Lecture 11 - Networking
IAT267 Introduction to Technological Systems

Fall 2011

Organizational items

- Assignment 3 – will be posted by the end of this week, due in week 13

- Quiz this week – optional
  - Will replace one of your lowest marks in the other quizzes
  - Will contain questions from all course content

Final exam – December 16, 2011, 8:30 – 11:30

- Closed book exam

- Questions from the entire course content
  - Lectures
  - Workshops
  - Assignments

- You will not be given to write code from scratch, or draw circuits – however you will be asked to explain or modify the functionality of a given circuit or code

- No multiple choice questions
- The exam has three parts, corresponding to three parts of the course
  - 1. Technological systems – computer systems
  - 2. Sensors, Arduino, Processing
  - 3. Networking
- Help available in office hours in the week before the exam - TBA

Final exam questions

- Will ask you to explain /describe various topics from the course
  - For example:
    - explain the piezoelectric effect and how it is used in sensors
    - Describe the von Neumann computer architecture (each component)
    - Describe the characteristics of the UDP protocol

- Explain and/or modify given code – Processing, Arduino

- For a given circuit, explain the functionality, calculate some values, and maybe modify the circuit
  - For example:
    - circuit containing a voltage divider – calculate voltage
    - Series/parallel circuits – calculate equivalent resistance
    - Circuits containing sensors – modify the circuit

Project – Milestone 4

- Posted on webct
  - In-class project presentation and demo, each team
  - Final project report and code

Networking

- Network organization
- Network identification
- Protocols in networking

Network organization

- Examples of networks:
  - Two computers linked together
  - Computers attached to the internet
  - Machines connected to a wireless router

- Common characteristics of all networks:
  - Machines on the the network need to be **identifiable** to each other and themselves
  - Machines need to know **how to connect** with one another
  - Machines need to know **what protocol** the other machines are using

Network identification

- Over a network, all the machines are identified by their **Internet Protocol (IP)** addresses.

- Mac OS X: Preferences / Network
- Windows: Control Panel / Properties / Manage Network Settings

- Sample IP address: 192.168.0.25

Addressing

- In order for an application on one machine to send a message to an application on another machine, the sending application must identify the receiving application.

- To identify the receiving application, one must typically specify two pieces of information:
  - The name or address of the host machine
  - The identity of the receiving process on the destination host
    - Receiving process = application

- The destination host's address is specified by its IP address and the application is specified by the port number.

Ports

- A computer has a single physical connection to the network. All data destined for a particular computer arrives through that connection.
- The data may be intended for different applications running on the computer. The computer knows to which application to forward the data through the use of ports.
- The computer is identified by its 32-bit IP address, which IP uses to deliver data to the right computer on the network. Ports are identified by a 16-bit number.

Analogy

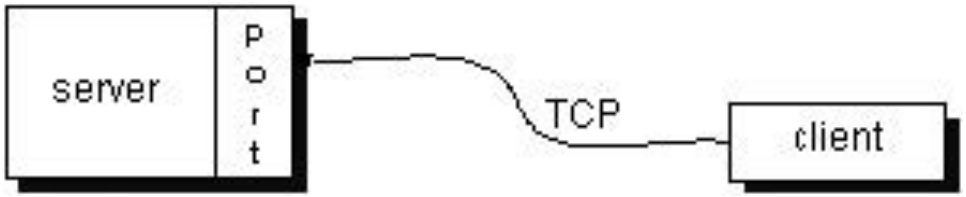
- IP addresses = street address of an apartment building
- Port number = the number of a particular apartment within that building.
- If a letter (a data packet) is sent to the apartment (IP) without an apartment number (port number) on it, then nobody knows who it is for (which service it is for).
- In order for the delivery to work, the sender needs to include an apartment number along with the address to ensure the letter gets to the right domicile.

