HTTP

CS 241

April 23, 2014

University of Illinois

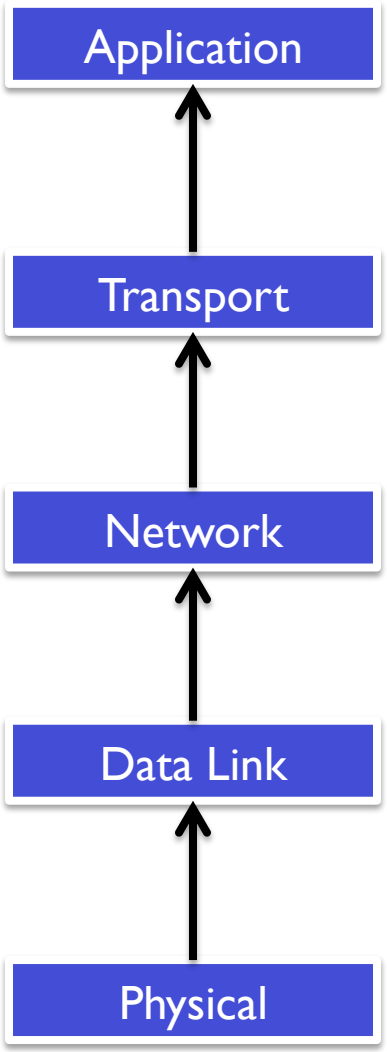
Announcements

Brighten out of town Friday

- Guest lecture: Ankit Singla
- DNS and networking research

MP6 due today, MP7 out today

Onward to Applications



All networked applications use "application layer" protocols to communicate

Examples

- HTTP
- FTP
- SMTP
- IMAP
- Telnet
- SSH
- …

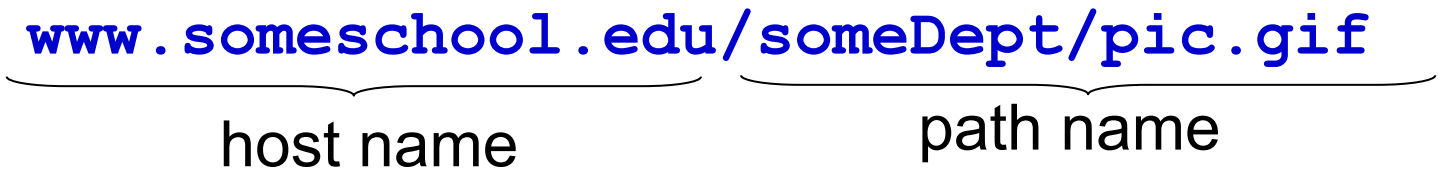
Web page structure

Web pages consist of

- Objects
  - HTML files, JPEG images, Java applets, audio files,…
- Base HTML file
  - Includes several referenced objects (How many on a typical modern site?)

Each object is addressable by a URL

Example URL:

