# Computer Systems Principles

Web Services

UMASSCS
SCHOOL OF COMPUTER SCIENCE

# Learning Objectives

- What do we mean by the “Web”
- HTTP an application Layer protocol
- Understand web terminology: URLs, Cookies, caches and proxies
- Understand the basic workings of the HTTP protocol

# What is the Web?

- The Web is
  - "an information space with resources identified by global identifiers URIs (Uniform Resource Identifiers)" - http://www.w3.org/

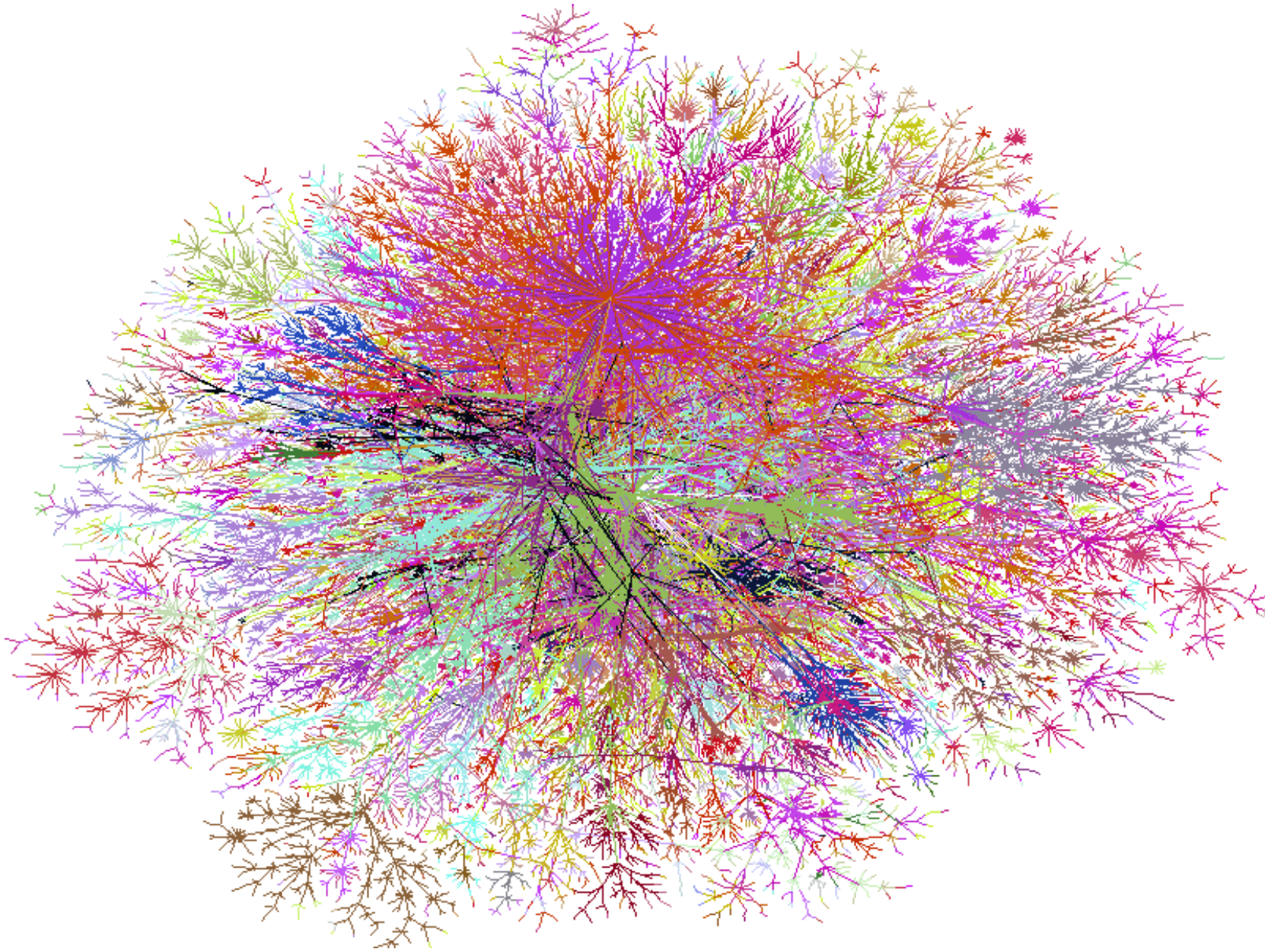
- How is it related to the Internet?
  - "the Internet is a network of networks, defined by the TCP/IP standards"

