CS 43: Computer Networks

HTTP

September 10, 2018

Announcements

- Sigma-Xi Poster Session in the Eldridge Commons
  - Thursday, Sept 13, from 3:00 to 5:00
  - Friday, Sept 14, from 11:30 to 2:00
- Clicker participation counts from today onwards.
- Readings are mandatory
  - Reading quizzes from next class (points based on participation not correctness)

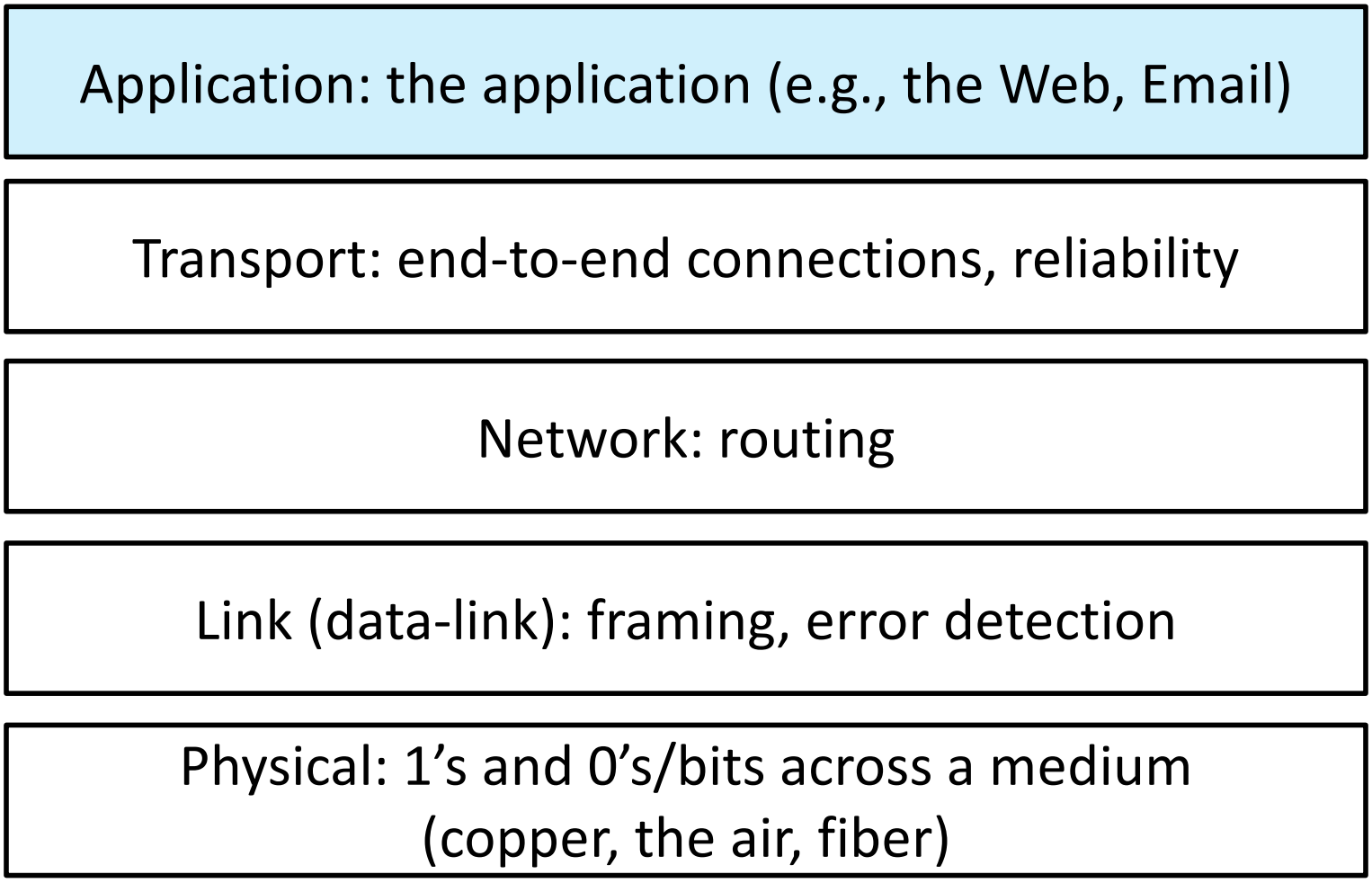
Last class

- End-to-end argument
- Five-layer protocol stack
  - Protocols at each layer
- Example HTTP Request

Today

- HTTP
  - GET vs. POST
  - response messages
  - Persistence vs. Non-persistence
- HTTP Performance and Cookies
- Server-side Socket Programming

Last class: Five-Layer Internet Model



Last class: HTTP Overview

1. User types in a URL.
2. Browser establishes connection with server.
3. Browser requests the corresponding data.
4. Server responds with the requested data.
5. Browser renders the response, fetches other objects, and closes the connection.

It's a document retrieval system, where documents point to (link to) each other, forming a "web".