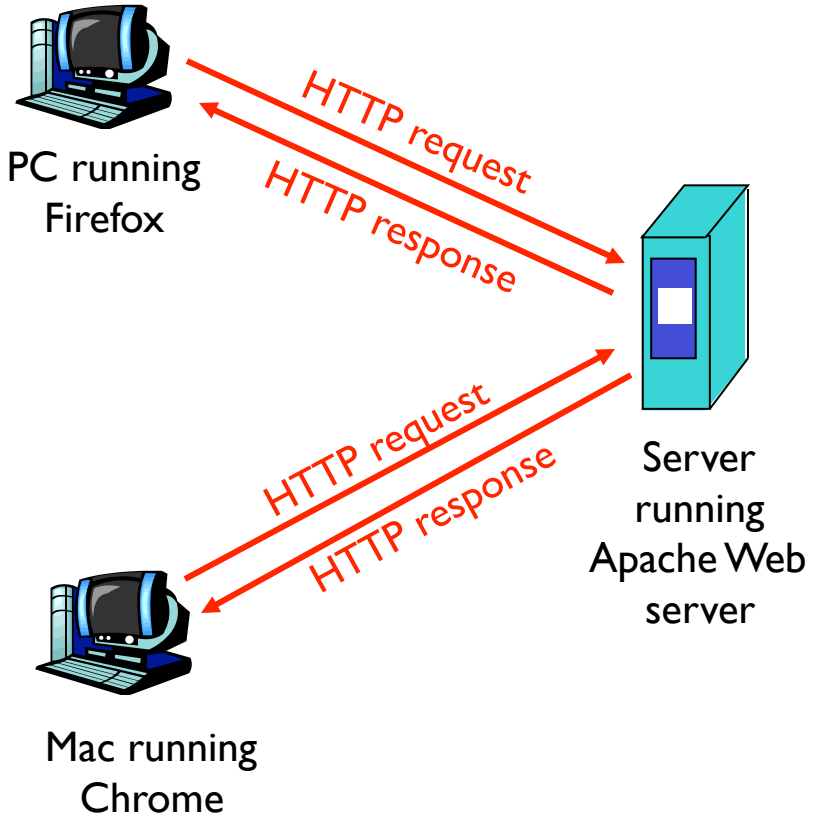


HTTP (Hypertext Transfer Protocol)

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

Uses TCP

Basic steps