DrAfter123/Vetta/Getty Images

# Web History

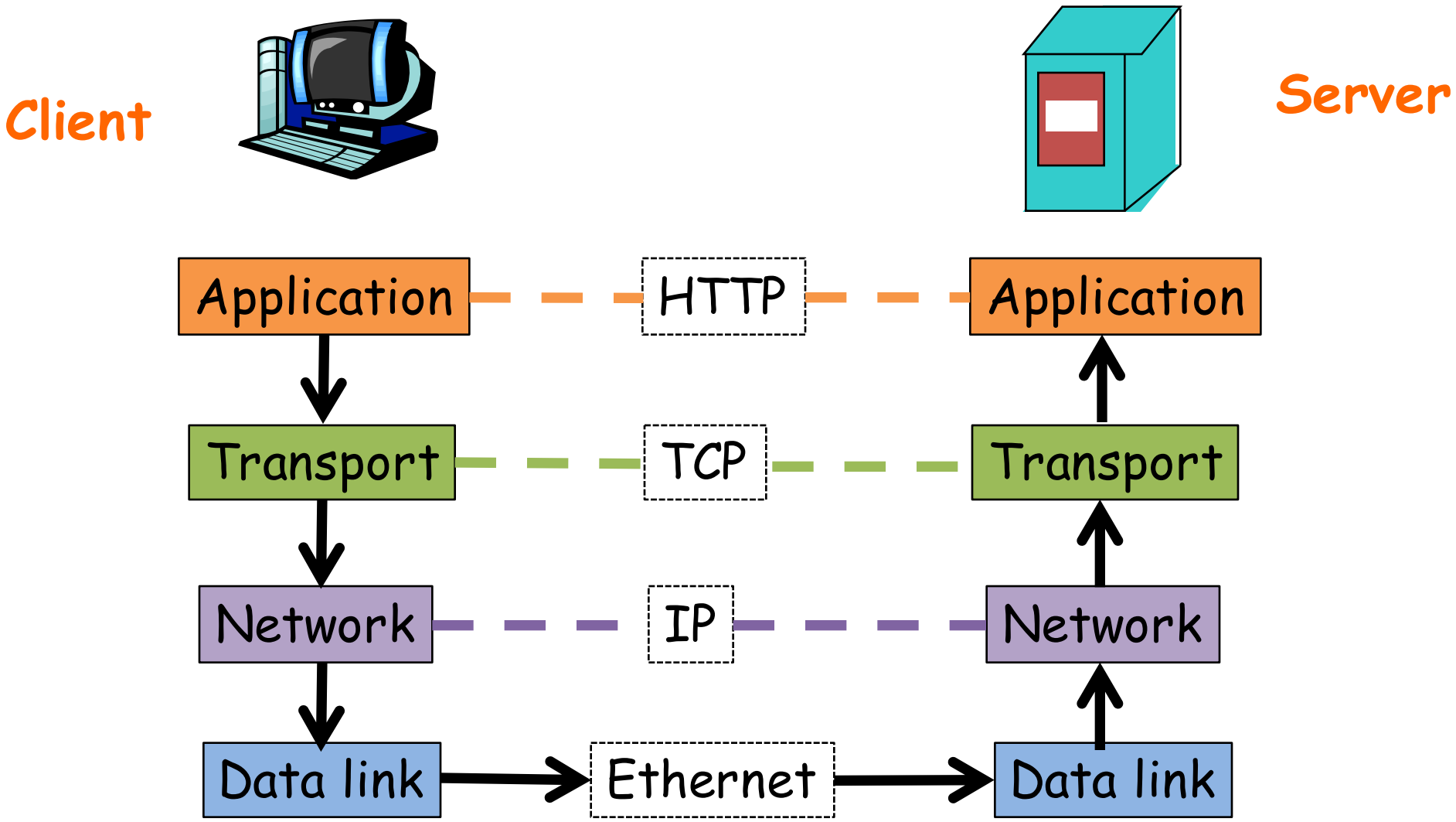
- 1989
  - Tim Berners-Lee (CERN) writes internal proposal to develop a distributed hypertext system
  - Connects “a web of notes with links”
- 1990-92
  - Tim Berners-Lee writes a graphical browser for Next machines
  - NCSA server released
- 1993-94
  - Marc Andreessen releases first version of NCSA Mosaic Browser
  - Andreessen founded “Mosaic Communications Corp” (predecessor to Netscape)

