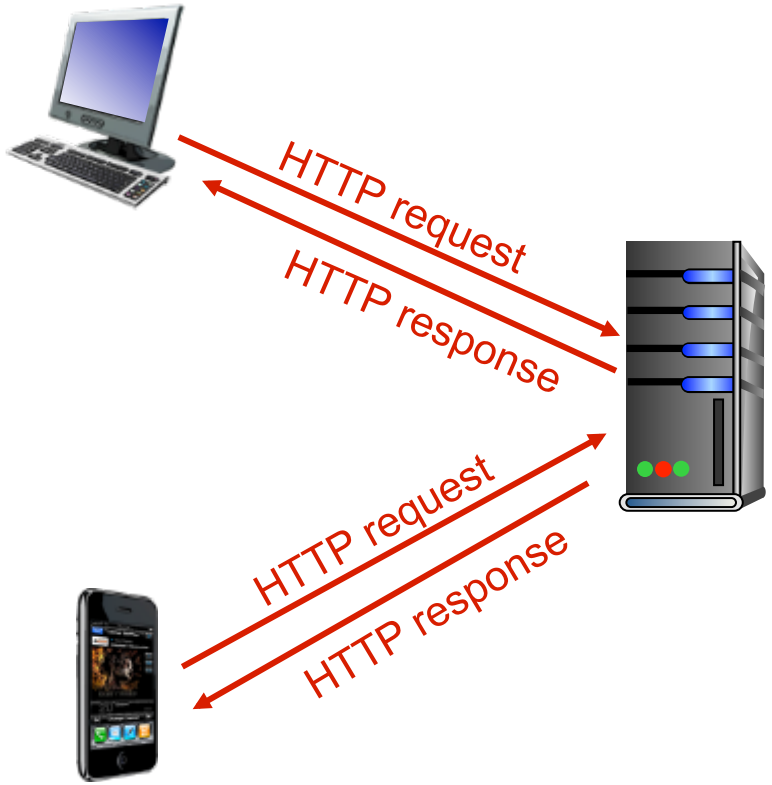
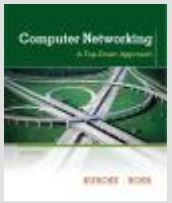
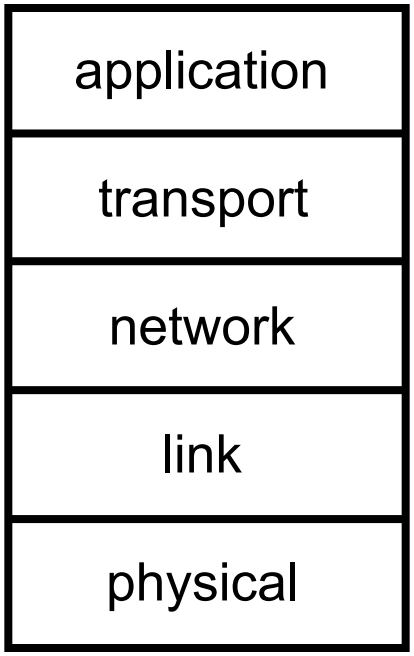
HTTP, cookies, and caching



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley



Overview



- Chapter 2: Application Layer
  - Many familiar services operate here
    - Web, email, Skype, P2P file sharing
  - Socket programming
- HTTP
  - Statelessness
  - Non-persistent vs. persistent connections
  - Cookies
  - Caching

HTTP: TCP + stateless

Uses TCP:

- Client initiates TCP connection (creates socket) to server, port 80
- Server accepts TCP connection from client
- HTTP messages (application-layer 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

Trying out HTTP for yourself

1. Telnet to your favorite Web server:

telnet cis.poly.edu 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:

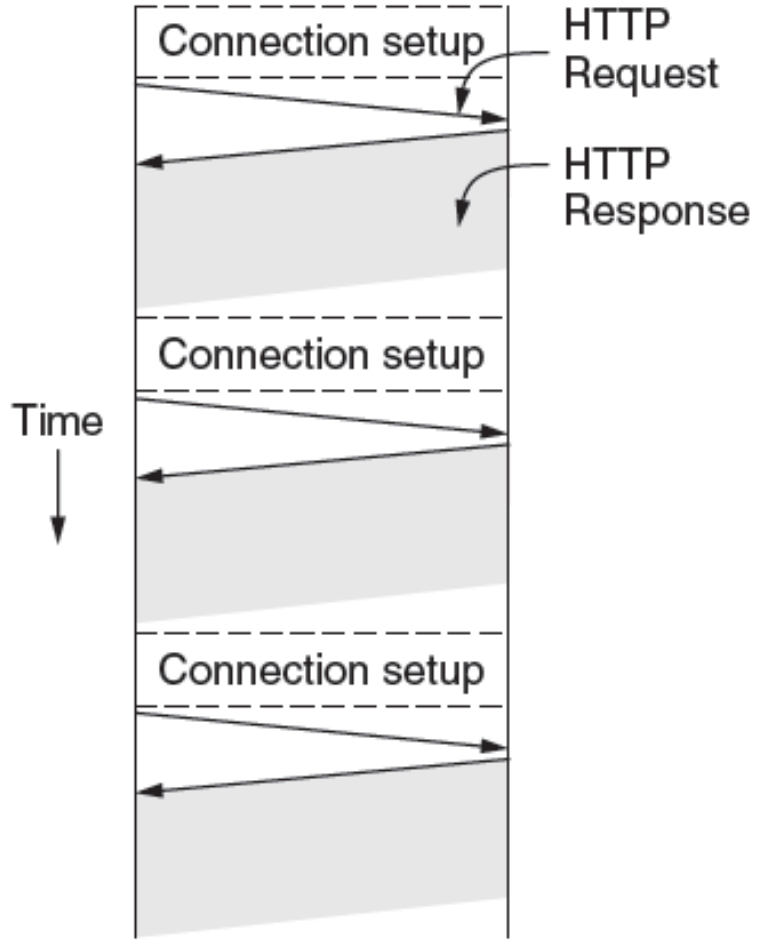
GET /~ross/ HTTP/1.1
Host: cis.poly.edu

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

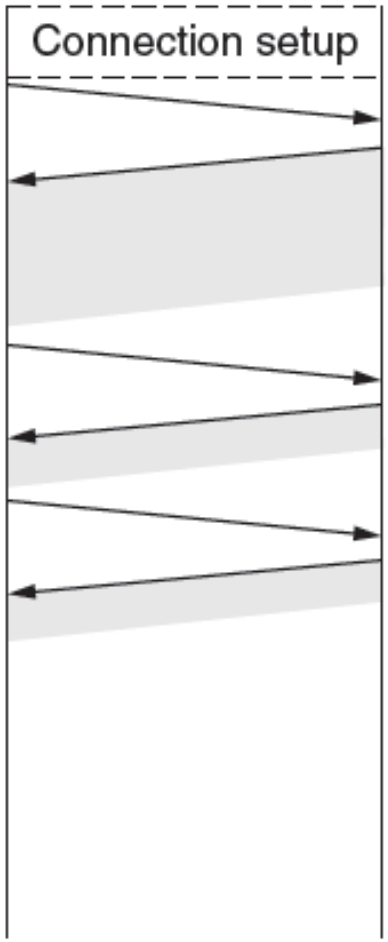
3. Look at response message sent by HTTP server!

(or use Wireshark to look at captured HTTP request/response)

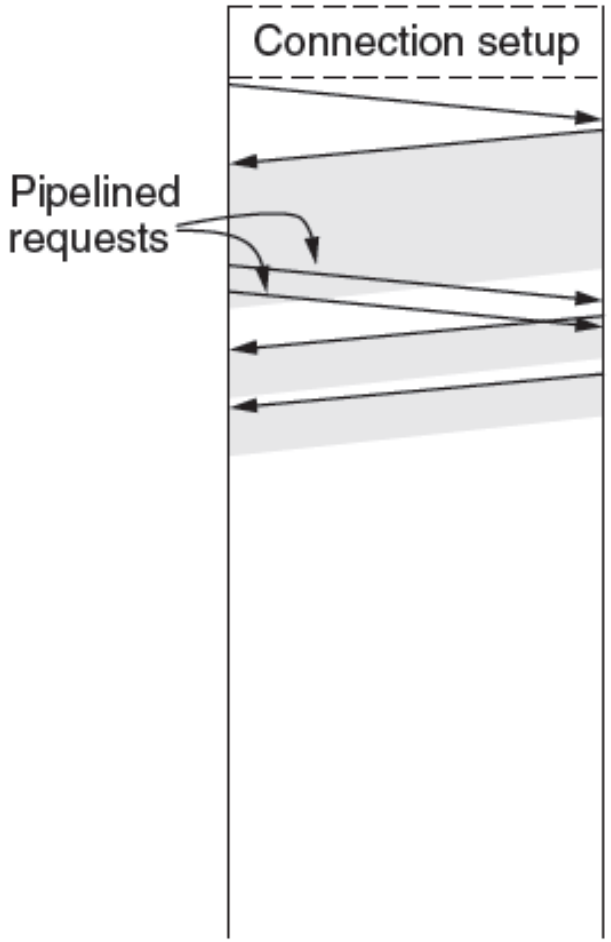
HTTP communication options



Multiple connections and sequential requests.