- 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

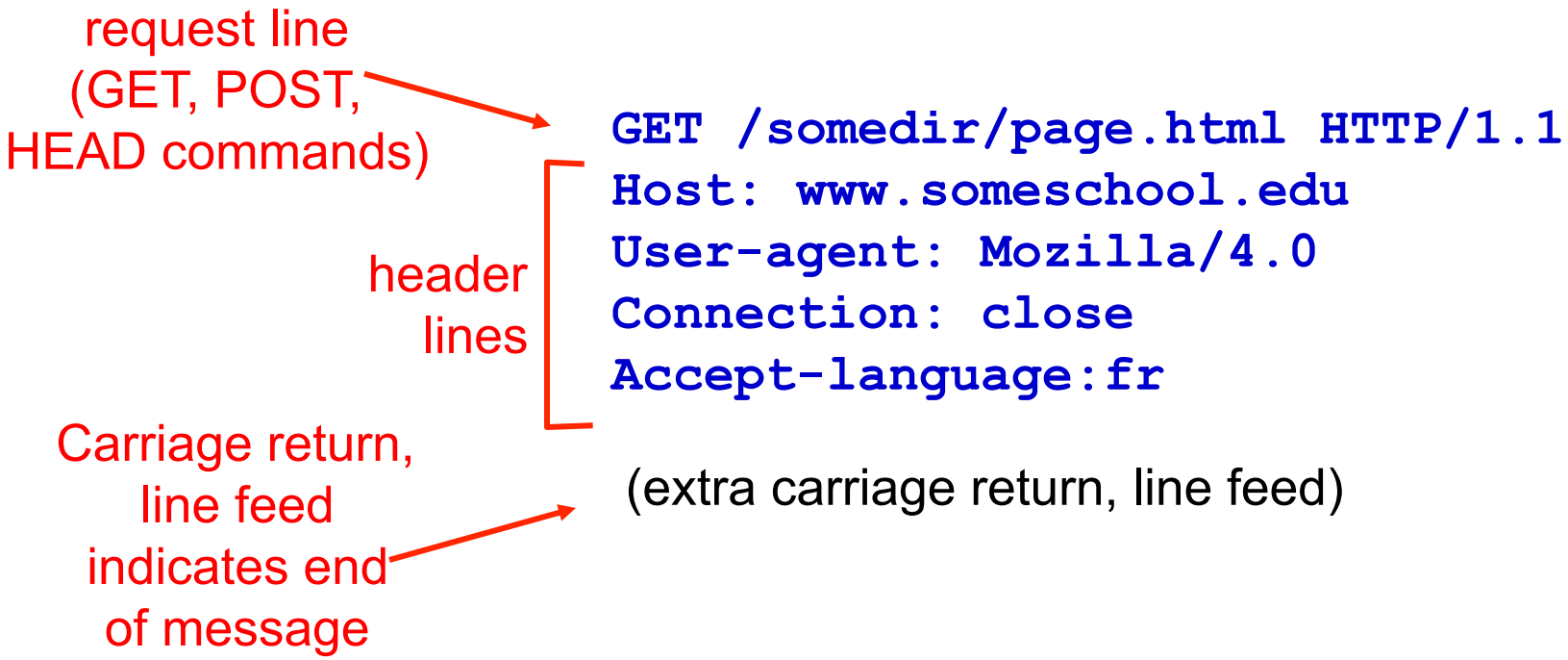
- Server need not maintain information about past client requests

HTTP Request Message

Two types of HTTP messages: request, response

- ASCII (human-readable format)

HTTP request message:



Method Types

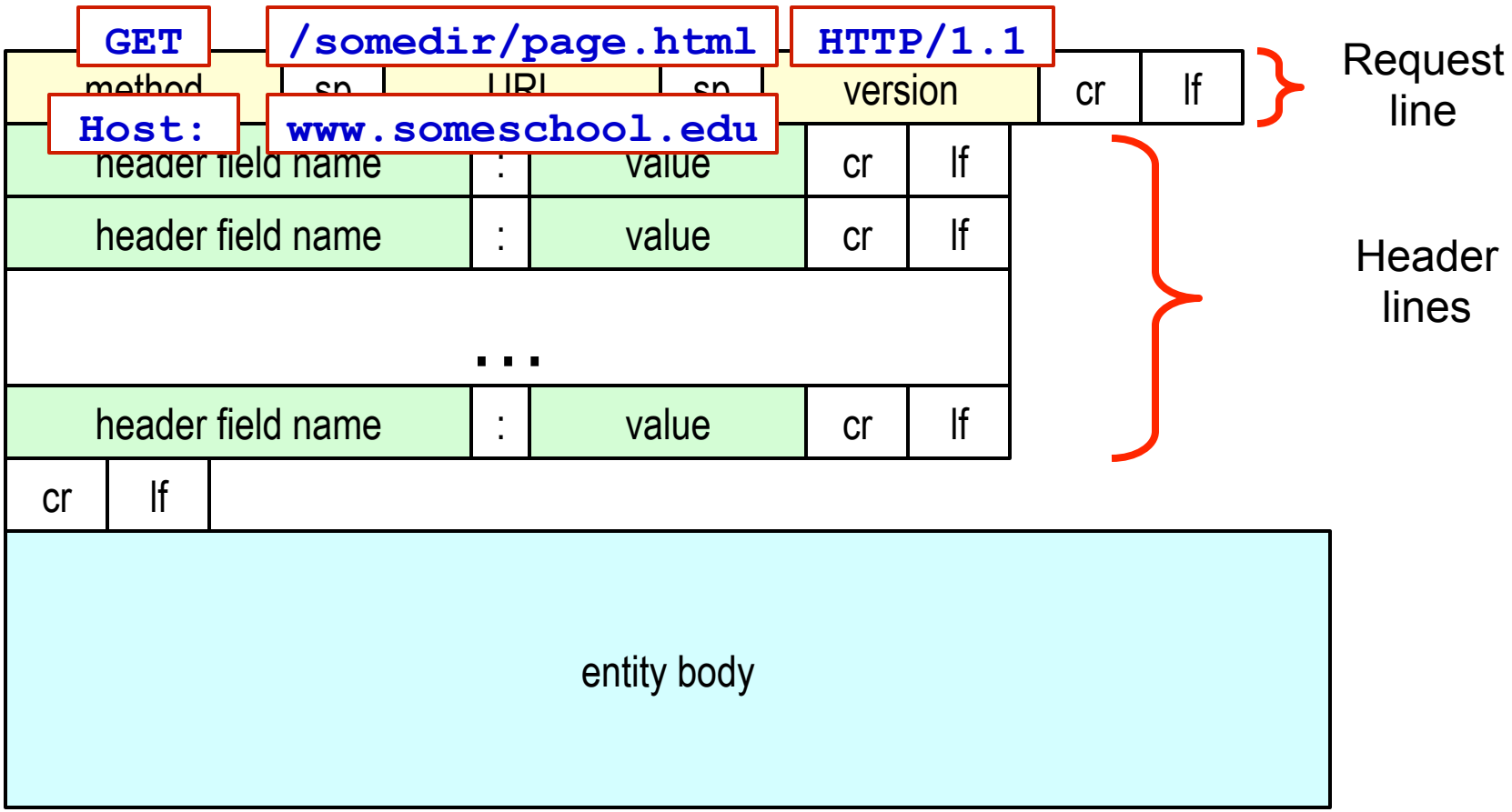
HTTP/1.0

- GET
- POST
- HEAD
  - Asks server to leave requested object out of response

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 Request Message: General Format