# Internet



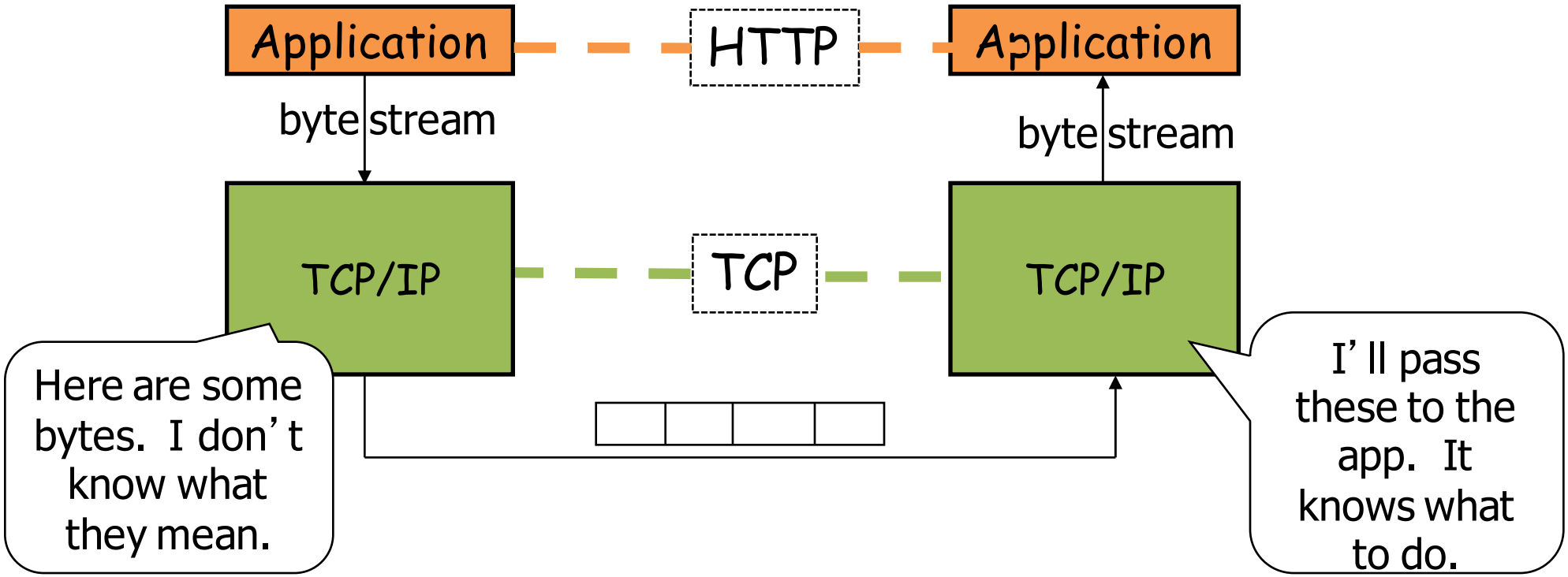
https://www.flickr.com/photos/jurvetson

# How do Applications communicate?



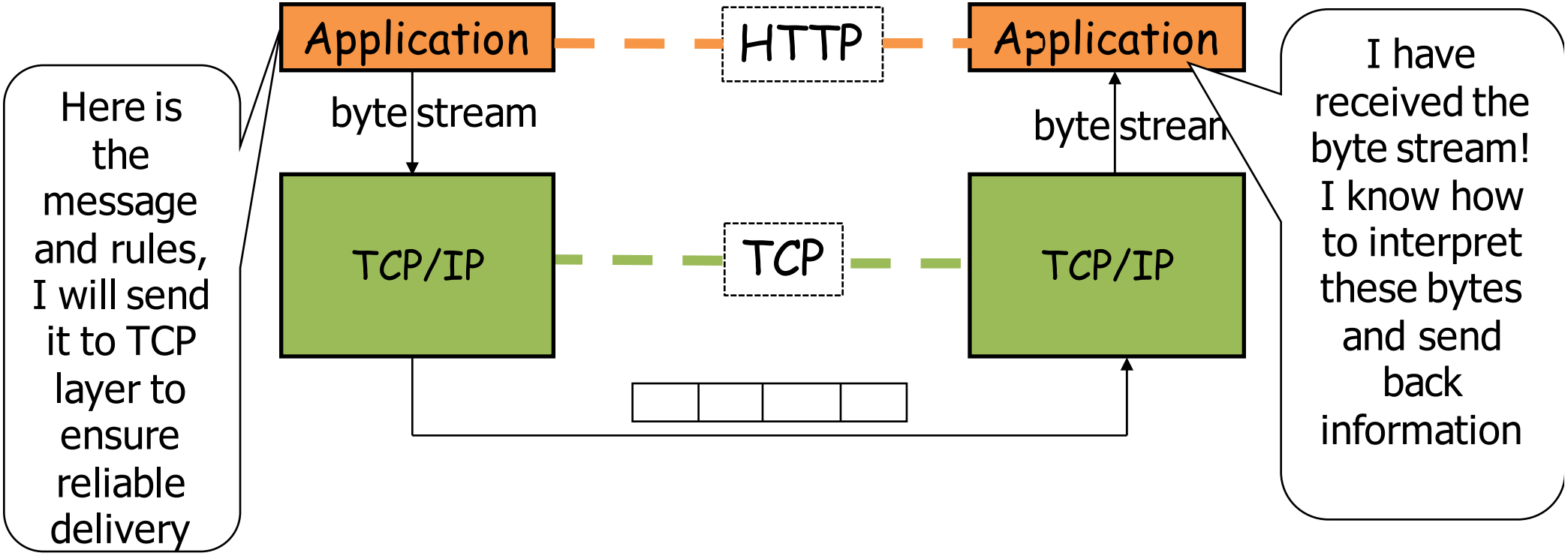
# Application Layer Protocols

- Network processes communicate – sending messages into sockets.
  - How are these messages structured?
  - Meaning of various message fields?
  - When do processes send messages?
- An Application layer protocol defines how application's processes, running on different end systems, pass messages to each other



# Application Layer Protocols

- An Application layer protocol defines how application's processes, running on different end systems, pass messages to each other.
  - types of messages: request, response
  - syntax: field delineation,
  - semantics: meaning of information in each filed
  - rules for sending and responding to messages

