World Wide Web

TCP/IP class

Jim Binkley
outline

- intro - the big picture (elephant in breadbox)
- HTML - Hypertext Markup Language
  - we can hopefully ignore this as you know it
- HTTP - Hypertext Transfer Protocol
- etc. (short, but there is a lot of etc)
  - elephant makes room for rhino, hippo, and giraffe
what is the World Wide Web?

an information system that links data from different protocols under one umbrella

it allows pages to be linked together so that you can jump from one to another in a non-continuous way (hypertext over the Internet) (...end of linear thinking ...)

it allows display of graphics (2D images) plus small doses of audio and even video -- tolerates heterogeneous datatypes and very-well may slice bread

no, it is NOT the Internet, just one more meta-network. It is a loose collection of technologies though.

Jim Binkley
key technologies/buzzwords

- html and http
- html - Hypertext-markup-language
  - you probably know some of this (or all?)
- http - Hypertext-transfer-protocol
  - end to end transport on top of TCP
  - uses MIME like SMTP/email, FTP-like error messages
  - http used to transfer html and/or other file types
- URLs - addresses
1989, Tim Berners-Lee at CERN (European Laboratory for Particle Physics) proposed World Wide Web protocols

W3 consortium now “leading” effort, includes CERN, MIT, INRIA, see http://www.w3c.org

early browser called Mosaic, done at NCSA (National Center for Supercomputing Applications), 1993

then netscape, then browser wars (netscape vs. IE)

plus a blizzard of possible add-on technologies to extend the web on the client or server sides
  – java/CGI&perl/JavaScript/dynamic html/plugs-ins, blah, blah
this slide is wrong? - standards

- html 4.0, see http://www.w3.org for updates
  - not the whole picture of course given netscape vs IE hooks
- rfc2616, http/1.1
- many other possible documents including security-related (SSL) + email/MIME RFCS

Jim Binkley
intro - concepts

- web client supports > 1 protocol for fetching documents, HTTP (native web), ftp, gopher, USENET, WAIS, telnet(?), etc.
- HTML page == formatted (graphics+text+links)
- key here is tying **graphics** and **text** together, along with **hypertext** links; i.e., a discontinuous jump to other material anywhere on net
- link = to ftp, to telnet, to more HTML hypertext, to arbitrary program at web server

Jim Binkley
basic client/server architecture

browser

http get file/tcp to port 80

MIME type + .html/.png file to client tcp port

web server/httpd

read file from file system

~jrb/index.html + jrb.png

web servers serve html, web browsers may do more than http/html

Jim Binkley

blah, blah

netscape/IE as CLIENT
hypertext links - URLs

- a link will normally include a WWW network address for a page or something...
- called an URL, Uniform Resource Locator
- syntax = protocol://dns name[:tcp port]/file
- examples:
  - ftp://zymurgy.cs.pdx.edu
  - file:/some/where/local.txt
  - telnet://somewhere.mud.edu:8000
  - gopher://some.gopher.server.edu/
  - news:alt.fan.cecil-adams (note: no dns name)

Jim Binkley
this is a GREAT idea

* in the history of GREAT ideas ...  
* and so simple
web browser may speak more than http; e.g., ftp client too

and it may do email, make coffee, change baby diapers, etc...

Jim Binkley
file format extensibility

- web has native set of file formats including HTML docs, and graphics but clients and servers can be extended to handle other file formats (and other code ...)
- fetch of .ps file may invoke GNU ghostscript postscript viewer
- client/server communicate file format info via mail MIME encoding format
- client may be taught to invoke external application to handle new format
file heterogeneity

- MIME encodings exist for many kinds of data or can make new one up on fly
- e.g., windows pc netscape viewer can be taught to invoke powerpoint for “.ppt” files
- can invoke audio/video viewers for sound and video files
- client may not know how to display a server-side file, should just do download
client/server code extensibility

- for many reasons, desirable to extend both client/server functionality
  - just a few examples of MANY possible technologies
- server-side
  - common-gateway-interface with perl
  - basically API between web server and some program to pass parameters coming in over the web
  - could invoke database OR whatever
client-side extensibility

- may wish better gui/formatting than with just plain HTML OR
- wish to offload work from busy server (server scalability issues)
- can use java/JavaScript, etc
  - java can be used on server-side for that matter
- our goal here is NOT to explore these issues (basically just http ...)