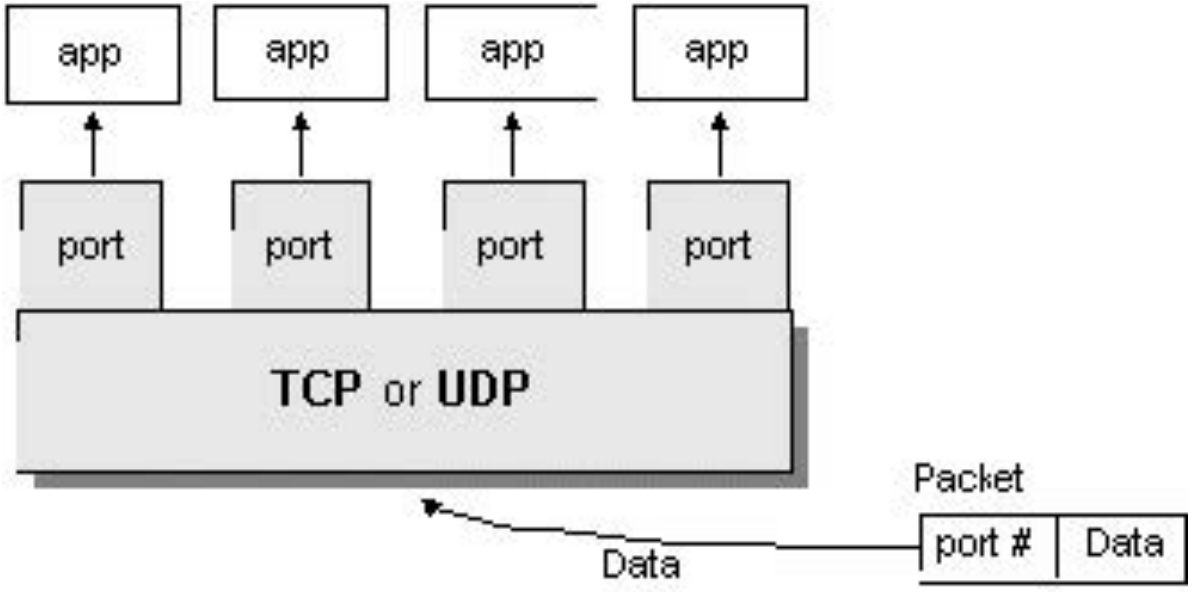
Ports

In computer hardware, a 'port' serves as an interface between the computer and other computers or devices. Physically, a port is a specialized outlet on a piece of equipment to which a plug or cable connects.

A **software port** (usually just called a 'port') is a virtual data connection that can be used by programs to exchange data directly, instead of going through a file or other temporary storage location. The most common of these are TCP and UDP ports which are used to exchange data between computers on the Internet.

