IOWA

CS3640

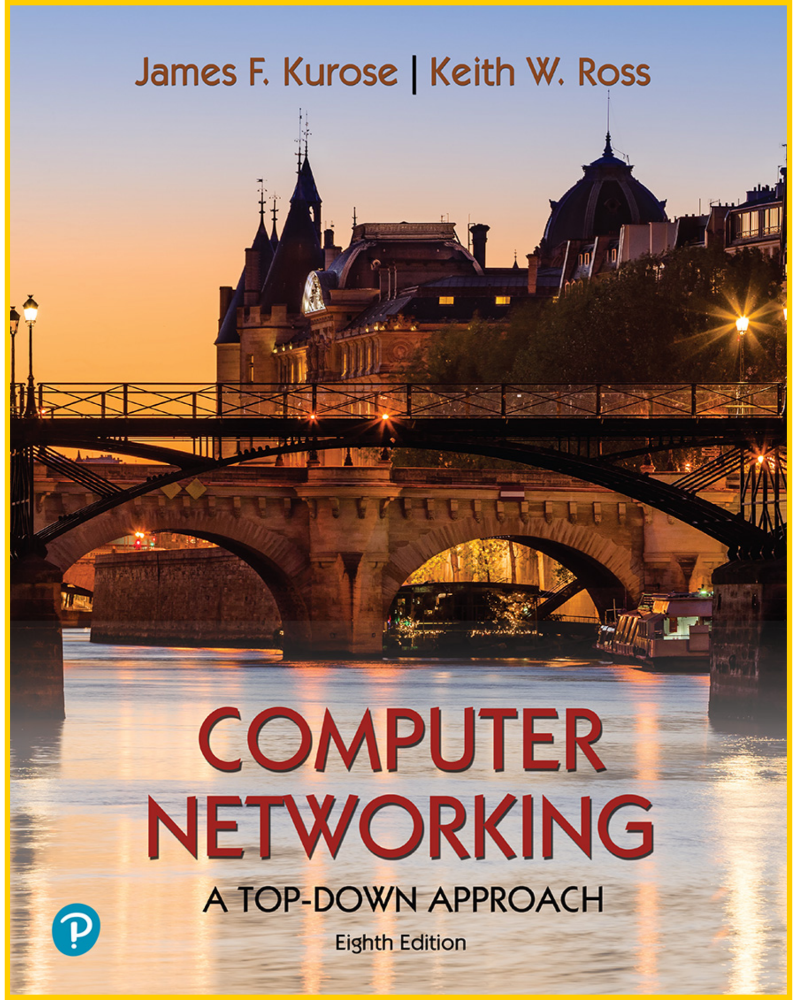
Application Layer (2): The Web & HTTP

Prof. Supreeth Shastri
Computer Science
The University of Iowa

Lecture goals

Deep dive into the design and operation of the world wide web

- *HTTP*
- *Web cookies*
- *Web caches*
- *HTTP/2*



Chapter 2.2

World Wide Web (WWW)

*WWW is an information system where documents and other resources are **identified** by Uniform Resource Locators (URLs), which may be **interlinked** by hypertext, and are **accessible** over the Internet.*

URL

$$\underbrace{\texttt{www.uiowa.edu}}_{\text{host name}}\texttt{/}\underbrace{\texttt{index.html}}_{\text{path name}}$$

HTML

Language for creating hypertext documents

HTTP

Application protocol for transferring web resources

*A **web browser** procures pages and objects from web servers, and displays them to the users*

- a web page consists of **base HTML file**, which typically hyperlinks other **web objects**, each addressable by a URL
- web objects can be a HTML file, image, scripts, audio, video, etc., and can be stored on **same** or **different** web servers

HTTP Overview

Protocol specs

- RFC1945 (v1), RFC2616 (v1.1)
- ASCII (human readable) format
- Two messages: request & response

Client - Server model

- clients request and receive web objects via HTTP, and then display them
- servers store and send web objects in response to HTTP requests