Last class: Example

```
$ telnet demo.cs.swarthmore.edu 80
Trying 130.58.68.26...
Connected to demo.cs.swarthmore.edu.
Escape character is '^]'.
GET / HTTP/1.1
Host: demo.cs.swarthmore.edu
```

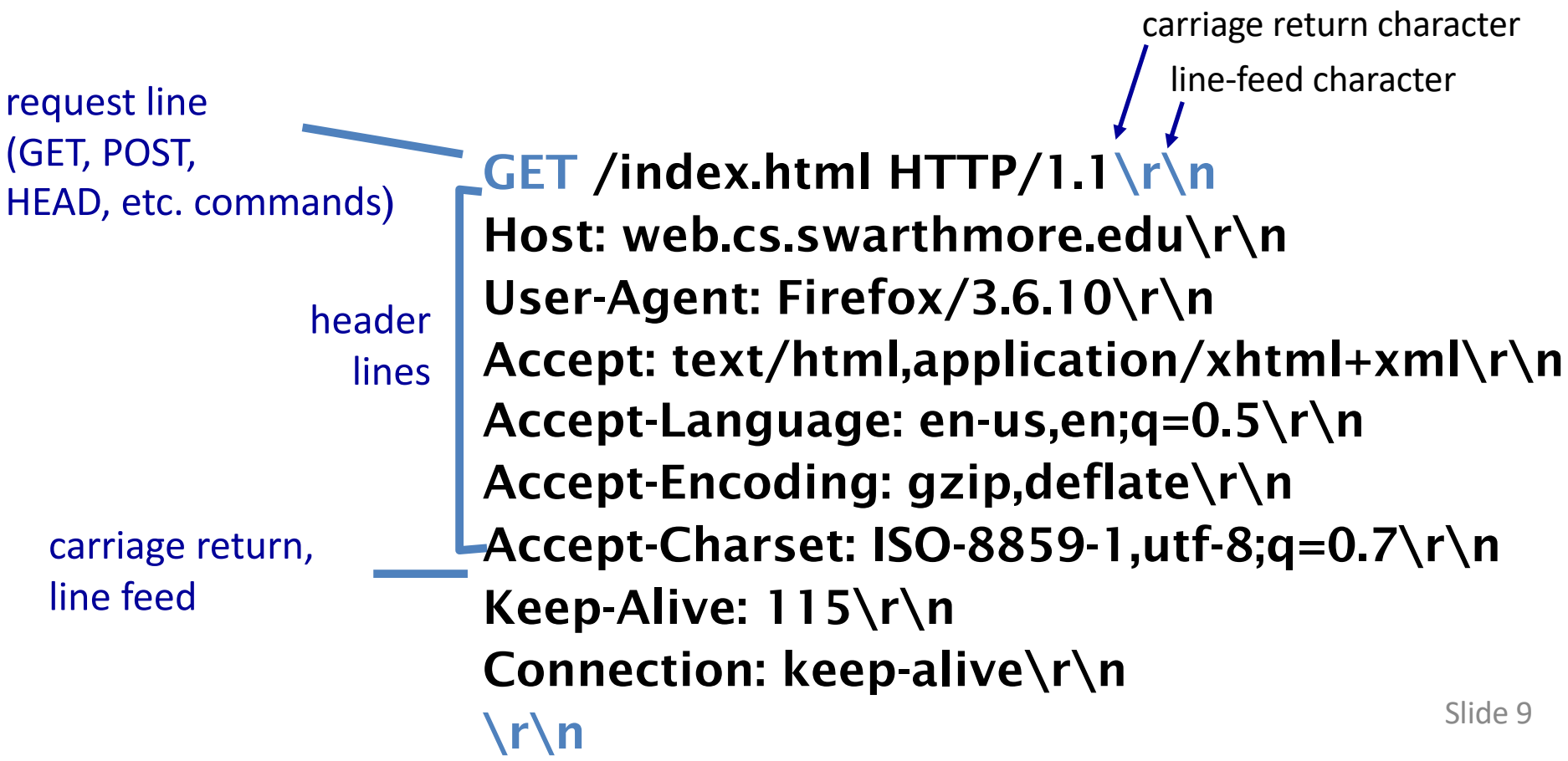
Response headers

```
<html><head><title>Demo Server</title></head>
<body>
.....
</body>
</html>
```

Response body (This is what you should be saving in lab 1.)

HTTP request message

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



Why do we have these \r\n (CRLF) things all over the place?

```
GET /index.html HTTP/1.1\r\n
Host: web.cs.swarthmore.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

A. They're generated when the user hits 'enter'.
B. They signal the end of a field or section.
C. They're important for some other reason.
D. They're an unnecessary protocol artifact.

Why do we have these \r\n (CRLF) things all over the place?

```
GET /index.html HTTP/1.1\r\n
Host: web.cs.swarthmore.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

A. They're generated when the user hits 'enter'.

B. They signal the end of a field or section.

C. They're important for some other reason.

D. They're an unnecessary protocol artifact.

How else might we delineate messages?

A. There's not much else we can do.

B. Force all messages to be the same size.

C. Send the message size prior to the message.

D. Some other way (discuss).

HTTP is all text…

- Makes the protocol simple
  - Easy to delineate message (\r\n)
  - (Relatively) human-readable
  - No worries about encoding or formatting data
  - Variable length data
- Not the most efficient
  - Many protocols use binary fields
    - Sending “12345678” as a string is 8 bytes
    - As an integer, 12345678 needs only 4 bytes
  - The headers may come in any order
  - Requires string parsing / processing

Request Method Types ("verbs")

HTTP/1.0 (1996):

- GET:
  - Requests page.
- POST:
  - Uploads user response to a form.
- HEAD:
  - asks server to leave requested object out of response

HTTP/1.1 (1997 & 1999):

- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field
- TRACE, OPTIONS, CONNECT, PATCH
- Persistent connections

Uploading form input

GET (in-URL) method:

- uses GET method
- input is uploaded in URL field of request line:

  www.somesite.com/animalsearch?monkeys&banana

POST method:

- web page often includes form input
- input is uploaded to server in request entity body

GET vs. POST

GET can be used for idempotent requests

- Idempotence: an operation can be applied multiple times without changing the result (the final state is the same)

GET vs. POST

GET can be used for idempotent requests

- Idempotence: an operation can be applied multiple times without changing the result (the final state is the same)

Q: How many of the following operations are idempotent?

