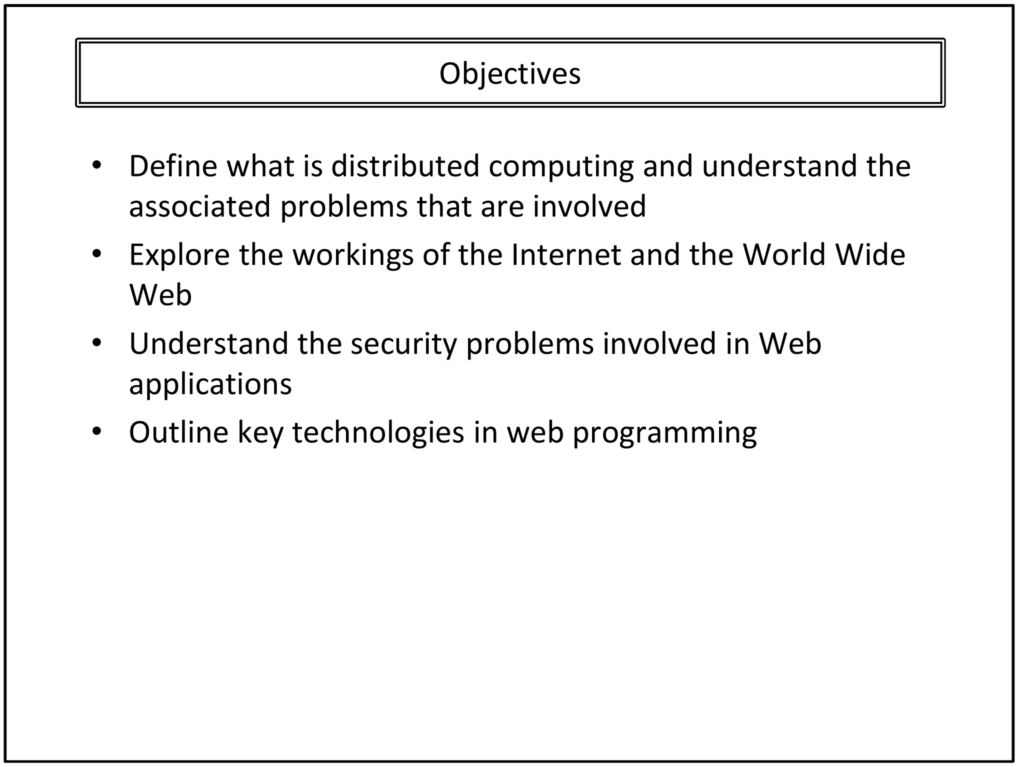
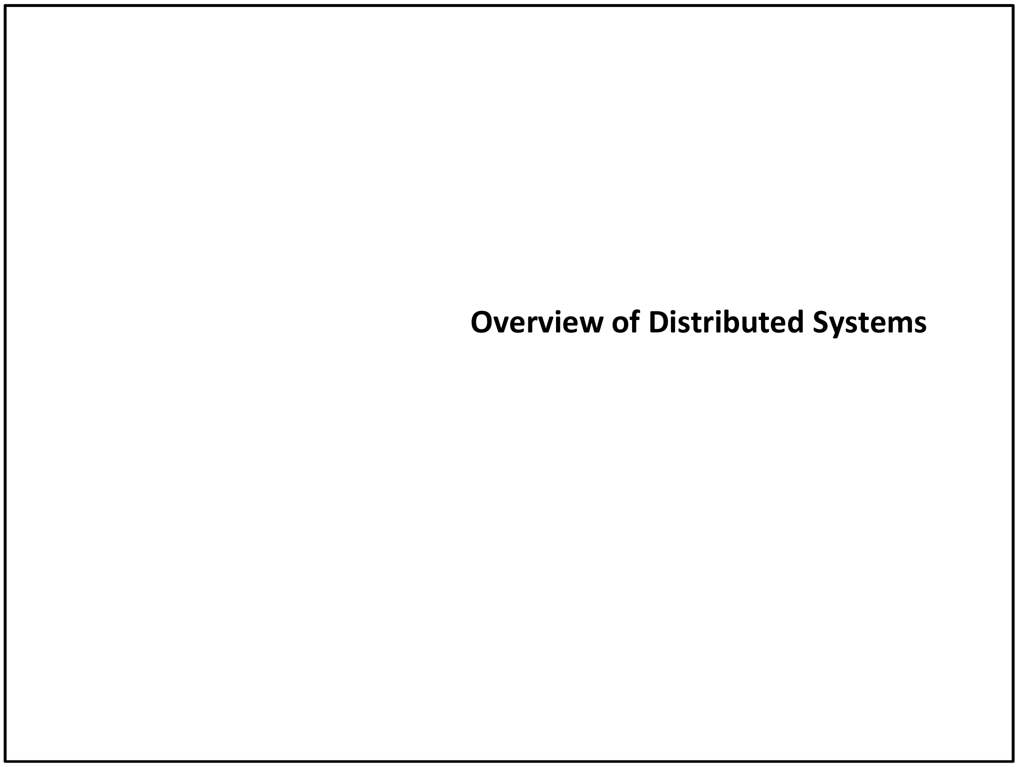
Introduction to Web and Distributed Programming



Objectives

- Define what is distributed computing and understand the associated problems that are involved
- Explore the workings of the Internet and the World Wide Web
- Understand the security problems involved in Web applications
- Outline key technologies in web programming



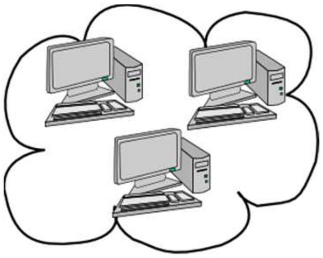
Overview of Distributed Systems

Why Distributed/Web Computing?

World is filled with computing machines

Why not choose a single machine?

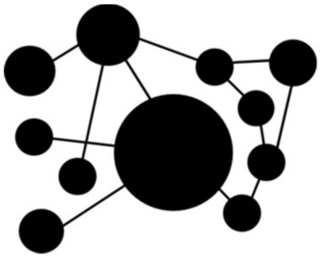
- **Data separation**: need to communicate
- **Performance/Scaling**: single machine not fast enough
- **Cost-effectiveness**: increasing processing power of single machine is hard and expensive
- **Reliability**: single machine means single point of failure

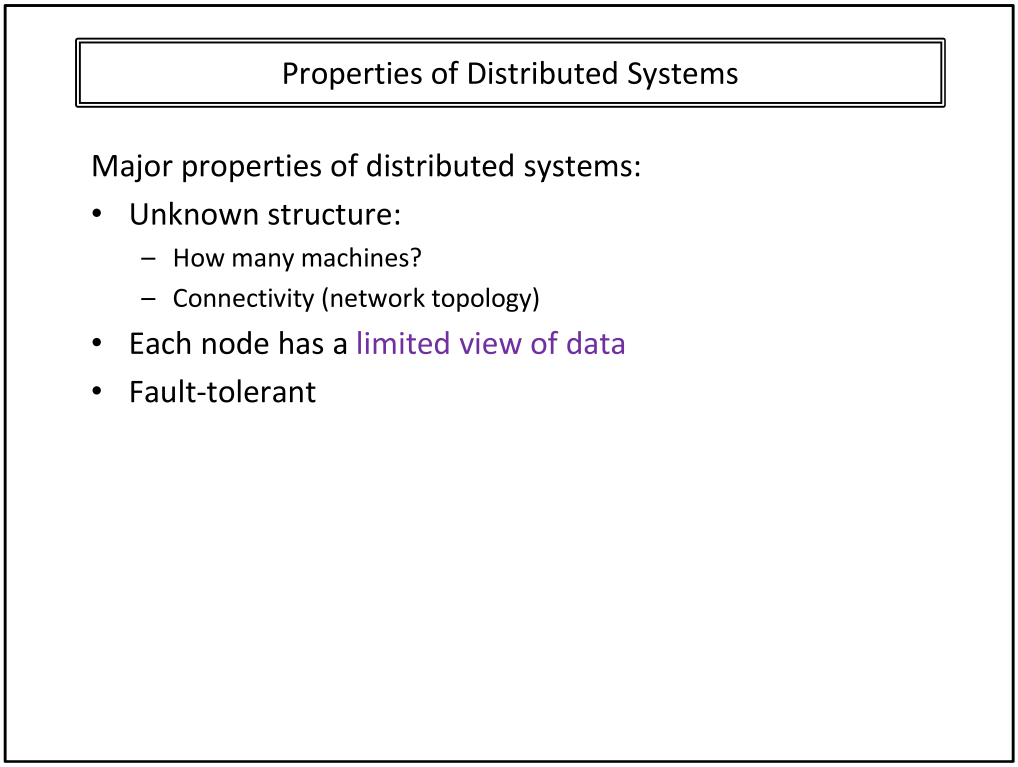


Distributed System Definition

A ***distributed computing system*** is an

- Autonomous set of nodes
- Linked together by a computer network
- Communicating by passing messages

