Lecture 1

WWW Architecture Overview
Web Technologies

• Architecture Basics
• Presentation Markup - HTML
• XML
• IP – UDP – TCP
• HTTP – SMTP – FTP – BEEP
• XML
The Internet

• see http://www.w3.org/TR/

• A set of technology standards
  – IP, TCP, SMTP, HTTP, HTML.
  – Browsers – InternetExplorer, Netscape, Opera, Mosaic, ...
  – Web servers
  – The Internet

• A set of users

• See
  http://www.wired.com/wired/archive//3.06/xanadu.html?person=ted_nelson&topic_set=wiredpeople
What does the Internet look like?

• A collection of interconnected local area networks
• All are based on IP
• Packet based network
• UDP and TCP
Backbones in North America

• AGIS
• ANS
• ATMnet
• BBNplanet
• Compuserve
• CRL
• CWIX
• DataXchange
• DIGEX

• Epoch
• GetNet
• GlobalCenter
• GoodNet
• GridNet
• IBM
• Interconnect
• InternetMCI

• iSTAR
• MCIWorldcom 2000
• NapNet
• Netrail
• NFS
• PsiNet
• Savvis
• Sprint
• UUNET
The World Wide Web

• “Invented” by Tim Berners – Lee at CERN 1989
• Runs on the Internet
• WWW = HTML + HTTP + Browser + DNS
HTML – HyperText Markup Language

• A way of describing output in a browser
• Things like:
  – fonts, typefaces, bold, italics, underlining
  – lists (Bulleted, numbered)
  – background and foreground colors
  – hyperlinks
  – images and graphics
  – tables
  – metadata (information about the data in the page)
Sample HTML page in a browser

This is a sample HTML text

This is a sample HTML text - BOLDED

This is a sample HTML text - larger size 5

A bulleted list

- Item 1
- Item 2
- Item 3

A link to the class page
<html>
<head>
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>This is the Title of the HTML page</title>
</head>
<body>
<p>This is a sample HTML text</p>
<p><b>This is a sample HTML text - BOLDED</b></p>
<p><font size="5">This is a sample HTML text - larger size 5</font></p>
<p><font size="5">A bulleted list</font></p>
<ul>
<li><font size="5">Item 1</font></li>
<li><font size="5">Item 2</font></li>
<li><font size="5">Item 3</font></li>
</ul>
<p><font size="5">A link to the class</font></p>
<a href="http://www.cs.nyu.edu/courses/fall02/G22.3033-009/index.htm">page</a></p>
<p><img border="0" src="images/sub2asubway%20map..gif" width="293" height="191"></p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
<p>&nbsp;</p>
</body>
</html>
HTML uses tags

- `<html>` - first tag in an HTML doc
- `<head>` - heading (title bar info & metadata)
- `<meta>` - meta data about the page
- `<title>` - title for the title bar
- `<body>` - body of the document
- `<p>` - paragraph
- `<ul>` - list
- `<li>` - list item
- `<font>` - set the font
- `<font size="5">` - size is an attribute of the tag/element
- `<a>` - anchor (hyperlinks)
- `<img>` - embeds an image in the text
HTML – HyperText Markup Language

• A human readable text format that describes what a page should look like when displayed
• Loosely based on SGML – an application of a markup language
• Tags and values
HTML - Presentation Oriented

• Text oriented not data or objects
• No clue about the identity of that text.
• An HTML document is just a bundle of strings, not an object or even a data structure. No “name” given to a “value” i.e. string of text. Display attributes are supported, like bold, color, and font.
• Need XML to understand the structure of the text in a semantic way
• What you want is (name, value, type) tuples. See XML schemas for this.
• Maybe what you want is (name, value, type, presentation attributes, behavior including time related behavior)?
XML — eXtensible Markup Language

• A language for defining markup languages
• XML can be used to define an XML schema for HTML compliant documents. Thus, HTML could be an application written in XML.
• Defining your own markup language is creating your own DTD or XSD (Schema)
XML Terms

- Well Formed Document – has matching tags and follows the XML syntax rules
- Validating – passes the rules of a DTD or XSD.
- DTD is a simple template that defines what a document must “look” like. What it a can contain, can’t contain and where it can occur. No typing of values, everything is string.(not in an XML doc!)
- XSD (Schema) – a more robust and flexible template for defining an XML document. Supports typing of the values in an XML document e.g. int, long, date, string, …
- XSL – eXtensible Stylesheet Language – defines a “program” that takes an XML and transforms it to a display format like HTML. Supports conversions of XML (n,v,t) into other (n,v,t) via XSLT transforms.
XML Terms (2)

- Namespaces – supports the same tag name e.g. Name, Address, Phone being used in a document without collisions. Allows bringing together two different schemas in a single XML doc without creating problems referencing the correct data. Ex: a customer schema and a supplier schema that have name, address and phone elements. You can see a Customer and Supplier namespace in the combined document.
XML

- `<my tag>this is the value of my tag</my tag>`
- That’s is legal well formed XML, not validating XML though.
XHTML in a browser

This is a sample of text.

