java.io.*; import java.util.*;

public class DayTimeClient {
  static int dayTimePort = 13;

  public static void main(String argv[]) {
    try {
      Socket sock = new Socket(argv[0], dayTimePort);
      BufferedReader din = new BufferedReader(
        new InputStreamReader(sock.getInputStream()));
      String rTime = din.readLine();
      System.out.println(rTime);
      sock.close();
    }
    catch (exception e) {}
  }
}
```



# A Java Daytime Server

- It is easy to create a daytime server in Java (the only real problem is that your Java server will not be able to use port 13).
- The server version of the program will use a `ServerSocket` to communicate with a client.
- A `ServerSocket` will open a TCP port and wait for a connection.
- Once a request is detected, a new port will be created, and the connection will be established between the client's source port and this new port.
- Most servers listen for requests on a particular port, and then service that request on a different port.
- This makes it easy for the server to accept and service requests at the same time.

# Class ServerSocket

```
// Constructors (partial list)

public ServerSocket(int port);
public ServerSocket(int port, int count);

// Methods (partial list)

public Socket accept();
public void close();

public InetAddress getInetAddress();
public int getLocalPort();

public String toString();
```



# Class ServerSocket

- A `ServerSocket` waits for requests to come in over the network. It performs some operation based on that request, and then possibly returns a result to the requester.
- The actual work of the `ServerSocket` is performed by an instance of the `SocketImpl` class.
- The abstract class `SocketImpl` is a common superclass of all classes that actually implement sockets. It is used to create both client and server sockets.
- A *plain* socket implements the SocketImpl methods exactly as described, without attempting to go through a firewall or proxy.

# DayTimeServer

```
import java.net.*; import java.io.*; import java.util.*;

public class DayTimeServer {
  public static void main(String argv[]) {
    try {
      ServerSocket listen = new ServerSocket(0);
      System.out.println("Listening on port:  "+listen.getLocalPort());

      for(;;) {
        Socket clnt = listen.accept();
        System.out.println(clnt.toString());
        PrintWriter out = new PrintWriter(clnt.getOutputStream(), true);
        out.println(new Date());
        clnt.close();
      }
    } catch(Exception e) {}}}
```



# DayTimeServer in Action

The output from the daytime server looks like this:

```
kiev> java DayTimeServer
Listening on port:  36109
Socket[addr=cobalt/129.21.37.176,port=32875,localport=36109]
Socket[addr=localhost/127.0.0.1,port=36112,localport=36109]
```

The client output looks like this:

```
cobalt> telnet kiev 36109
Trying 129.21.38.145...
Connected to kiev.
Escape character is '^]'.
Fri Feb 06 09:53:00 EST 1998
Connection closed by foreign host.
```

# Multi-Threaded Servers

- It is quite easy, and natural in Java, to make a server multi-threaded.
- In a multi-threaded server a new thread is created to handle each request.
- Clearly for a server such as the daytime server this is not necessary, but for an FTP server this is almost required.
- The code for the multi-threaded version of the server consists of a new class called Connection.
- An instance of this class handles the clients request.



# Connection.java

```
import java.net.*; import java.io.*; import java.util.*;

class Connection extends Thread {
  protected Socket clnt;
  public Connection(Socket sock) {
    clnt = sock;
    this.start();
  }

  public void run() {
    Date today = new Date();
    try {
      PrintWriter out = new PrintWriter(clnt.getOutputStream(), true);
      out.println(today);
      client.close();
    } catch (IOException e) {} }}
```

# TDayTimeServer.java

```
import java.net.*; import java.io.*; import java.util.*;

public class TDayTimeServer {
  public static void main(String argv[]) {
    try {
      ServerSocket listen = new ServerSocket(0);
      System.out.println("Listening on:  "+listen.getLocalPort());

      for(;;) {
        Socket client = listen.accept();
        System.out.println(client.toString());
        Connection c = new Connection(client);
      }
    }
    catch(Exception e) { System.out.println("Server terminated"); }
  }}
```



# Datagrams

- Datagram packets are used to implement a connectionless, packet based, delivery service.
- Each message is routed from one machine to another based solely on information contained within that packet.
- Multiple packets sent from one machine to another might be routed differently, and might arrive in any order.
- Packets may be lost or duplicated during transit.
- The class `DatagramPacket` represents a datagram in Java.

# Class DatagramPacket

```
//Constructors
public DatagramPacket(byte ibuf[], int ilength);
public DatagramPacket(
  byte ibuf[], int ilength, InetAddress iaddr, int iport);

// Methods
public synchronized InetAddress getAddress();
public synchronized int getPort();
public synchronized byte[] getData();
int getLength();

void setAddress(InetAddress iaddr);
void setPort(int iport);
void setData(byte ibuf[]);
void setLength(int ilength);
```



# Class DatagramSocket

- This class represents a socket for sending and receiving datagram packets.
- Addressing information for outgoing packets is contained in the packet header.
- A socket that is used to read incoming packets must be bound to an address (sockets that are used for sending must be bound as well, but in most cases it is done automatically).
- There is no special datagram server socket class.
- Since packets can be lost, the ability to set timeouts is important.

# Class DatagramSocket

```
// Constructors
DatagramSocket()
DatagramSocket(int port)
DatagramSocket(int port, InetAddress iaddr)

// Methods
void close()
InetAddress getLocalAddress()
int getLocalPort()
int getSoTimeout()
void receive(DatagramPacket p)
void send(DatagramPacket p)
setSoTimeout(int timeout)
```



# Echo Services

- A common network service is an echo server
- An echo server simply sends packets back to the sender
- A client creates a packet, sends it to the server, and waits for a response.
- Echo services can be used to test network connectivity and performance.
- There are typically different levels of echo services. Each provided by a different layer in the protocol stack.

# UDPEchoClient.java

```
import java.net.*; import java.io.*; import java.util.*;

public class UDPEchoClient {
  static int echoPort = 7; static int msgLen = 16; static int timeOut=1000;

  public static void main(String argv[]) {
    try {
      DatagramSocket sock = new DatagramSocket();
      DatagramPacket pak;
      byte msg[] = new byte[msgLen];

      InetAddress echoHost = InetAddress.getByName(argv[0]);
      pak = new DatagramPacket(msg,msgLen,echoHost,echoPort);

      sock.send(pak);
      sock.setSoTimeout(timeOut);
      sock.receive(pak);
    }
    catch (InterruptedIOException e) {System.out.println("Timeout");}
    catch (Exception e) {}}}
```



# UDPEchoServer.java

```
import java.net.*;import java.io.*;import java.util.*;

public class UdpEchoServer {
  static int echoPort = 7000; static int msgLen = 1024;

  public static void main(String args[]) {
    try {
      DatagramSocket sock = new DatagramSocket(echoPort);
      DatagramPacket p,reply;
      byte msg[] = new byte[msgLen];
      pak = new DatagramPacket(msg,msgLen);

      for (;;) {
        sock.receive(p);
        System.out.println(p.getAddress());
        reply =
          new DatagramPacket(p.getData(),p.getLength(),p.getAddress(),p.getPort());
        sock.send(reply);
      }
    } catch (Exception e) {} }}
```