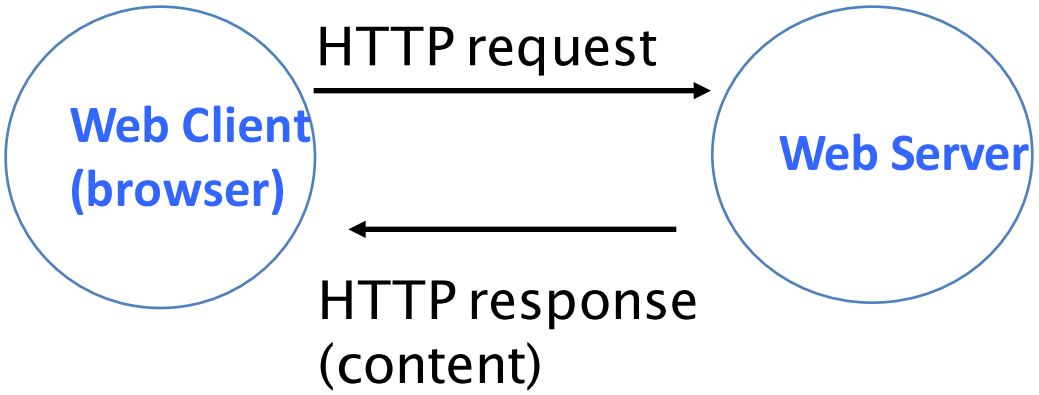


# Application Layer Protocols

- Application layer protocols in RFCs are public domain.
  - The Web's application-layer protocol is HTTP (Hyper Text Transfer Protocol).
  - If a Web browser and a Web server follow the rules of HTTP they can talk to each other (send/retrieve web pages).
- What are some others?
  - Email application: SMTP (pass messages, mailboxes, mail client)
  - Directory Service for the Internet: DNS (network name to network address translation)

# Web Servers and Web Clients

- Clients and servers communicate using the HyperText Transfer Protocol (HTTP)
  - Client and server establish TCP connection
  - Client requests content
  - Server responds with requested content
  - Client and server (may) close connection




- Current version is HTTP/1.1 (RFC 2616, June, 1999)

# Web Servers and Web Clients

- Web browsers implement the client side of HTTP
  - use browser and client interchangeably

- Web servers implement the server side of HTTP
  - house web objects addressable by a URL
  - popular web servers: Apache, Microsoft Internet Information Server


# iClicker Question

Internet Explorer, Firefox, Chrome, and Safari all implement the same protocol for transferring web content:

A) True

B) False

True: HTTP = HyperText Transfer Protocol

# Another iClicker Question

Apache and Microsoft Internet Information Server implement the same server protocol for offering web content to clients

A) True

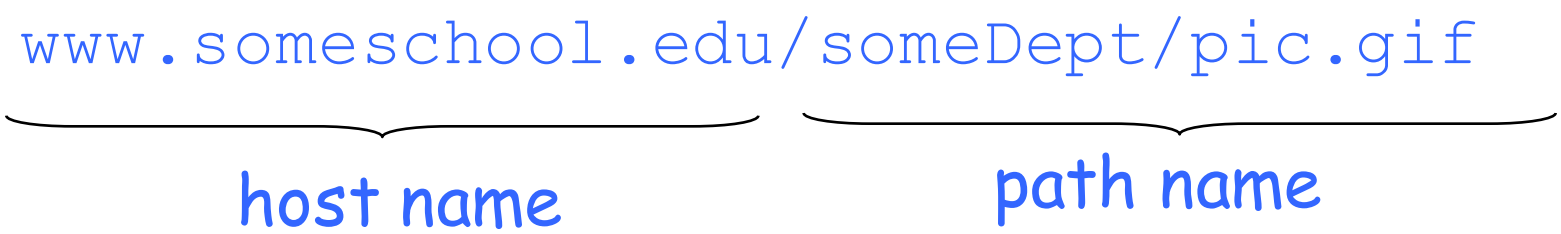
B) False

True: HTTP = HyperText Transfer Protocol

# Web Terminology

First some jargon

- Web page consists of objects
- Object can be HTML file, JPEG image, Java applet, audio file,…
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL (Uniform Resource Locator)
- Example URL:



# Web Content

- Web servers return content to clients content:
  - a sequence of bytes with an associated MIME (Multipurpose Internet Mail Extensions) type
- Example MIME types
  - text/html HTML document
  - text/plain Unformatted text
  - application/postscript Postcript document
  - image/gif Binary image (GIF format)
  - image/jpeg Binary image (JPEG format)