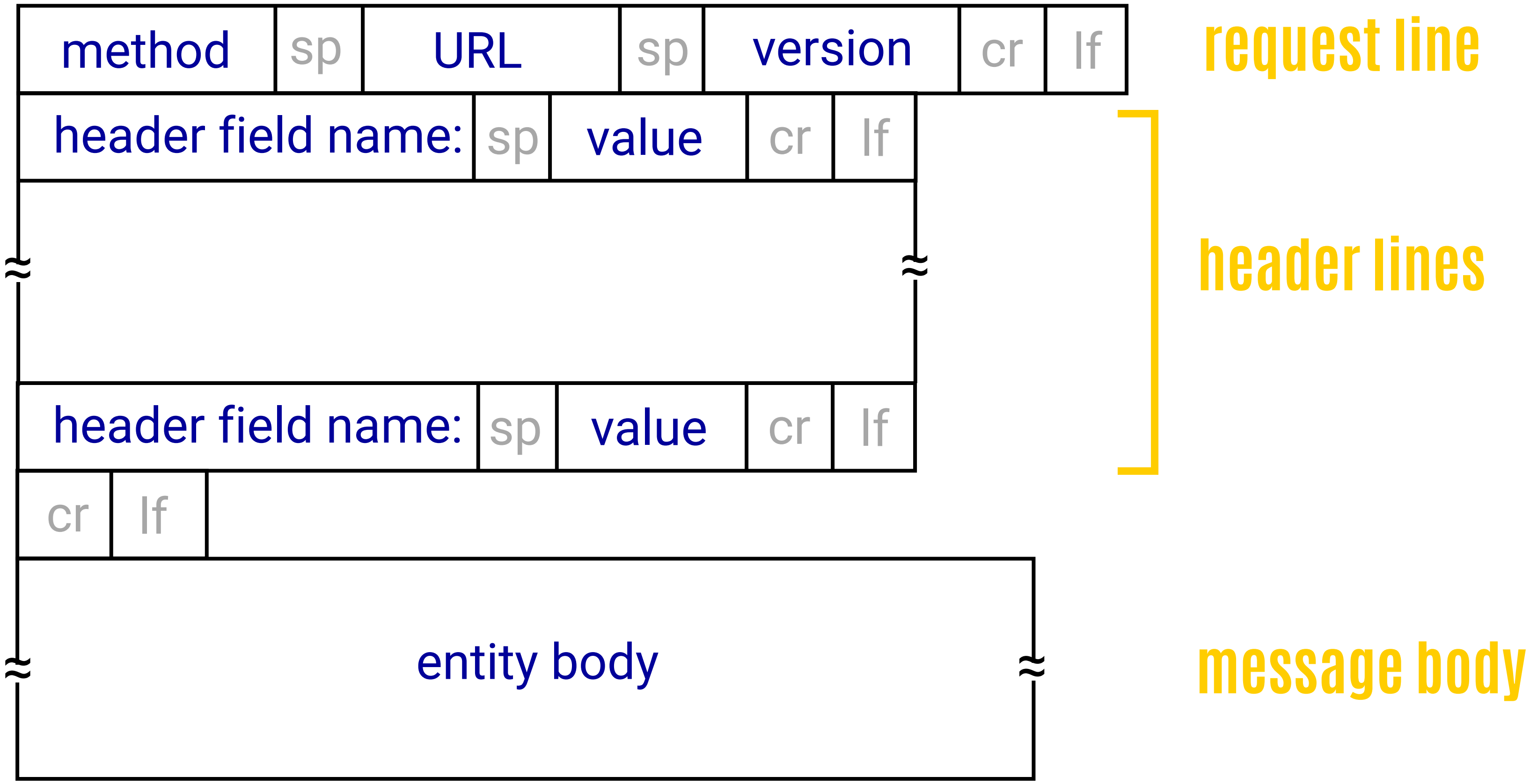
HTTP uses TCP

- Server listens on port 80
- Client initiates TCP handshake, exchanges HTTP messages, and closes the connection

HTTP is stateless

- server maintains no information about past client requests
- Why stateless? ∵ protocols that maintain state tend to be complex

HTTP request format



method sp URL sp version cr lf

header field name: sp value cr lf

header field name: sp value cr lf

cr lf

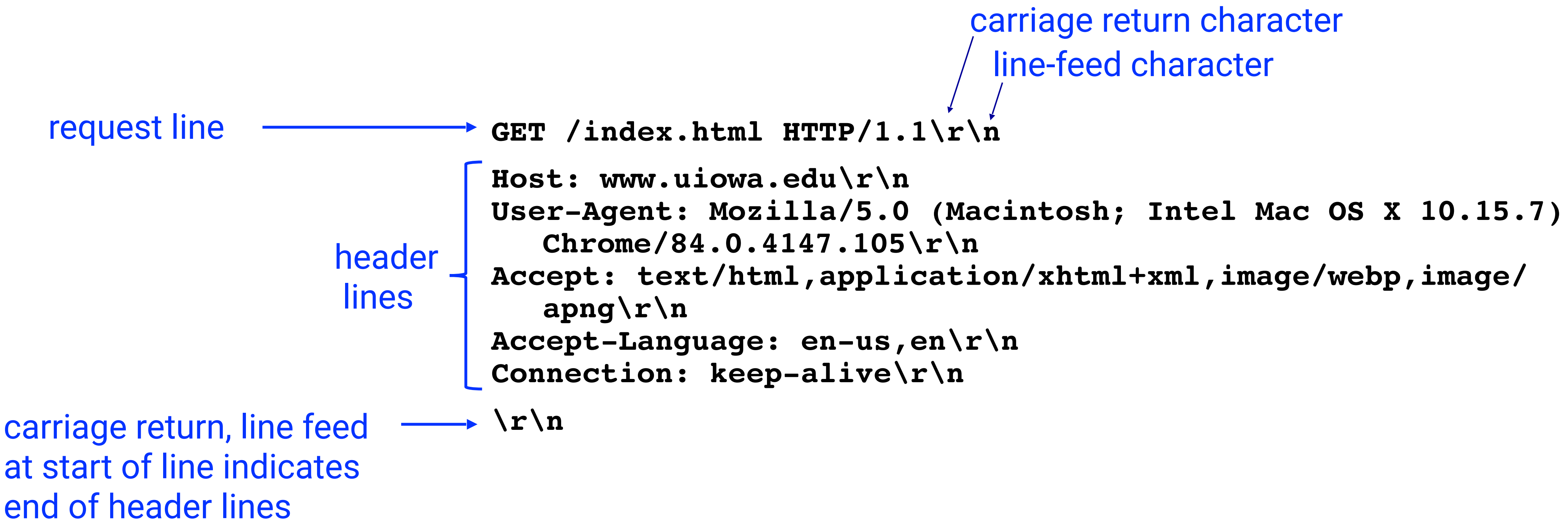
entity body

request line

header lines

message body

HTTP request message



carriage return character

line-feed character

request line

```
GET /index.html HTTP/1.1\r\n
Host: www.uiowa.edu\r\n
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15.7)
   Chrome/84.0.4147.105\r\n
Accept: text/html,application/xhtml+xml,image/webp,image/
   apng\r\n
Accept-Language: en-us,en\r\n
Connection: keep-alive\r\n
\r\n
```

header lines

carriage return, line feed at start of line indicates end of header lines

Five methods of HTTP request

GET method

- requests data from the server
- could include user data in the URL field (following a ?). E.g., `www.google.com/search?q=uiowa`