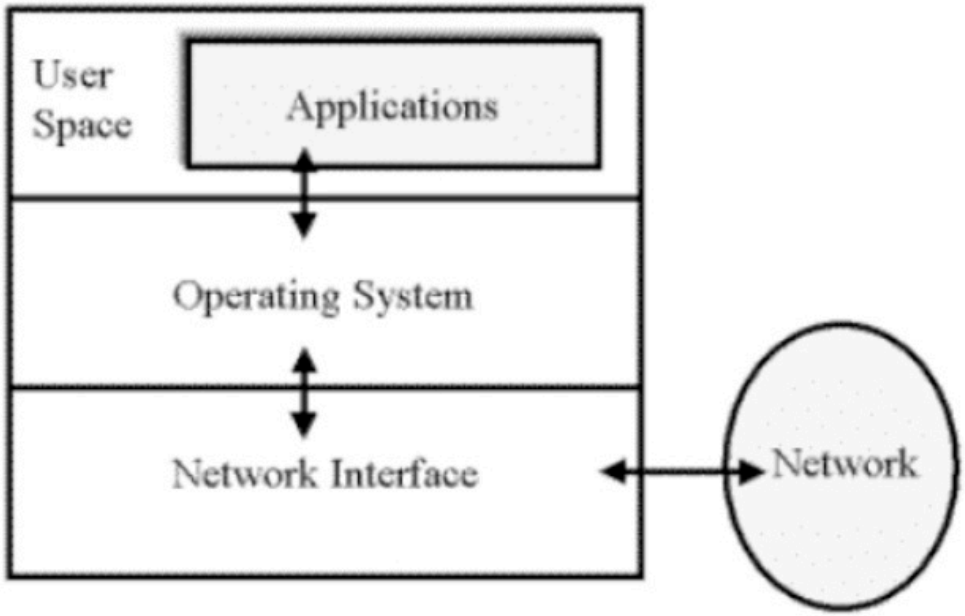
Ports


server
port
TCP
client

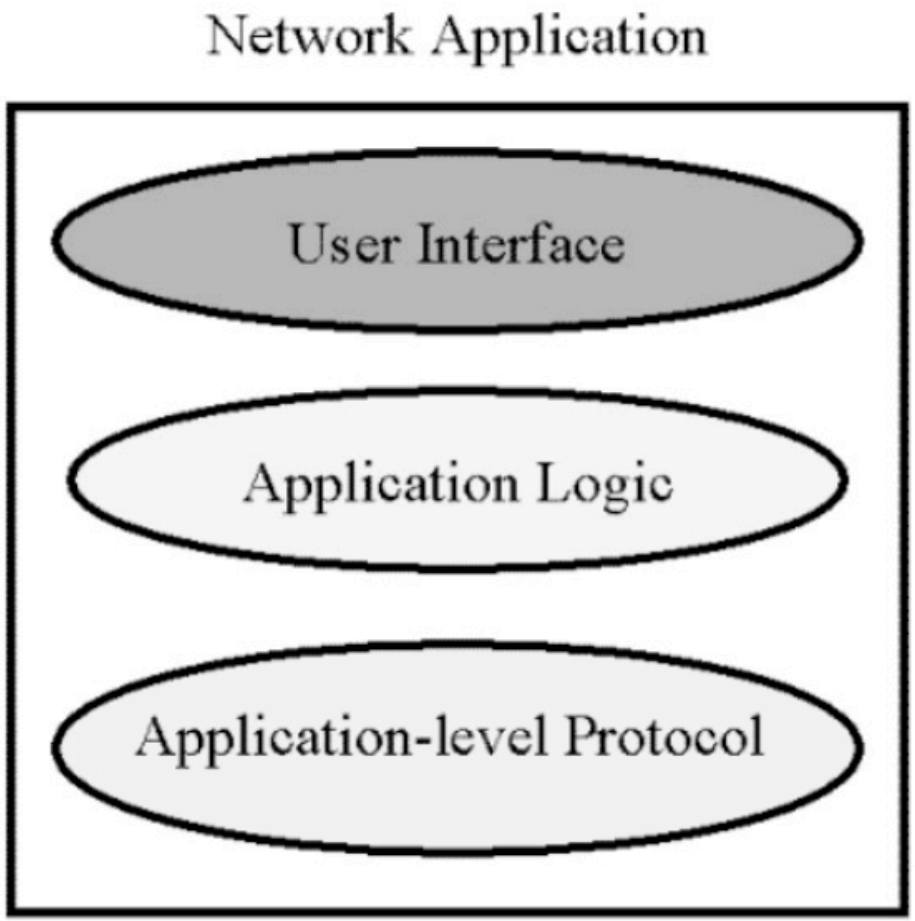

app
app
app
app
port
port
port
port
TCP or UDP
Packet
port #
Data
Data

Applications

- Computer programs running on network attached computers



Structure of a network application



Structure of a network application

- **User interface**: allows the user to invoke and control application functionality
- **Application logic**: the software code, processing instructions that provide functionality
- **Application-level protocol**: define the format and order of the messages exchanged between processes, as well as the actions taken on the transmission or receipt of a message

User interface

- The user interface is an interface between the user and the network application

- Examples:
  - Web browser's interface
  - Electronic mail application (the user interface is a 'mail reader')

Application-Level Protocols

- Define:
  - The types of messages exchanged: e.g., request and response messages
  - The syntax of various message types, that is, the fields in the message and how the fields are delineated
  - The semantics of the fields – the meaning of the information in the fields
  - Rules for determining when and how a process sends messages and responds to messages

Communication mode

- Two network processes can communicate as:
  - client/server
  - peer-to-peer

Client-server communication

- Communication is asymmetrical, the client program is different from the server program
- The server provides a service to the client
- It is usually the client who initiates the communication or transaction
- A server can be communicating to multiple clients at the same time

Peer-to-peer communications

- Communication is symmetrical, and the two communication processes are based on the same application program
- Either end can initiate communication or transaction
- Examples: some chat programs
- One peer can be communicating to multiple peers at the same time

Network Application Programming Interface (API)

- In order for two applications to communicate across the network, the operating system provides an application programming interface (API) to allow applications to invoke network services.