I. Incrementing a variable
II. Assigning a value to a variable
III. Allocating Memory
IV. Compiling a program

A. None of them
B. One of them
C. Two of them
D. Three of them
E. All of them

GET vs. POST

GET can be used for idempotent requests

- Idempotence: an operation can be applied multiple times without changing the result (the final state is the same)

Q: How many of the following operations are idempotent?

I. Incrementing a variable
II. Assigning a value to a variable
III. Allocating Memory
IV. Compiling a program

A. None of them
B. One of them
C. Two of them
D. Three of them
E. All of them

GET vs. POST

GET can be used for idempotent requests.

- Idempotence: an operation can be applied multiple times without changing the result (the final state is the same)

POST should be when:

- A request changes the state of the server or DB
- Sending a request twice would be harmful: (Some) browsers warn about sending multiple post requests
- Users are inputting non-ASCII characters
- Input may be very large
  - You want to hide how the form works/user input

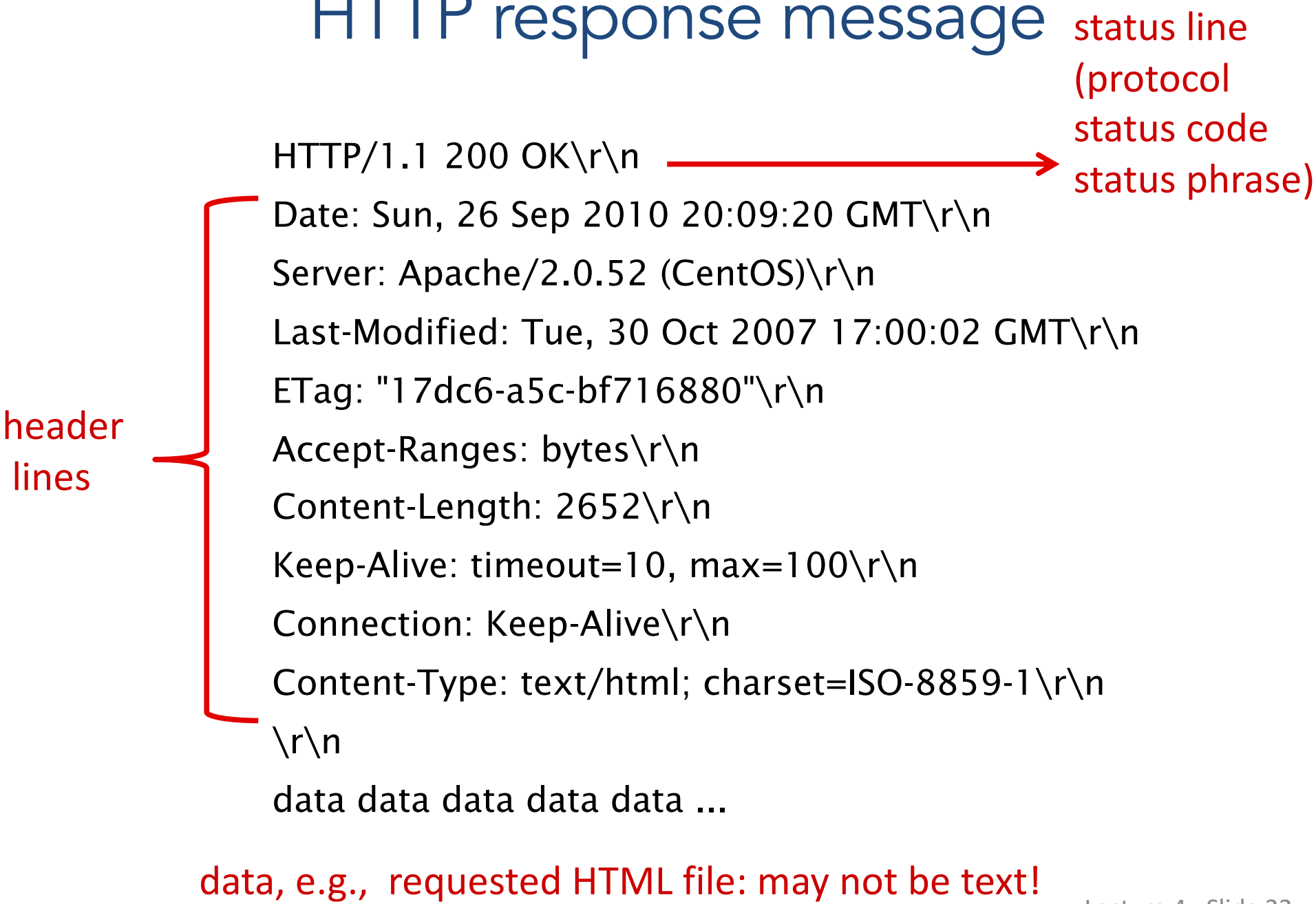
When might you use GET vs. POST?

	GET	POST
A.	Forum post	Search terms, Pizza order
B.	Search terms, Pizza order	Forum post
C.	Search terms	Forum post, Pizza order
D.	Forum post, Search terms, Pizza Order	
E.		Forum post, Search terms, Pizza Order

When might you use GET vs. POST?

	GET	POST
A.	Forum post	Search terms, Pizza order
B.	Search terms, Pizza order	Forum post
C.	Search terms	Forum post, Pizza order
D.	Forum post, Search terms, Pizza Order	
E.		Forum post, Search terms, Pizza Order

HTTP response message



HTTP response status codes

Status code appears in first line of server-to-client response message.