PUT method

- uploads a new object to the server
- completely replaces file that exists at specified URL with content in entity body of POST

POST method

- for transmitting a web form filled out by a user
- server returns a web page based on what users entered in the form

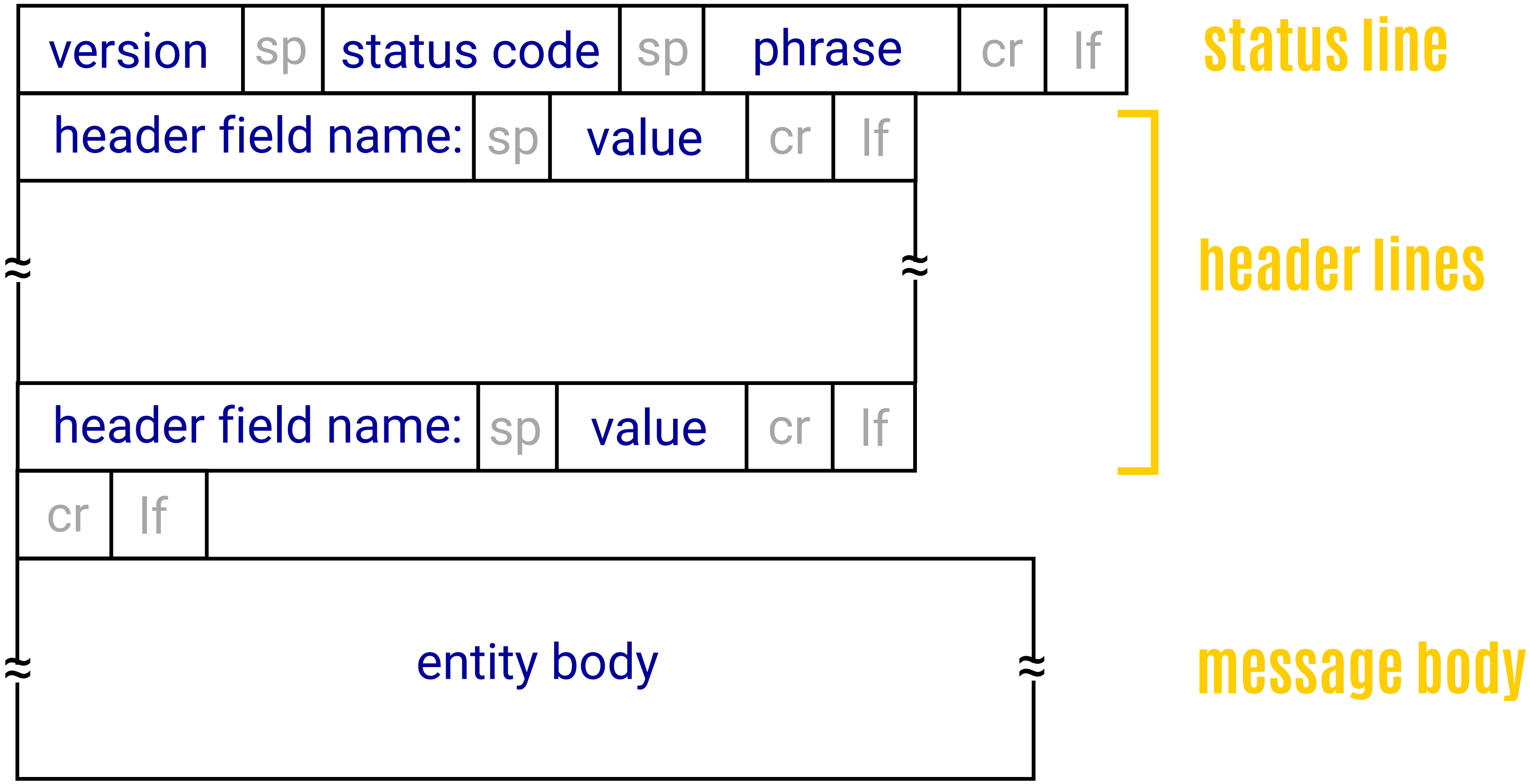
HEAD method

- requests the server to send only headers pertaining to the URL (i.e., no msg body)
- commonly used for debugging

