

1.1 A Brief Intro to the Internet

- *Origins*

- ARPAnet - late 1960s and early 1970s**
 - Network reliability**
 - For ARPA-funded research organizations**
 - BITnet, CSnet - late 1970s & early 1980s**
 - email and file transfer for other institutions**
 - NSFnet - 1986**
 - Originally for non-DOD funded places**
 - Initially connected five supercomputer centers**
 - By 1990, it had replaced ARPAnet for non-military uses**
 - Soon became the network for all (by the early 1990s)**
 - NSFnet eventually became known as the Internet**
-
- ### **- *What the Internet is:***
- A world-wide network of computer networks**
 - At the lowest level, since 1982, all connections use TCP/IP**

1.1 A Brief Intro to the Internet (continued)

- ***Internet Protocol (IP) Addresses***
 - Every node has a unique numeric address
 - Form: 32-bit binary number
 - New standard, IPv6, has 128 bits (1998)
 - Organizations are assigned groups of IPs for their computers
- ***Domain names***
 - Form: host-name.domain-names
 - First domain is the smallest; last is the largest
 - Last domain specifies the type of organization
 - ***Fully qualified domain name*** - the host name and all of the domain names
 - DNS servers - convert fully qualified domain names to IPs
- ***Problem:*** By the mid-1980s, several different protocols had been invented and were being used on the Internet, all with different user interfaces (Telnet, FTP, Usenet, mailto)

1.2 The World-Wide Web

- **A possible solution to the proliferation of different protocols being used on the Internet**
- ***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**
 - **Document form: hypertext**
 - **Hypermedia – more than just text – images, sound, etc.**
- ***Web or Internet?***
 - **The Web uses one of the protocols, `http`, that runs on the Internet--there are several others (`telnet`, `mailto`, etc.)**

1.3 Web Browsers

- **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 are clients - always initiate, servers react (although sometimes servers require responses)**
- **Most requests are for existing documents, using HyperText Transfer Protocol (HTTP)**

1.4 Web Servers

- **Provide responses to browser requests, either existing documents or dynamically built documents**

1.4 Web Servers (continued)

- **All communications between browsers and servers use Hypertext Transfer Protocol (HTTP)**
- **Web servers run as background processes in the operating system**
- **All current Web servers came from either**
 - 1. The original from CERN**
 - 2. The second one, from NCSA**
- **Web servers have two main directories:**
 - 1. Document root (servable documents)**
 - 2. 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**
- **Virtual document trees**
- **Virtual hosts**

1.4 Web Servers (continued)

- **Proxy servers**
- **Web servers now support other Internet protocols**
- **Apache (open source, fast, reliable)**
 - **Began as the NCSA server, `httpd`**
 - **Maintained by editing its configuration file**
- **IIS**
 - **Maintained through a program with a GUI interface**

1.5 URLs

- General form:

scheme:object-address

- The scheme is often a communications protocol, such as `telnet` or `ftp`
- For the `http` protocol, the object-address is: fully qualified domain name/doc path
- For the `file` protocol, only the doc path is needed
- Host name may include a port number, as in `zeppo:80` (80 is the default, so this is silly)
- URLs cannot include spaces or any of a collection of other special characters (semicolons, colons, ...)
- The doc path may be abbreviated as a *partial path*
 - The rest is furnished by the server configuration

1.6 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)
- Type specifications
 - Form:
`type/subtype`
 - Examples: `text/plain`, `text/html`, `image/gif`,
`image/jpeg`
- *Experimental types*
 - Subtype begins with `x-`
e.g., `video/x-msvideo`
 - Experimental types require the server to send a helper application or plug-in so the browser can deal with the file

1.7 The HyperText Transfer Protocol

- The protocol used by ALL Web communications

- *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

1.7 The HyperText Transfer Protocol (continued)

- Four categories of header fields:

General, request, response, and entity

- Common request fields:

`Accept: text/plain`

`Accept: text/*`

`If-Modified_since: date`

- Common response fields:

`Content-length: 488`

`Content-type: text/html`

- Can communicate with HTTP without a browser

`> telnet blanca.uccs.edu http`

`GET /user1 /respond.html HTTP/1.1`
`Host: blanca.uccs.edu`

1.7 The HyperText Transfer Protocol

(continued)

- **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

(Current version is 1.1)

- **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**

1.7 The HyperText Transfer Protocol (continued)

- 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

1.8 Security

- There are many kinds of security problems with the Internet and the Web
- One fundamental problem is getting data between a browser and a server without it being intercepted or corrupted in the process

1.8 Security

- Security issues for a communication between a browser and a server:
 1. Privacy
 2. Integrity
 3. Authentication
 4. Nonrepudiation
- The basic tool to support privacy and integrity is *encryption*
- Originally, a single key was used for both encryption and decryption, which requires the sender of an encrypted document to somehow transmit the key to the receiver
 - *Solution: (1976, Diffie and Hellman)*
 - Public-key encryption
 - *Use a public/private key pair*
 - Everyone uses your public key to encrypt messages sent to you
 - You decrypt them with your matching private key
 - It works because it is virtually impossible to compute the private key from a given public key

1.8 Security

- RSA is the most widely used public-key algorithm
- *Another security problem:* destruction of data on computers connected to the Internet
- Viruses and worms
- Yet another common security problem:
Denial-of-Service (DoS)

1.9 The Web Programmer's Toolbox

- *HTML*

- 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**
 - Shortcuts to typing tag names, spell-checker,**
 - WYSIWYG HTML editors**
 - Need not know HTML to create HTML documents**

1.9 The Web Programmer's Toolbox

(continued)

- ***Plug ins***

- Integrated into tools like word processors, effectively converting them to WYSIWYG HTML editors

- ***Filters***

- Convert documents in other formats to HTML

- ***Advantages of both filters and plug-ins:***

- Existing documents produced with other tools can be converted to HTML documents
 - Use a tool you already know to produce HTML

- ***Disadvantages of both filters and plug-ins:***

- HTML output of both is not perfect - must be fine tuned
 - HTML may be non-standard
 - You have two versions of the document, which are difficult to synchronize

1.9 The Web Programmer's Toolbox

(continued)

- ***XML***

- **A meta-markup language**
- **Used to create a new markup language for a particular purpose or area**
- **Because the tags are designed for a specific area, they can be meaningful**

- ***JavaScript***

- **A client-side HTML-embedded scripting language**
- **Provides a way to access elements of HTML documents and dynamically change them**

1.9 The Web Programmer's Toolbox

(continued)

- ***Flash***

- **A system for building and displaying text, graphics, sound, interactivity, and animation (movies)**

- ***Two parts:***

- 1. **Authoring environment**

- 2. **Player**

- **Supports both motion and shape animation**

- ***PHP***

- **A server-side scripting language**

- **Great for form processing and database access through the Web**

1.9 The Web Programmer's Toolbox

(continued)

- ***Ajax***

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

1.9 The Web Programmer's Toolbox

(continued)

- *ASP.NET*

- Does what JSP and JSF do, but in the .NET environment**
- Allows .NET languages to be used as server-side scripting language**

- *Ruby*

- 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**

1.9 The Web Programmer's Toolbox

(continued)

- ***Rails***

- **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**