Uploading Form Input

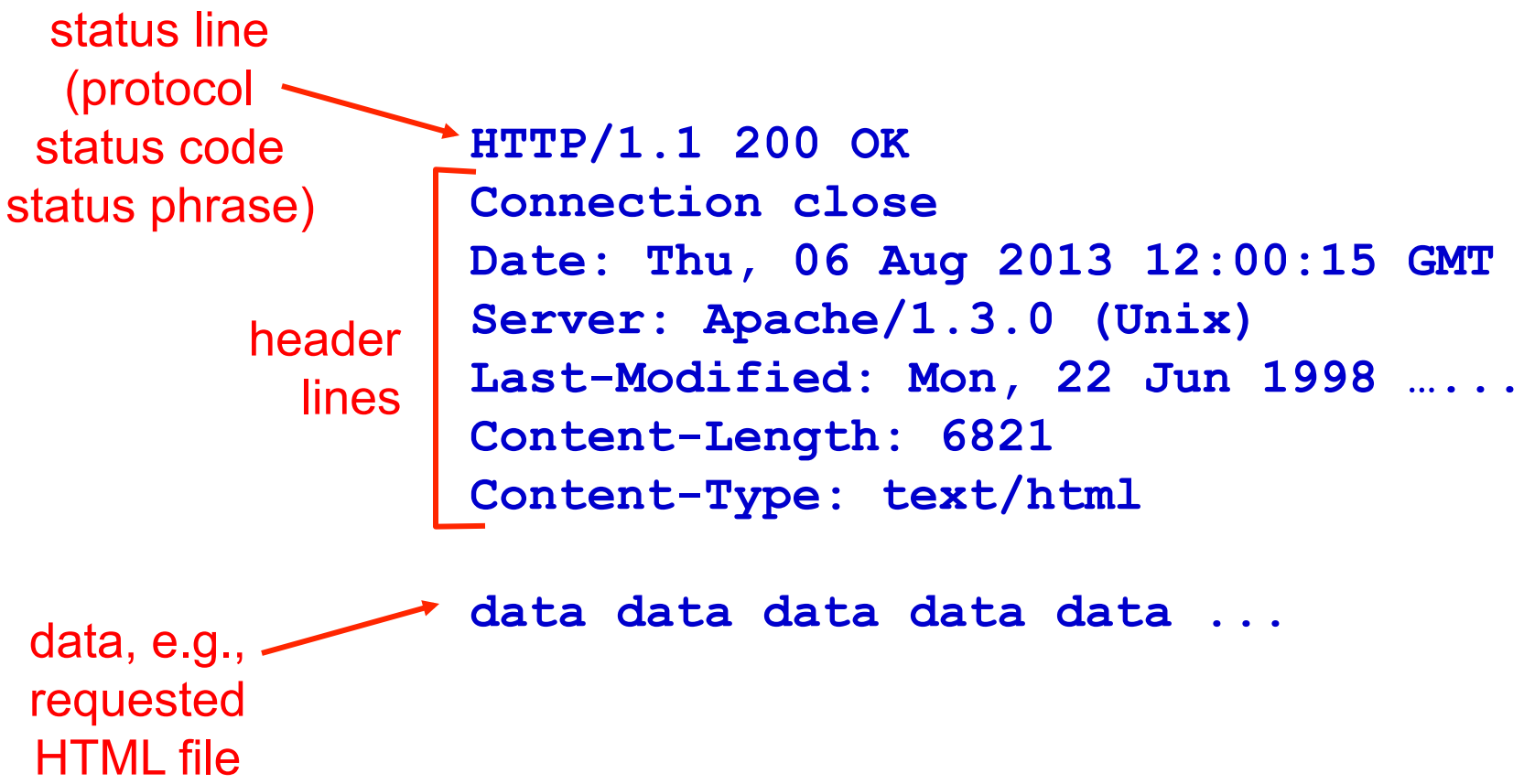
Post method

- Web page often includes form of 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

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:), client automatically retrieves new URL
400	Bad Request	request message not understood by server
404	Not Found	requested document not found on this server
505	HTTP Version Not Supported	

HTTP response status codes

In first line in server->client response message

A few sample codes

More in the illustrated guide...

- http://tinyurl.com/cvyepwt

Trying out HTTP (client side) For Yourself

1. Telnet to your favorite Web server

```
telnet courses.engr.illinois.edu 80
```

Opens TCP connection to port 80 (default HTTP server port) at `www.cs.illinois.edu`. Anything typed in sent to port 80 at `cs.illinois.edu`

2. Type in a GET HTTP request

```
GET /class/sp12/cs241/index.html
HTTP/1.0
```

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!

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

- Alice 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

What cookies can bring

- Authorization
- Shopping carts
- Recommendations
- User session state (Web e-mail)

