

Web Systems & Protocols

Info 341

1

Objectives

- What are 'components' of the web system?
- What are the types of content in the web?
- What is hypertext? Who coined the term?
- Who is credited with developing the World Wide Web? Where was it developed?
- Where was the first popular WWW browser developed? What was its name?
- What is HTTP?
- What is the specification of an HTTP request? An HTTP response?
- What are some common HTTP protocol commands?
- What are some common HTTP protocol status codes?
- What is the basic structure of an httpd.conf file?

2

The Web System

- Protocols & Standards
 - HTTP, WebDAV, SSL
 - URL, URI, URN
 - HTML, XHTML, DHTML, CSS
- Software
 - Web servers - Apache, IIS
 - Server-side code
 - CGI, PHP, JSP, ASP, Apache modules
 - Web clients - IE, Netscape, Mozilla, Opera
 - Client-side code
 - JavaScript, DHTML

3

What makes the web useful?

- News, weather, stocks, sports, etc
- World wide reach via the Internet
- Browser features- bookmarking, tabs
- Complex data via plug-ins
- Client and server side scripting
- Bill pay, shopping
- Marketing (cookies, tracking images ..)
- And a whole lot more!

4

What makes the web useful?

- Content, content, content
 - With out content, the web is really nothing
- What are the types of content?

5

What makes the web useful?

- Content, content, content
 - With out content, the web is really nothing
- What are the types of content?
 - Static - content in files (web pages)

6

[What makes the web useful?]

- Content, content, content
 - With out content, the web is really nothing
- What are the types of content?
 - Static - content in files (web pages)
 - Dynamic - content generated on the fly by the server

7

[What makes the web useful?]

- Content, content, content
 - With out content, the web is really nothing
- What are the types of content?
 - Static - content in files (web pages)
 - Dynamic - content generated on the fly by the server
 - Active - content is a script or code that changes on the client side (e.g.. PHP)

8

[Where are we heading?]

- Background
- Mini-history of hypertext & web
- HTTP in some depth
- Apache
 - Installation
 - Configuration
- Extensions to HTTP
 - WebDAV
 - SSL, STL
- Dynamic Content
 - CGI, PHP, JSP, ASP

9

A brief history of hypertext 1

- The Memex
 - Vannavar Bush's concept of a digital desk
 - Vast array of micro-filmed resources with shared annotations
 - Describe in "As we may Think"
 - Published in *Atlantic Monthly* 1945
- Introduced key concepts of relationships (links) among artifacts (nodes) as the essence of intellectual work
- It took a while for the concept to catch on and technology to catch up

10

A brief history of hypertext 2

- Ted Nelson
 - 1965 in "Literary Machines" defined the term 'hypertext' as a way to write and publish in a non-linear format.
 - Described project Xanadu
- Douglas Engelbart
 - 1968 Demo of NLS (oN Line System)
 - The first: mouse, windows, video conferencing, shared data, tele-work ...
- Explosion of hypertext research in the 1980s
- Early popular hypertext software
 - HyperCard, SuperCard
 - None of these hypertext systems were networked

11

Early history of the web 1

- The world of CERN
 - European research center for particle physics
 - In 1980 a developer took a consulting job, Tim Berners-Lee, stays 6 months (is hired back to a start-up company, printer drivers)
 - 1983 Berners-Lee applies to a CERN for a fellowship program
 - Three years has made CERN a different place
 - Different equipment everywhere, IBM, DEC, Contro-Data, and a bunch of small PCs, Mac, IBM-PC
 - Nothing really works together, information sharing among the physicists is a mess

12

Early history of the web 2

- Concern for information sharing and interoperability
- Tim Berners-Lee experimented with stuff
 - Unix
 - TCP/IP
 - RPC
 - Why was this novel?
 - CERN was late to adopt the cultural imperialist "Internet" - They were ISO protocol compliant.
- Explored documentation systems, help systems, shared file systems ...
- Nothing quite solved the problem, many solutions were proprietary

13

Early history of the web 3

- 1989 Berners-Lee writes a proposal to CERN to integrate disparate information sources, text pages, news groups
- 1990 project is shelved
- 1990 CERN buys some NeXT cubes
- 1990 Berners-Lee goes to the European Conference on Hypertext (ECHT) - Finds some like minds, but not the right system
- 1990 October begins some exploratory coding
- 1990 November
 - HTTP - has the first crude web server working
 - HTML - has a basic browser and editor working
 - URI - defined (Uniform Resource Identifier)
 - Called it WorldWideWeb
 - Credits the simplicity of using NeXTStep



14

Early history of the web 4

- 1990-1991 CERN slowly adopts Berners-Lee's system
- 1991 National Center for Supercomputing Applications (NCSA) at University of Illinois Urbana-Champaign recognize the value, but concern that the browser only runs on NeXT
- 1992 (late) Two programmers Marc Andreessen and Eric Bina decide to try and create a general browser
- 1993 March Mosaic is released
- 1993 NCSA also has a new version of the CERN server
- 1993 December Marc Andreessen graduates from college
- 1994 Browsers are the thing
 - MacWeb, WinWeb, InternetWorks, SlipKnot, Cello, NetCruiser, Lynx, WinTapestry, WebExplorer ...

15

[HTTP: Hypertext Transfer Protocol]

- HTTP two RFCs
 - HTTP/1.0 rfc1945 - May 1996 (Berners-Lee, Fielding, Frystyk)
 - HTTP/1.1 rfc2068, rfc2616 - June 1999 (Fielding, Gettys, Mogul, Frystyk, Masinter, Leach, Berners-Lee)
- Reading
 - rfc2616 sections 5, 6, & 9
 - <http://www.ietf.org/rfc/rfc2616.txt>

16

[Looking at the protocol...]

- We're going to look a little more deeply at the protocol - but first let's think about how it needs to work
- What has to happen?

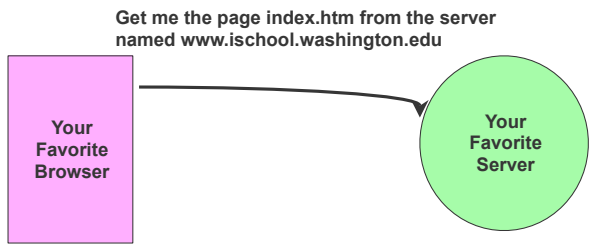
17

[Protocol Schema]



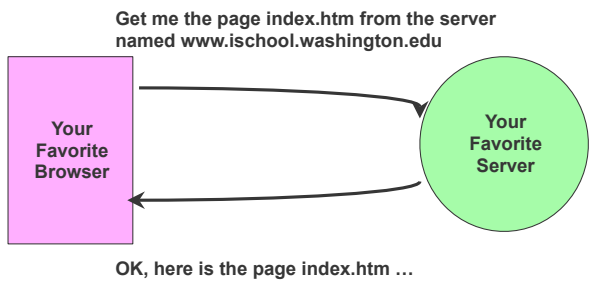
18

[Protocol Schema]



19

[Protocol Schema]



20

[Protocol Problems]

- What are the possible problems?

21

[Protocol Problems]

- What are the possible problems?
 - server crash or internal error
 - page not there or unknown
 - page access restricted
 - page not really 'mine' (moved, proxy)
 - content (page) can expire
- The protocol must specify how to handle these situations and many more

22

[Deeper look at HTTP]

- HTTP Protocol
 - HTTP Request format
 - HTTP Response format
 - Common response codes
 - HTTP Commands
- Try little proxy demo

23

[HTTP Protocol:Request]

```
Request = Request-Line
        *(( general-header
          | request-header
          | entity-header ) CRLF)
        CRLF
        [ message-body ]

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Method = "OPTIONS"
        | "GET"
        | "HEAD"
        | "POST"
        | "PUT"
        | "DELETE"
        | "TRACE"
        | "CONNECT"
        | extension-method
        extension-method = token

Request-URI = "*" | absoluteURI | abs_path | authority

HTTP-Version = "HTTP/1.0" | "HTTP/1.1"
```

24

[HTTP Protocol:Request-URI]

Request-URI = "*" | **absoluteURI** | **abs_path** | **authority**

URIs have been known by many names: WWW addresses, Universal Document Identifiers, Universal Resource Identifiers (URI), and finally the combination of Uniform Resource Locators (URL) and Names (URN).

As far as HTTP is concerned, Uniform Resource Identifiers are simply formatted strings which identify--via name, location, or any other characteristic--a resource.

You should think of URIs in a form that you probably know:

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

`http://www.ischool.washington.edu/mcdonald/`

25

[HTTP Protocol:Request Header]

```
request-header = Accept
                | Accept-Charset
                | Accept-Encoding
                | Accept-Language
                | Authorization
                | Expect
                | From
                | Host
                | If-Match
                | If-Modified-Since
                | If-None-Match
                | If-Range
                | If-Unmodified-Since
                | Max-Forwards
                | Proxy-Authorization
                | Range
                | Referer
                | TE
                | User-Agent
```

26

[HTTP Protocol:Response]

```
Response = Status-Line
          *(( general-header
            | response-header
            | entity-header ) CRLF)
          CRLF
          [ message-body ]
```

Status-Line = **HTTP-Version** SP **Status-Code** SP **Reason-Phrase** CRLF

27

[HTTP Protocol:Status Codes]

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

- 1xx: Informational - Request received, continuing process
- 2xx: Success - The action was successfully received, understood, and accepted
- 3xx: Redirection - Further action must be taken in order to complete the request
- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
- 5xx: Server Error - The server failed to fulfill an apparently valid request

28

[HTTP Protocol:Status Codes]

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

```
Status-Code =  
| "200" ; OK  
| "201" ; Created  
| "202" ; Accepted  
| "203" ; Non-Authoritative Information  
| "204" ; No Content  
| "301" ; Moved Permanently  
| "307" ; Temporary Redirect  
| "400" ; Bad Request  
| "401" ; Unauthorized  
| "402" ; Payment Required  
| "403" ; Forbidden  
| "404" ; Not Found  
| "500" ; Internal Server Error  
| "501" ; Not Implemented  
| "502" ; Bad Gateway  
| extension-code  
extension-code = 3DIGIT
```

29

[HTTP Protocol:Commands]

- OPTIONS
- GET
- HEAD
- POST
- PUT
- DELETE
- TRACE
- CONNECT

30

HTTP Protocol:Commands

OPTIONS

- Request for information on the communication options available
- Version compliance HTTP/1.0 or HTTP/1.1
- Deal with proxy forwarding
- Close or Keep-Alive connections

31

HTTP Protocol:Commands

GET

- Get the entity specified by the Request-URI
- Modifiable by the Request-header options:

If-Match
If-Modified-Since
If-None-Match
If-Unmodified-Since

HEAD

- Semantically identical to GET but only sends the response header

32

HTTP Protocol:Commands

POST

- Requests that the server handle the sent entity as a subordinate to the Request-URI
 - In general that Request-URI is going to be some kind of active entity (e.g. a script)
 - Annotate a URI
 - Post a message, say to a newsgroup
 - Provide a block of data, say submitting a form

PUT

- Requests that the server store the sent entity under the Request-URI
 - Often used to upload a file to a server
 - The semantics of this are server dependent

33

[HTTP Protocol:Commands]

- DELETE
 - Specifies that the Request-URI be deleted
 - Human interaction can override this request
 - A server should not respond with “200 OK” if it does not intend to delete the item
 - Can use “202 Accepted” or “204 No content” instead
 - Often useful for deleting items from a proxy cache

34

[HTTP Protocol:Commands]

- TRACE
 - Implements a loopback so that clients can see what is being received by a server
 - Useful for testing
- CONNECT
 - Reserved - not implemented
 - Idea is to use this for switching between regular and tunneled or proxied connections
 - Possible use for SSL connections

35

[Where is the protocol? (*)]

36

[Where is the protocol?]

- Browsers (clients)
 - Firefox
 - Safari
 - Netscape
 - Mozilla
 - Opera
 - iCab
 - Lynx
 - Internet Explorer ...

37

[Where is the protocol?]

- Browsers (clients)
- Proxies
- Web Crawlers
- "bots"
 - `wget -r http://www.site.com`
 - curl (another CLI like wget)
 - sitesucker (GUI version of wget)

38

[Where is the protocol?]

- Browsers (clients)
- Proxies
- Web Crawlers
- Servers - Apache, IIS
 - You already have experience with IIS
 - Let's look a little at Apache

39

Apache: a short history

- 1993-1994 NCSA actively develops a version of the CERN web server (httpd)
- 1994 (mid-year) Rob McCool, NCSA's main httpd developer leaves project stops
- 1995 February - NCSA httpd server was the most popular in the world
- 1995 February - Small number of webmasters share patches in email (known bug fixes, popular features)
- 1995 April - First public release (0.6.2)
- 1995 June - New architecture, significant rewrite
- 1995 December - Apache 1.0 released
- 1995-2002 Numerous releases (1.3.24 is current)
- 2001 Apache Foundation begins work on Apache 2.0
- 2006 Apache web server 2.2.0 available

40

Making an HTTP server work

- Early NCSA Philosophy
 - Anyone can do this... Configuration was simple, worked out of the box
- Current Apache Philosophy
 - 'We don't want just anybody setting up a server. If you can't figure it out you don't have any business setting it up...'

41

Apache Configuration

- The httpd.conf file
 - Basic sections
 - server identification
 - file locations
 - process creation
 - network configuration
 - resource utilization
 - log files
 - Set Global Environment
 - Set 'Main' Server
 - Set Virtual Hosts

42

Global Environment

```
ServerType standalone
ServerRoot "/usr/local/apache_1.3.24"
# Timeout: The number of seconds before receives and sends time out.
Timeout 300
# KeepAlive: Whether or not to allow persistent connections (more than
# one request per connection). Set to "Off" to deactivate.
KeepAlive On
MaxKeepAliveRequests 100
MinSpareServers 5
MaxSpareServers 10
# Modules for the server
# Example: LoadModule foo_module libexec/mod_foo.so
LoadModule ssl_module modules/libssl.so
LoadModule env_module modules/mod_env.so
# + many more modules
#
# numerous other options
#
```

43

Main Server

```
# Port: The port to which the standalone server listens. For
# ports < 1023, you will need httpd to be run as root initially.
Port 2086

# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.
#User nobody
#Group nobody

# ServerAdmin: Your address, where problems with the server should be
# e-mailed. This address appears on some server-generated pages, such
# as error documents.
ServerAdmin dwmc@u.washington.edu

# ServerName allows you to set a host name which is sent back to
# client for your server if it's different than the one the program
# would get (i.e., use "www" instead of the host's real name).
ServerName mead02.u.washington.edu:2086
```

44

Main Server

```
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory,
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "~/dwmc/apache_1.3.24/htdocs"

# Each directory to which Apache has access, can be configured with
# respect to which services and features are allowed and/or disabled in
# that directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of
# permissions.
#
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
```

45

Main Server

```
# This should be changed to whatever you set DocumentRoot to.
#
<Directory "~/dwmc/apache_1.3.24/htdocs">
# This may also be "None", "All", or any combination of "Indexes",
# "Includes", "FollowSymLinks", "ExecCGI", or "MultiViews".
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
    Options Includes Indexes ExecCGI FollowSymLinks MultiViews
# This controls which options the .htaccess files in directories can
# override. Can also be "All", or any combination of "Options",
# "FileInfo", "AuthConfig", and "Limit"
    AllowOverride None
# Controls who can get stuff from this server.
    Order allow,deny
    Allow from all
</Directory>
```

46

Virtual Host Configuration

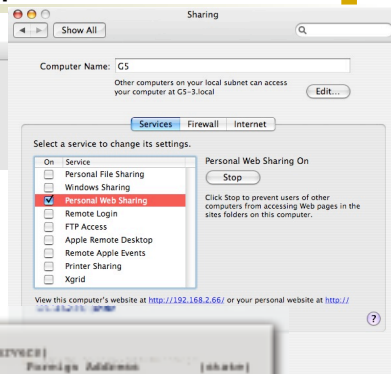
```
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost container.
# The first VirtualHost section is used for requests without a known
# server name.
#
<VirtualHost *>
    ServerAdmin dwmc@u.washington.edu
    DocumentRoot "~/dwmc/apache_1.3.24/htdocs2"
    ServerName bogus.server.com

    ErrorLog logs/dummy-host.example.com-error_log
    CustomLog logs/dummy-host.example.com-access_log common
</VirtualHost>
```

47

Monitoring httpd

```
Terminal — bash (tty5) — 50x7 — 962
G5-3:~ johnb$ ps ax | grep http
18312 ?? Ss  0:00.15 /usr/sbin/httpd
18313 ?? S   0:00.01 /usr/sbin/httpd
18319 p5 S+  0:00.01 grep http
G5-3:~ johnb$
```



```
johnb@G5 ~ % ss -tln
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address          Foreign Address         (state)
tcp4      0      0 *.http                 *.*                      LISTEN
```

48