DELETE method

- to delete a specified object at the server
- not commonly supported by servers

HTTP response format



version sp status code sp phrase cr lf

header field name: sp value cr lf

header field name: sp value cr lf

cr lf

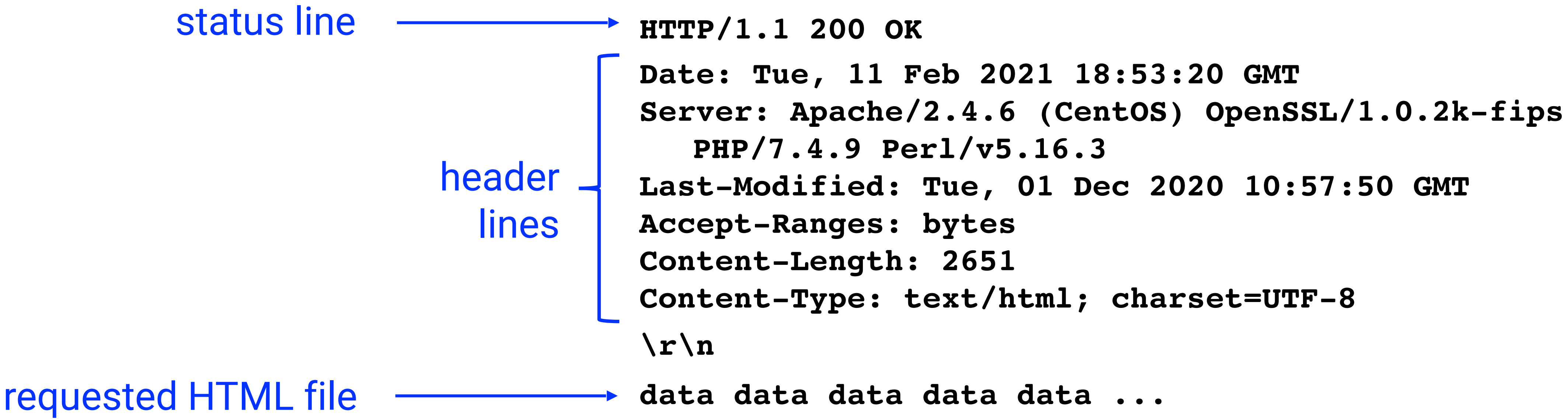
entity body

status line

header lines

message body

HTTP response message



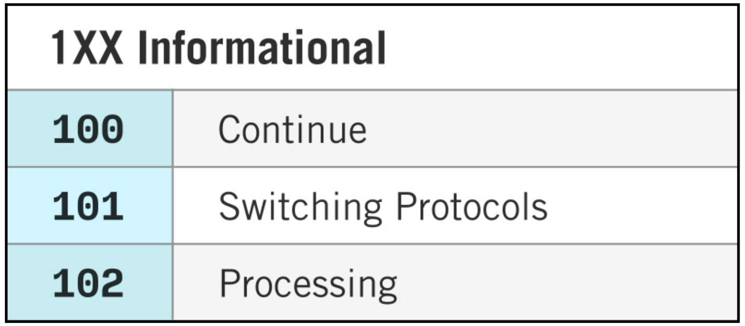
status line

header lines

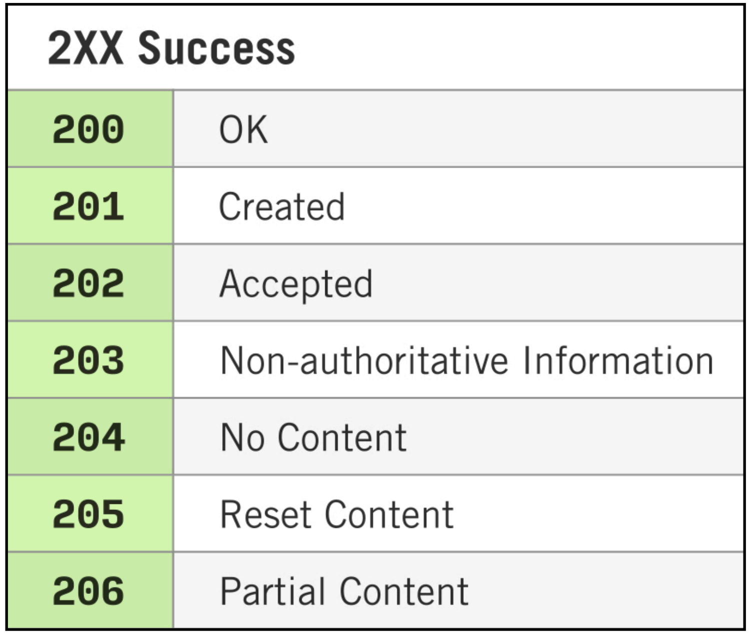
requested HTML file

```
HTTP/1.1 200 OK
Date: Tue, 11 Feb 2021 18:53:20 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips
   PHP/7.4.9 Perl/v5.16.3
Last-Modified: Tue, 01 Dec 2020 10:57:50 GMT
Accept-Ranges: bytes
Content-Length: 2651
Content-Type: text/html; charset=UTF-8
\r\n
data data data data data ...
```

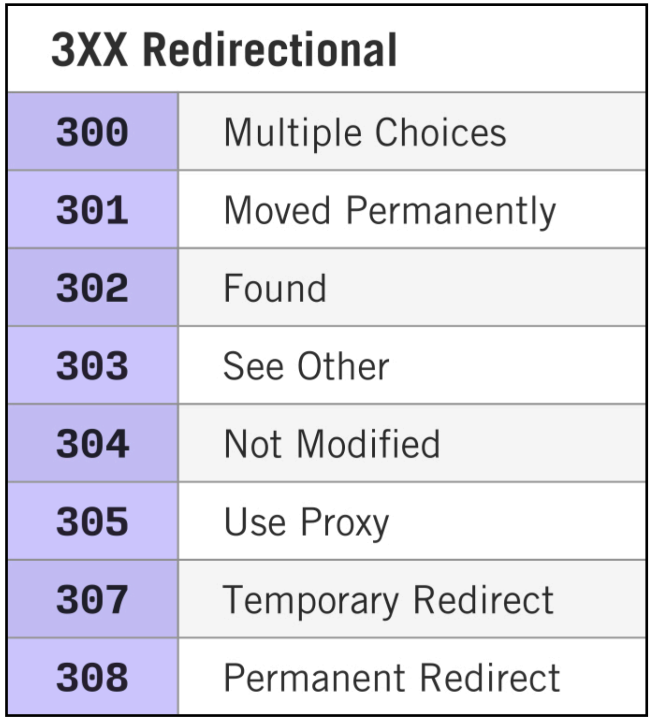
HTTP response codes



1XX Informational	
100	Continue
101	Switching Protocols
102	Processing

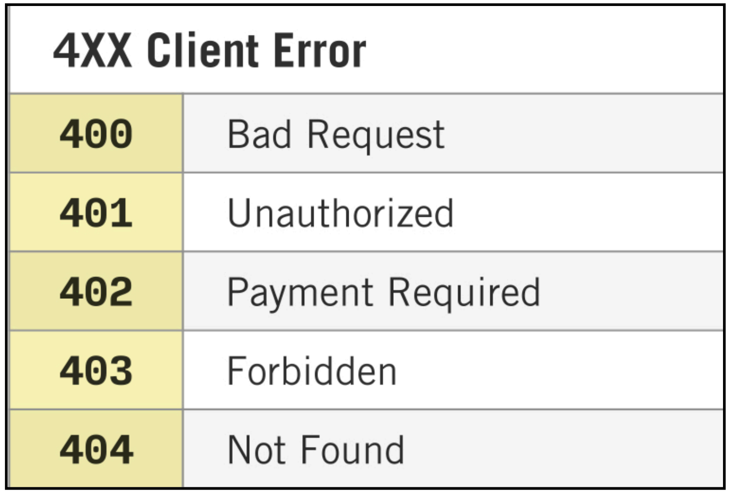


2XX Success	
200	OK
201	Created
202	Accepted
203	Non-authoritative Information
204	No Content
205	Reset Content
206	Partial Content

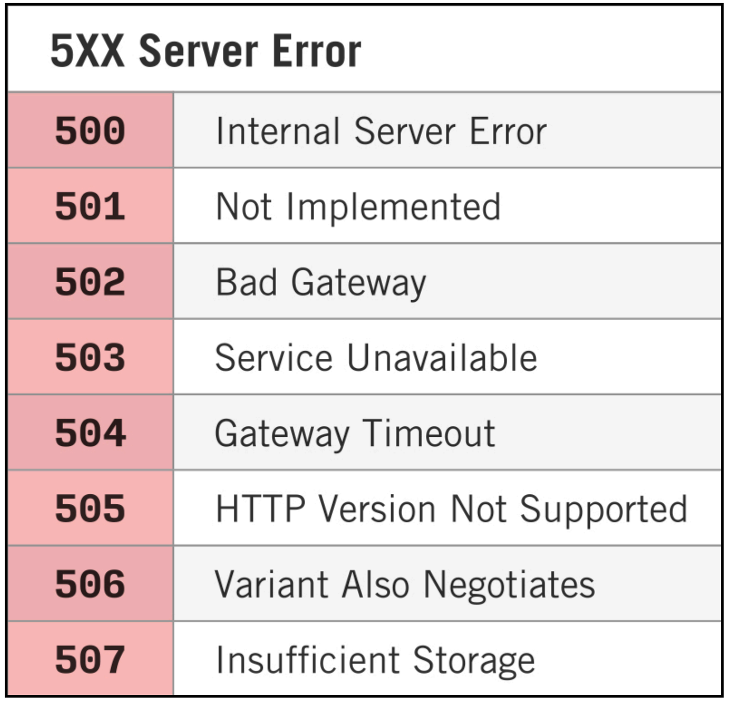


3XX Redirectional	
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
307	Temporary Redirect
308	Permanent Redirect

