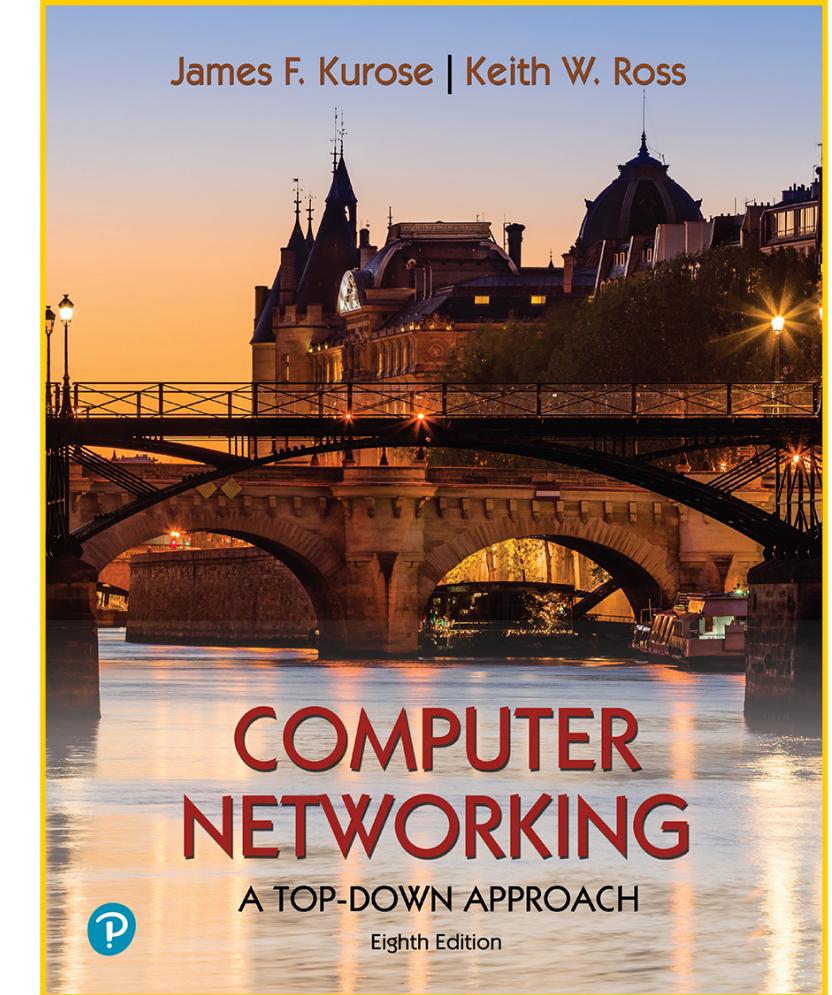


CS3640

---

# 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*

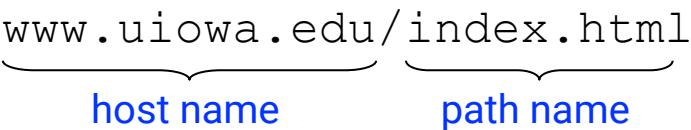
- *HTTP*
- *Web cookies*
- *Web caches*
- *HTTP/2*

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

www.uiowa.edu/index.html  
host name                  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**

# ~~Tim Berners-Lee's~~ World Wide Web



Sir Tim Berners-Lee invented the World Wide Web in 1989.