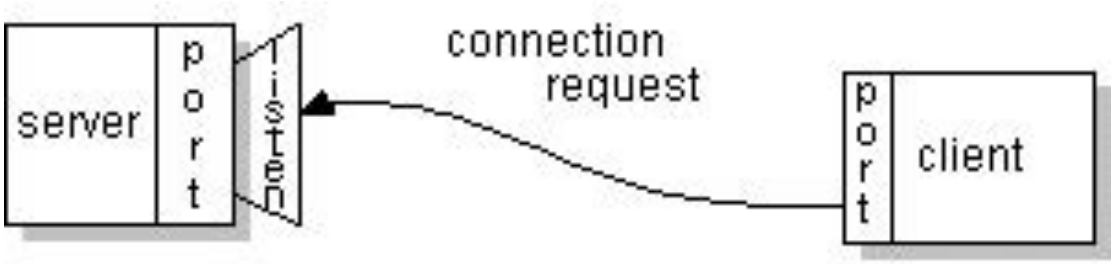
- In modern operating systems, the network API is known as the “Socket API”.

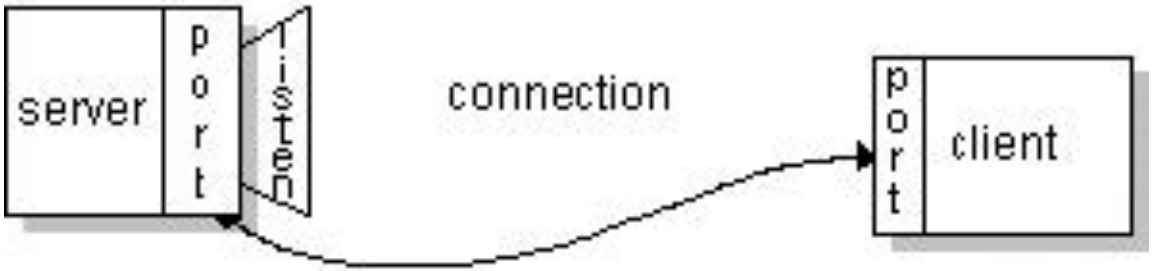
- In essence, the two processes communicate with each other by sending and receiving messages through their sockets.

Socket

- A socket is one end-point of a two-way communication link between two programs running on the network.

- An endpoint is a combination of an IP address and a port number.


server
port
listen
connection
request
port
client


server
port
listen
connection
port
client

What data is being sent over the Internet?