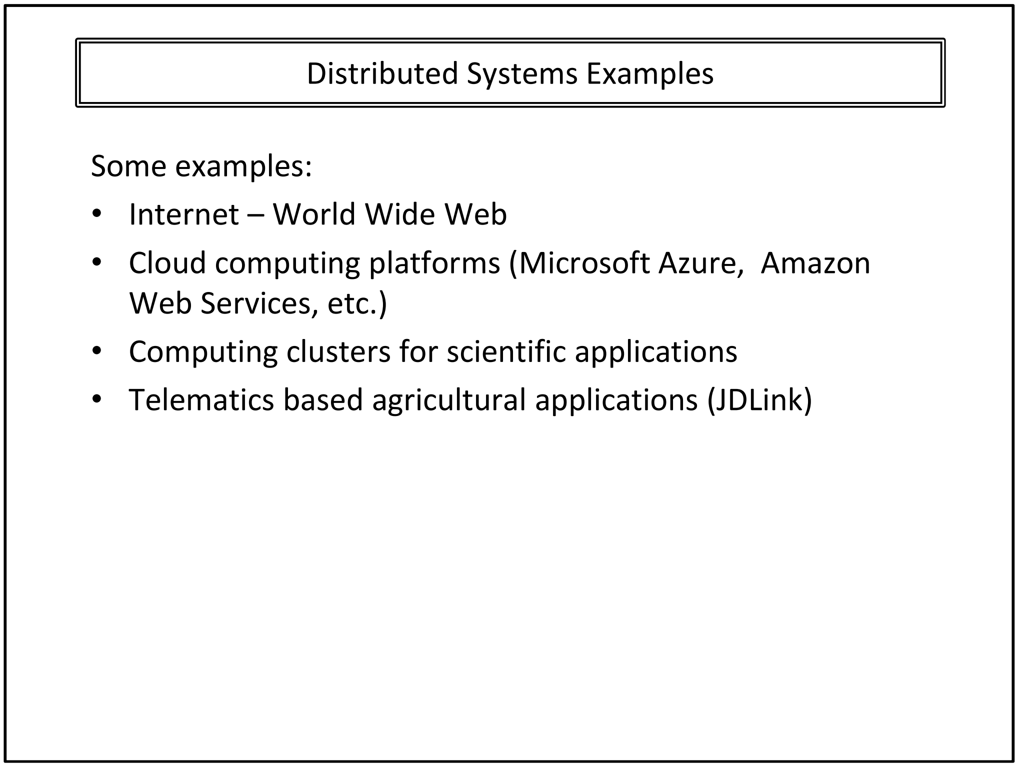




Properties of Distributed Systems

Major properties of distributed systems:

- Unknown structure:
  - How many machines?
  - Connectivity (network topology)
- Each node has a limited view of data
- Fault-tolerant



Distributed Systems Examples

Some examples:

- Internet – World Wide Web
- Cloud computing platforms (Microsoft Azure, Amazon Web Services, etc.)
- Computing clusters for scientific applications
- Telematics based agricultural applications (JDLink)

Issues in Distributed Systems

Distributed computing introduces additional complications/issues:

Heterogeneity

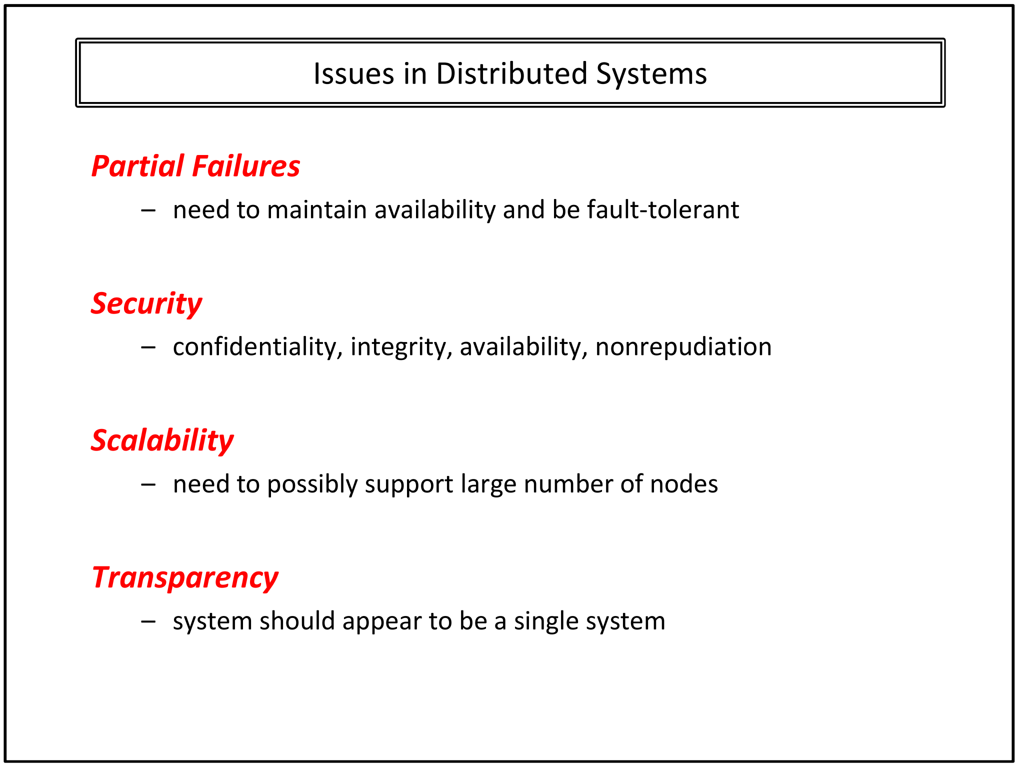
- different types of hardware, software, protocols

Openness

- every service should be equally accessible to every client

Communication

- latency
- synchronization



Issues in Distributed Systems

Partial Failures

- need to maintain availability and be fault-tolerant

Security

- confidentiality, integrity, availability, nonrepudiation

Scalability

- need to possibly support large number of nodes

Transparency

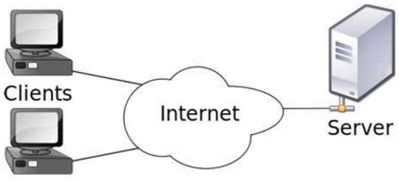
- system should appear to be a single system

Also Legalities, Privacy, and Data Ownership can be a challenge

Communication Architectures

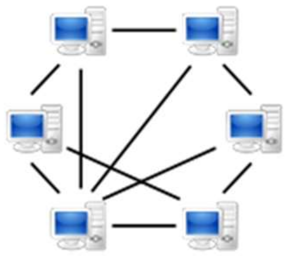
Client / Server

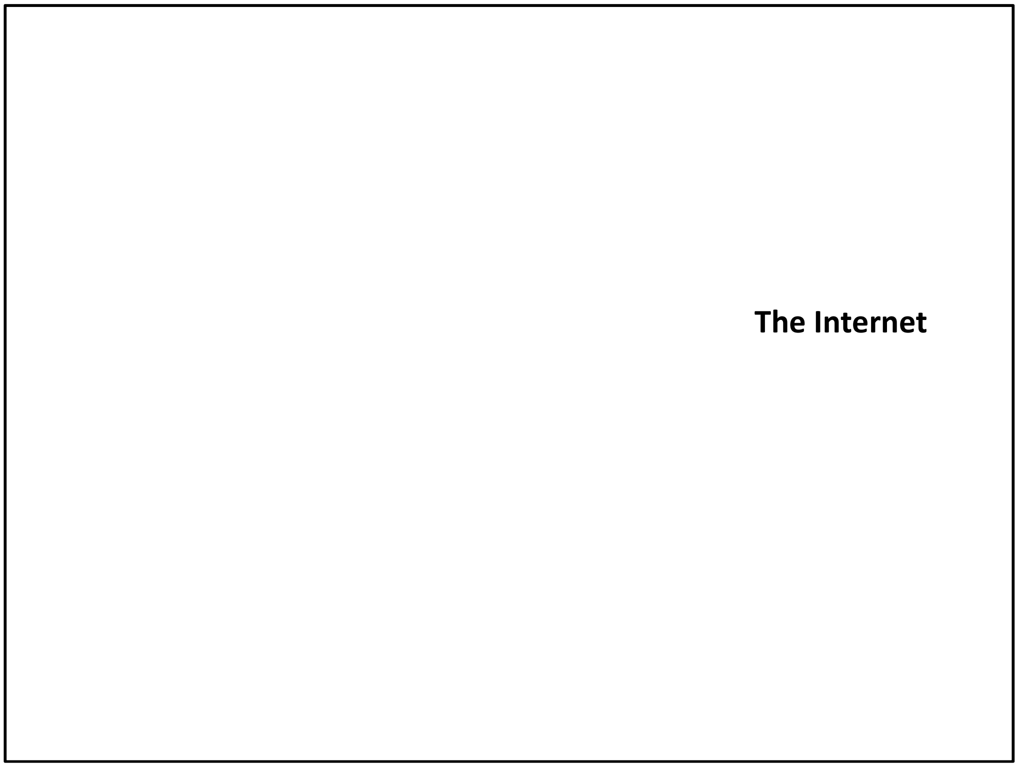
- One machine (the ***server***) is a distinguished node
- All other machines (the ***clients***) communicate with the server



Peer-to-Peer

- All machines can act as both client and server





The Internet

The Internet (History)

The Internet has roots in a ***Department of Defense*** project

- Needed a ***robust*** network that will withstand attacks
- Led to creation of ARPAnet (1960s)
- First node: UCLA – 1969

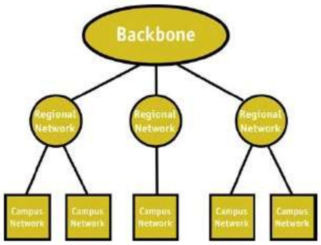
Led to other projects:

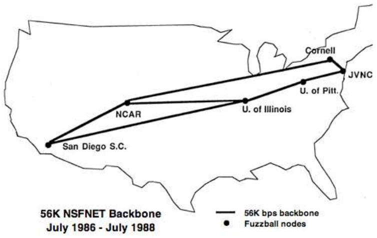
- BITnet, Csnet (1970s, 1980s): email, file transfer