Image: © CERN

*Read more: <https://webfoundation.org/about/vision/history-of-the-web/>*

# 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

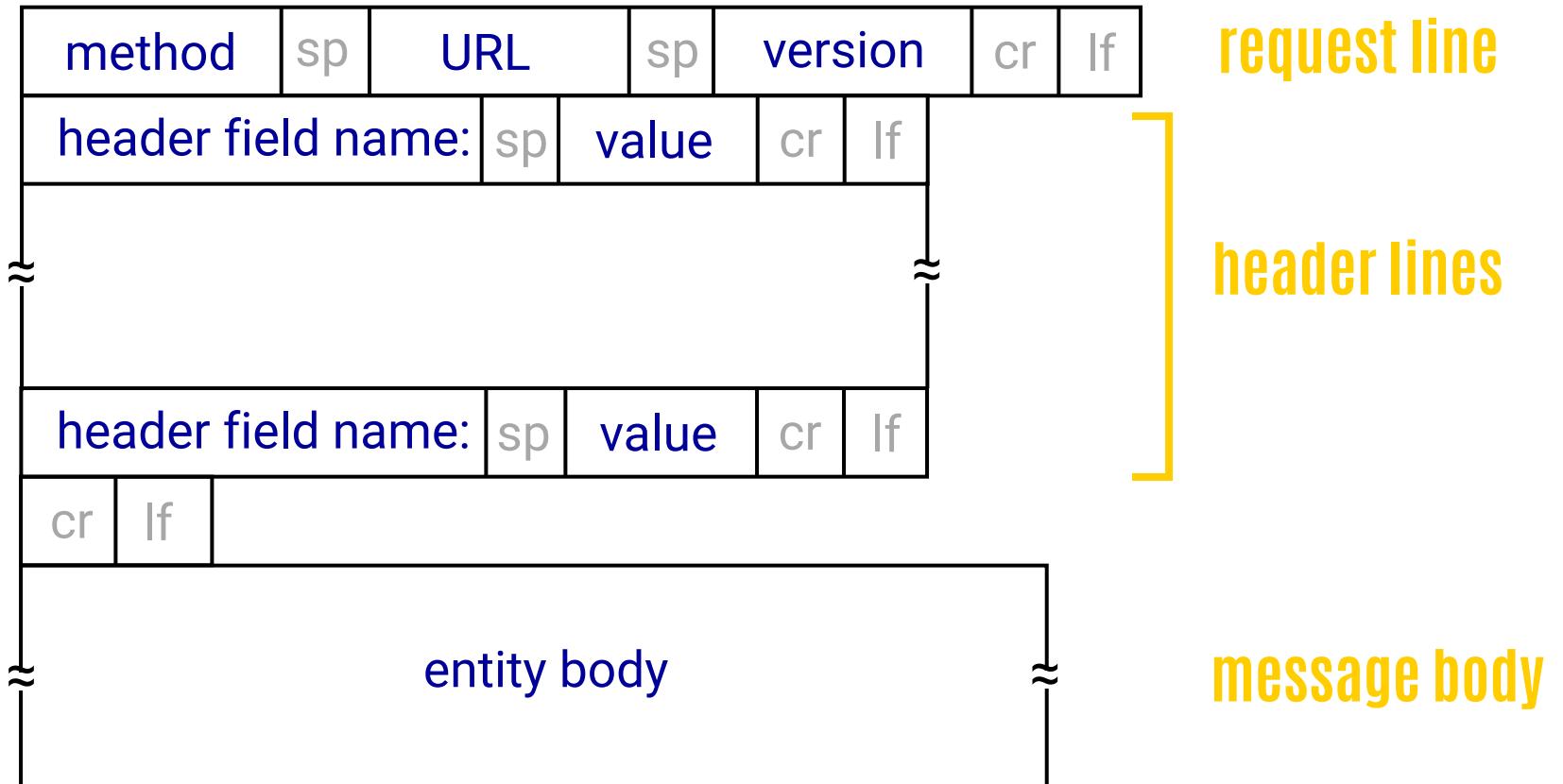
## HTTP uses TCP

- 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



# HTTP request message

request line → GET /index.html HTTP/1.1\r\n

header lines → Host: www.uiowa.edu\r\nUser-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15.7)  
Chrome/84.0.4147.105\r\nAccept: text/html,application/xhtml+xml,image/webp,image/  
apng\r\nAccept-Language: en-us,en\r\nConnection: keep-alive\r\n

carriage return, line feed  
at start of line indicates  
end of header lines → \r\n

carriage return character  
line-feed character

# 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](http://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

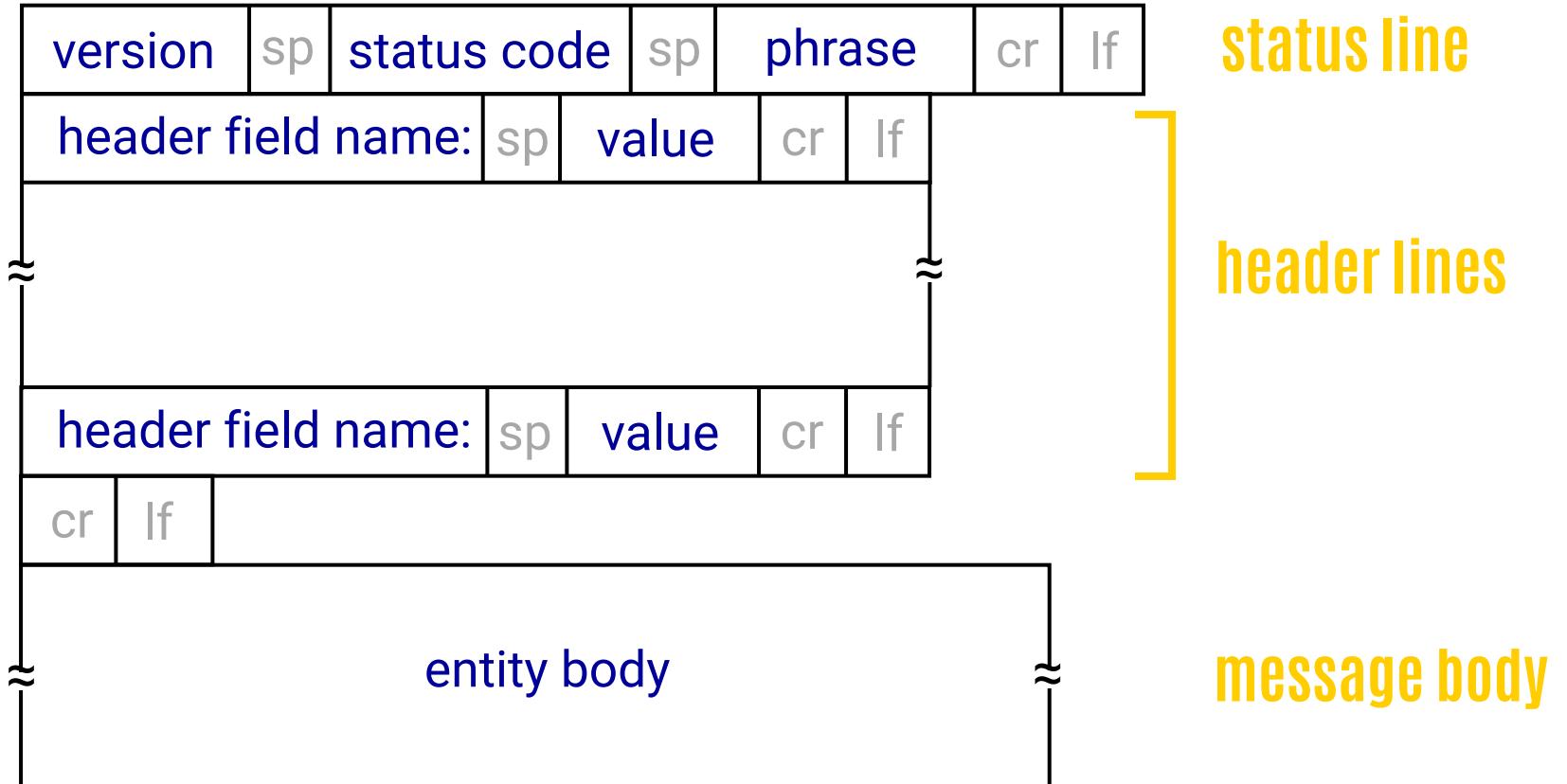