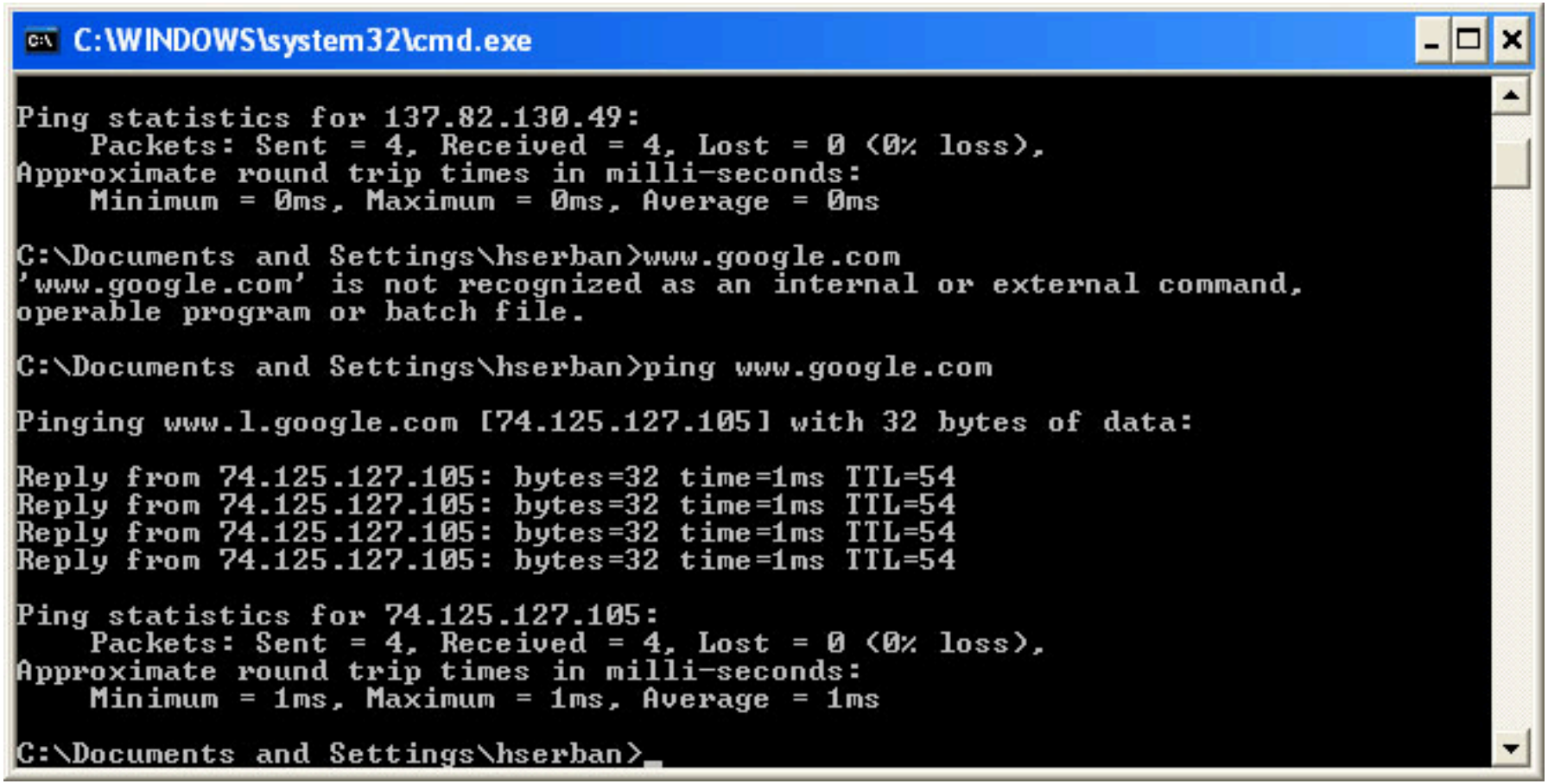
- **Packets**

- Packet = the data, broken into smaller pieces, for easier sending

- A way of seeing this at work is through the **ping** command

ping command

- The name "ping" is taken from the sonar operation to locate undersea objects.
- The ping program tests whether an Internet host is reachable.
- This simple utility also measures the round-trip time to the host, thus providing some indication of how "far away" the remote host is.


C:\WINDOWS\system32\cmd.exe
Ping statistics for 137.82.130.49:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 0ms, Maximum = 0ms, Average = 0ms
C:\Documents and Settings\hserban>www.google.com
'www.google.com' is not recognized as an internal or external command,
operable program or batch file.
C:\Documents and Settings\hserban>ping www.google.com
Pinging www.l.google.com [74.125.127.105] with 32 bytes of data:
Reply from 74.125.127.105: bytes=32 time=1ms TTL=54
Reply from 74.125.127.105: bytes=32 time=1ms TTL=54
Reply from 74.125.127.105: bytes=32 time=1ms TTL=54
Reply from 74.125.127.105: bytes=32 time=1ms TTL=54
Ping statistics for 74.125.127.105:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 1ms, Maximum = 1ms, Average = 1ms
C:\Documents and Settings\hserban>

Network data flow

- How does a network request work if you want to get an HTML file, for example?
  - Your application is running on your local computer, connected to the Internet through a wireless router
  - 1. app sends its data to the router
  - 2. router sends message to the network provider
  - 3. message is routed to the correct network provider for the server you want to make a request from
  - 4. The server will receive the request for the file

Handling network communication in Processing

- Communicating over a network is a very important capability of Processing

- The 'Network' library
  - Your application can communicate with other Processing applications
  - Download data from the internet
  - Send data to be stored or processed remotely

Processing: the 'Network' library

- Two classes:
  - the Client class
  - the Server class

Network

The Network library makes it easy to read and write data across machines on the Internet. It allows the creation clients and servers. A server connects to a list of clients for reading and writing data. A client is able to read and write data to a server.