This is a sample hyperlink

- This is a bullet
- Another bulleted item
- Yet a third bullet

<table>
<thead>
<tr>
<th>This is a cell in table</th>
<th>cell 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>cell 3</td>
<td>cell 4</td>
</tr>
</tbody>
</table>

this is a submit button
XHTML – HTML that is an XML document

<?xml version="1.0" encoding="iso-8859-1"?><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"><html xmlns="http://www.w3.org/1999/xhtml"><head>
  <title>No title</title>
  <meta name="generator" content="amaya 6.2, see http://www.w3.org/Amaya/"/>
</head>
<body><p>This is a sample of text.</p>
<form method="get" action="POST">
  <p>This is a sample <a href="http://www.cs.nyu.edu/courses/fall02/G22.3033-009/index.htm">hyperlink</a></p>
</form>
<form method="get" action="POST">
  <ul>
    <li>This is a bullet</li>
    <li>Another bulleted item</li>
    <li>Yet a third bullet</li>
  </ul>
</form>
<table border="1" cellpadding="0" cellspacing="0">
  <tbody>
    <tr><td>This is a cell in table</td><td>cell 2</td></tr>
    <tr><td>cell 3</td><td>cell 4</td></tr>
  </tbody>
</table>
<p><form method="get" action="POST">
  <p><button name=""/>this is a submit button</button></p>
</form></p></body></html>
Client & Server Applications communicate via the protocols

Internet/WWW Protocol Layers

<table>
<thead>
<tr>
<th>SNMP</th>
<th>DNS</th>
<th>NFS</th>
<th>FTP</th>
<th>TelNET</th>
<th>POP</th>
<th>SMTP</th>
<th>BEEP</th>
<th>HTTP</th>
<th>LDAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP (Unreliable)</td>
<td>TCP (Reliable)</td>
<td>IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethernet 802</td>
<td>SLIP/PPP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cat 1-5</td>
<td>FDDI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATM</td>
<td>ISDN</td>
<td>ADSL</td>
<td>X.25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Web Browser using Protocol Layers

Web Browser

Web Server

HTTP

TCP (Reliable)

IP

Ethernet 802

Cat 1-5
IP

- Internetworking Protocol – 1982?
- Runs on top of Ethernet (IEEE802) protocol and others (ATM, FDDI, X.25)
- Provides sending packets between addresses
- “Fire and forget” protocol
- IPv4 addresses are 4 numbers separated by dots
  - 10.255.0.128 (all numbers are between 0-255, or 0x00 - 0xFF hexadecimal or a 32 bit binary number. It takes 4 bytes to hold the address.)
  - Means $2^{32} = 4,294,967,296 - 1 \approx 4.29 \times 10^9$ possible addresses!
  - Like “phone numbers” for internet hosts
IP (IPv4)

- Addressing subdivides “spaces” that may be granted to a group, corporation, university, individual, etc.
  - Class A – 1.a.c.d to 126.b.c.d (16+ million nodes)
  - Class B – 128.x.c.d 191.x.c.d (65,534 nodes)
  - Class C – 192.255.x.d 233.255.c.d (254 nodes)
  - Class D – 224.255.c.d 239.255.c.d (?? nodes)
  - Class E – 240.255.c.d 255.255.c.d – reserved
    *X is a number assigned to you!

- Because of class addressing far fewer than $2^{32}$ addresses, in reality. CIDR and IPv6 will fix this.
IP

- Connectionless (like sending a letter not like a phone call)
- Addressing and route selection
- Packet assembly and disassembly
- Header
  - Version
  - Source Address – 4 bytes
  - Destination Address – 4 bytes
  - Checksum
  - TTL (Time To Live) – number of router hops allowed
  - Total Length
  - Payload – between 1 – 64K bytes long
IP Addresses

• Static – Never changes unless you change it in your OS setup. Assigned to the MAC address (Unique ID embedded in the NIC) of your NIC (Network Interface Card) on a machine.

• Dynamic – assigned by a piece of software on the LAN that maintains a pool of addresses and hands them out as needed. They expire and you must get a new one (automatic). Using DHCP (Dynamic Host Configuration Protocol) your machine gets new IP Addresses. Efficient and flexible on a network with machines moving around or attaching and unattaching.
How do you find out a host’s IP address?