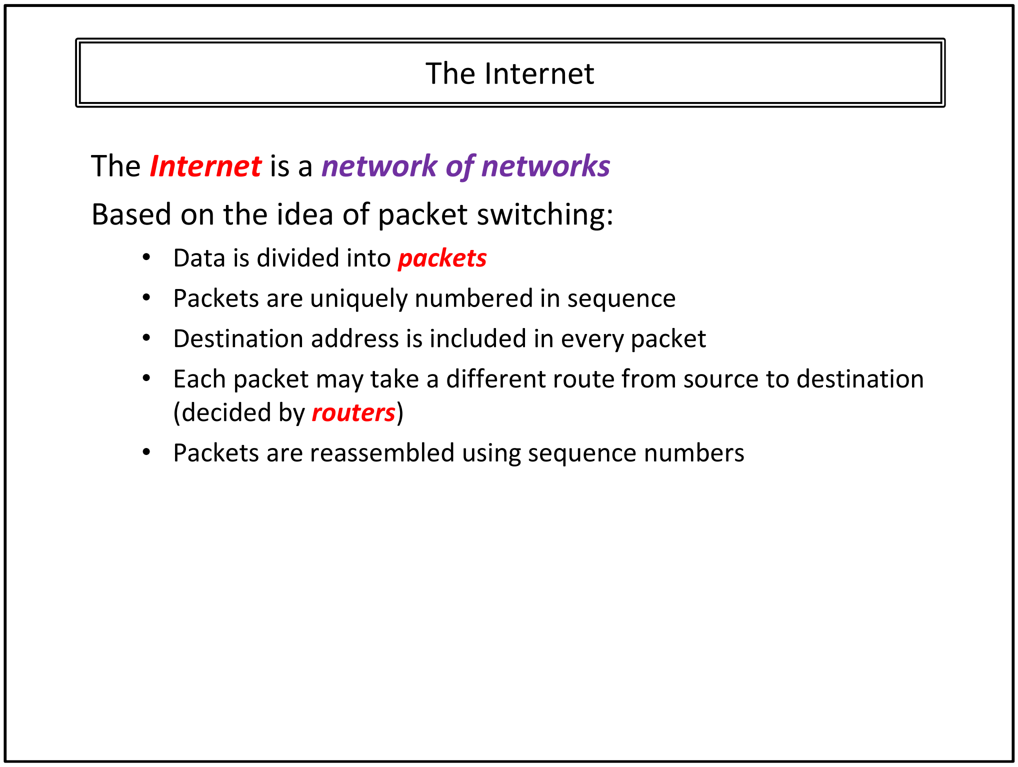
The Internet (History cont.)

Later, led to **NSFnet** (1986)

- Originally connected 5 supercomputer centers
- Used a three-tiered network architecture
- 1990 – replaced ARPAnet for non-military uses
- Became available for all to use
- Eventually became the known as the Internet





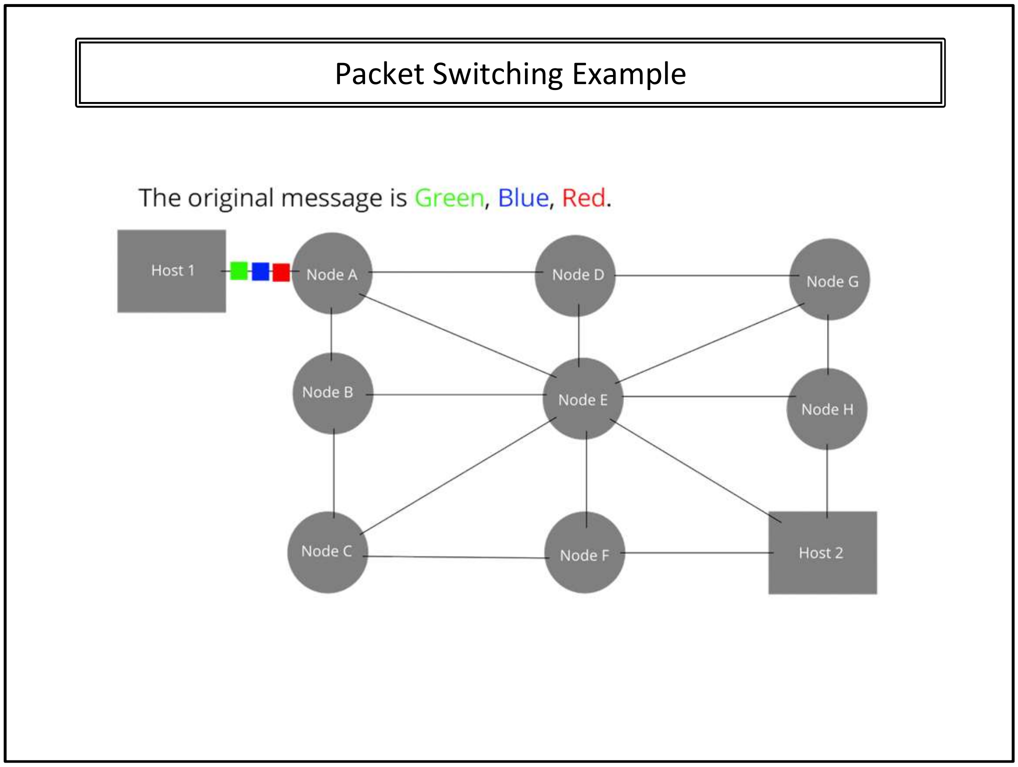


The Internet

The ***Internet*** is a ***network of networks***

Based on the idea of packet switching:

- Data is divided into ***packets***
- Packets are uniquely numbered in sequence
- Destination address is included in every packet
- Each packet may take a different route from source to destination (decided by ***routers***)
- Packets are reassembled using sequence numbers



Packet Switching Example

The original message is Green, Blue, Red.

Packet Switching

Advantages of packet switching:

- No centralized control (robust): can reroute around failed nodes
- Efficient: packets can be routed along most efficient path
- Lower cost compared to dedicated lines (circuits)

Internet Protocols

Mechanism for implementing the Internet:

TCP/IP protocol(s)

Protocol: set of rules

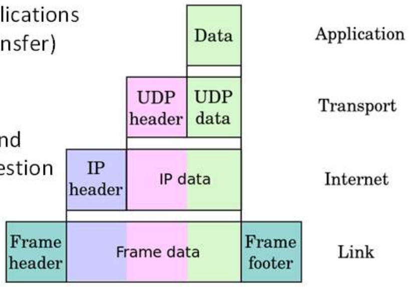
TCP/IP Protocols

- 4 layers: application, transport, internet, and link
- Each layer hides the implementation of the bottom layer

TCP/IP

Application Layer Protocols - Top Layer

- Describes format of data for specific applications
- HTTP (WWW), Telnet, email, FTP (file transfer)



Transport Layer Protocols

- Describe how data is sent from end-to-end
- Handles connections, flow control, congestion
- TCP, UDP

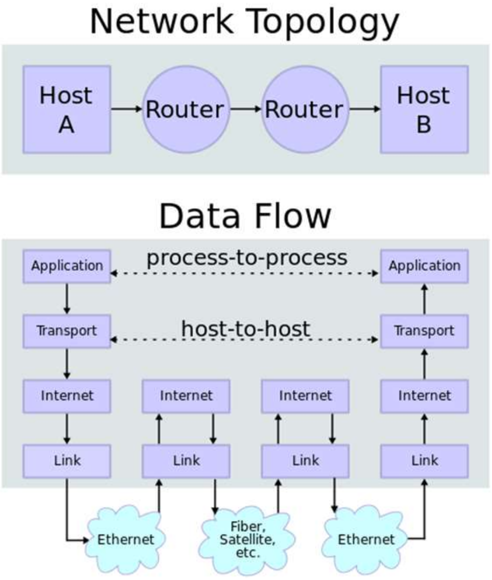
Internet Layer Protocols

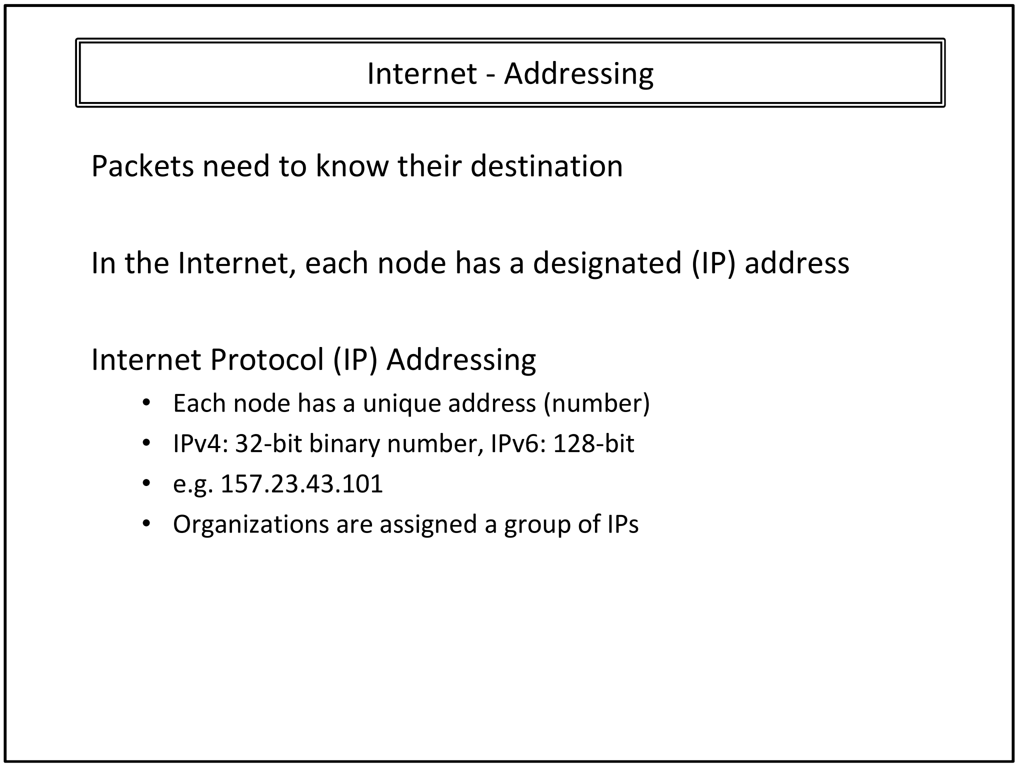
- Provides an addressing scheme
- Implements routing algorithms for transporting data to destination address
- IPv4, IPv6

Link Layer Protocols - Bottom Layer

- How two machines are physically connected
- Wired (Ethernet), Wireless (WiFi), PPP, Fiber