200 OK

- Request succeeded, requested object later in this msg

301 Moved Permanently

- Requested object moved, new location specified later in this msg (Location:)

400 Bad Request

  - Request msg not understood by server

403 Forbidden

  - You don't have permission to read the object

404 Not Found

  - Requested document not found on this server

505 HTTP Version Not Supported

HTTP response status codes

Status code appears in first line of server-to-client response message.

Many others! Search “list of HTTP status codes”

420 Enhance Your Calm (twitter)

- Slow down, you’re being rate limited

451 Unavailable for Legal Reasons

- Censorship?

418 I’m a Teapot

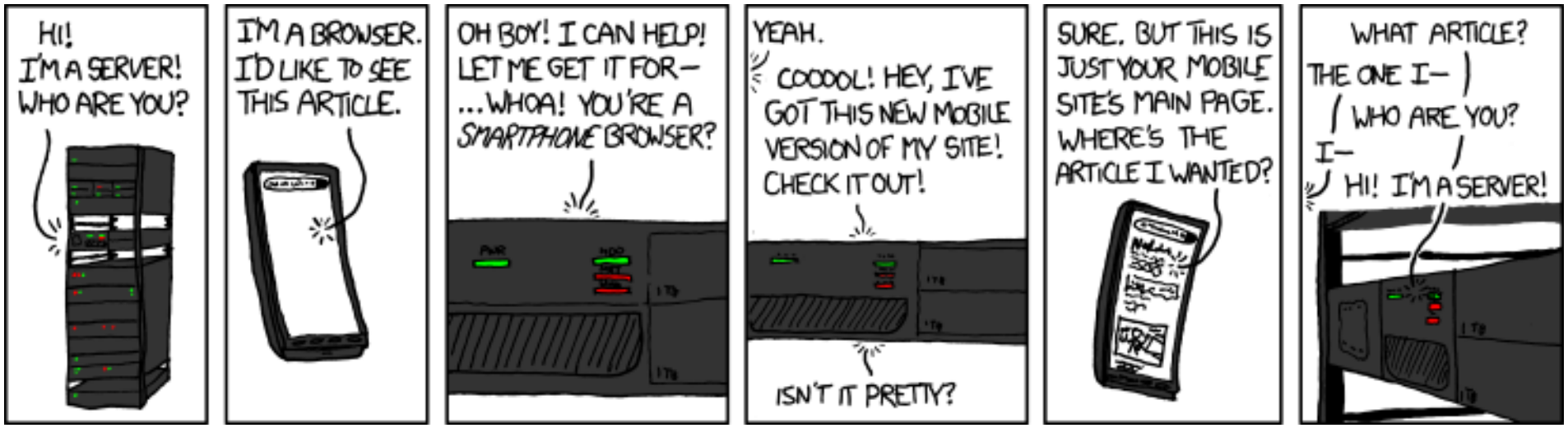
- Response from a teapot requested to brew a beverage (announced Apr 1)

HTTP State

Does the HTTP protocol, allow for a server to keep track of every client?

A. Yes, it's required to
B. No, it would not scale
C. That's against privacy rules!
D. Something else

State(less)



(XKCD #869, “Server Attention Span”)

State(less)

- Original web: simple document retrieval
- Server is not required to keep state between connections
  - often it might want to though
- Client is not required to identify itself
  - server might refuse to talk otherwise though