Server Class

The server class is used to create server objects which send and receives data to and from its associated clients (other programs connected to it).

Server
write()
available()
stop()
disconnect()

Client Class

The client class is used to create client Objects which connect to a server to exchange data.

Client

Client class

- Is used to create client objects that connect to a server to exchange data
  - Server name
  - Port number

```
import processing.net.*
Client networkClient = new Client(this, "www.oreilly.com", 80);
    // Connect to server on port 80
```

Characteristics of a Client

- Sends the request
- Initiates requests
- Waits for and receives replies
- Usually connects to a small number of servers at one time
- Typically interacts directly with end-users using a graphical user interface

Server Class

- Is used to create server objects that can send and receive data to and from any client connected to it.
- You create a new server object by passing the port number that the server should be available on:

```
Server srv = new Server(this, 5204);
```

Characteristics of a Server

- Receiver of request which is send by client is known as server
- Passive
- Waits for requests from clients
- Upon receipt of requests, processes them and then serves replies
- Usually accepts connections from a large number of clients
- Typically does not interact directly with end-users

Protocols in Networking

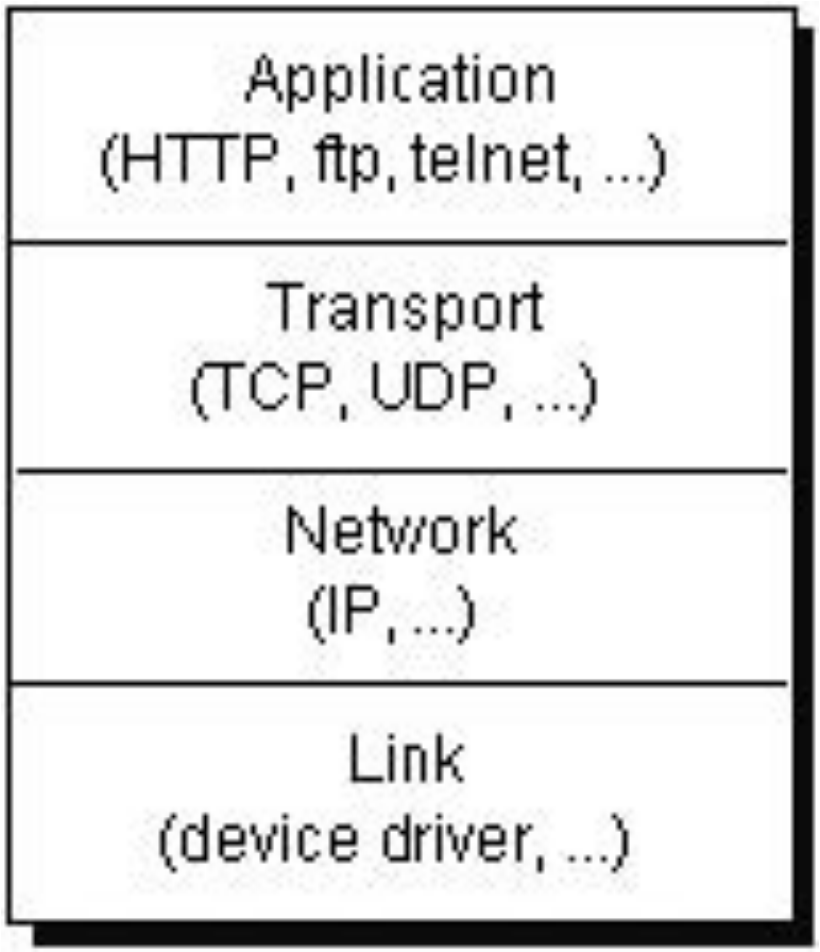
- Internet communication: a machine makes a request to a server for a particular file and the server responds with that file
- That request uses a certain protocol to tell the server what file it wants, where the request is coming from and how the file should be returned
- The response uses a similar protocol
- Hypertext Transmission Protocol

HTTP

What is HTTP?