Jim Binkley
intro - summary

- platform independent (HTML)
- protocol opaque (HTTP/ftp/gopher, etc.)
- ties docs together over net with hypertext links (HTML/links)
- 2-d graphics (HTML)
- can tolerate file heterogeneity (MIME)
- client/server extensions via various programming languages/techniques
intro - HTML

- HTML is a language that consists of ways of “marking up” text and including pictures and links
- the markup symbols are called **tags** and are not displayed at the viewer, rather they are interpreted as suggestions as how to format the display
- clients format HTML as best they can - interpretation is not the same from client to client
- tags include ways to include pictures in GIF/JPEG/png format, links, paragraphs, lists, GUI objects like buttons and fill-in fields (**forms**)  
- note: html really is NOT a networking protocol, just a display language somewhat akin to postscript/NROFF/Tex

Jim Binkley
HTML example

- <p> blah blah blah </p>
  <p> blegh <b>foo!</b> </p>
- when we fetch and display the HTML:

  blah blah blah
  blegh foo!
HTML < SGML

- HTML is subset of SGML, Structured Generalized Markup Language
- SGML used by US DOD/ISO developed
- Software exists for SGML
- Key is that HTML is simplified over SGML
- Another key: “trust the client”
- Tradeoff: platform independence versus authoring control

Jim Binkley
religion and html

◆ some want absolute control over how their data is displayed, want “physical” control
◆ some want platform independence, want “logical” suggestions where client does best job it can according to local circumstances
◆ <strong>Do It My Way! </strong> 
◆ <b>Do It My Way! </b>
**some basic html tags**

<table>
<thead>
<tr>
<th>element</th>
<th>type</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>container</td>
<td>src/dest of link</td>
</tr>
<tr>
<td>B</td>
<td>container</td>
<td>bold text</td>
</tr>
<tr>
<td>LINK</td>
<td>empty</td>
<td>link from this doc</td>
</tr>
<tr>
<td>BR</td>
<td>empty</td>
<td>line break</td>
</tr>
<tr>
<td>H1...H6</td>
<td>container</td>
<td>heading level</td>
</tr>
<tr>
<td>IMG</td>
<td>empty</td>
<td>image</td>
</tr>
<tr>
<td>LI</td>
<td>empty</td>
<td>list item</td>
</tr>
<tr>
<td>UL</td>
<td>container</td>
<td>unordered list</td>
</tr>
<tr>
<td>P</td>
<td>empty</td>
<td>paragraph</td>
</tr>
<tr>
<td>HR</td>
<td>empty</td>
<td>horizontal rule</td>
</tr>
</tbody>
</table>

Jim Binkley
html example - the basic skeleton

<html>
<head>
<title>Simple Web Page</title>
<link rel="MADE" href="mailto:jrb@cs.pdx.edu">
</head>
<body>

THE BODY GOES HERE

</body>
</html>
**html body - the inside**

<h1> Simple Web Page - first level header </h1>

Here is a picture of my friend, Bev Kramlich, hope she never hears about this. <p>
<img src="bevk.png"> Bev Kramlich <p>

<h2> A Second level header. Plus Interesting Web Places to Visit </h2>

<!-- <b> you didn’t see this -->

<UL>
  <LI> <A href="http://www.NCSA.uiuc.edu/SDG/People/robm/sg.html">A Typical System Administrator</A>
</UL>

<hr>

<address> somebody@somewhere.org </address>

Jim Binkley
Simple Web Page – first level header

Here is a picture of my friend, Bev Kramlich, hope she never hears about this.

Bev Kramlich

A 2nd level header. Plus Interesting Web Places to Visit

- A Typical System Administrator
- Jim Binkley’s Home Page
- NASA Home Page
- A Beginner’s Guide to HTML
- FreeBSD
- Internet RFCs

Phineas Phogg Phreaker phogg@universe.org
HTTP protocol - encapsulation

<table>
<thead>
<tr>
<th>link</th>
<th>ip</th>
<th>tcp</th>
<th>http</th>
<th>data</th>
</tr>
</thead>
</table>

http on top of tcp on top of IP

data == ASCII text, html, image,
typed with MIME type
HTTP - Hypertext Transfer Protocol

- protocol web clients use to talk to “web” servers (use http/fetch html)
- TCP-based, typically to server port 80
- simple request/response protocol
- client makes request, tells server what it can handle for file types
- server responds with MIME type + data file, type info usually gained from file suffix (foo.png)
- commands done in ASCII, errors in ASCII

Jim Binkley
HTTP, cont.

- commands called “methods”, but for the most part, just a variation on “get file”
- server status and errors similar to error strings found in ftp/email
  - 200 - successful
  - 300 - not done yet; e.g., 301 is moved permanently
  - 400 - client error; e.g., 403 forbidden (server refuses)
  - 500 - server error; 503 service unavailable at the moment
- http 1.0 being replaced by http 1.1

Jim Binkley
protocol overview

**client**
- TCP connect
- HTTP get file
- accept filetypes x, y, ...

**server**
- TCP/socket accept (port 80)
- server status + header
- info
- MIME type
- return file data

<read and display data>
TCP close

note: typically DNS before TCP ...

Jim Binkley
HTTP 1.0 Methods (3)

- **GET** file - used for fetching most HTML documents, file is URL minus protocol/DNS portions.
  - may use conditional **if-Modified-Since** time
get is done only if object is newer than time
  - also used for one form of cgi-bin forms ("get")

- **HEAD** file - get server-side header info about file but not file itself. Used for link test, cache test.

- **POST** cgi-bin/file - another way to do forms
  - theoretically used to annotate/append/”post” message or send record to database
HTTP methods, cont.

- but other methods have been proposed; e.g.,
- **PUT** - put new URL and overwrite old one
- **DELETE**
- question is: how to authorize remote file access; i.e., how to make it secure so can do PUT/DELETE, therefore less available
- original designers hoped that **annotation** of pages would be possible (yellow-sticky analogy along with hypertext) - unsuccessful idea at this point

Jim Binkley
protocol trace - GET method

% telnet localhost 80

request:  GET /foo.html HTTP/1.0 <cr> <cr>

header:  HTTP/1.0 200 OK
          Date: Tuesday, 22-Nov-94 18:10:58 GMT
          Server: NCSA/1.3
          MIME-version: 1.0
          Content-type: text/html
          Last-modified: Wednesday, 16-Nov-94 21:18:37 GMT
          Content-length: 1115

body:    <HTML> <TITLE> Joe FooBar’s Home ... </TITLE>
          HTML...etc., etc...

Jim Binkley
server-side note (file mapping)

- in previous slide /foo.html is mapped on server side to server documents file tree
- e.g., with server on UNIX, file tree maybe /usr/local/httpd/htdocs/index.html
- GET / -> root is mapped to index.html
- /foo.html would be in above htdocs directory
on UNIX server, users may have home pages

http://foo.org/~bob

cd ~bob; mkdir public_html;

make it world readable, chmod 664 public_html

make your home page public_html/index.html

make it world readable too

so http://foo.org/~bob as URL is mapped to
protocol trace - HEAD method

% telnet localhost 80

HEAD / HTTP/1.0 <cr> <cr>

HTTP/1.0 200 OK
Date: Tuesday, 22-Nov-94 18:13:45 GMT
Server: NCSA/1.3
MIME-version: 1.0
Content-type: text/html
Last-modified: Wednesday, 16-Nov-94-21:18:37 GMT
Content-length: 1115
<connection closed>