Courtesy: https://www.steveschoger.com/status-code-poster/



4XX Client Error	
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found

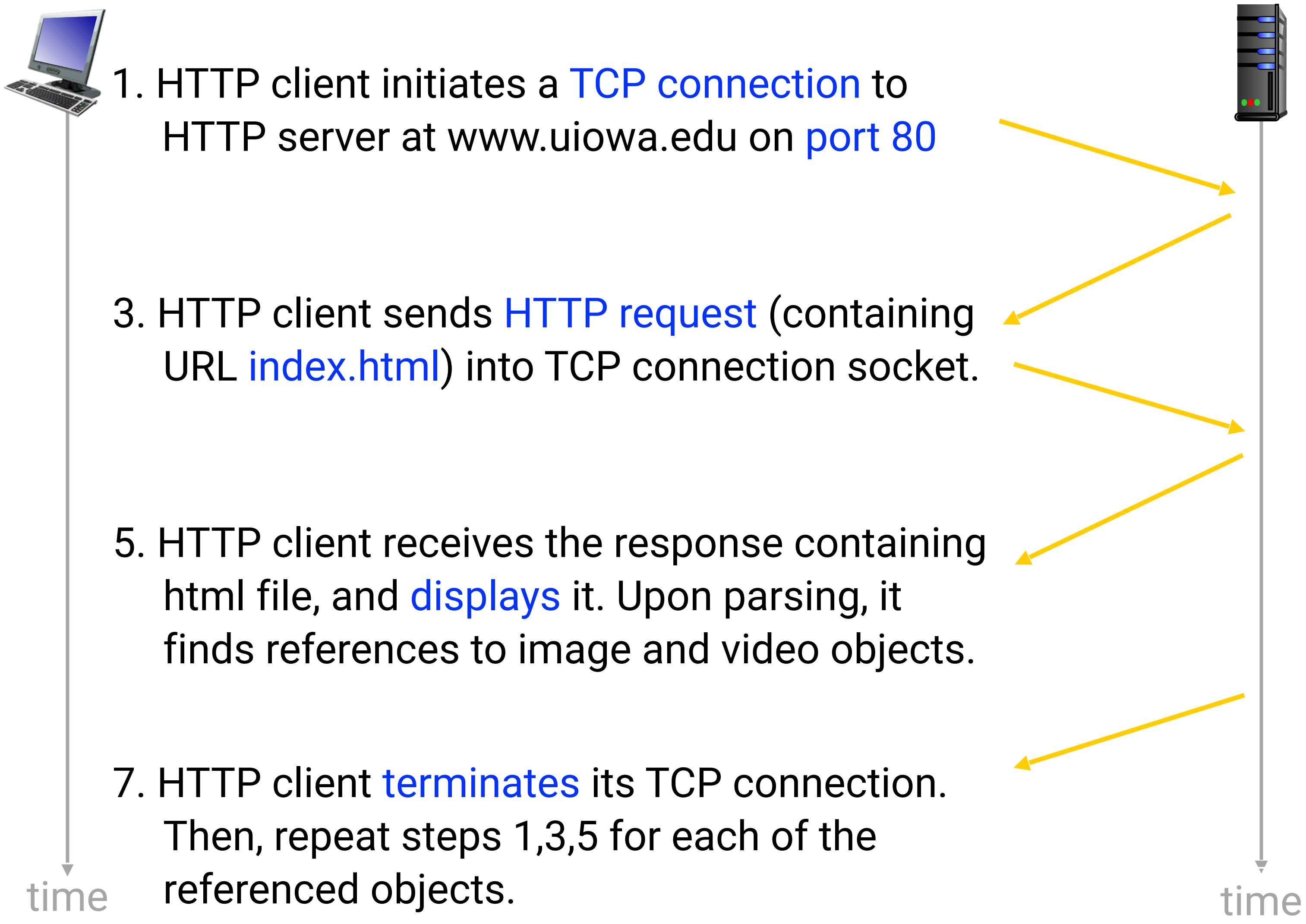


5XX Server Error	
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported
506	Variant Also Negotiates
507	Insufficient Storage

HTTP Connection **Persistence**

HTTP/1.0 message exchange

User enters URL: `www.uiowa.edu/index.html` (containing text, images, and videos)



1. HTTP client initiates a TCP connection to HTTP server at www.uiowa.edu on port 80

2. HTTP server at host www.uiowa.edu waiting for TCP connection at port 80 "accepts" connection, notifying client

3. HTTP client sends HTTP request (containing URL index.html) into TCP connection socket.

4. HTTP server receives the request, forms response with contents of index.html, and sends HTTP response into its socket

5. HTTP client receives the response containing html file, and displays it. Upon parsing, it finds references to image and video objects.

6. HTTP server closes the TCP connection

7. HTTP client terminates its TCP connection. Then, repeat steps 1,3,5 for each of the referenced objects.

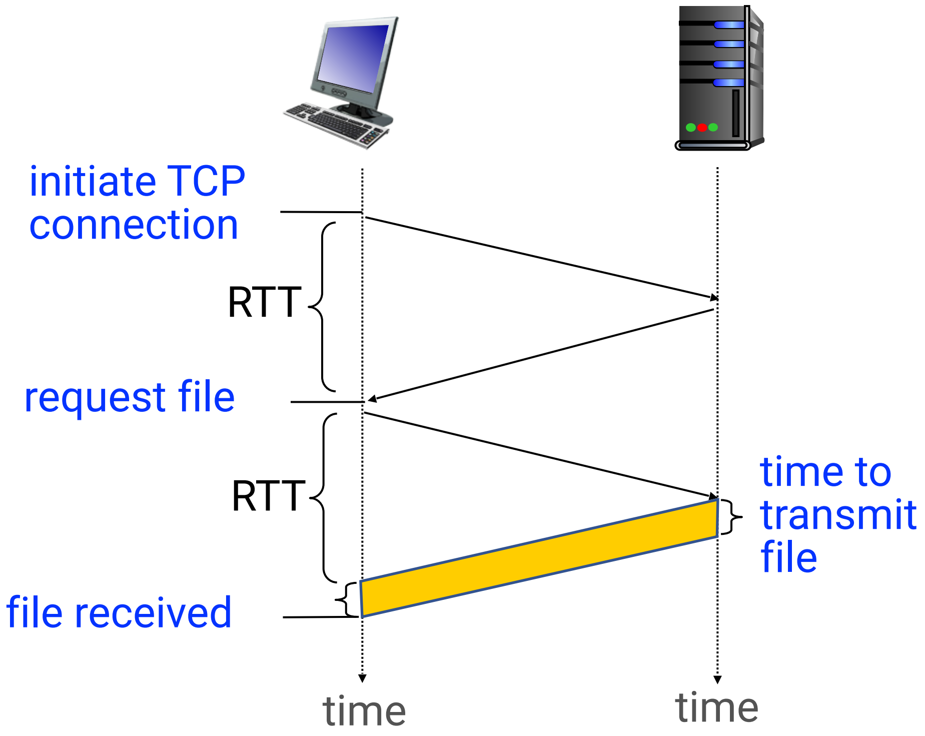
Characterizing non-persistent of HTTP/1.0

Round trip time (RTT)

time for a small packet to travel from client to server and back

HTTP response time (per object):

- one RTT to initiate TCP connection
- one RTT for HTTP request and for the initial bytes of HTTP response to return
- transmission time for the remaining of the object



Total response time per object ≃ 2RTT+ file transmission time

Shortcomings of the non-persistent HTTP

- requires 2 RTTs per object
- OS overhead for each TCP connection
- modern browsers often open multiple parallel TCP connections in parallel to fetch referenced objects

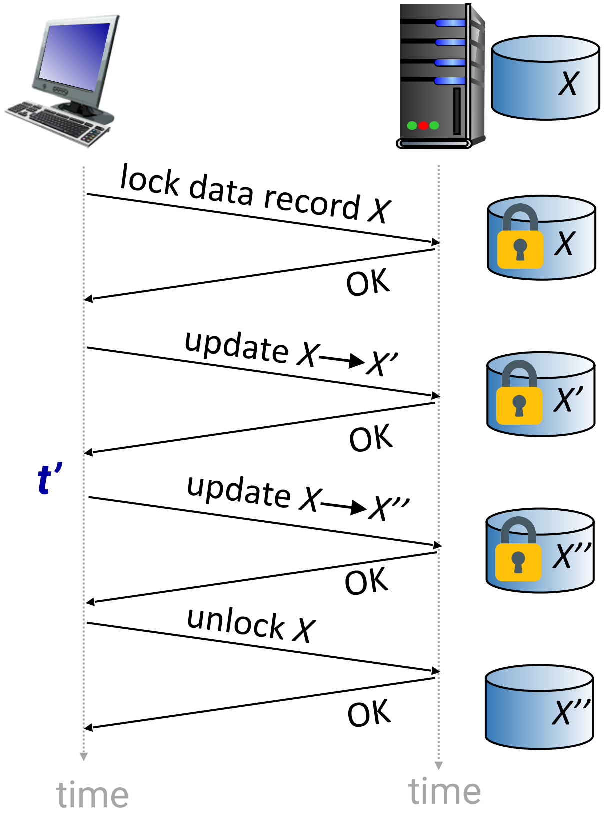
Persistent HTTP (HTTP/1.1)

- server leaves the TCP connection open after sending its response
- subsequent HTTP messages between the same client-server sent over the existing open connection
- client can send requests as soon as it encounters a referenced object
- **response time** is close to one RTT for all but the first referenced objects (cutting the average response time in half)

Web **Cookies**

An illustrative stateful protocol

client makes two changes to X, or none at all



What happens if network connection or client crashes at time *t'* ?

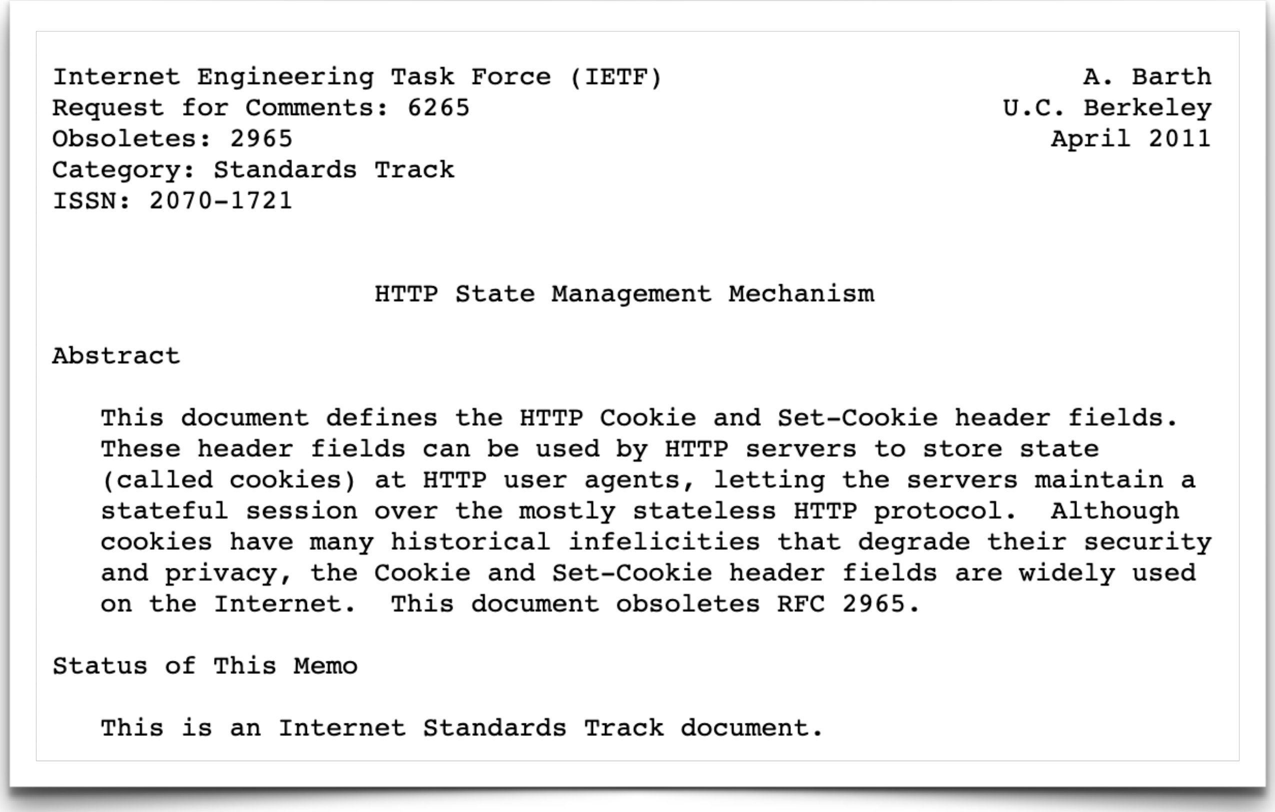
HTTP interaction is stateless

i.e., no notion of multiple HTTP messages completing a web "transaction"

- all HTTP requests are independent of each other
- allows HTTP servers to be simple and high-performant since they don't have to "recover" from a partially-completed-but-failed transactions
- Yet, many emerging and commercial use cases of the web required maintaining the state. E.g., shopping cart

Web Cookies

A mechanism for web servers and client browsers to maintain state across HTTP transactions

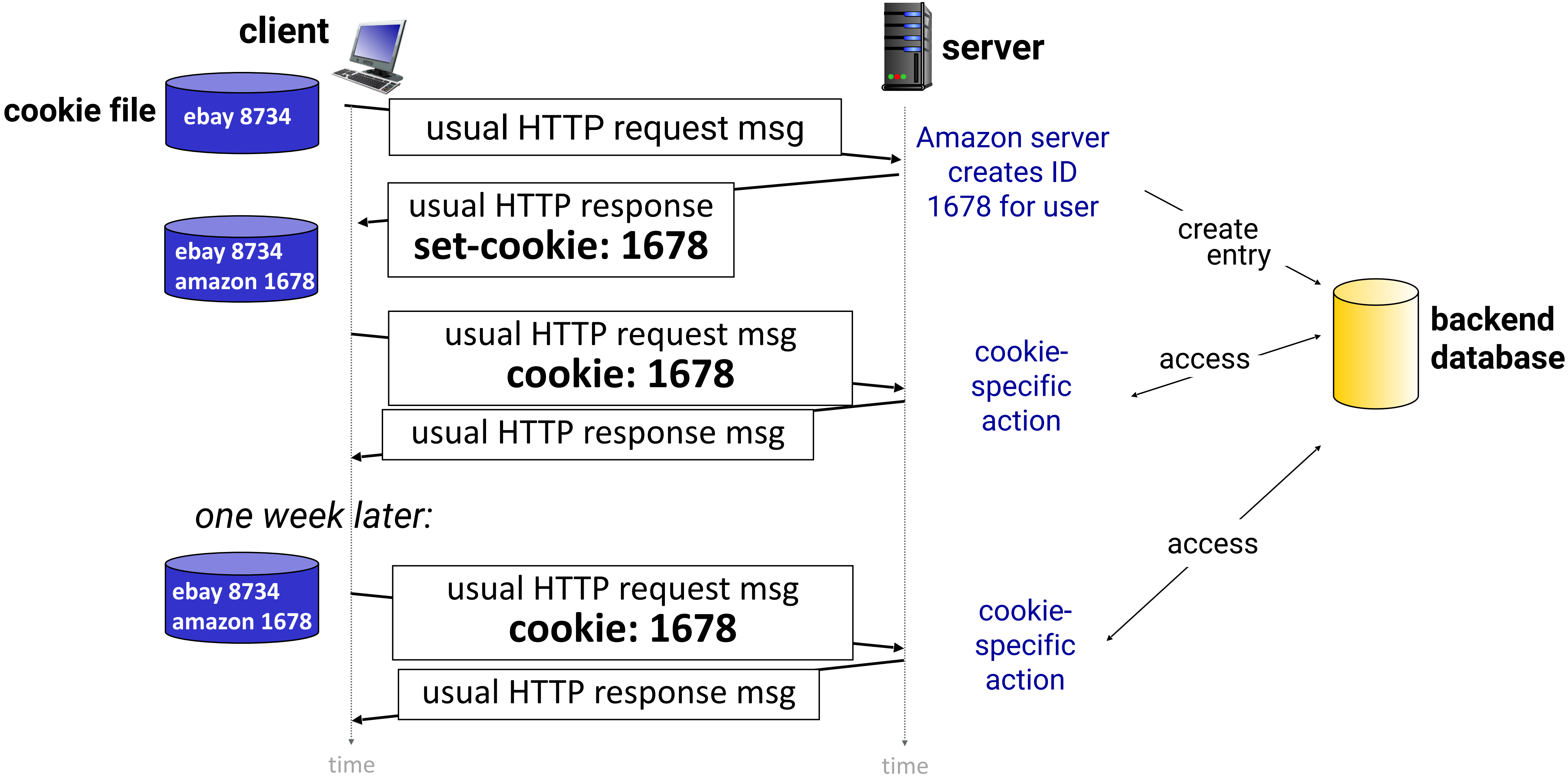


```
Internet Engineering Task Force (IETF)                          A. Barth
Request for Comments: 6265                                U.C. Berkeley
Obsoletes: 2965                                              April 2011
Category: Standards Track
ISSN: 2070-1721


                    HTTP State Management Mechanism

Abstract

   This document defines the HTTP Cookie and Set-Cookie header fields.
   These header fields can be used by HTTP servers to store state
   (called cookies) at HTTP user agents, letting the servers maintain a
   stateful session over the mostly stateless HTTP protocol.  Although
   cookies have many historical infelicities that degrade their security
   and privacy, the Cookie and Set-Cookie header fields are widely used
   on the Internet.  This document obsoletes RFC 2965.

Status of This Memo

   This is an Internet Standards Track document.
```

Four key components

- **HTTP response**: a cookie header line in the first response from the server
- **HTTP request**: cookie header line in all subsequent requests from the client
- **Browser**: cookie file kept on user’s host and managed by user’s browser
- **Web server**: back-end database for cookie management

Maintaining user/server state: cookies



Cookies: the good, the bad, the ugly

Cookies are useful in

- authorization
- shopping carts
- recommendations
- generic session state

Challenges

- at HTTP endpoints: maintain state at sender/ receiver over multiple transactions
- in messages: cookies in HTTP messages carry state

Privacy considerations

- cookies permit sites to learn a lot about you on their site
- third party persistent cookies (aka, tracking cookies) allow persistent identity beyond one website, and thus, enable unlimited tracking across the web

Spot Quiz (ICON)