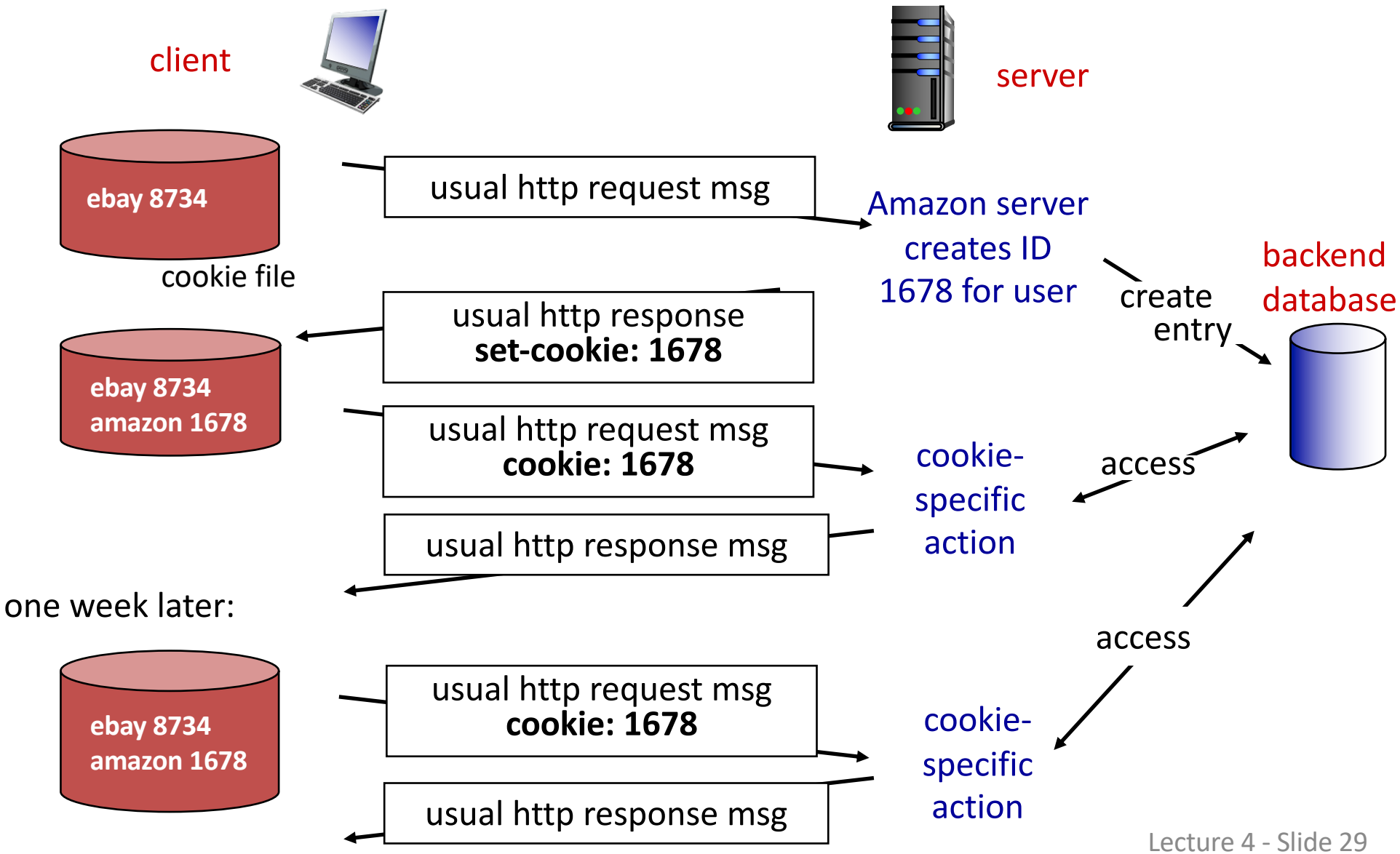
User-server state: cookies

Many web sites use cookies

Four components:

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

Cookies: keeping "state" (cont.)



Cookies (continued)

What cookies can be used for:

- 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 (and more!)
- Third-party cookies (from ad networks, etc) can follow you across multiple sites.
  - Ever visit a website, and the next day ALL your ads are from them?
- You could turn them off
  - But good luck doing anything on the internet!

HTTP connections

Non-persistent HTTP

- at most one object sent over TCP connection
  - connection then closed
- downloading multiple objects requires multiple connections

Persistent HTTP

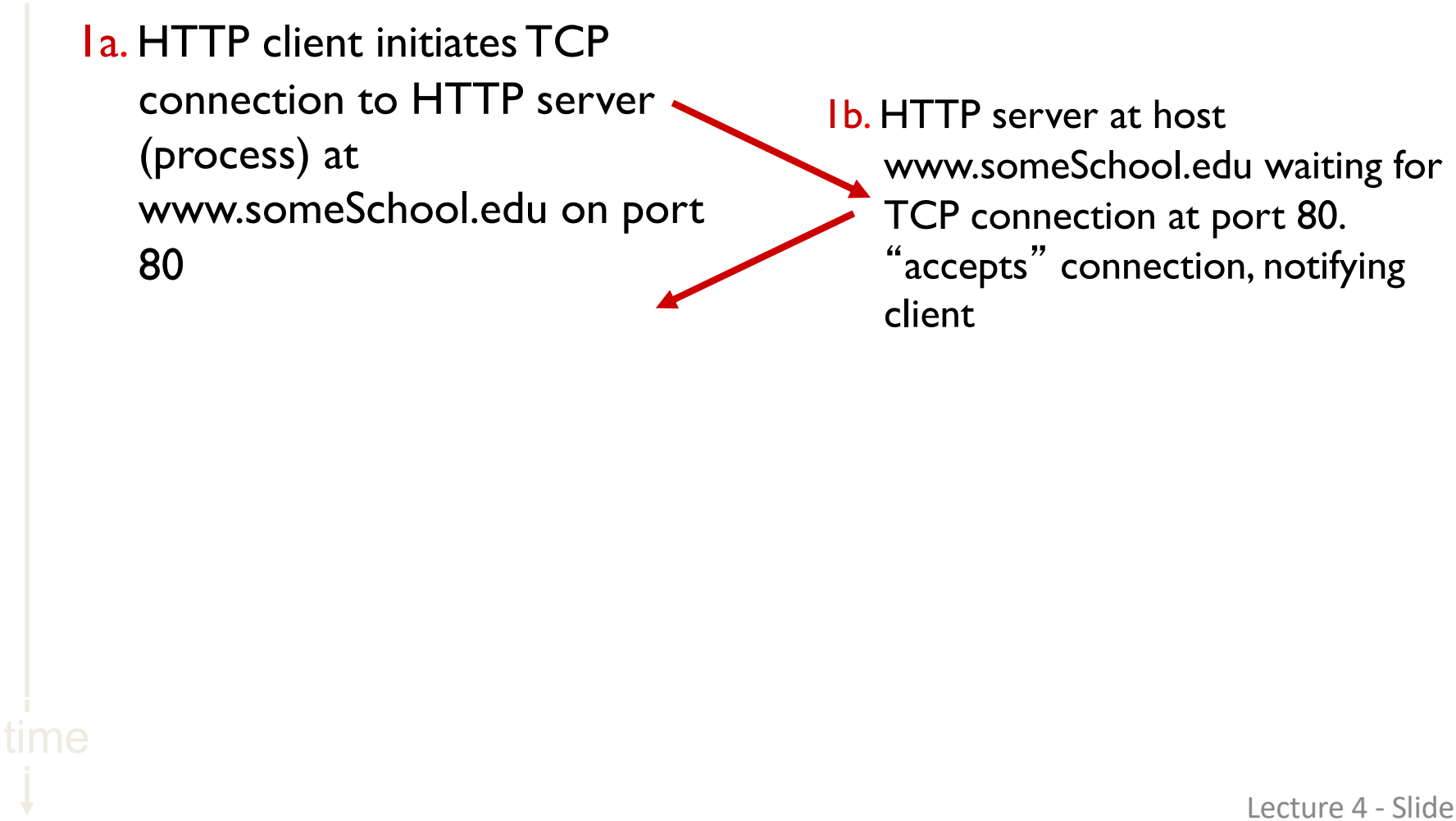
- multiple objects can be sent over single TCP connection between client, server

object: image, script, stylesheet, etc.