## HEAD method

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

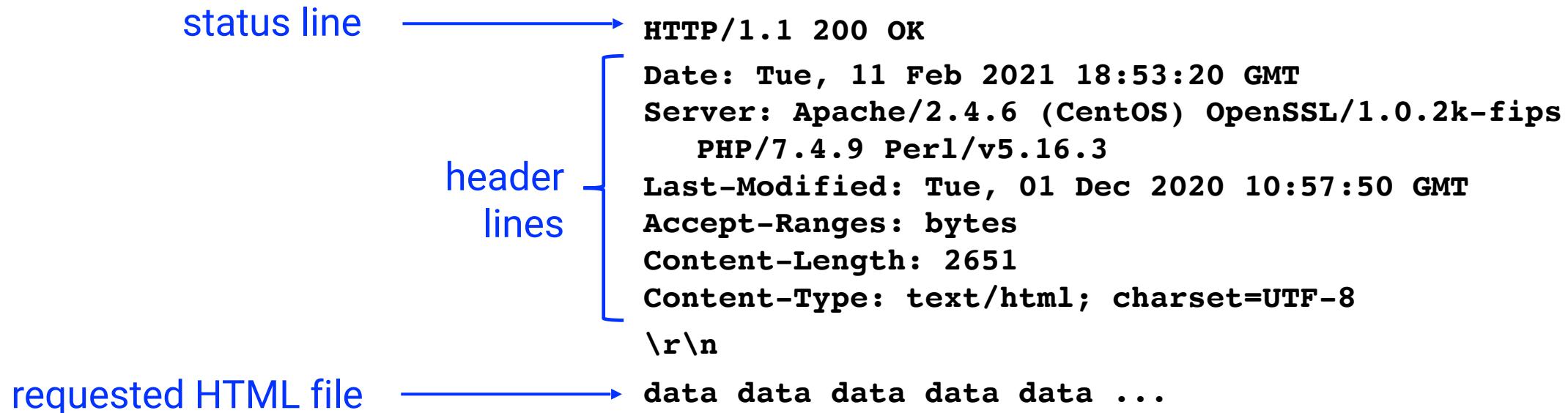
## DELETE method

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

# HTTP response format



# HTTP response message



# HTTP response codes

1XX Informational	
<b>100</b>	Continue
<b>101</b>	Switching Protocols
<b>102</b>	Processing

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

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

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

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

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

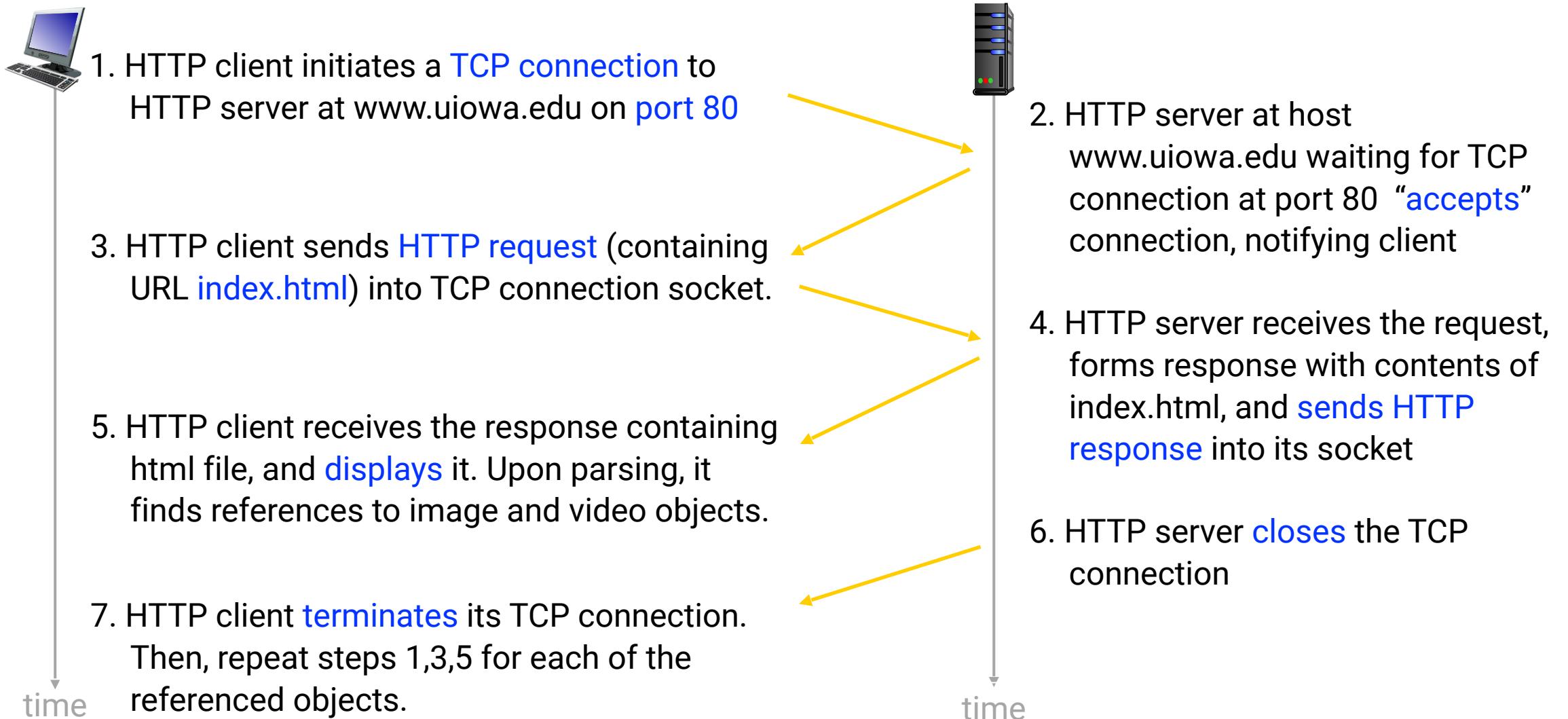
# Common HTTP headers

Header	Type	Contents
User-Agent	Request	Information about the browser and its platform
Accept	Request	The type of pages the client can handle
Accept-Charset	Request	The character sets that are acceptable to the client
Accept-Encoding	Request	The page encodings the client can handle
Accept-Language	Request	The natural languages the client can handle
Host	Request	The server's DNS name
Authorization	Request	A list of the client's credentials
Cookie	Request	Sends a previously set cookie back to the server
Date	Both	Date and time the message was sent
Upgrade	Both	The protocol the sender wants to switch to
Server	Response	Information about the server
Content-Encoding	Response	How the content is encoded (e.g., gzip)
Content-Language	Response	The natural language used in the page
Content-Length	Response	The page's length in bytes
Content-Type	Response	The page's MIME type
Last-Modified	Response	Time and date the page was last changed
Location	Response	A command to the client to send its request elsewhere
Accept-Ranges	Response	The server will accept byte range requests
Set-Cookie	Response	The server wants the client to save a cookie

# HTTP Connection Persistence

# HTTP/1.0 message exchange

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



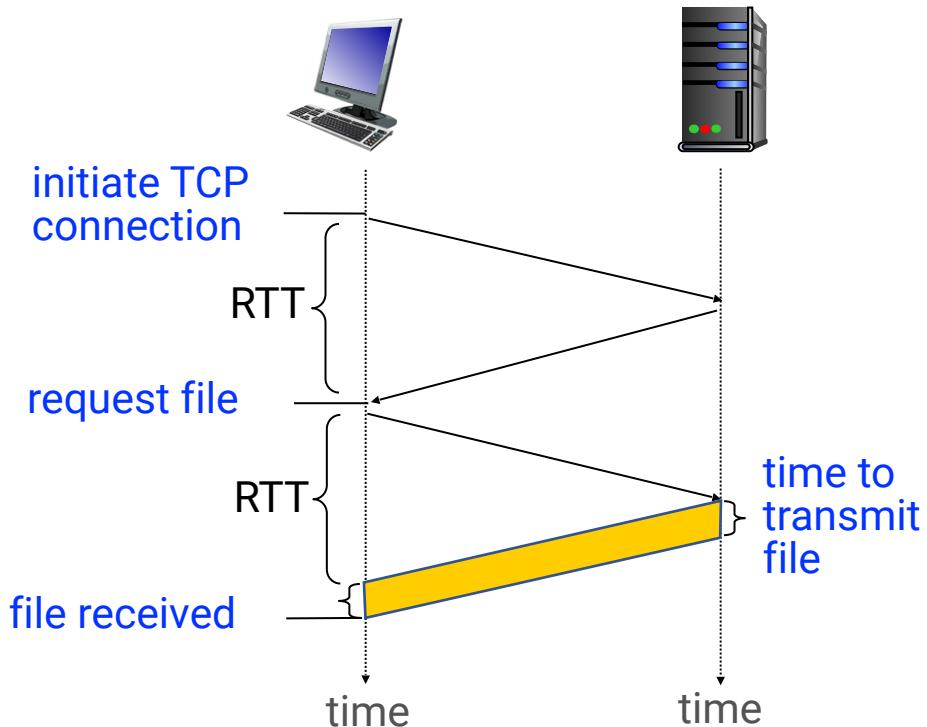
# Characterizing non-persistent of HTTP/1.0

## 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



$$\text{Total response time per object} \approx 2\text{RTT} + \text{file transmission time}$$

## 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

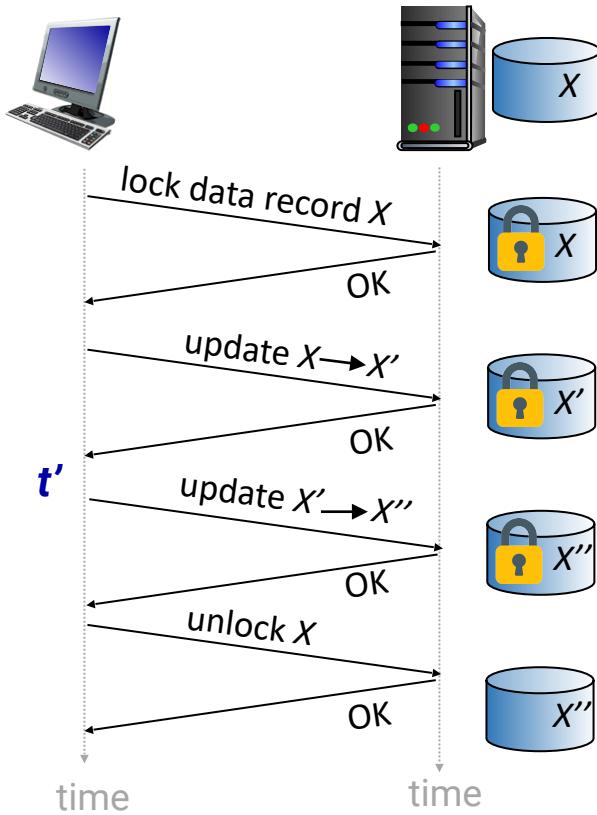
## Persistent HTTP (HTTP/1.1)

- 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)  
Request for Comments: 6265  
Obsoletes: 2965  
Category: Standards Track  
ISSN: 2070-1721

A. Barth  
U.C. Berkeley  
April 2011

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.

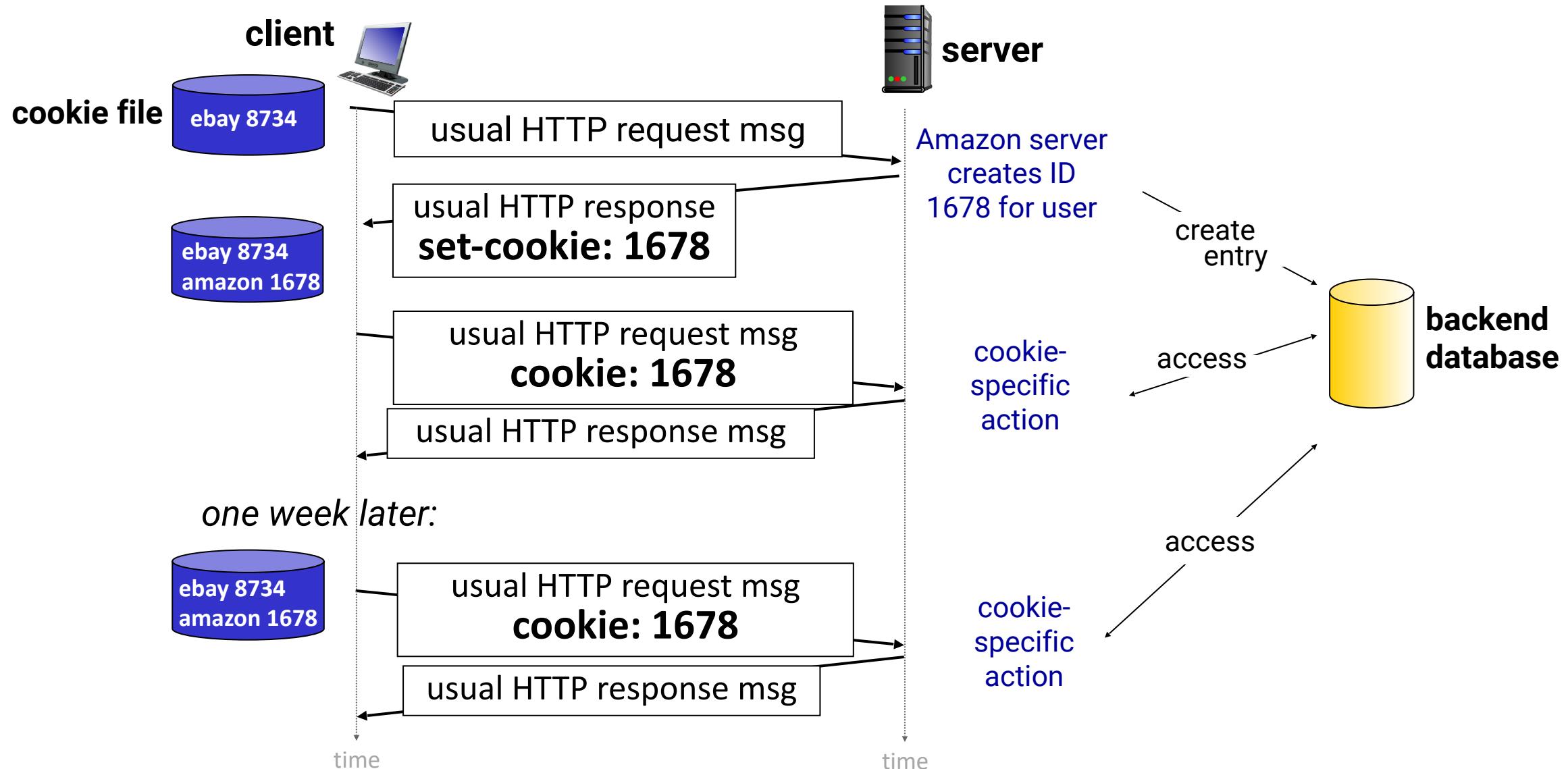
**Status of This Memo**

This is an Internet Standards Track document.

## Four key components

- **HTTP response:** a set-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)**