# Concerning HTML

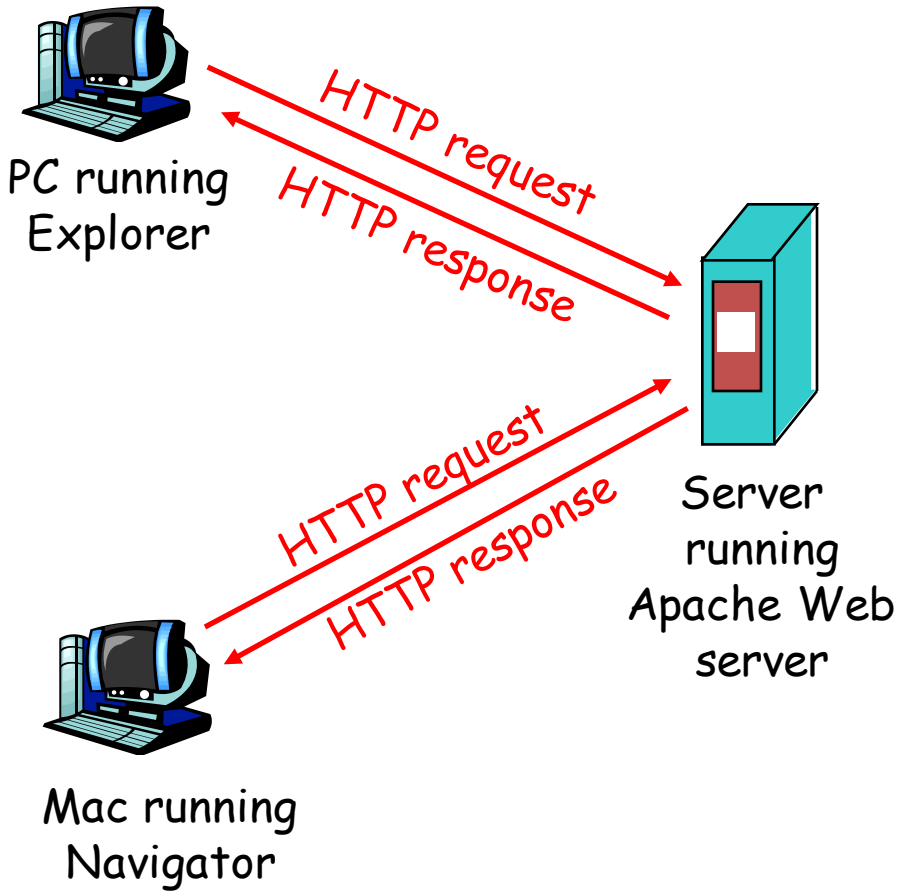
HTML = HyperText Markup Language

- “Markup” generically means formatting specifications such as font, type style, indentation, …
- Includes provisions for referring to other objects
- Can interact with executable scripts, etc.
- Continues to evolve substantially
- HTTP, in contrast, does not need to change much over time – it’s only about getting at content

# HTTP overview

## HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - client: browser that requests, receives, "displays" Web objects
  - server: Web server sends objects in response to requests



# HTTP overview

## HTTP Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

## HTTP is "stateless"

- server maintains no information about past client requests

**Protocols that maintain "state" are complex!**

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled

# HTTP connections

## Nonpersistent HTTP

- At most one object is sent over a TCP connection.

## Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server.

# Uploading form input

Post method:

- Web page often includes form input
- Input is uploaded to server in entity body

URL method:

- Uses GET method
- Input is uploaded in URL field of request line:

**www.somesite.com/animalsearch?monkeys&banana**

# URL (Uniform Resource Locator)

- Each file managed by a server has a unique name called a URL (Universal Resource Locator)
- URLs for static content:
  - https://www.cs.umass.edu:80/ugrad-education/courses
  - https://www.cs.umass.edu/ugrad-education/courses
    - Identifies a file, managed by a Web server at www.cs.umass.edu that is listening on port 80
- URLs for dynamic content:
  - http://www.jacelridge.com:3000/calculator/adder?x=15000&y=213
    - Identifies an executable file called calculator, managed by a Web server at jacelridge.com that is listening on port 3000, and is given a path of adder and two argument strings: x=15000 and y=213

# How Clients and Servers Use URLs

- Example URL: http://www.google.com:80/index.html
- Clients use prefix (http://www.google.com:80) to infer:
  - What kind of server to contact (http (Web) server)
  - Where the server is (www.google.com)
  - What port the server is listening on (80)
- Servers use suffix (/index.html) to:
  - Determine if request is for static or dynamic content
    - No hard and fast rules for this
  - Find file on file system
    - Initial "/" in suffix denotes home directory for requested content
    - Minimal suffix is "/", which servers expand to some default home page (e.g., index.html)

# Static and Dynamic Content

The content returned in HTTP responses can be either static or dynamic

- Static content: content stored in files and retrieved in response to an HTTP request
  - Examples: HTML files, images, audio clips
- Dynamic content: content produced on-the-fly in response to an HTTP request
  - Example: content produced by a program executed by the server on behalf of the client (i.e., search results)

# HTTP Method types

HTTP/1.1

- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field

HTTP/1.1

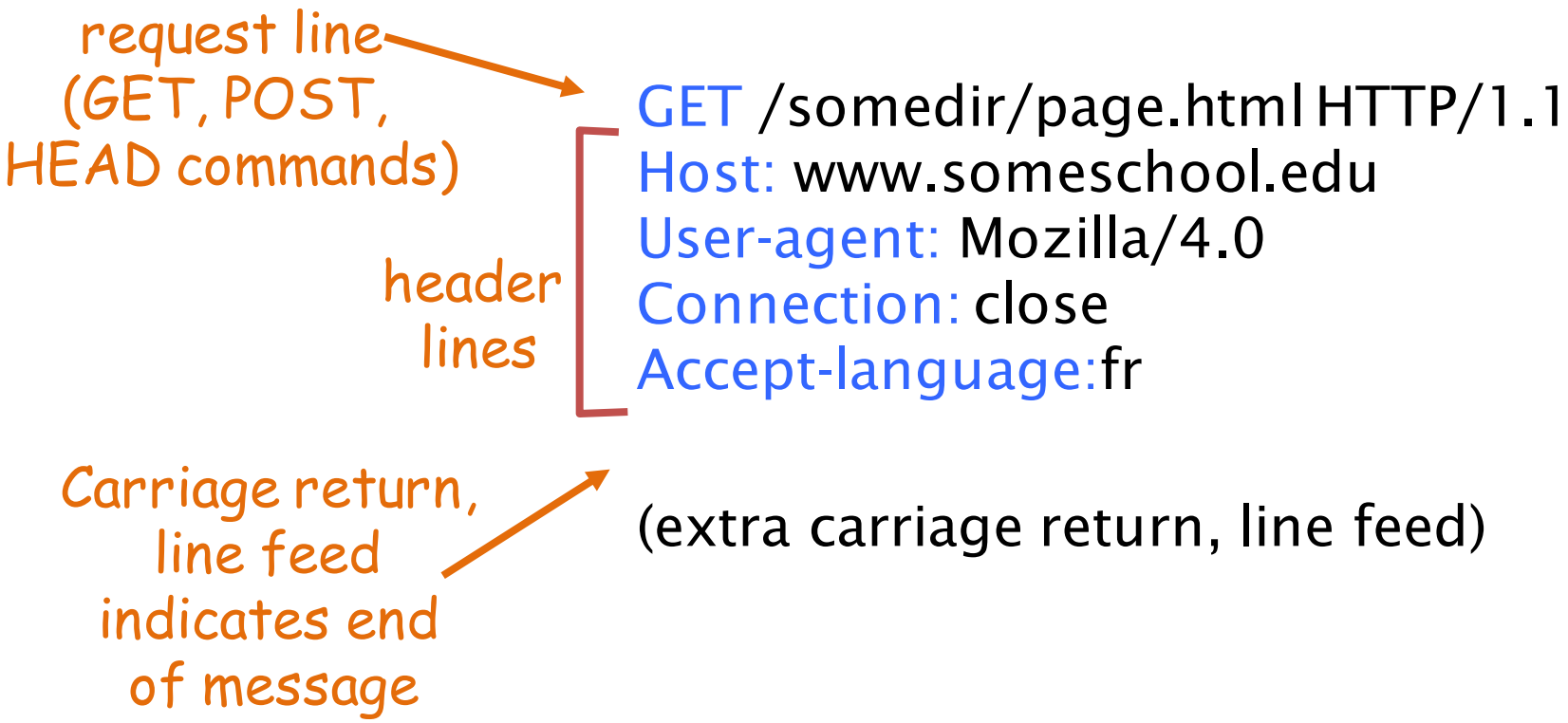
- TRACE
  - Echoes request msg from server
- OPTIONS
  - Returns HTTP methods that the server supports
- CONNECT
  - TCP/IP tunnel for HTTP