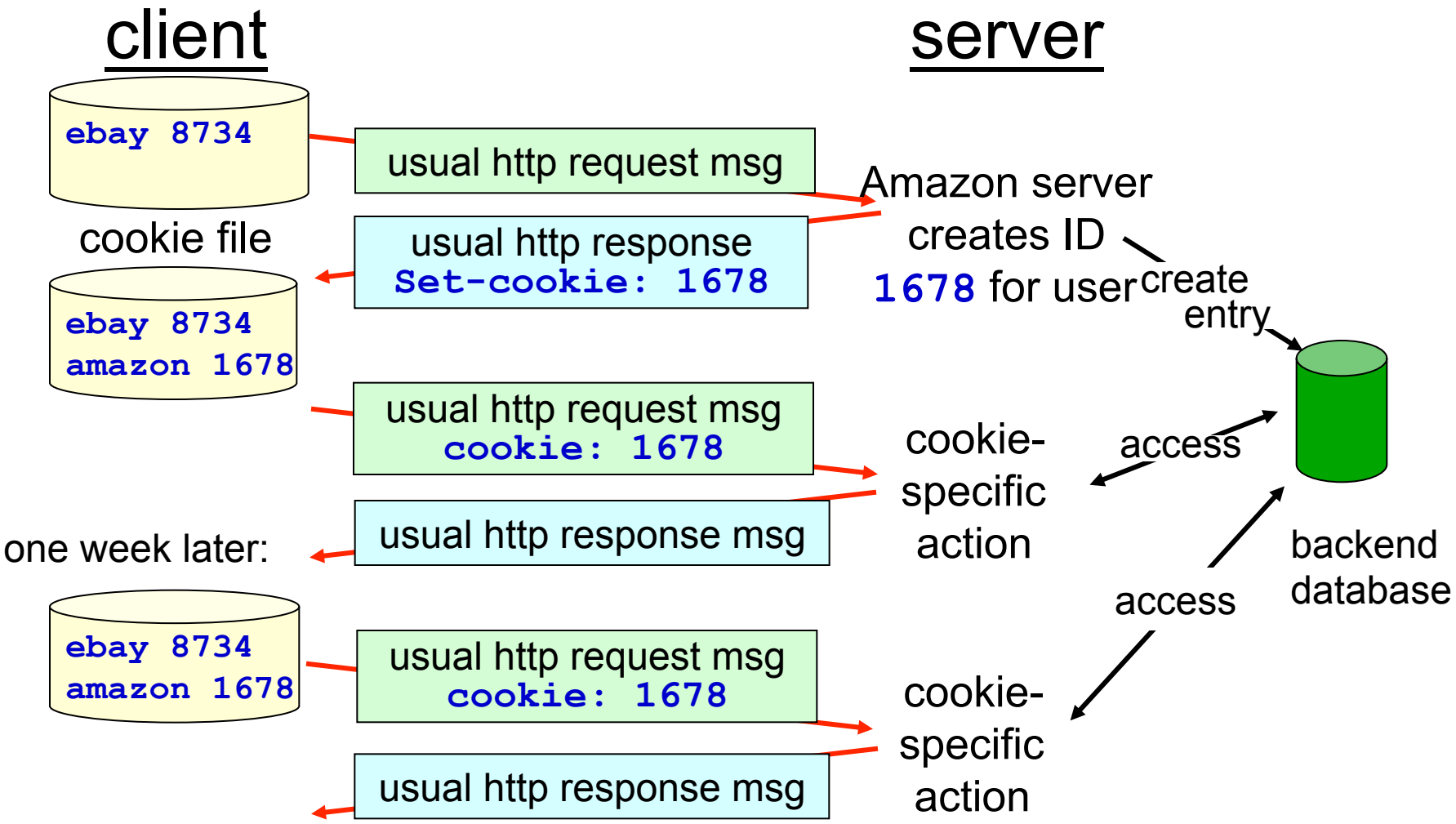
How to keep “state”

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

Cookies and privacy

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

Cookies: Keeping "State"



Optimizing HTTP

How long does it take?

Consider loading a website:

- You request a single HTML page
- The HTML page contains 5 images
- (Since the HTML page contains the 5 images, you don't know about them before you receive the images.)

Q: how long would it take to completely load the webpage given:

- You have to make a new connection for every requested object,
  and
- You can only have one active connection at any time
  and
- The actual HTML and images are very small in size ... but the round trip time (RTT) is reasonably large

HTTP step by step

Optimizing HTTP

In HTTP, the **`Connection`** header requests the server to keep the TCP connection active, or persistent.

- **`Connection: keep-alive`**
- **`Connection: close`**

Using **`Connection: keep-alive`**, how many RTTs would it take to load the same website?

Response time for whole web page

Nonpersistent HTTP

- Requires 2 RTTs per object plus file transmission time
- Plus, OS overhead for each TCP connection

Persistent HTTP

- Server leaves TCP connection open after sending response
- Subsequent HTTP messages between same client/server sent over same open connection

Optimizing HTTP

Nearly all modern browsers make at least 2 connections to every web server. This allows for requests to be made in parallel.

Using Connection: keep-alive and three parallel connections, how many RTTs would it take to load the same website?

Optimizing HTTP

In HTTP/1.1, a new feature called HTTP Pipelining allowed multiple requests to be made in one header.

- Request all five images at once!

Using Connection: keep-alive, HTTP pipelining, and only one connection, how many RTTs would it take to load the same website?

Aside:
Do a few RTTs matter?

Collective experiment

```
ping your_favorite_domain.foo
```

MP8 Overview

Goal: Build a simple HTTP web server.