Non-persistent HTTP

suppose user enters URL:
`www.someSchool.edu/someDepartment/home.html`

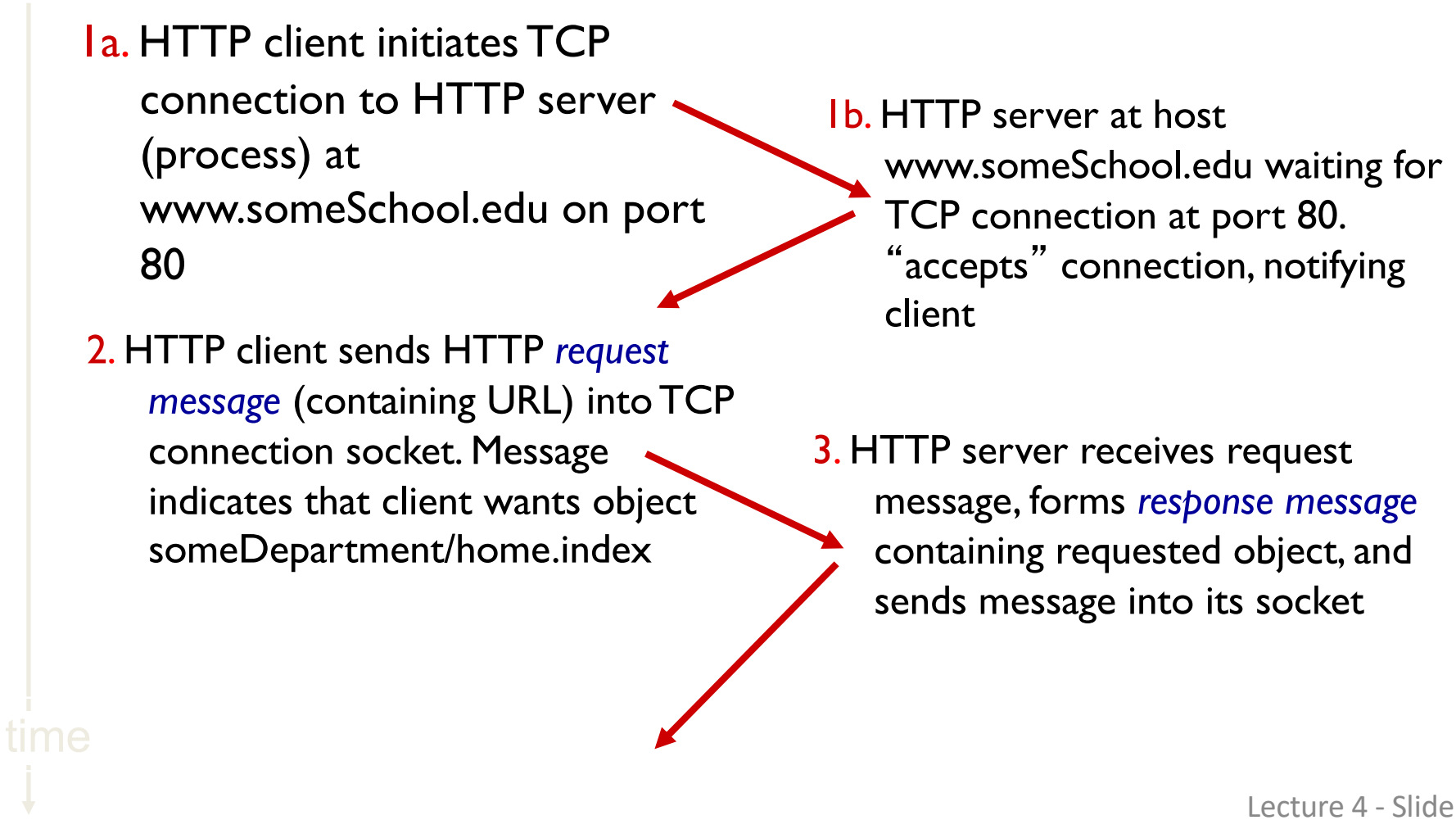
(contains text, references to 10 jpeg images)