# HTTP Requests

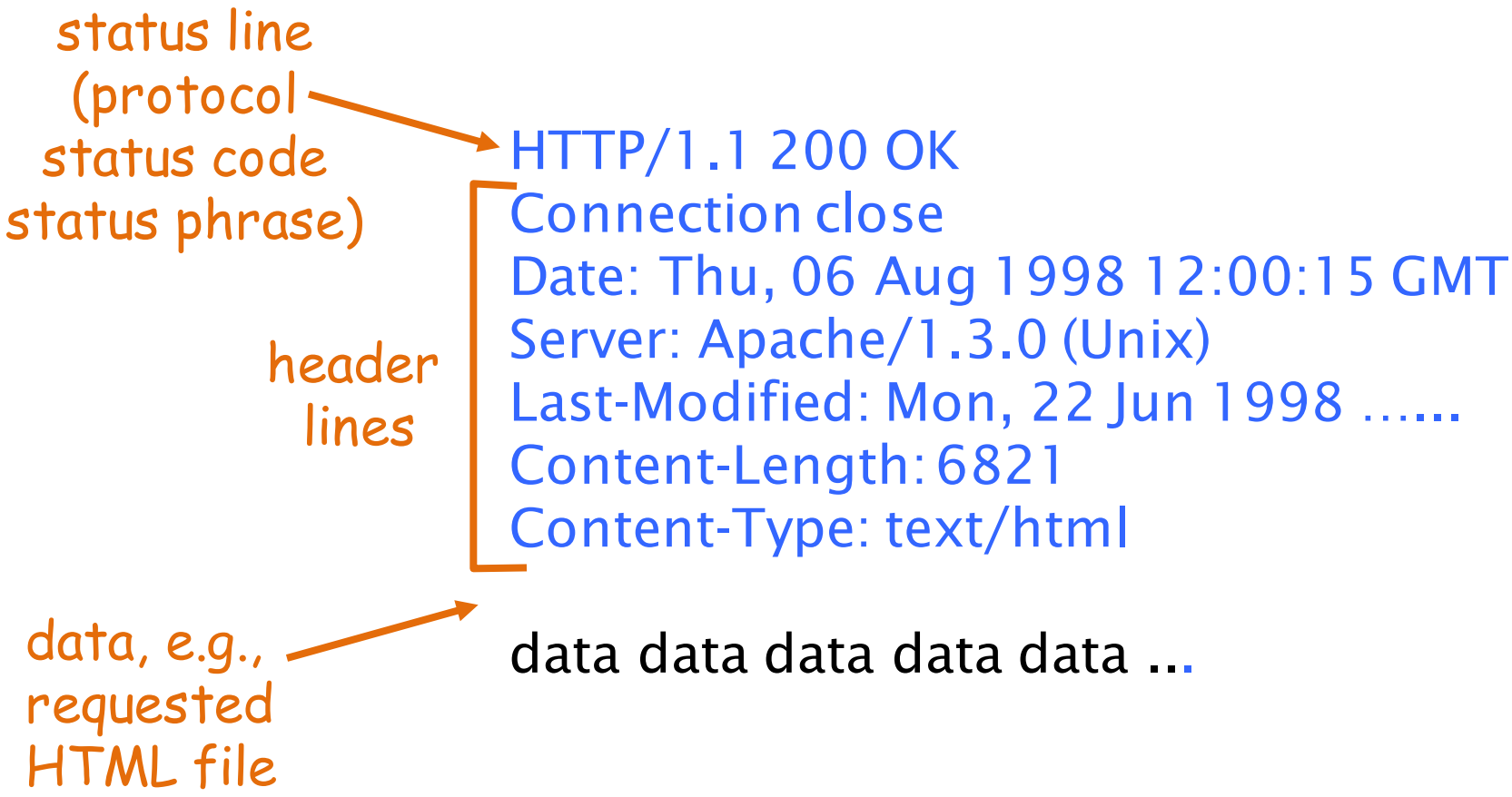
- HTTP request is a request line, followed by zero or more request headers
- Request line: <method> <uri> <version>
  - <method> : GET, POST, OPTIONS, HEAD, PUT, DELETE, or TRACE
  - <uri> : URL for proxies, URL suffix for servers
  - <version> : HTTP/1.0 or HTTP/1.1
- HTTP/1.1 and HTTP/1.0
  - HTTP/1.1 : supports persistent connections
    - Multiple transactions over the same connection
    - Connection: Keep-Alive
  - HTTP/1.1 requires HOST header
    - Host: www.yahoo.com
  - HTTP/1.1 adds additional support for caching

# HTTP request message

- two types of HTTP messages: request, response
- HTTP request message:
  - ASCII (human-readable format)



# HTTP response message



# HTTP response status codes

In first line in server->client response message.
A few sample codes:

**200 OK**

- request succeeded, requested object later in this message

**301 Moved Permanently**

- requested object moved, new location specified later in this message (Location:)

**400 Bad Request**

- request message not understood by server

**404 Not Found**

- requested document not found on this server

**505 HTTP Version Not Supported**

# Telnet

- “Telnet is a network protocol used on the Internet or local area networks to provide a bidirectional interactive text-oriented communication facility using a virtual terminal connection.”
- Client-server protocol on the TCP/IP stack

# Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet www.google.com 80
```

Opens TCP connection to port 80 (default HTTP server port) at cis.poly.edu. Anything typed in sent to port 80 at cis.poly.edu

2. Type in a GET HTTP request:

```
GET / HTTP/1.1
Host: www.google.com
```

By typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server

3. Look at response message sent by HTTP server!

# Anatomy of an HTTP Transaction

| | |
|---|---|
| unix> telnet www.google.com 80 | Client: open connection to server |
| Trying 209.85.164.104... | Telnet prints 3 lines to the terminal |
| Connected to www.l.google.com. | |
| Escape character is '^]'. | |
| GET / HTTP/1.1 | Client: request line |
| host: www.google.com | Client: required HTTP/1.1 HOST header |
| | Client: empty line terminates headers |
| HTTP/1.1 200 OK | Server: response line |
| Cache-Control: private | Server: followed by six response headers |
| Content-Type: text/html; charset=ISO-8859-1 | |
| Set-Cookie: PREF=ID=<..snip..> | |
| Server: gws | |
| Transfer-Encoding: chunked | |
| Date: Tue, 13 Nov 2007 17:25:04 GMT | |
| | Server: empty line ("\r\n") terminates hdrs |
| <html> | Server: first HTML line in response body |
| <..snip..> | Server: HTML content not shown. |
| </html> | Server: last HTML line in response body |
| Connection closed by foreign host. | Server: closes connection |
| unix> | Client: closes connection and terminates |

# User-server state: cookies


https://www.flickr.com/photos/kalexanderson

# User-server state: cookies

Many major Web sites use cookies

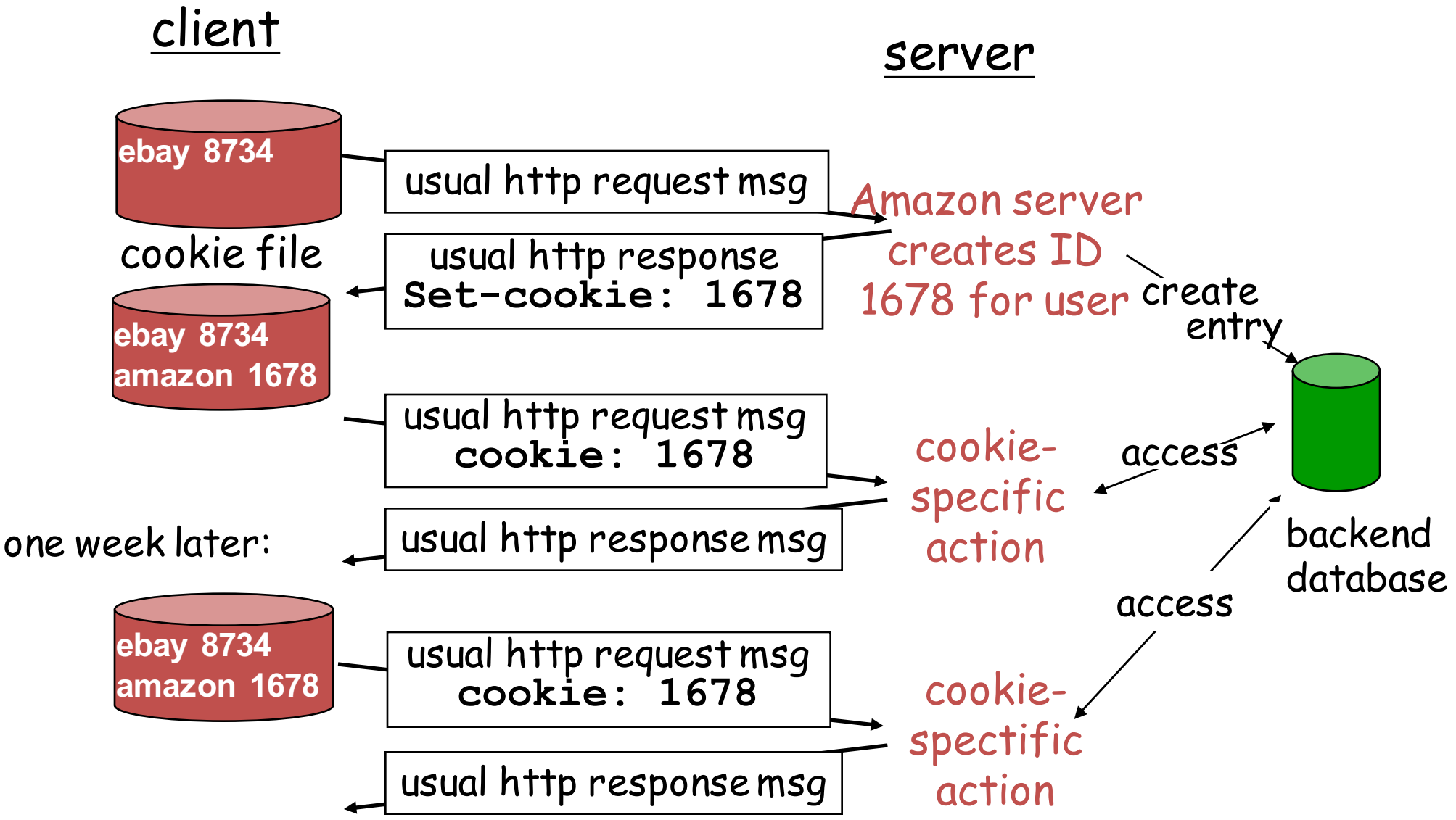
Four components:

1) cookie header line of HTTP *response* message
2) cookie header line in HTTP *request* message
3) cookie file kept on user's host, managed by user's browser
4) back-end database at Web site

Example:

- Susan always accesses Internet from PC
- visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
  - unique ID
  - entry in backend database for ID

# Cookies: keeping "state" (cont.)



# Cookies (continued)

## What cookies can bring:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

**Cookies and privacy:**

- cookies permit sites to learn a lot about you
- you may supply name and e-mail to sites

## How to keep "state":

- protocol endpoints: maintain state at sender/receiver over multiple transactions
- cookies: http messages carry state

Set-Cookie: Name=content data; expires=Fri, 21-Nov-2014 23:59:59 GMT; path=/; domain=.example.net