Overview

• The Internet and the World Wide Web
• HTML, SGML, and XML
• The Protocol
  – Requests and Responses
  – CGI
  – Javascript

The Internet

• The internet is a set of communicating machines
• The basis for communications is:
  – a shared machine address space (IP)
  – A name lookup mechanism -- Domain Name Space (DNS)
  – A protocol for integral messaging (TCP)
  – A protocol for doing business (http)
  – Software to interpret the messages exchanged
The Internet Generically

Internet Domain

168.245.13.1

136.142.78.4

168.245.13.13

An Internet Transaction

Domain Name Server

Xyz.com=168.245.13.13

136.142.78.4

1. Where is xyz.com?

2. Request to xyz.com

3. Request to 168.245.13.13

4. Response to Request

The World Wide Web (History)

- 1989, March Tim Berners-Lee (TBL), working at the Swiss Institute for Particle Physics (CERN) wrote “Information Management: A Proposal”
- 1990, Oct. TBL starts work on a hypertext GUI browser-editor using a Next Machine TBL coins the term WWW
- 1990, Dec the system is demonstrated
- 1992, Jan. Line mode browser available by FTP.
- 1993, Jan. X and Mac browsers released. 50 known servers.
- 1993, February NCSCA release Andreessen’s Mosaic for X
- 1993, October Over 200 known HTTP servers.
- 1994, March Marc Andreesen and colleagues leave NCSCA to form “Mosaic Communications Corp” (now Netscape).
The World Wide Web (Parts)

- Built on top of the Internet
- A simple protocol
  - GET, POST
  - PUT, HEAD, OPTIONS, TRACE, DELETE
- A simple message
  - Here is some data
  - Here is a “document”
- An increasingly complex server (state, authentication, encryption, application serving)
- An increasing complex client (parse a variety of documents, trace links, spawn applications)

The http protocol

- The web protocol is very robust and very simple
- For each request, the client:
  - Does a DNS lookup if needed
  - Opens a connection to the server
  - Sends a request for a resource
- The server
  - Checks the availability of the resource
  - Returns the resource or an error message
  - Closes the connection

The structure of requests and responses

- Requests have a header and a body
- The header has many lines but:
  - Begins with one of seven standard types
  - The body is null for five of the request types and contains data for the POST and PUT types
- Responses have a header and a body
- The header has many lines but:
  - Begins with a status
  - Ends with a content type
  - The body contains either the resource or an explanatory message
3 September 2001  WWW Document Technology  M.B. Spring

**Web Technology**

- Javascript
- Vbscript
- HTML
- CSS
- XML
- Applet
- Plugin capable browser

- http request
- Server analyzes request and gets page or runs program
- Pages for delivery
- Programs that produce pages

- Java Application
- Java Application

**HTML and SGML**

- The body of an http message may be anything, but frequently it is a document encoded using a markup language known as HTML.
- HTML is in reality simply an SGML “Document Type Definition” (DTD).
- SGML is the “Standard Generalized Markup Language”
  - SGML (ISO 8879) is a standard for document interchange
  - SGML divorces structure and appearance
  - SGML defines the rules for defining documents

**SGML Structured Documents**

- SGML is important in that it defines the rules for constructing structured documents
- Under SGML, a document is defined as a directed acyclic graph — i.e. tree consisting of a series of nested elements
- Elements consist of start and stop tags with the associated content
  - `<name>` is a start tag for element name
  - `</name>` is an end tag for element name
- Elements, through their tags, may have associated attribute sets.
  - `<name attributename= stringvalue>` associates stringvalue with attribute attributename for its particular instance of element name
**HTML and XML**

- HTML is a technically weak DTD
  - It defines a very weak structure (e.g. H3 anywhere)
  - Some tags (e.g. bold) are too procedural
  - HTML 1.1 is better than 1.0
- XML is gaining momentum as a replacement
  - XML is a language, like SGML but simpler for defining DTDs
  - XML companion standards are appearing very rapidly

**XML**

- XML, or eXtended Markup Language was developed to replace HTML on the Web
- It is a “simplified” version of SGML
- It is extended in that it offers more capability than HTML
- XML more complex document forms
- XML is also being used to “wrap” records.
  - XML datatypes and schema allows XML to wrap DBMS records and EDI transaction data

**An Structure of an HTML Doc**

- An HTML document has a `<head>` and a `<body>`
  - Don’t confuse with protocol the header and body
- The `<head>` of an html document contains control information (meta tags, title, keywords, scripts, etc.)
- The `<body>` of an html document contains all of the elements that will normally appear in the browser window
HTML Elements

- HTML elements fall into ten categories:
  - Overall document structure – head and body
  - Text level formatting – bold, italic
  - Block level – quote
  - List tags
  - Hyperlink tags
  - Image related tags
  - Table Tags
  - Form Tags
  - Frame Tags
  - Executable Content tags