TCP/IP - Example





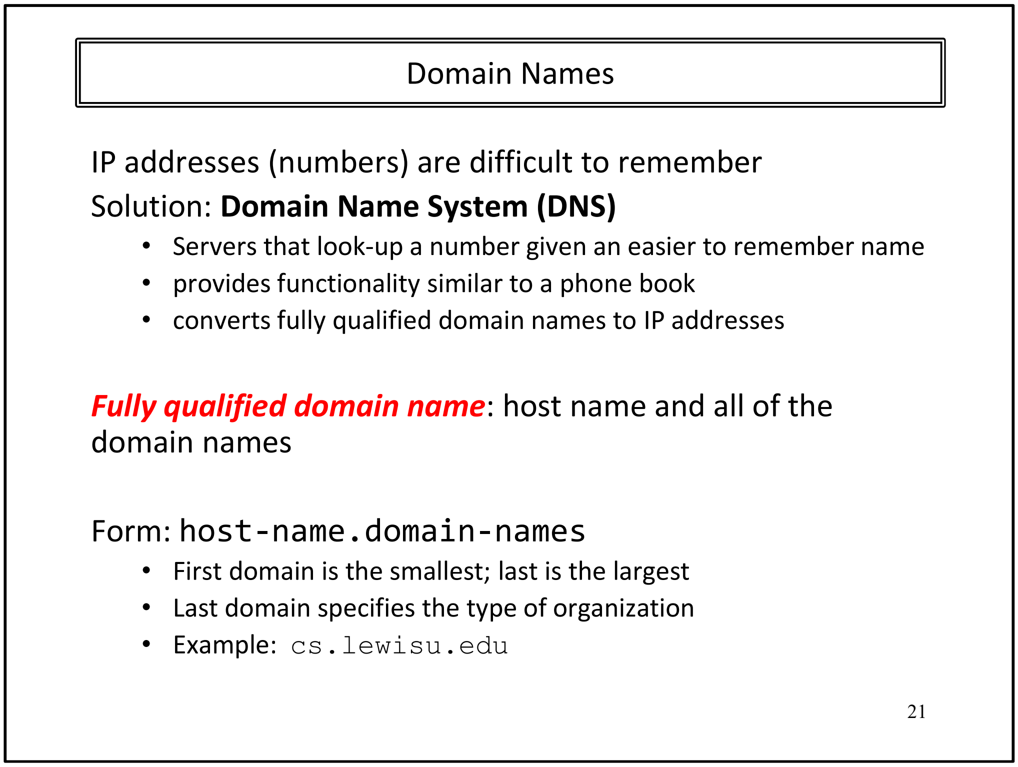
Internet - Addressing

Packets need to know their destination

In the Internet, each node has a designated (IP) address

Internet Protocol (IP) Addressing

- Each node has a unique address (number)
- IPv4: 32-bit binary number, IPv6: 128-bit
- e.g. 157.23.43.101
- Organizations are assigned a group of IPs



Domain Names

IP addresses (numbers) are difficult to remember

Solution: **Domain Name System (DNS)**

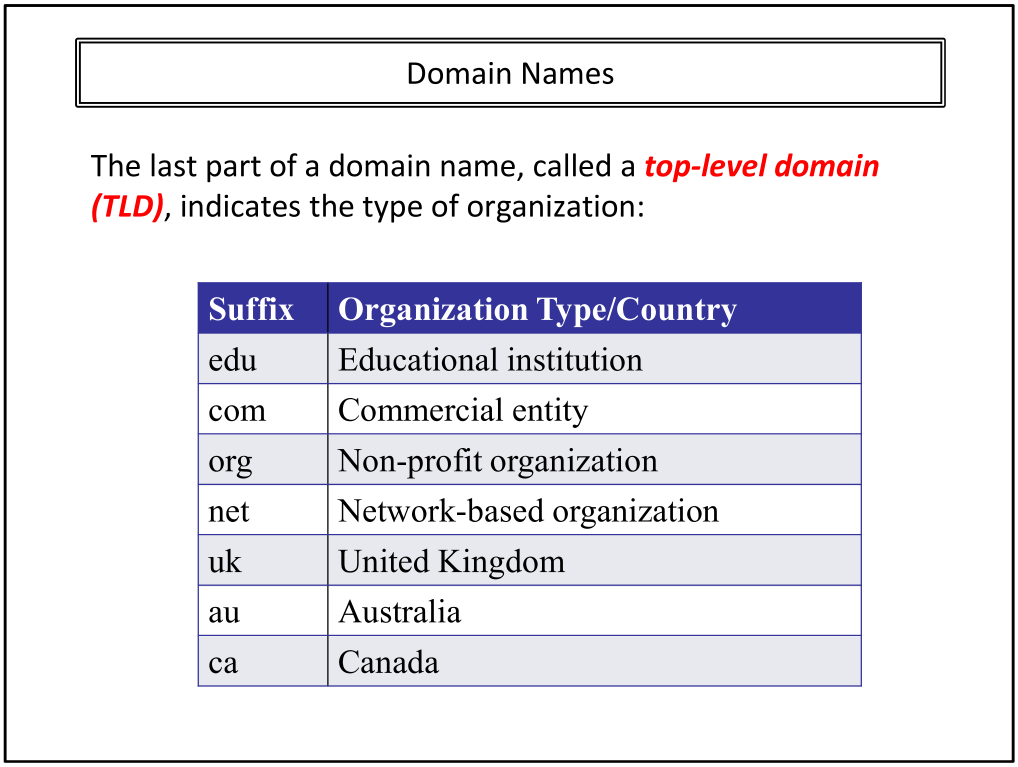
- Servers that look-up a number given an easier to remember name
- provides functionality similar to a phone book
- converts fully qualified domain names to IP addresses

Fully qualified domain name: host name and all of the domain names

Form: `host-name.domain-names`

- First domain is the smallest; last is the largest
- Last domain specifies the type of organization
- Example: `cs.lewisu.edu`

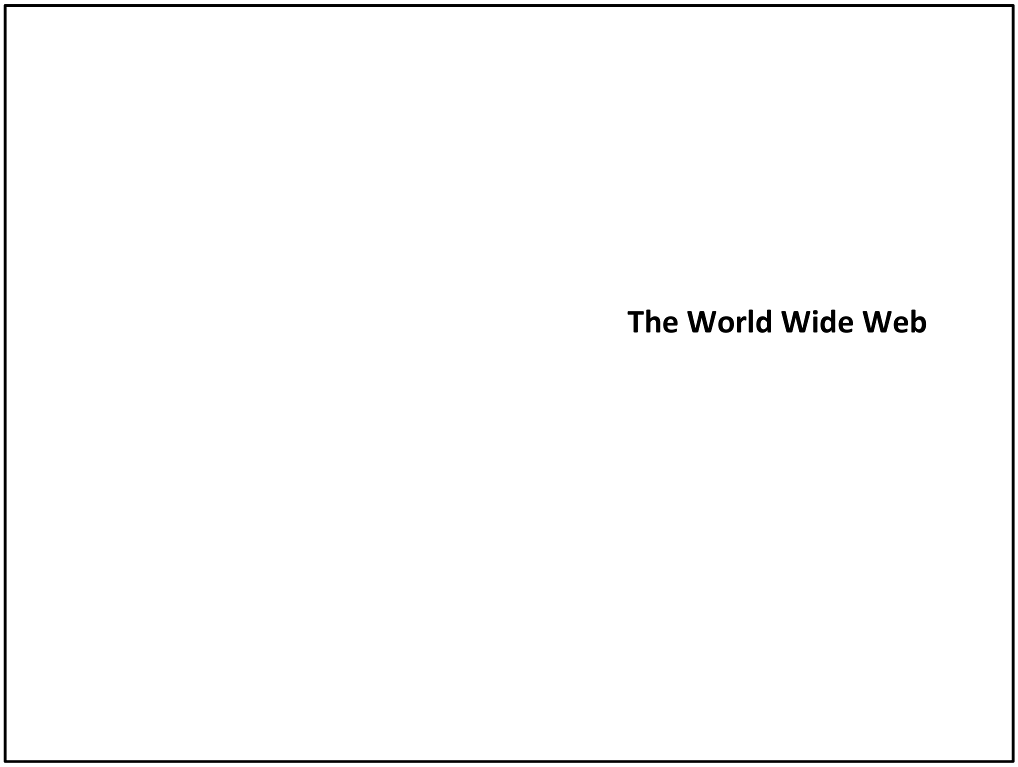
Share Bob Parsons DNS / GoDaddy learning moment.



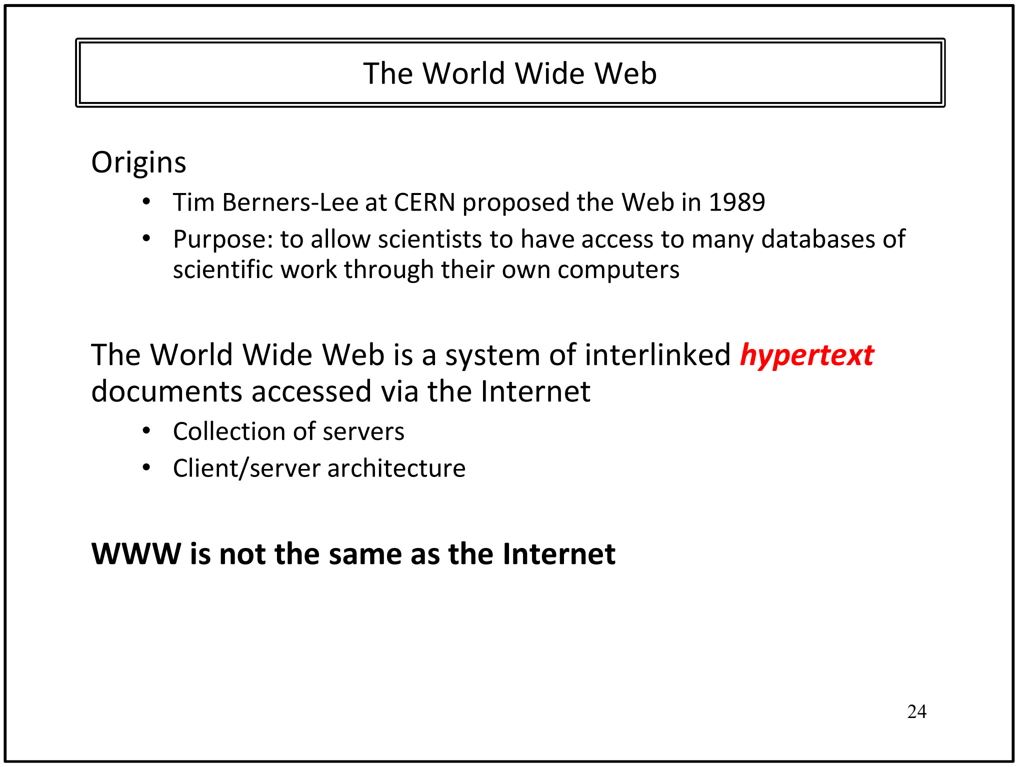
Domain Names

The last part of a domain name, called a ***top-level domain (TLD)***, indicates the type of organization:

Suffix	Organization Type/Country
edu	Educational institution
com	Commercial entity
org	Non-profit organization
net	Network-based organization
uk	United Kingdom
au	Australia
ca	Canada



The World Wide Web



The World Wide Web

Origins

- Tim Berners-Lee at CERN proposed the Web in 1989
- Purpose: to allow scientists to have access to many databases of scientific work through their own computers

The World Wide Web is a system of interlinked ***hypertext*** documents accessed via the Internet

- Collection of servers
- Client/server architecture

WWW is not the same as the Internet

Web Browsers

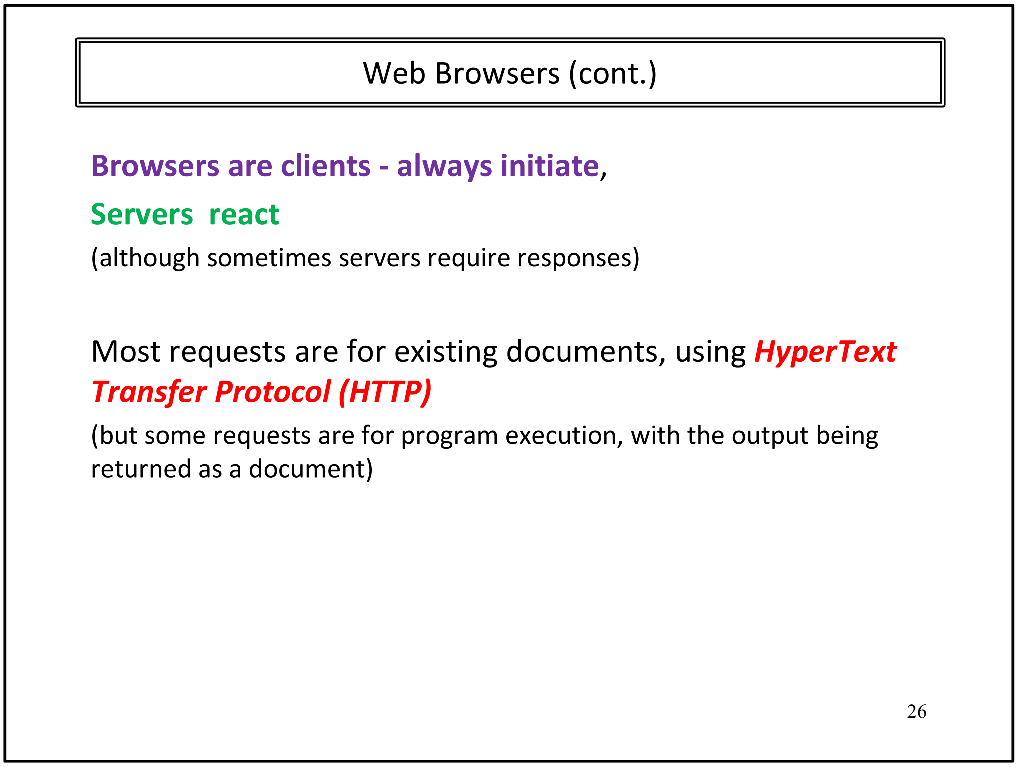
A ***browser*** is a program which accesses and presents information

- text, graphics, video, sound, audio, executable programs

Mosaic - NCSA (Univ. of Illinois), in early 1993

- First to use a GUI, led to explosion of Web use
- Initially for X-Windows, under UNIX, but was ported to other platforms by late 1993

Browsers request documents from ***Web Servers***



Web Browsers (cont.)

Browsers are clients - always initiate,

Servers react

(although sometimes servers require responses)

Most requests are for existing documents, using ***HyperText Transfer Protocol (HTTP)***

(but some requests are for program execution, with the output being returned as a document)

Web Servers

Web servers provide responses to browser requests

Either existing documents or **dynamically built documents**

Browser-server connection is now maintained through more than one **request-response cycle**

Web Servers (continued)

Communications between browsers and servers use HTTP

Web servers run as background processes in the operating system

- Monitor a communications port on the host, accepting HTTP messages when they appear

Web servers have two main directories:

- ***Document root*** (servable documents)
- ***Server root*** (server system software)

Document root is accessed indirectly by clients

- Its actual location is set by the server configuration file
- Requests are mapped to the actual location

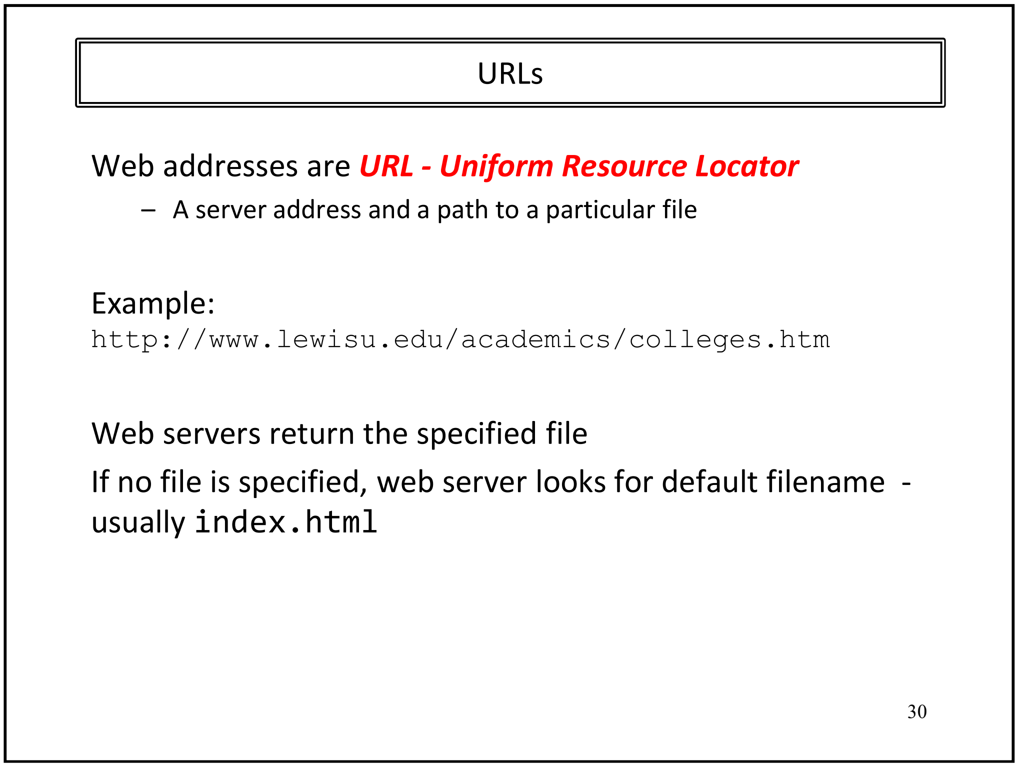
Common Web Servers

Apache (open source, fast, reliable)

- Currently the most popular web server
- Open source
- Cross-platform: UNIX, Linux, Mac OS X, Windows, etc.
- Installed on the computer science web server
- Maintained by editing its configuration file

IIS (Microsoft)

- Maintained through a program with a GUI interface



URLs

Web addresses are ***URL - Uniform Resource Locator***

- A server address and a path to a particular file

Example:

`http://www.lewisu.edu/academics/colleges.htm`

Web servers return the specified file

If no file is specified, web server looks for default filename - usually `index.html`

The HyperText Transfer Protocol

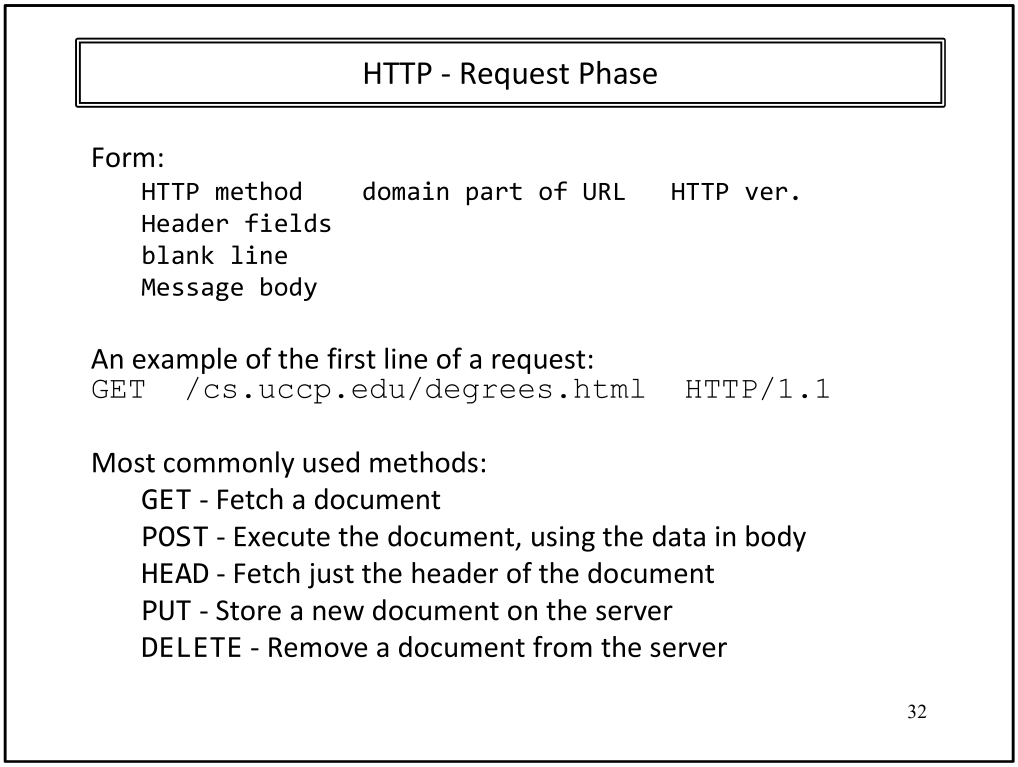
HTTP is the protocol used by ALL Web communications

Consists of two phases

- ***Request Phase***
- ***Response Phase***

Each communication (request or response) has two parts

- Header
- Body



HTTP - Request Phase

Form:

```
   HTTP method    domain part of URL   HTTP ver.
   Header fields
   blank line
   Message body
```

An example of the first line of a request:

```
GET  /cs.uccp.edu/degrees.html  HTTP/1.1
```

Most commonly used methods:

- GET - Fetch a document
- POST - Execute the document, using the data in body
- HEAD - Fetch just the header of the document
- PUT - Store a new document on the server
- DELETE - Remove a document from the server

HTTP - Header Fields

Four categories of ***header fields***

- General
- Request
- Response
- Entity

Common **request fields**:

- `Accept: text/plain`
- `Accept: text/*`
- `If-Modified_since: date`

Common **response fields**:

- `Content-length: 488`
- `Content-type: text/html`

Multipurpose Internet Mail Extensions (MIME)

Within some fields of the form, there are specified types

e.g.:

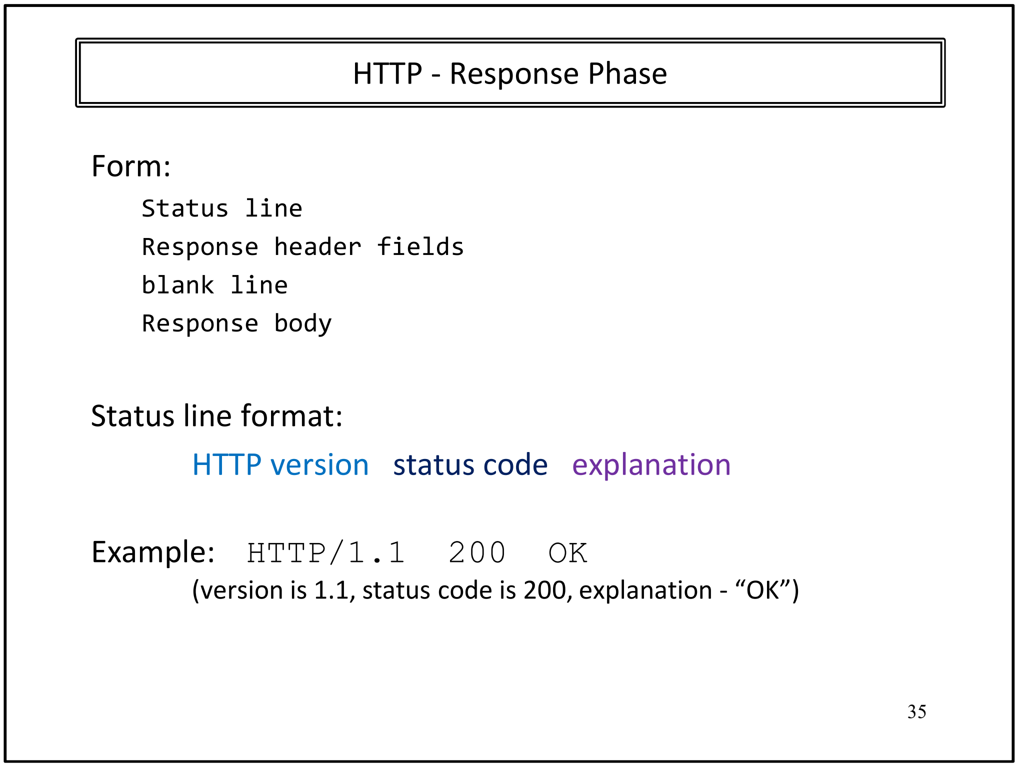
```
Content-type: text/html
Accept: text/plain
```

These are examples of ***Multipurpose Internet Mail Extensions (MIME)***

- Originally developed for email
- Used to specify to the browser the form of a file returned by the server (attached by the server to the beginning of the document)
- Form: type/subtype
- Examples: `text/plain, text/html, image/gif, image/jpeg`

Server gets type from the requested file name's suffix

- `.html` implies `text/html`
- Browser gets the type explicitly from the server



HTTP - Response Phase

Form:

```
Status line
Response header fields
blank line
Response body
```

Status line format:

HTTP version   status code   explanation

Example: `HTTP/1.1  200  OK`

(version is 1.1, status code is 200, explanation - “OK”)

Response Phase (cont.)

Status code is a three-digit number

First digit specifies the general status:

1 => Informational

2 => Success

3 => Redirection

4 => Client error

5 => Server error

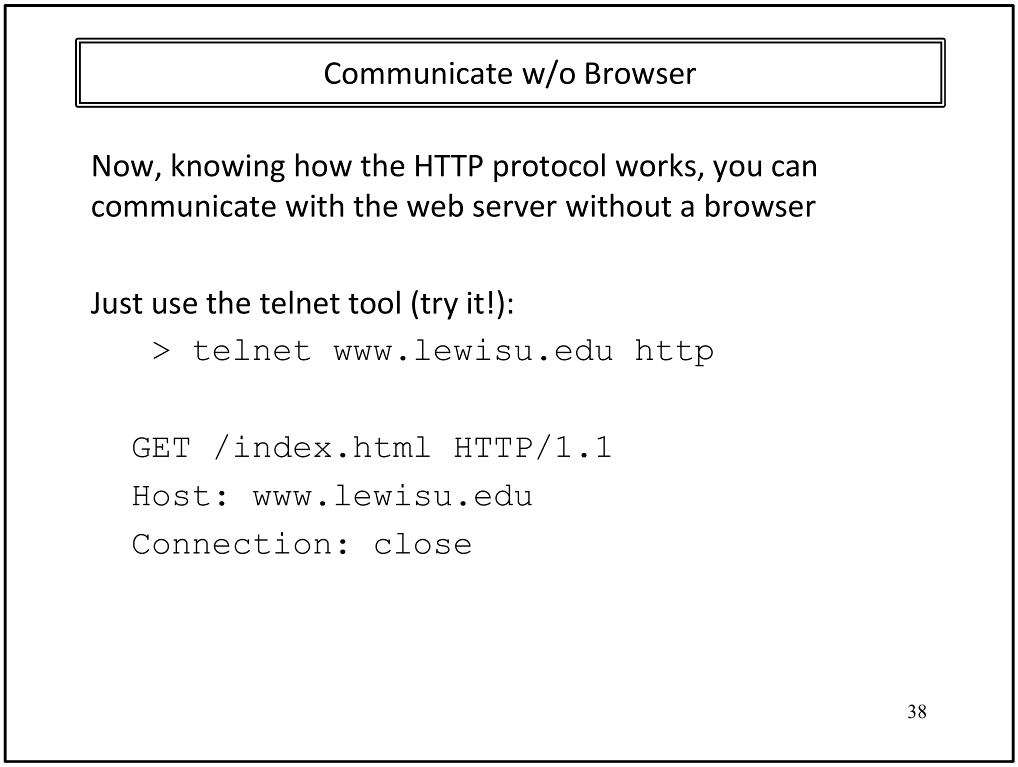
The header field, `Content-type`, is required

Response Phase (cont.)

An example of a complete response header:

```
HTTP/1.1  200  OK
Date: Sat, 25 July 2009 20:15:11 GMT
Server: Apache /2.2.3 (CentOS)
Last-modified: Tues, 18 May 2004 16:38:38 GMT
Etag: "1b48098-16a-3dab592dc9f80"
Accept-ranges: bytes
Content-length: 364
Connection: close
Content-type: text/html, charset=UTF-8
```

Both request headers and response headers must be followed by a blank line



Communicate w/o Browser

Now, knowing how the HTTP protocol works, you can communicate with the web server without a browser

Just use the telnet tool (try it!):

```
   > telnet www.lewisu.edu http

 GET /index.html HTTP/1.1
 Host: www.lewisu.edu
 Connection: close
```

Security Issues

Security

There are many kinds of security problems with the Internet and the Web

One fundamental problem:

Getting data between a browser and a server without it being intercepted or corrupted in the process

Security

There are 4 general security problems for a communication between a browser and a server:

Confidentiality (privacy): message contents should not be visible to unwanted parties

Integrity: message contents should remain unchanged during transmission

Authentication: messages should be sent to the appropriate (i.e. authorized) party

Nonrepudiation: legally prove message was sent and received

Security

The basic tool to support privacy and integrity is encryption

Encryption: changing a message in such a way that only an authorized party can read it

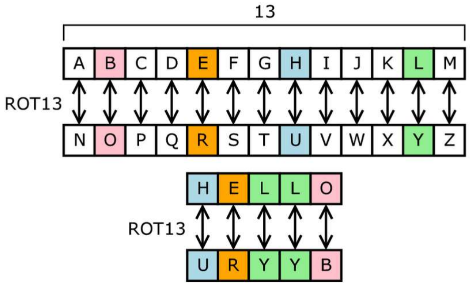
Decrypting the message requires the knowledge of the ***cipher*** used and the ***key***

Security

Example of a simple cipher (called the Caesar cipher):

- shift each letter by X positions in the alphabet
- X is the key (e.g. 13)

Example of encrypting “HELLO”:



13

A B C D E F G H I J K L M

ROT13

N O P Q R S T U V W X Y Z

H E L L O

ROT13

U R Y Y B

Security

Many encryption algorithms use symmetric-key ciphers: same key for encryption and decryption

Problem: If the sender and the receiver both use the same key, the key must be transmitted from the sender to the receiver

Solution: ***Public-key encryption***

- Use a **public/private key pair**
- It works because it is virtually impossible to compute the private key from a given public key

Public-Key Encryption

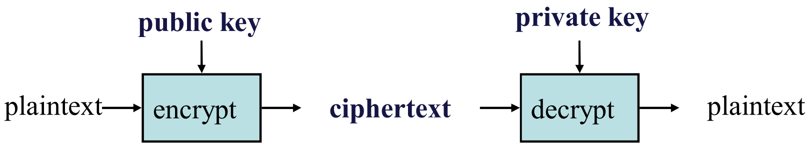
Assume Bob wants to send a message to Alice

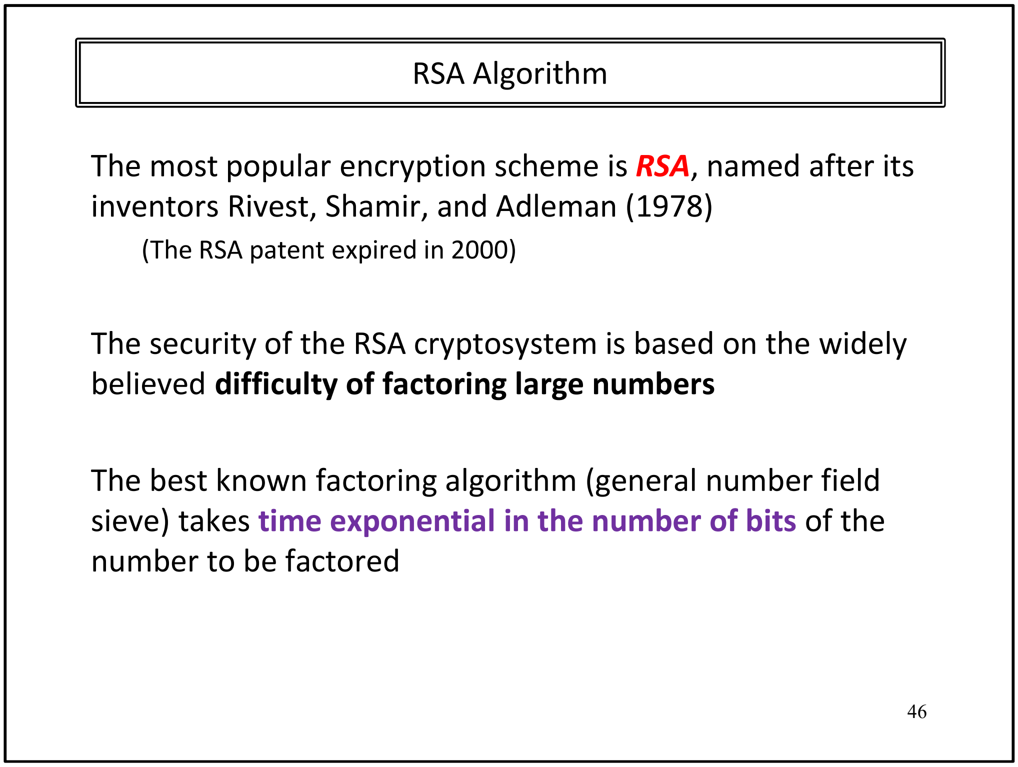
Alice uses a pair of keys (K_E,K_D) and

- makes key K_E public
- keeps key K_D private

Anyone can use the public key K_E to encrypt a ***plaintext*** into a ***ciphertext*** sent to Alice

Only Alice can decrypt the ciphertext using the private key K_D





RSA Algorithm

The most popular encryption scheme is ***RSA***, named after its inventors Rivest, Shamir, and Adleman (1978)

(The RSA patent expired in 2000)

The security of the RSA cryptosystem is based on the widely believed **difficulty of factoring large numbers**

The best known factoring algorithm (general number field sieve) takes **time exponential in the number of bits** of the number to be factored

Other security problems

Destruction of data on computers connected to the Internet

- Viruses and worms
- Other forms of malware

Denial-of-Service Attacks (DoS)

- Created by flooding a Web server with requests
- Even more powerful: Distributed DoS

Web Development Technologies

The Web Programmer’s Toolbox

Client Side

- **HTML**
- **XML**
- **JavaScript**
- Flash

Server Side

- **PHP**
- **Ajax**
- Servlets and JavaServer Pages
- ASP.NET
- Ruby
- Rails

HTML

HTML is used to describe the general form and layout of documents

An HTML document is a mix of **content** and **controls**

Controls are ***tags*** and their ***attributes***

- Tags often delimit content and specify something about how the content should be arranged in the document
- Attributes provide additional information about the content of a tag

Tools for creating HTML documents

- HTML editors - make document creation easier (Aptana Studio)
  (shortcuts to typing tag names, spell-checker, etc.)
- WYSIWYG HTML editors (Word)
  (need not know HTML to create HTML documents)

XML

XML is a ***meta-markup language***

XML is used to create a new markup language for a particular purpose or area

(since the tags are designed for a specific area, they can be meaningful)

No presentation details

A simple and universal way of representing and transmitting data of any textual kind

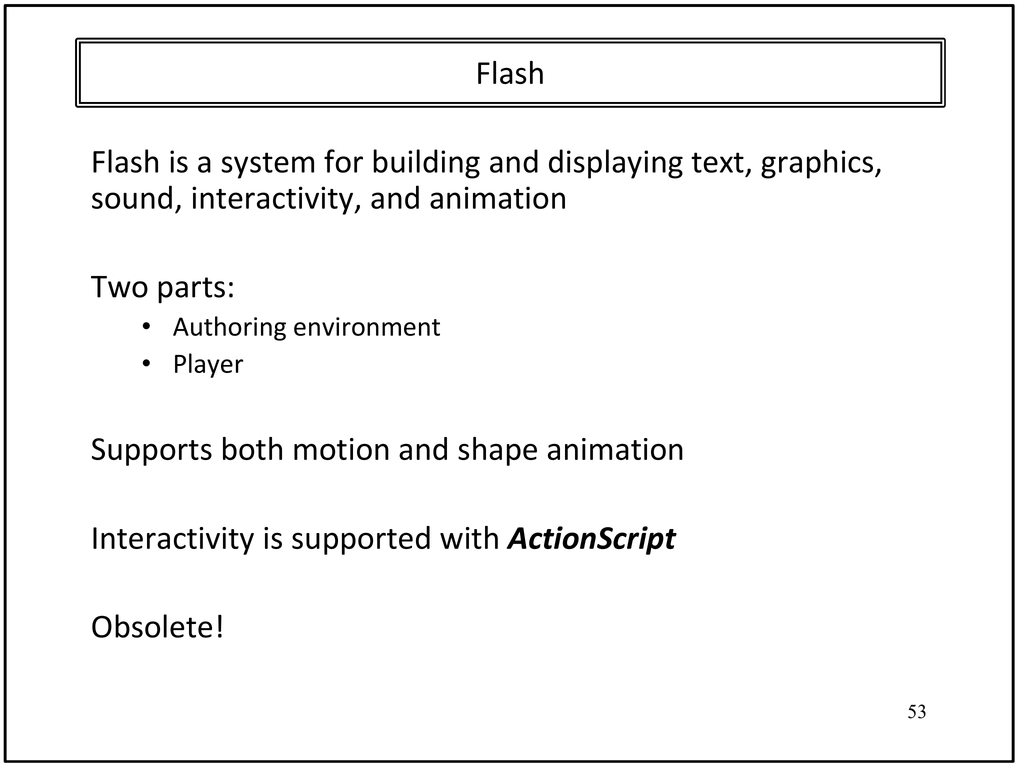
JavaScript

JavaScript is a client-side, HTML-embedded, ***scripting language***

Provides a way to access elements of HTML documents and dynamically change them

Only related to Java through syntax

Dynamically typed and not object-oriented



Flash

Flash is a system for building and displaying text, graphics, sound, interactivity, and animation

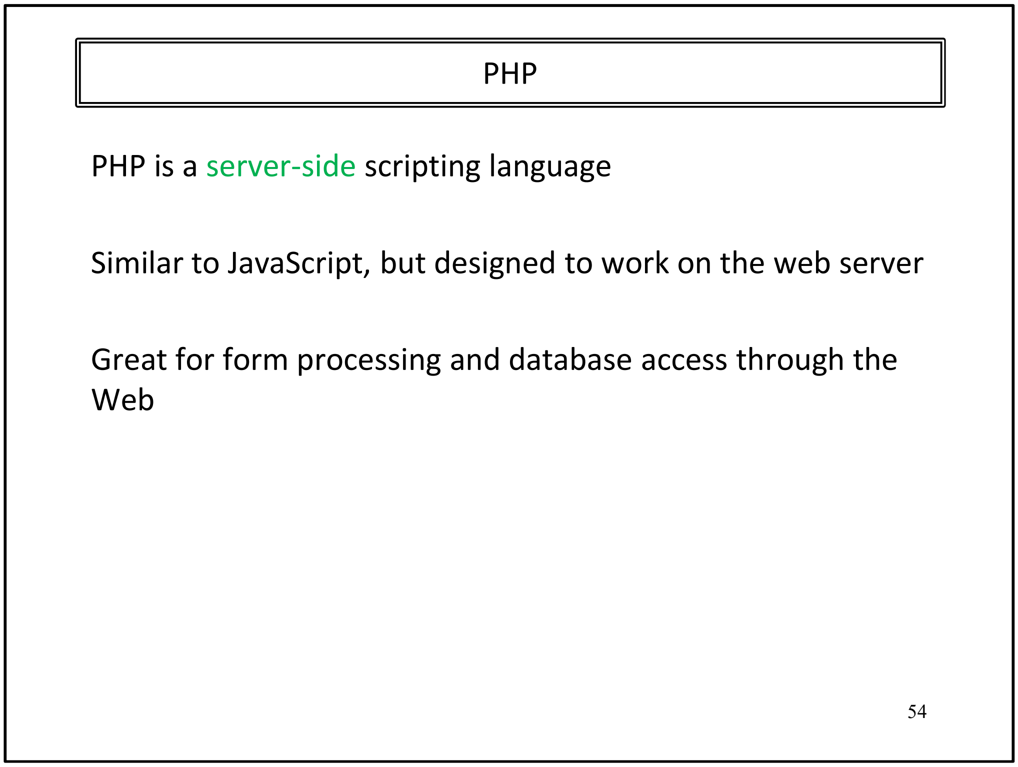
Two parts:

- Authoring environment
- Player

Supports both motion and shape animation

Interactivity is supported with ***ActionScript***

Obsolete!



PHP

PHP is a server-side scripting language

Similar to JavaScript, but designed to work on the web server

Great for form processing and database access through the Web

Ajax

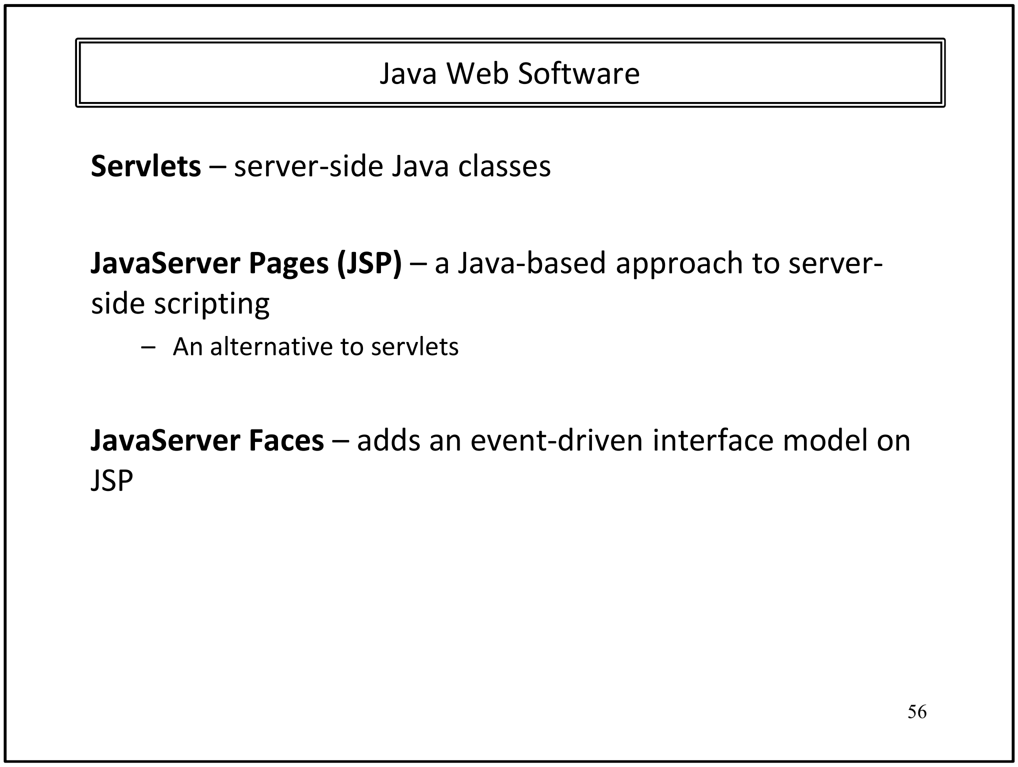
Ajax is ***Asynchronous JavaScript + XML***

- No new technologies or languages

Much faster for Web applications that have extensive user/server interactions

Uses ***asynchronous requests*** to the server

- Requests and receives small parts of documents, resulting in much faster responses



Java Web Software

Servlets – server-side Java classes

JavaServer Pages (JSP) – a Java-based approach to server-side scripting

- An alternative to servlets

JavaServer Faces – adds an event-driven interface model on JSP

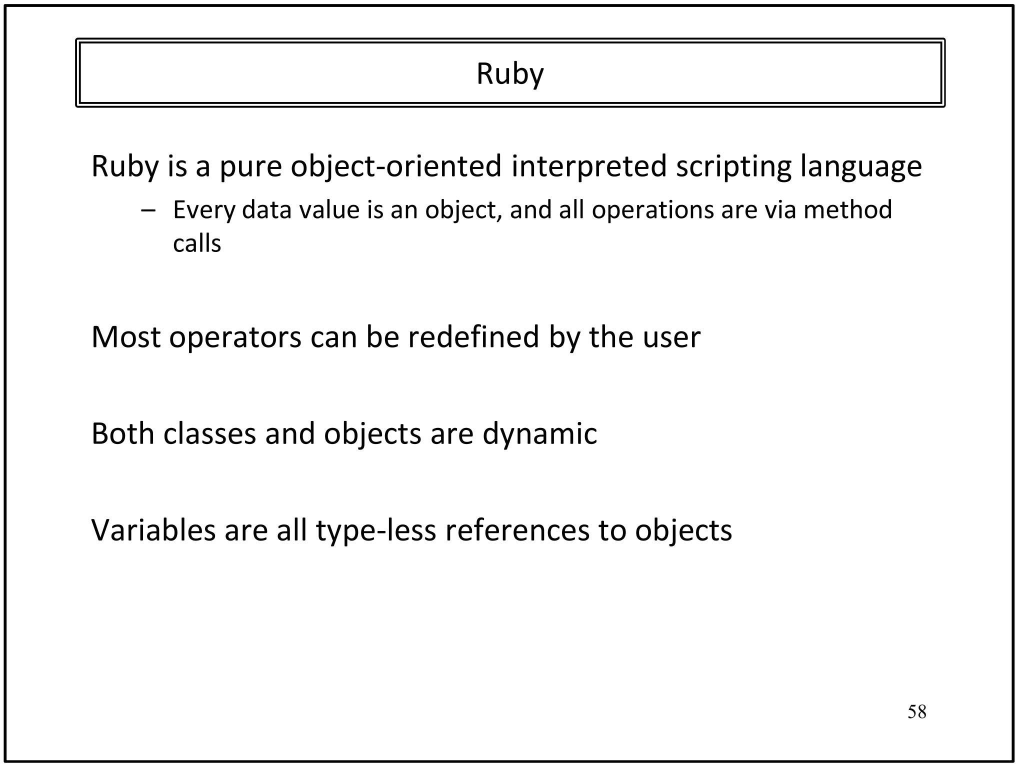
Active Server Pages

ASP does what JSP and JSF do

- But in the **.NET environment**

Allows any .NET language to be used as a server-side scripting language

ASP.NET documents are compiled into **classes**



Ruby

Ruby is a pure object-oriented interpreted scripting language

- Every data value is an object, and all operations are via method calls

Most operators can be redefined by the user

Both classes and objects are dynamic

Variables are all type-less references to objects

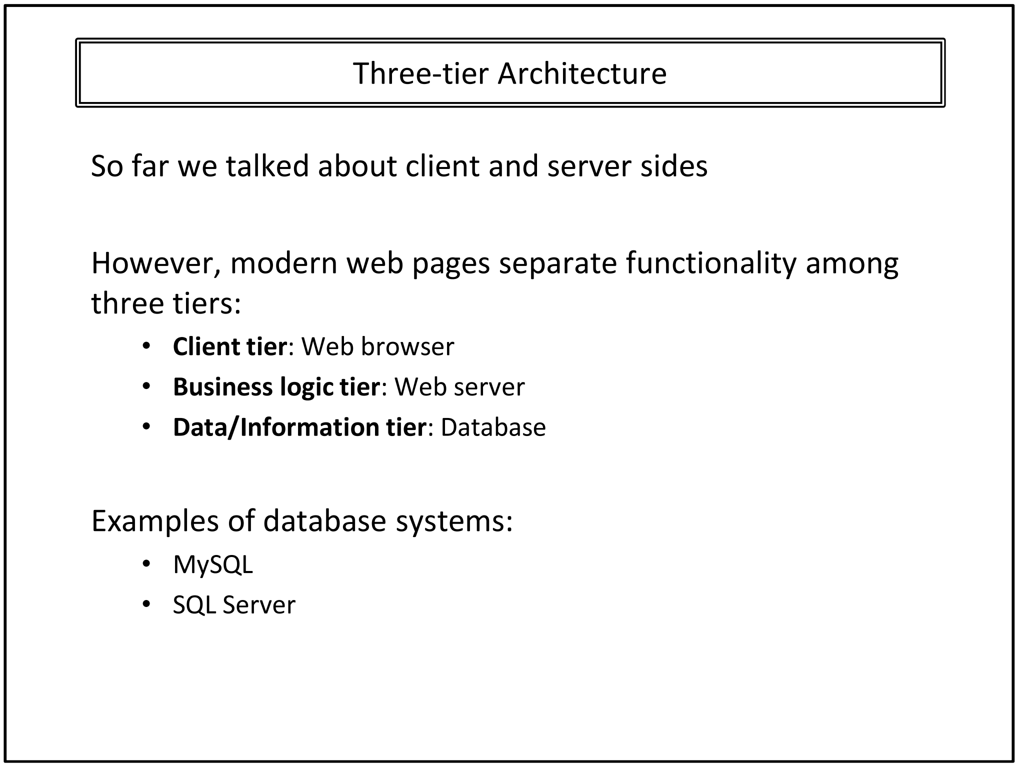
Rails

Rails is a development framework for Web-based applications

Particularly useful for Web applications that access databases

Written in Ruby and uses Ruby as its primary user language

Based on the ***Model-View-Controller architecture***



Three-tier Architecture

So far we talked about client and server sides

However, modern web pages separate functionality among three tiers:

- **Client tier**: Web browser
- **Business logic tier**: Web server
- **Data/Information tier**: Database

Examples of database systems:

- MySQL
- SQL Server

Summary

- Distributed systems are used for performance, cost-effectiveness, reliability, and when data is separated
- The Internet with the World Wide Web is the most famous distributed system
- The Internet is based on the idea of packet switching and uses the TCP/IP protocol suite
- The World Wide Web is based on the client/server architecture and uses the HTTP protocol
- The four security issues (confidentiality, integrity, authentication, and nonrepudiation) can all be addressed using encryption
- Common web development technologies include HTML, JavaScript (client side), PHP (server side), and MySQL (database)