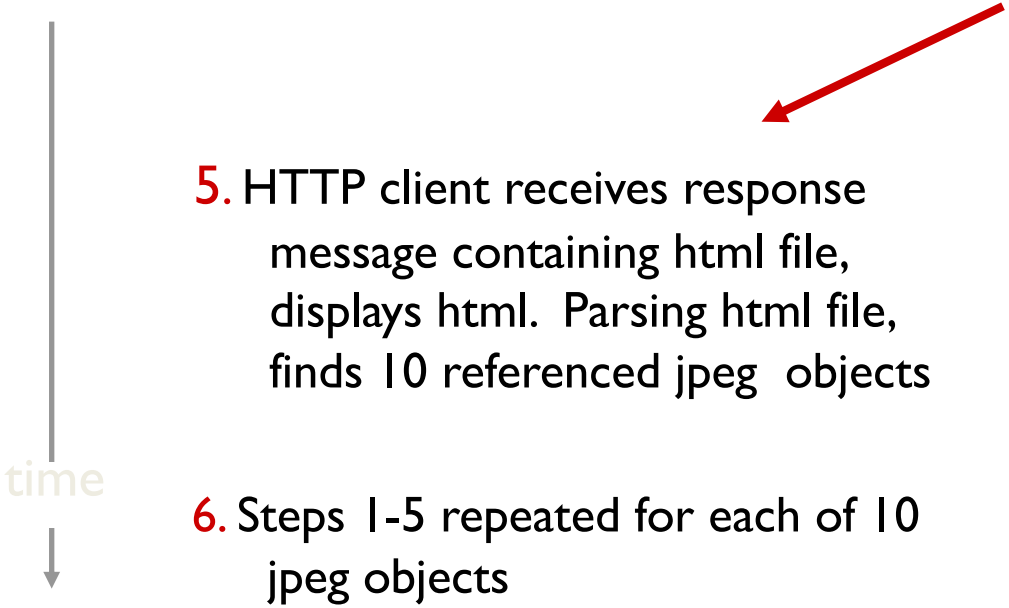
Non-persistent HTTP

suppose user enters URL:
`www.someSchool.edu/someDepartment/home.html`

(contains text, references to 10 jpeg images)



Non-persistent HTTP (cont.)



4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

Pseudocode Example

non-persistent HTTP

```
for object on web page:
    connect to server
    request object
    receive object
    close connection
```

persistent HTTP

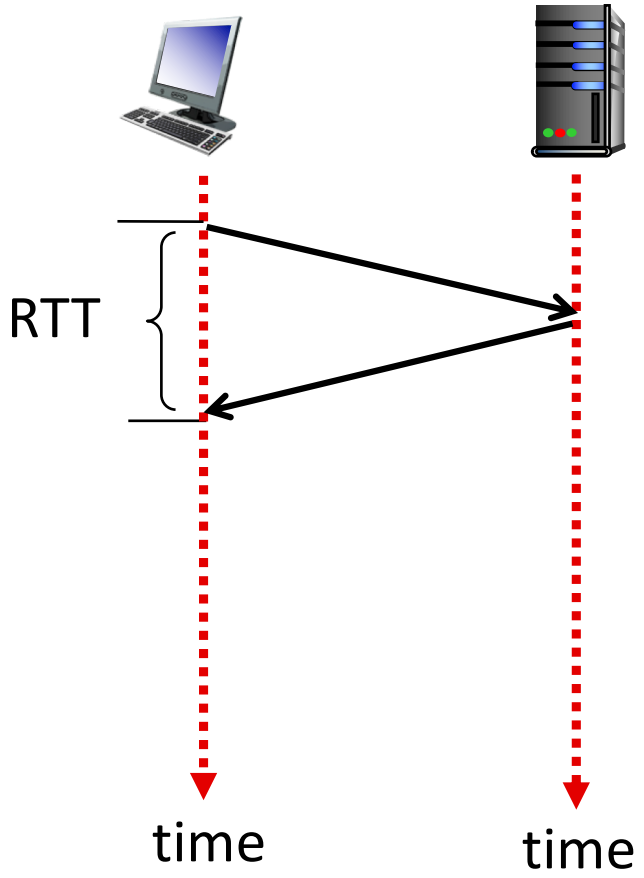
```
connect to server
for object on web page:
    request object
    receive object
close connection
```

Round Trip Time

Round Trip Time (RTT):

- time for a small packet to travel from client to server and response to come back.

- Connection establishment (via TCP) requires one RTT.



Non-Persistent HTTP Connections can download a website with several objects in…

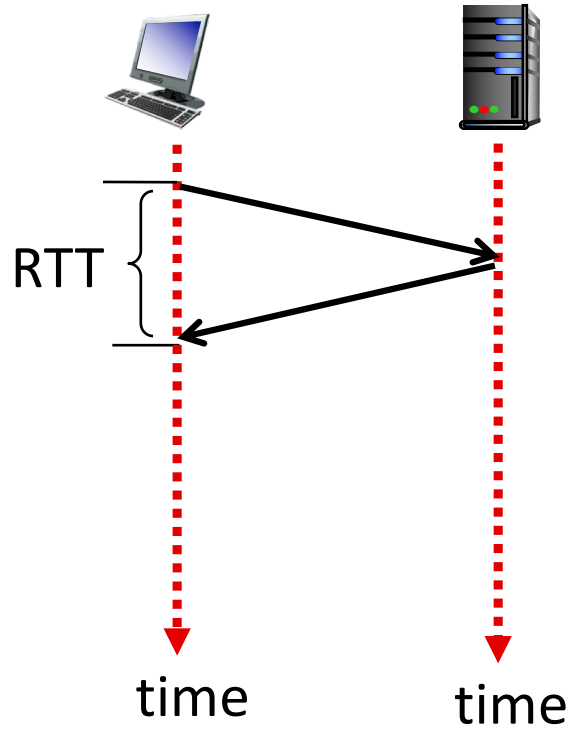
A. One RTT + (File transfer time per object)