Jim Binkley
some example MIME types

<table>
<thead>
<tr>
<th>MIME type</th>
<th>viewer action</th>
</tr>
</thead>
<tbody>
<tr>
<td>text/plain</td>
<td>no formatting</td>
</tr>
<tr>
<td>text/html</td>
<td>display as HTML (.html)</td>
</tr>
<tr>
<td>application/postscript</td>
<td>fireup ps viewer (.ps)</td>
</tr>
<tr>
<td>application/powerpoint</td>
<td>fireup powerpoint (.ppt)</td>
</tr>
<tr>
<td>image/jpeg</td>
<td>jpeg image, inline display (.jpeg)</td>
</tr>
<tr>
<td>image/gif</td>
<td>gif image, inline display (.gif)</td>
</tr>
<tr>
<td>audio/basic</td>
<td>u-law format, fireup audio playback (.au)</td>
</tr>
<tr>
<td>video/mpeg</td>
<td>short “movie”, fireup mpeg player (.mpeg)</td>
</tr>
<tr>
<td>audio/x-midi</td>
<td>MIDI file format (.mid)</td>
</tr>
</tbody>
</table>

Jim Binkley
MIME type extensibility

- on server, add types to server config file, server associates file extension with MIME type
- on client, teach client about local viewer apps; e.g., windows NCSA mosaic.ini file

```
[Viewers]
TYPE10="audio/x-midi"
audio/x-midi="mplayer %ls"
```

Jim Binkley
http 1.1 - just an introduction

- RFC 2616 - fundamental redefinition of HTTP 1.0
  - 176 pages long ...
- host request header - next slide
- must support persistent connections
  - one TCP connection, many itty-bitty image files
  - not one connection per file
- good for TCP and good for the Inet
host request

- client may send:
  GET /pub/WWW/TheProject.html HTTP/1.1
  Host: www.w3.org

- (absolute URL or path) + host info (may include port)

- may help eliminate wasteful binding of IP addresses to ONE server, since this info is now available to server (not buried in stack)

- can now bind multiple names to one IP address

Jim Binkley
etc. section - a few more tricks

- proxies
- cgi-bin, quick overview
- security
- server-side scalability IS A PROBLEM
- there is no end to this ...
  - use the web to learn about the web
  - after all, WWW put the Internet on the map

Jim Binkley
http - proxy extension

- Internet-capable server can act as proxy for clients not on Internet - useful in firewall situations
- client simply sends http request with real URL (ftp/http/gopher, whatever) encapsulated in http request
- server proxies as real client to internet
- sends info back to client
- server can cache results - useful for Internet-wide efficiency
- can do gopher, http, ftp, can’t do telnet of course
proxy picture

client

(proxy
cache)

remote
Inet server

ftp get

Jim Binkley
cgi-bin: server-side extensibility

- cgi - Common Gateway Interface
- server-side app lives in /<server-path>/cgi-bin, invoked by web server.
- can be coded in C, perl, C++, shellscript, java
- needs to be able to write to stdout, read environment variables or stdin
- conventions exist (GET/POST) for passing parameters from form to cgi applet
- cgi app can send more HTML back to client as output, which may in turn have more form tags/cgi references
forms + cgi-bin apps

- one can invoke “forms” on the client-side
- forms consist of a limited set of GUI objects, text fill areas, fill-in fields, select menus, buttons,
- all expressed as HTML tags in the HTML src
- when form is complete, user “sends” via embedded URL to backend **cgi-bin** app located at http server
- server-side cgi-bin app processes form
cgi-bin architecture

client

form shown here

GET/POST + params

form output

web server

pipe
cgi-bin app

Jim Binkley
security

- end/end exists, authentication and/or encryption
- plaintext password and/or IP address authentication exist
  - not ideal for the usual reasons
- SSL offers easy server-side encryption
  - client-side authentication less-easy
  - leads to issues of Public Key Infrastructure
- beware: download of code from strangers
- privacy issues; e.g., cookies which are ASCII state stored by server at client
server-side scalability is challenging issue

◆ 1 server - 100 million clients want ONE PAGE RIGHT NOW!

◆ intranet solutions include:
  – round-robin DNS
  – NAT-like remapping of local addresses, 1 to many

◆ Internet solutions
  – try to determine “nearest” server and bounce request
    (e.g., use BGP routing info)
  – try to build large web of smart servers and clever
    rewrite/caching schemes at application layer

Jim Binkley