Anchors and Hyperlinks

- HTML defines an element known as an Anchor:
  - `<A>This is an anchor</A>`
- A property or attribute of an anchor is its HREF – HyperText Reference
- Web HREF values are Universal Resource Locator
- `<A href= http://www.sis.pitt.edu:80/~spring/index.html> Home page Michael B. Spring</A>`
- A URL is made up four parts:
  - A service identifier – e.g. http://
  - An Internet Address – e.g. www.sis.pitt.edu
  - A port overriding the default service specification – e.g. 8080
  - An absolute path – spring/index.html

A Sample Request

- The user types the following in their client:
  - `http://www.sis.pitt.edu/~cascade/index.html`
- The client sends only a header:
  - `GET /~cascade/index.html HTTP/1.0`
  - `If-Modified-Since: Fri, 10 Oct 1997 17:35:54 GMT;`
  - `User-Agent: Mozilla4.7 [en] (X11; i; SunOS 5.6 sun4u)`
  - `Pragma: no-cache`
  - `Host: www.sis.pitt.edu`
  - `Accept: image/gif, image/jpeg, image/pjpeg, image/png, */*`
  - `Accept-Encoding: gzip`
  - `Accept-Language: en-US, en`
  - `Accept-Charset: iso-8859-1, *`
Request/Response Headers

- Authorization: encoding, name and password
- Content-Encoding: how the body is encoded
- Content-Length: length of the body
- Content-Type: type(mime) of the body
- Date: the date and time the request was generated
- From: email address of the requestor
- Last-Modified: date/time of last modification
- Pragma: directives to the client – e.g. no-cache
- Server/User Agent: server/browser type
- Referer: the address of the resource of the link

A Sample Response

HTTP/1.1 200 OK
Date: Wed, 01 Dec 1999 16:11:19 GMT
Server: Apache/1.3.1 (Unix)
Last-Modified: Wed, 12 May 1999 20:31:56 GMT
ETag: "7a108-16c2-3739e53c"
Content-Length: 5826
Connection: close
Content-Type: text/html

<html>
<head>
<title>CASCADE</title>
</head>
<body>

Status Codes

- Five categories of status code
  - 1xx: informational – used for development
  - 2xx: Successful response
  - 3xx: Redirection
  - 4xx: Client Error
  - 5xx: Server Error

- Frequently used codes:
  - 200 – success
  - 301 and 302 – moved permanently or temporarily
  - 400 – bad request
  - 401 – unauthorized
  - 403 – forbidden
  - 404 – not found
Development of web capability

• With time, it became clear that web was too static
• The Common Gateway Interface (CGI)
  – CGI created a capability to develop dynamic pages based on server program execution. Perl became the language of choice.
• Scripting Languages
  – As the CGI load on networks and servers grew, scripting languages were developed to offload some of the demand to the client
  – Full client side applications – applets emerged as well
  – Stylesheets were also added for clients
• Active Server Pages (ASP)
  – ASPs are pages that call functions that yield specific pieces of text.
  – These provided an alternative to CGI – programs that wrote pages.
  – Java Server Pages (JSP) parallel Microsoft’s ASP

HTML Forms and CGI

• To make pages more dynamic, the Common Gateway Interface (CGI) was defined
• CGI defines the rules for passing data to and running and application of the server
• “Forms” are to pass data to a CGI program
• The server, takes the data and gives it to the program which it runs.
• The program processes the data and returns the results to the – most commonly an HTML doc

Forms Construction

• A form is an element in the body of an HTML document.
• A form element has two attributes – method and action
  – The method specifies which http protocol will be used
  – The action specifies the program that will process the data
• A form will have one or more inputs elements
A Sample Form

```html
<form METHOD="POST" ACTION="http://augment.sis.pitt.edu/cgi-bin/form.cgi">
  <P>Name: <input TYPE="text" SIZE="40" MAXLENGTH="80" NAME="name" VALUE="anonymous">
  <P>Subject: <input TYPE="text" SIZE="40" MAXLENGTH="80" NAME="subject" VALUE="None">
  <input TYPE="submit" NAME="ssc" VALUE="Send Comment">
  <input TYPE="reset" NAME="clr" VALUE="Clear Comment">
</form>
```

Scripts

- The use of CGI for data validation, given the overhead of the transactions proved costly.
- To reduce the time and cost of simple processing, client side scripting was introduced
  - Javascript is one of the many scripting languages
  - Javascript is a java-like language that combines HTML objects and java-like syntax
A Sample Javascript

```html
<html><head><title>Javascript Validation</title>
<script language="JavaScript">
<!-- begin script hide
function checknum(Obj,min,max)
{val = Obj.value
 if ((val<=min)||(val>max))
 window.alert("Value in "+Obj.name+": "+Obj.value+" is out of bounds. It must be between "+min+" and "+max);
 Obj.value="";
 Obj.focus();
}
// end script -->
</script></head>
<body><form name =myform method = post action =">
<p>Field1:<input type=TEXT NAME=Field1 VALUE=0 onchange="checknum(this,0,100)"></p>
<p>Field2:<input type=TEXT NAME=Field2 VALUE=0 onchange="checknum(this,1000,2000)"></p>
<p>Field3:<input type=TEXT NAME=Field3 VALUE=0 onchange="checknum(this,-200,100)"></p>
<input type = submit name=submit>
</form></body></html>
```