B. (One RTT + File transfer time) per object

C. Two RTTs

D. Two RTTs + (File transfer time per object)

E. (Two RTTS + File transfer time) per object



Non-persistent HTTP: response time

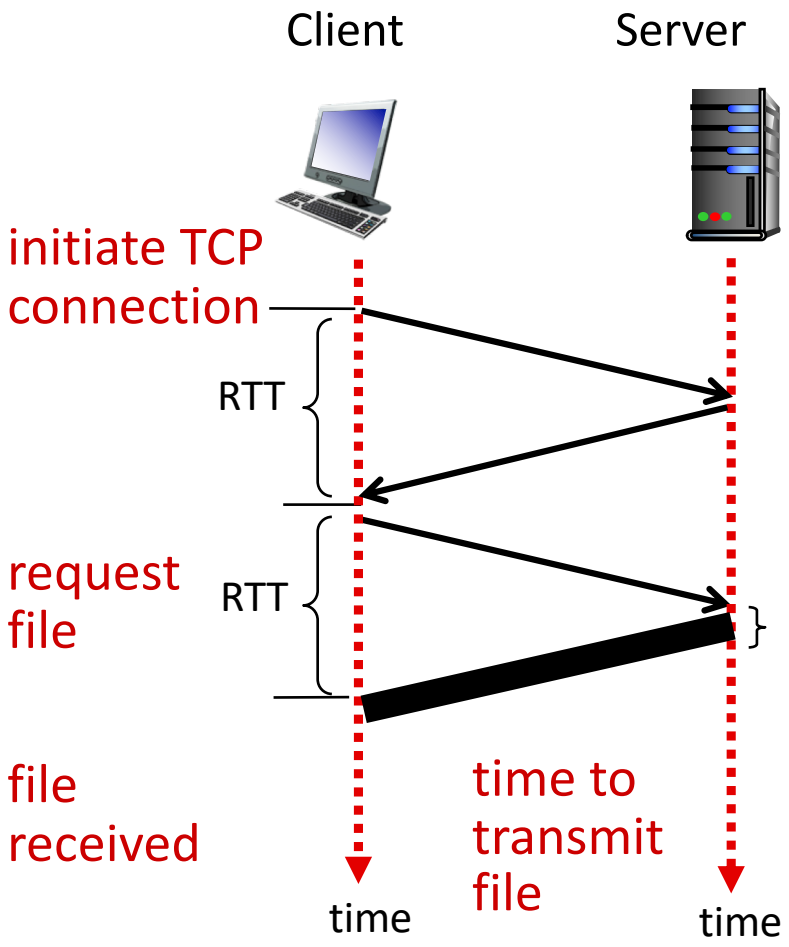
Round Trip Time (RTT): time for a small packet to travel from client to server and back

HTTP response time:

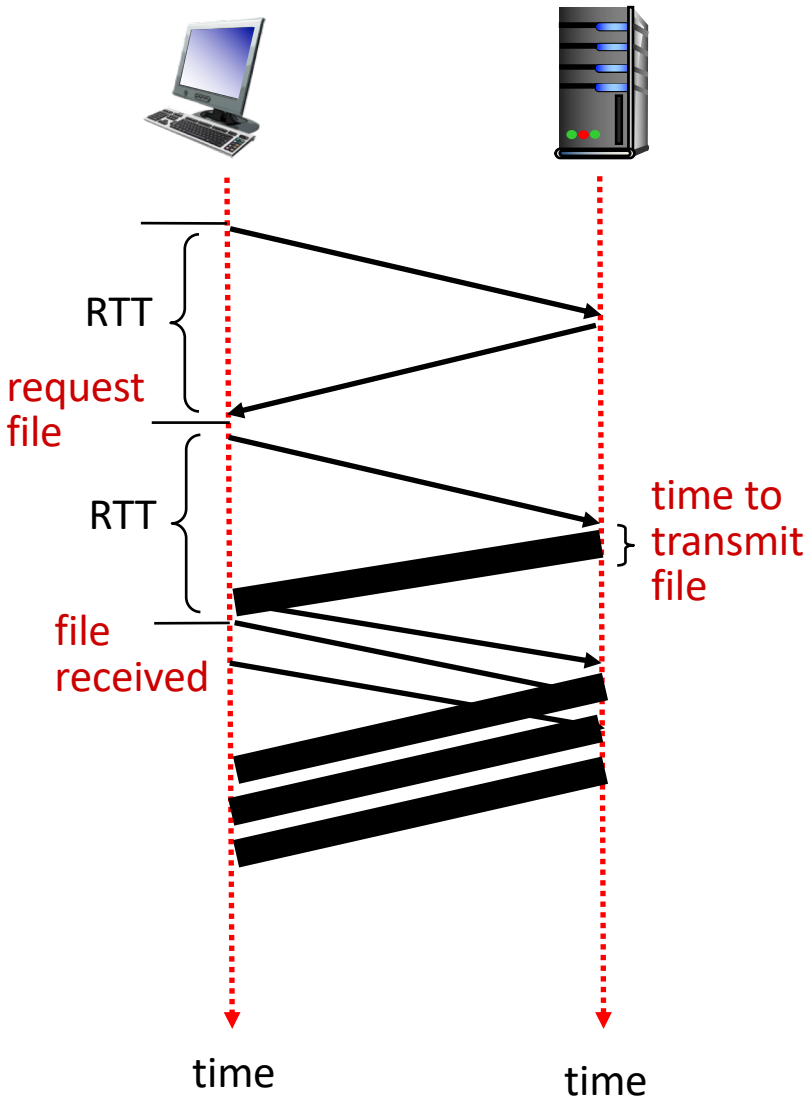
- 1-RTT to initiate TCP connection
- 1-RTT for HTTP request + first few bytes of HTTP response to return
- file transmission time
- non-persistent HTTP response time =

  2-RTT+ file transmission time

  For each object



Persistent Connection



Persistent HTTP

Non-persistent HTTP issues:

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

Persistent HTTP:

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects