Application Layer

Chapter 7

The World Wide Web

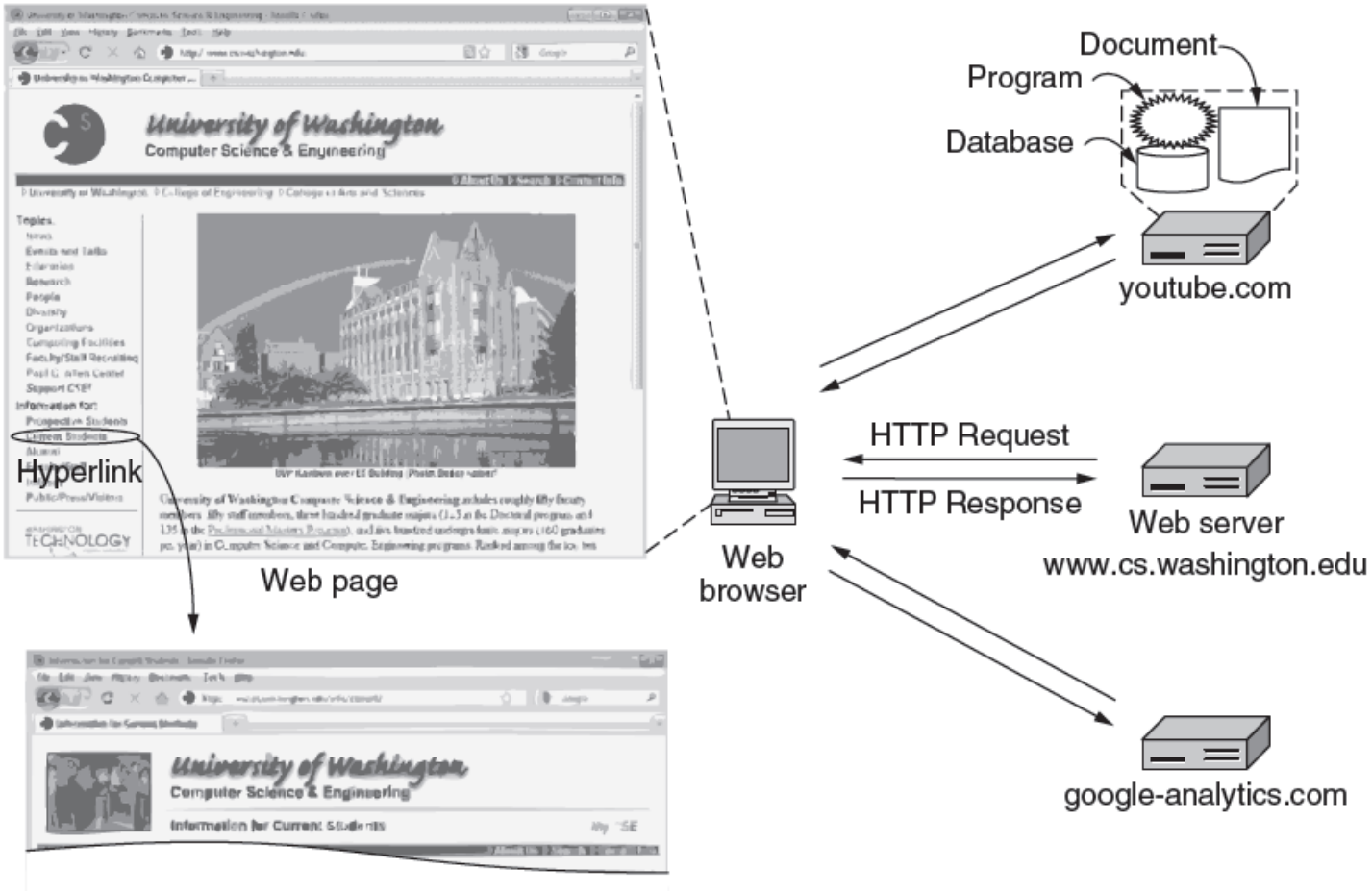
- Architectural overview »
- Static Web pages »
- Dynamic pages and Web applications »
- HTTP – HyperText Transfer Protocol »
- The mobile Web »
- Web search »

History

- 1945 – Vannevar Bush MIT.
- 1989 - Tim Berners-Lee (CERN)
- 1993 – Andeeseen Mosaic -> Netscape
- 1994-1997 – Browser War
- 1994 – Word Wide Web Consortium (W3C)
- WWW is NOT Internet
- The highest rate of growth

Architectural Overview (1)

HTTP transfers pages from servers to browsers



Architectural Overview (2)

What´s the name ? Where is it? How can access?

Pages are named with URLs (Uniform Resource Locators)

- Example: http://www.phdcomics.com/comics.php

Our focus

Name	Used for	Example
http	Hypertext (HTML)	http://www.ee.uwa.edu/~rob/
https	Hypertext with security	https://www.bank.com/accounts/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:///usr/suzanne/prog.c
mailto	Sending email	mailto:JohnUser@acm.org
rtsp	Streaming media	rtsp://youtube.com/montypython.mpg
sip	Multimedia calls	sip:eve@adversary.com
about	Browser information	about:plugins

Common URL protocols

Architectural Overview (3)

Steps a client (browser) takes to follow a hyperlink:

- Determine the protocol (HTTP)
- Ask DNS for the IP address of server
- Make a TCP connection to server
- Send request for the page; server sends it back
- Fetch other URLs as needed to display the page
- Close idle TCP connections

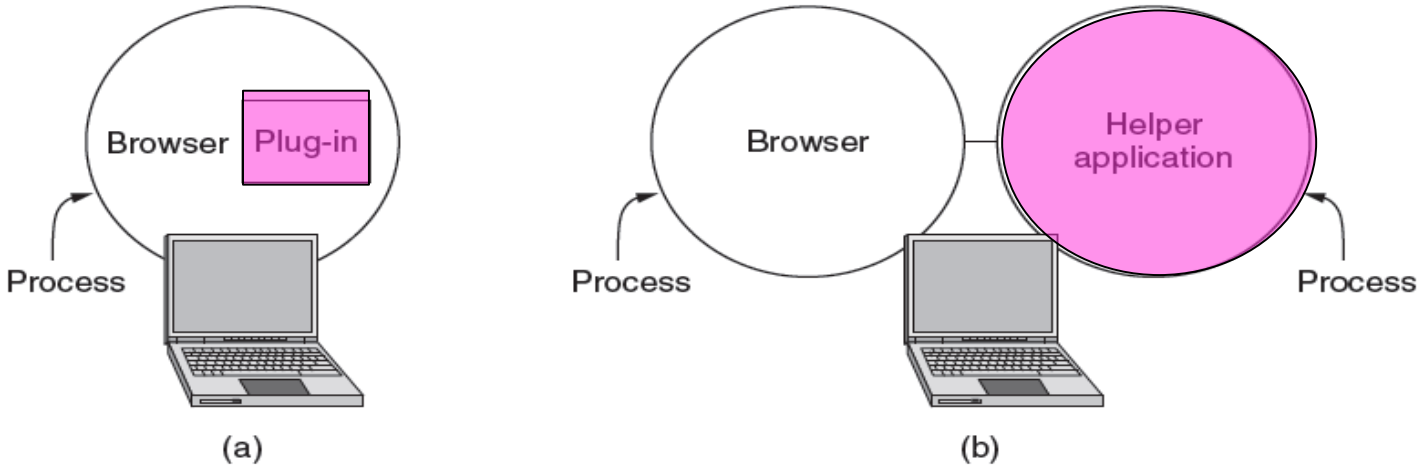
Steps a server takes to serve pages:

- Accept a TCP connection from client
- Get page request and map it to a resource (e.g., file name)
- Get the resource (e.g., file from disk)
- Send contents of the resource to the client.
- Release idle TCP connections

Architectural Overview (4)

Content type is identified by MIME types

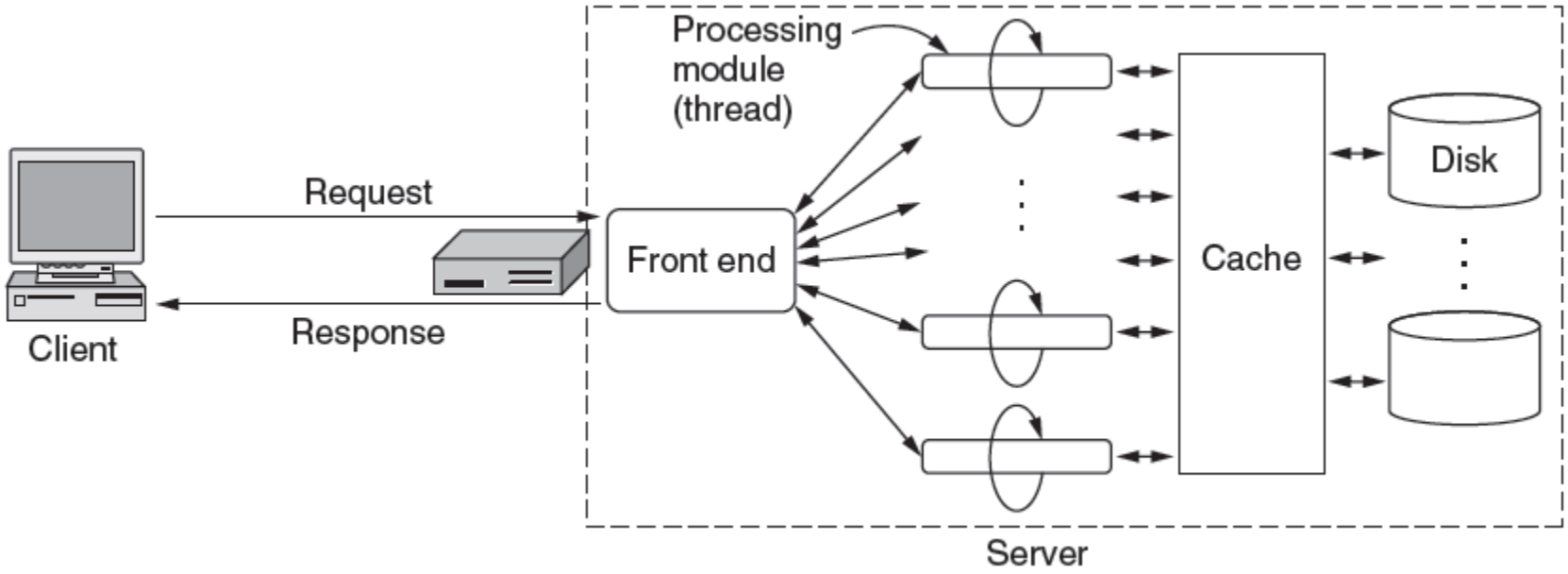
- Browser takes the appropriate action to display
- Plug-ins / helper apps extend browser for new types



Architectural Overview (5)

To scale performance, Web servers can use:

- Caching, multiple threads, and a front end



Architectural Overview (6)

Server steps, revisited:

- Resolve name of Web page requested
- Perform access control on the Web page
- Check the cache
- Fetch requested page from disk or run program
- Determine the rest of the response
- Return the response to the client
- Make an entry in the server log

Architectural Overview (7)

No concept of Session ! Ip ?

Cookies support stateful client/server interactions

- Server sends cookies (state) with page response
- Client stores cookies across page fetches
- Client sends cookies back to server with requests

Domain	Path	Content	Expires	Secure
toms-casino.com	/	CustomerID=297793521	15-10-10 17:00	Yes
jills-store.com	/	Cart=1-00501;1-07031;2-13721	11-1-11 14:22	No
aportal.com	/	Prefs=Stk:CSCO+ORCL;Spt:Jets	31-12-20 23:59	No
sneaky.com	/	UserID=4627239101	31-12-19 23:59	No

Examples of cookies

Static Web Pages (1)

Static Web pages are simply files

- Have the same contents for each viewing

Can be visually rich and interactive nonetheless:

- HTML that mixes text and images
- Forms that gather user input
- Style sheets that tailor presentation
- Vector graphics, videos, and more (over) . . .

Static Web Pages (2)

Progression of features through HTML 5.0

Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hyperlinks	x	x	x	x	x
Images	x	x	x	x	x
Lists	x	x	x	x	x
Active maps & images		x	x	x	x
Forms		x	x	x	x
Equations			x	x	x
Toolbars			x	x	x
Tables			x	x	x
Accessibility features				x	x
Object embedding				x	x
Style sheets				x	x
Scripting				x	x
Video and audio					x
Inline vector graphics					x
XML representation					x
Background threads					x
Browser storage					x
Drawing canvas					x

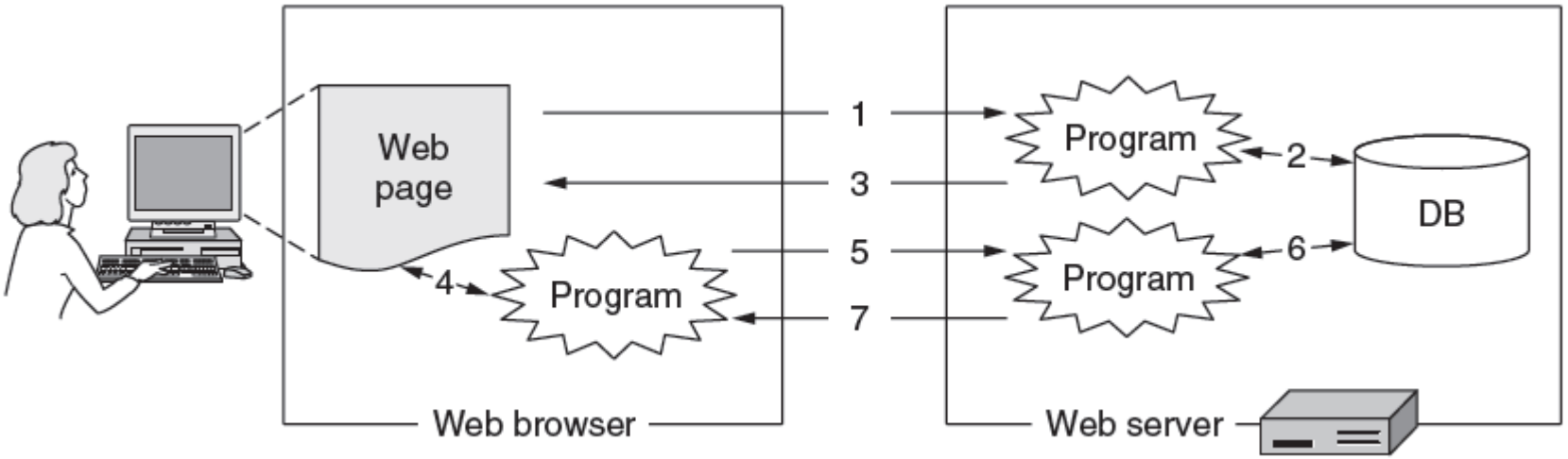
Dynamic Pages & Web Applications (1)

New application software -> run inside the browser

No needs to install new application, access from different places.

Dynamic pages are generated by programs running at the server (with a database) and the client

- E.g., PHP at server, JavaScript at client
- Pages vary each time like using an application



Dynamic Pages & Web Applications (2)

Web page that gets form input and calls a server program

```
<html>
<body>
<form action="action.php" method="post">
<p> Please enter your name: <input type="text" name="name"> </p>
<p> Please enter your age: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

(a)

PHP server program that creates a custom Web page

```
<html>
<body>
<h1> Reply: </h1>
Hello <?php echo $name; ?>.
Prediction: next year you will be <?php echo $age + 1; ?>
</body>
</html>
```

PHP calls

(b)

Resulting Web page (for inputs “Barbara” and “32”)

```
<html>
<body>
<h1> Reply: </h1>
Hello Barbara.
Prediction: next year you will be 33
</body>
</html>
```

Dynamic Pages & Web Applications (3)

JavaScript program produces result page in the browser

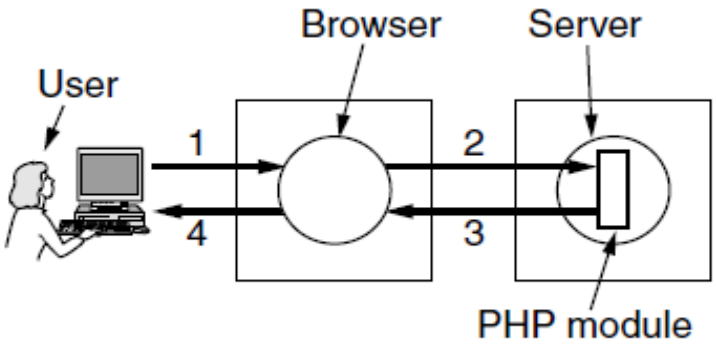
```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
   var person = test_form.name.value;
   var years = eval(test_form.age.value) + 1;
   document.open();
   document.writeln("<html> <body>");
   document.writeln("Hello " + person + ".<br>");
   document.writeln("Prediction: next year you will be " + years + ".");
   document.writeln("</body> </html>");
   document.close();
}
</script>
</head>
```

First page with form, gets input and calls program above

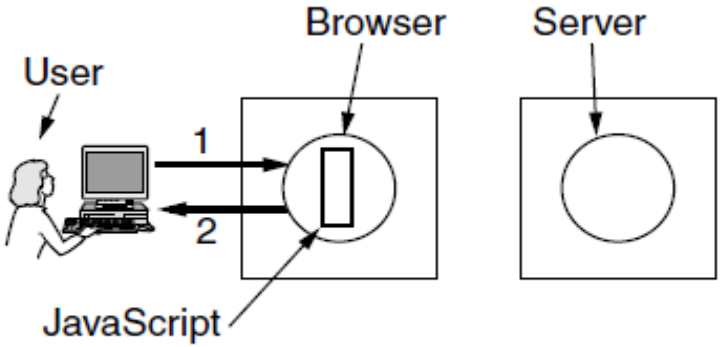
```
<body>
<form>
Please enter your name: <input type="text" name="name">
<p>
Please enter your age: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```

Dynamic Pages & Web Applications (4)

The difference between server and client programs



Server-side scripting with
PHP/CGI-BIN/JSP/ASP/.NET



Client-side scripting with
JavaScript/Applets/VBScripts/ActiveX

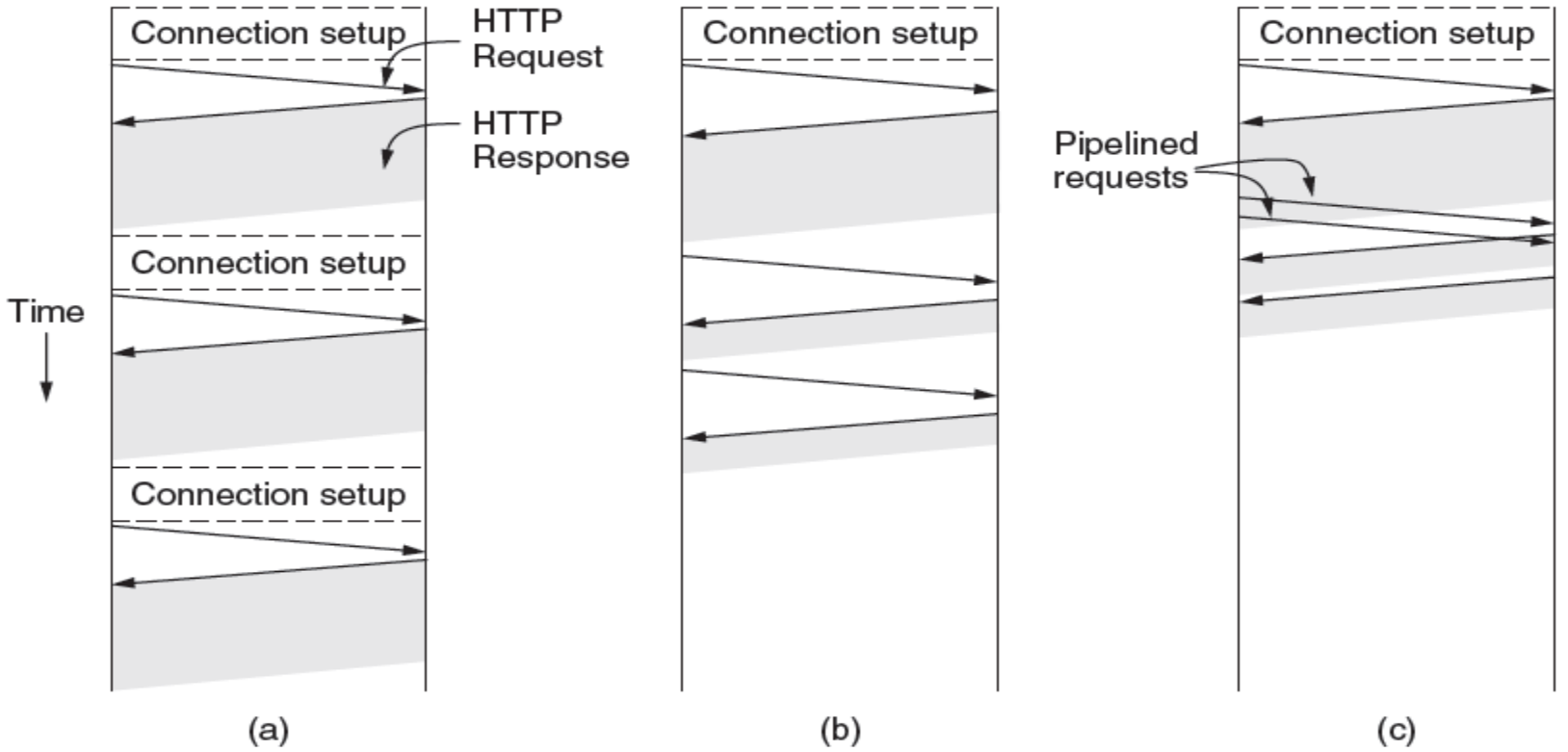
HTTP (1)

HTTP (HyperText Transfer Protocol) is a request-response protocol that runs on top of TCP

- Fetches pages from server to client
- Server usually runs on port 80
- Headers are given in readable ASCII
- Content is described with MIME types
- Protocol has support for pipelining requests
- Protocol has support for caching

HTTP (2)

HTTP uses persistent connections to improve performance



One connection for each request

Sequential requests on one connection

Pipelined requests on one connection

HTTP (3)

HTTP has several request methods.

Fetch a page →

Used to send input data to a server program →

Method	Description
GET	Read a Web page
HEAD	Read a Web page's header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTIONS	Query options for a page

HTTP (4)

Response codes tell the client how the request fared:

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

HTTP (5)

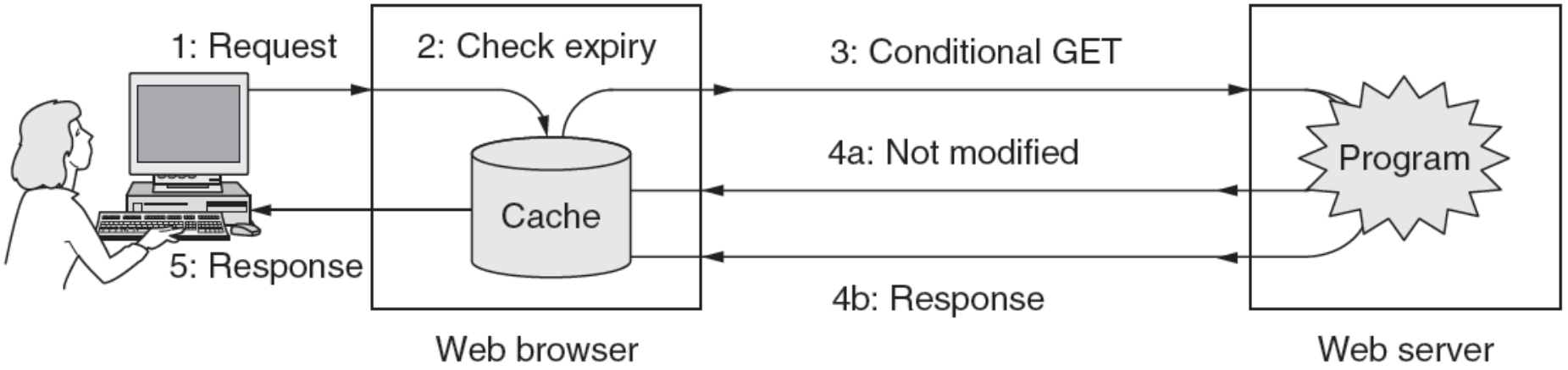
Many headers carry key information:

Function	Example Headers
Browser capabilities (client □ server)	User-Agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language
Caching related (mixed directions)	If-Modified-Since, If-None-Match, Date, Last-Modified, Expires, Cache-Control, ETag
Browser context (client □ server)	Cookie, Referer, Authorization, Host
Content delivery (server □ client)	Content-Encoding, Content-Length, Content-Type, Content-Language, Content-Range, Set-Cookie

HTTP (6)

HTTP caching checks to see if the browser has a known fresh copy, and if not if the server has updated the page

- Uses a collection of headers for the checks
- Can include further levels of caching (e.g., proxy)



The Mobile Web

Mobiles (phones, tablets) are challenging as clients:

- Relatively small screens
- Limited input capabilities, lengthy input.
- Network bandwidth is limited
- Connectivity may be intermittent.
- Computing power is limited

Strategies to handle them:

- Content: servers provide mobile-friendly versions; transcoding can also be used
- Protocols: no real need for specialized protocols; HTTP with header compression sufficient

Web Search

Search has proved hugely popular, in tandem with advertising that has proved hugely profitable

- A simple interface for users to navigate the Web

Search engine requires:

- Content from all sites, accessed by crawling. Follow links to new pages, but beware programs.
- Indexing, which benefits from known and discovered structure (such as XML) to increase relevance