HTTP: Short for ***H**yper**T**ext **T**ransfer **P**rotocol*, the underlying protocol used by the World Wide Web. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page.

Network Layering



Protocols: TCP and UDP

- When you write Java programs that communicate over the network, you are programming at the application layer.
- Typically, you don't need to concern yourself with the TCP and UDP layers.
- Instead, you can use the classes in the java.net package. These classes provide system independent network communication. However, to decide which Java classes your programs should use, you do need to understand how TCP and UDP differ.

Protocol: TCP

- TCP is a protocol used by HTTP (HyperText Transmission Protocol)

- TCP = Transmission Control Protocol
  - Provides transmission control
  - Presents the data in order
  - Provides error correction when packets get out of order

TCP

- Advantage: it is away to connect a client to a server that ensures that all the data is in correct order and notifies the client is something is broken in the message
- Downside of TCP: it can be slow because of error checking

There also is UDP

UDP (User Datagram Protocol)

- Much faster than TCP
- Has no concept of error checking
- Its messages are not ordered

TCP and UDP

- UDP = User Datagram Protocol
- TCP = Transmission Control Protocol
- When a developer creates a new application for the Internet, one of the first decisions that the developer must make is whether to use UDP or TCP
- Each of these protocols offers a different service model to the applications.

UDP Services

- **Lightweight** transport protocol with a minimalist service model
- **Connectionless**, no handshaking before the two processes start to communicate
- UDP provides an **unreliable** data transfer service. When a process sends a message into a UDP socket, UDP provides no guarantee that the message will ever reach the receiving socket
- Messages that do arrive to the receiving socket may arrive out of order.

UDP Services

- UDP does not include a flow control or congestion control mechanism, so a sending process can pump data into a UDP socket at any rate it pleases. Although all the data may not make it to the receiving socket, a large fraction of data may arrive.

- Because UDP does not use acknowledgments or retransmissions that can slow down the delivery of useful real-time data, developers of real-time applications often choose to run their applications over UDP.

- UDP provides no guarantee on delay.

TCP Services

- The TCP service model includes a **connection-oriented service** and a **reliable** data transfer service. When an application invokes TCP for its transport protocol, the application receives both of these services from TCP.

- Connection-oriented service: TCP has the client and server exchange control information with each other before the application-level messages begin to flow. This so-called handshaking procedure (part of the TCP protocol) alerts the client and server, allowing them to prepare for a transfer of packets.

TCP Services

- After the handshaking phase, a TCP connection is said to exist between the sockets of the two processes. The connection is a full-duplex connection, in that the two processes can send messages to each other over the connection at the same time.

- When the application is finished sending messages, it must tear down the connection. The service is referred to as "connection-oriented" (or a "virtual circuit" service), because the two processes are connected end-to-end in a very loose manner without any support from the intermediate nodes.

TCP services

- Reliable transport service: The communicating processes can rely on TCP to deliver all the messages sent without error and in the proper order. When one side of the application passes a stream of bytes into a socket, it can count on TCP to deliver the same stream of data to the receiving socket, with no missing or duplicate bytes.

- TCP includes an end-to-end flow control mechanism, which regulates sender transmission based on the availability of receiving buffer.

- TCP also includes a congestion control mechanism.

Network programming

- An application can open a UDP socket or a TCP socket.
- The TCP socket gives transport-level connection-oriented reliable byte-stream service to the application.
- On the other hand, the UDP socket provides transport-level connectionless unreliable datagram service to the application.

- Next week: Network programming

Resources

- Content and code on the slides based on:

Programming Interactivity/ Joshua Noble / O'reilly/ Cambridge, Massachusetts, London, England

Resources

- For the code and examples on the slides:
  - **Processing: a programming handbook for visual designers and artists**/ Casey Reas, Ben Fry/ The MIT Press/ Cambridge, Massachusetts, London, England (this book is available in the Library)
  - **Processing - Creative Coding and Computational Art** / Ira Greenberg
  - Code examples: www.processing.org

Thank you

Questions?