• They tell you

• You look it up
  – Hosts file on your machine – maps names to IP addresses. Names can be any form ex:
    – 121.6.7.8 www.thisthing.com
    – 121.6.7.9 foobardobar
    – use DNS
DNS

• Domain Name Server
• Converts from an web URL to an IP address (no ports that’s UDP/TCP)
  – www.nyu.edu is 128.122.108.9
  – on windows try ipconfig, winipconfig
  – on unix try ??
• Based on UDP on port 53
• Contains static IP addresses of hosts
DNS (cont)

• A set of servers that store all the top level domain names i.e. those that are:
  – www.xyz.com
  – .edu
  – .mil
  – .gov
  – .us / .uk / .tv / .de (or any other country code)
  – see www.alldomains.com/alltlds.html

• Returns IP addresses only
• DNS Servers share changes between them
• Your browser talks to a specified DNS server to look things up
• When you add a new host you must register the domain name and IP address with a DNS somewhere, usually through your ISP.
• Root servers are updated daily and managed by ICANN
TCP

- Transmission Control Protocol
- Connection oriented
- Guaranteed packet delivery – error free, in sequence sent, no dup,
- Runs on top of IP
- Slower than UDP
- Each IP Address also can have up to 65536 TCP Ports
  – Like phone extensions on a phone number.
TCP Socket

- IP Address + TCP Port Numbers of Client and Server
- Always PTP – Point to Point – like a phone call
- Handshaking between Client TCP Port and Server machine. Establishes a socket connection.
- Packets sent get acknowledged by the receiver.
- TCP handles congestion control
- Sockets system dependant implementation
- BSD Sockets machine independent
TCP Packet

- Source TCP Port – 2 bytes
  - Address is in the IP Header
- Target TCP Port - 2 bytes
  - Address is in the IP Header
- Sequence Number – 4 bytes = 0 – (2^{32}-1)
- Acknowledgement Number
- Window – size of the senders buffer
- Checksum – 2 bytes
- Control Bits
- Packet Size is in the IP header
- Payload

IP Packet

TCP Packet
Well Known TCP Ports (0-1024)

- 80     HTTP
- 443    SSL
- 20 & 21 FTP (20 control, 21 data)
- 25     SMTP
- 110    POP3
- 23     TelNet
- 161    SNMP
- 7      Echo
- 53     DNS
- others
UDP

• User Datagram Protocol
• Not connection oriented
• “Fire and forget” IP
• No guarantee – lost packets, dups, and out of sequence ok
• Faster than TCP – lower overhead
UDP Packet

• Inside the payload of an IP packet
• Source & Destination Port
  – Addresses are in the IP Header
• Checksum – 2 bytes
• Message Length is in IP Header
URL — Universal Resource Locator

- \textit{protocol} :// \textit{address} : port / \textit{path} #\textit{anchor}
- http://www.yahoo.com (defaults to :80)
- http://www.yahoo.com:80
- http://64.58.76.178 (same as yahoo.com)
- ftp://www.cs.nyu.edu (defaults to 20&21)
- file://
- smtp://
- beep://
HTTP

- HyperText Transfer Protocol
- Developed by Tim Berners-Lee at CERN in 1989-91
- Designed to share research information between users as HTML documents – Hyperlinks allowed navigation between parts of a document to other related documents

“\textit{The Hypertext Transfer Protocol (HTTP) is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object-oriented protocol which can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods (commands). A feature of HTTP is the typing of data representation, allowing systems to be built independently of the data being transferred.} “

- Tim Berners-Lee, et al. IETF Working Draft 1991 HTTP/1.0
http://www.w3.org/Protocols/HTTP/1.0/spec.html
HTTP Browser

- Initial browsers were text only, no images in the text.
- Port 80 is the default but can be configured on a different port say 8080 or 7000
- A client/server based protocol.
- Client (Usually a web browser) initiated the communication with a web server (HTTP server). Server on responds to requests made by the client, never initiates sending information to the client.
- Designed to static transfer documents (HTML pages).
- Not designed for dynamic applications.

See http://www.w3.org/Protocols/HTTP/1.0/spec.html
Browser Client – Web Server

Browser is the client

Client does a GET

Server sends a reply

Web Server is the server
HTTP Functions

• Invoked as:

  - http_URL = "http:" "//" host [ ":" port ] [ abs_path [ "?" query ]]

• HTTP Functions:

  - GET returns the item defined by the URN. Header and body. Metadata.
  - HEAD returns the header of the get not the body. Metadata about the item
  - POST sends data to the server from an HTML form
  - PUT puts the item on the server. Allows putting a HTML doc to a server.
  - OPTIONS returns the options supported by the HTTP server
  - DELETE removes an item from the server like a file delete via HTTP
  - TRACE returns the request back to the client like an HTTP echo
  - CONNECT used with proxy server (advanced)
  - extension-method allows adding new functions to HTTP server (proprietary)
“Structure” for a HTTP request

Request = Request-Line
  *(( general-header
  | request-header
  | entity-header ) CRLF)
CRLF
[ message-body ]
“Structure” for a HTTP response

Response = Status-Line

  *( ( general-header
       | response-header
       | entity-header ) CRLF)

CRLF

[ message-body ]
HTML page in a browser

This is a sample of text.

This is a sample hyperlink

- This is a bullet
- Another bulleted item
- Yet a third bullet

This is a cell in table  cell 2
cell 3  cell 4

this is a submit button
Sample HTML Form

<html>
<head>
<meta http-equiv="Content-Language" content="en-us">
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Sample HTML page that has a form</title>
<meta name="Microsoft Theme" content="global 110, default">
<meta name="Microsoft Border" content="tlb, default">
</head>

<body>
<p>Sample HTML page that has a form</p>
<form method="POST">
<p>This Text is in the form</p>
<p>Enter Your Name <input type="text" name="UserNameTextBox" size="20"></p>
<p>Enter Your State <select size="1" name="UserState">
<option value="AL">AL - Alabama</option>
<option value="NY" selected>NY - New York</option>
<option value="WI">WI - Wisconsin</option>
</select></p>
<p><input type="submit" value="Submit" name="SubmitButton"><input type="reset" value="Reset" name="ResetButton"></p>
</form>
<p>&nbsp;</p>
</body>
</html>
Sessionless and stateless

• Browser connects, makes request, gets response and disconnects.
• No information (no state) about the browser is maintained by the server between connections – no session.
• Efficient for operations where the server is waiting around a lot for browsers to make requests.
MIME

- Multimedia Internet Mail Extension (IETF 1992)
- A way to encode information send in email so that the receiver understands what the data is.
- Converts binary and non-US characters into 7 bit ASCII stream, then at the receiver can convert it back into the original data set
- mime header:
  - mime version
  - content type/sub type
  - content transfer encoding
  - content ID
MIME Header fragments

- MIME-Version: 1.0
  Content-type: text/plain; charset=us-ascii (Plain text)
- MIME-Version: 1.0
  Content-type: text/plain; charset="us-ascii"
- MIME-Version: 1.0
  Content-Type: text/plain; charset=ISO-8859-1
  Content-transfer-encoding: base64
S/MIME

• Secure MIME
• S/MIME was originally developed by RSA Data Security, Inc.
• Currently uses:
  – X509v3 Certificates
  – TRiple DES encryption
  – Diffie-Hellman with DSS Signatures
  – SHA-1 Hash Algorithm
• Encrypts and signs MIME messages
<table>
<thead>
<tr>
<th>WWW Protocol Layers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SNMP</strong></td>
</tr>
<tr>
<td>NFS</td>
</tr>
<tr>
<td><strong>Ethernet 802</strong></td>
</tr>
<tr>
<td>Cat 1-5</td>
</tr>
</tbody>
</table>
Architecture and WWW

- Application architectures can be broken down into layers. Common layering is a 3 tier (layer) architecture
  - Presentation Logic - does interaction with a User
  - Application Logic – business logic of an application
  - Data Logic – data access/manipulation logic

- WWW
  - Browser + WebServer = Presentation Layer
  - App Server (WebLogic, ASP Pages, PHP) = Application Logic Layer
  - Database/files = Data Logic
Why Tiers/Layers?

• Allow building a more flexible application
• Allows changing parts without changing the whole – changing the database from Oracle to SQL server for instance
• Allows reusing running functionality in a different application – Employee view and HR Administrator views of data in the enterprise applications that store the employees address, for instance. Could be two different applications that share an application component called “employee” and use a method of that component called “getEmployeeAddress()”
Web Application Tiers

- Web (HTTP) Server
  - Web Browsers
  - Servlets
  - EJB / App Server
  - PHP
  - ASP
  - .net App Server
  - CGI
  - Oracle Database
  - Files
  - ERP System
  - SQL Server Database
Tiers

• **Presentation** – formatting information to display it. Checking input with simple checks.

• **Business (Biz)/Application Logic** – calculations, rules that are business level
  
  \[\text{if age}<18 \text{ then don’t sell them beer }\]

• **Data Logic** – storage and manipulation of persistent data. Things like name, address, phone, invoices, orders, grades,…
Summary

• The WWW is based on HTML + HTTP + DNS + Internet
• HTTP is based on TCP based on IP …
• HTTP is sessionless and stateless
• HTTP uses MIME to encode information passed back and forth between the client and the server
• HTML is presentation oriented
• XML is data oriented and can be used to develop markup language including HTML
• The general architecture of WWW applications is 2 or 3 tier – presentation, application and data tiers