Persistent connection and sequential requests.



Persistent connection and pipelined requests.

Persistent HTTP

Non-persistent HTTP

- At most one object sent over TCP connection
  - connection then closed
- Downloading multiple objects required multiple connections

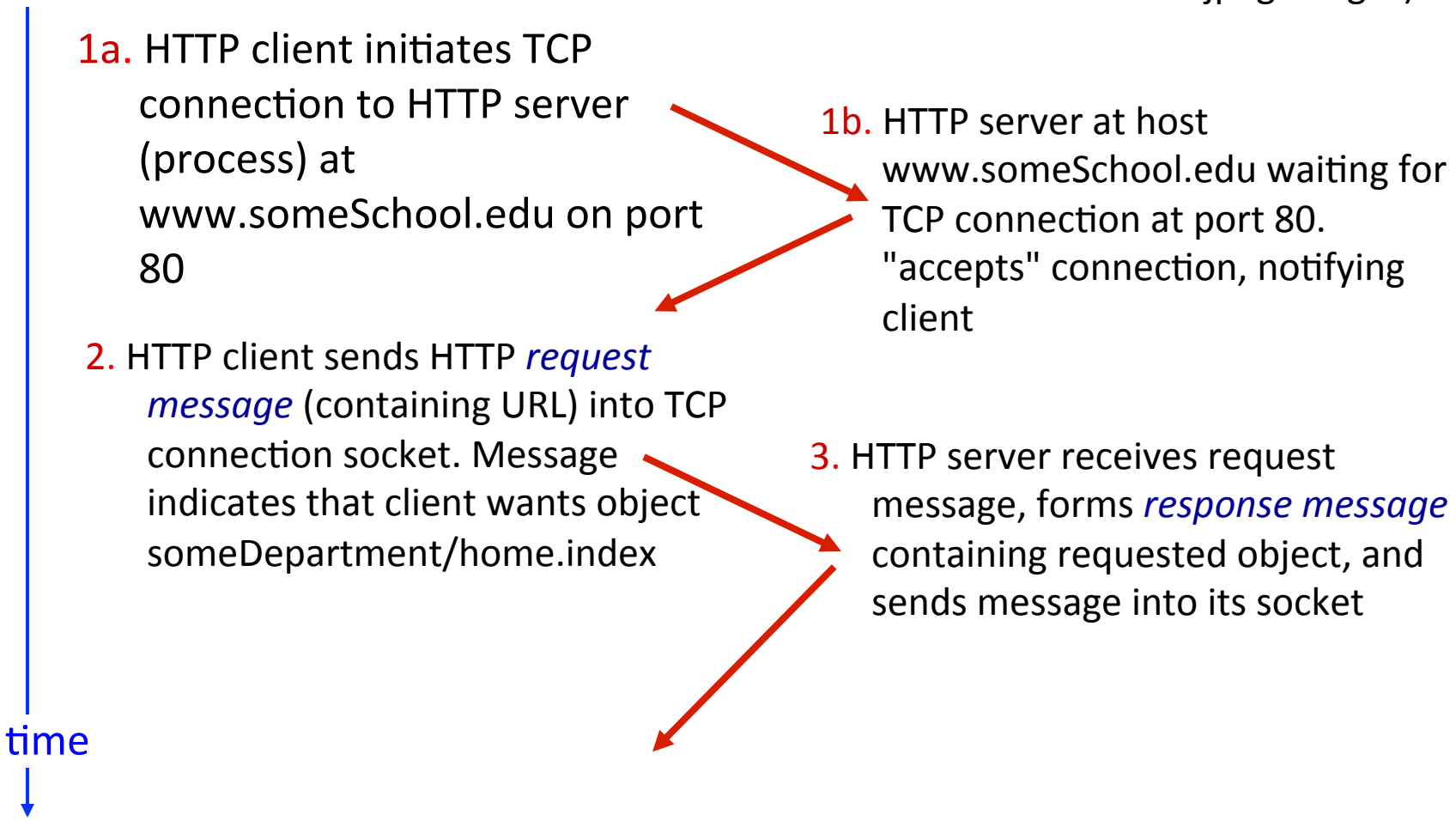
Persistent HTTP

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

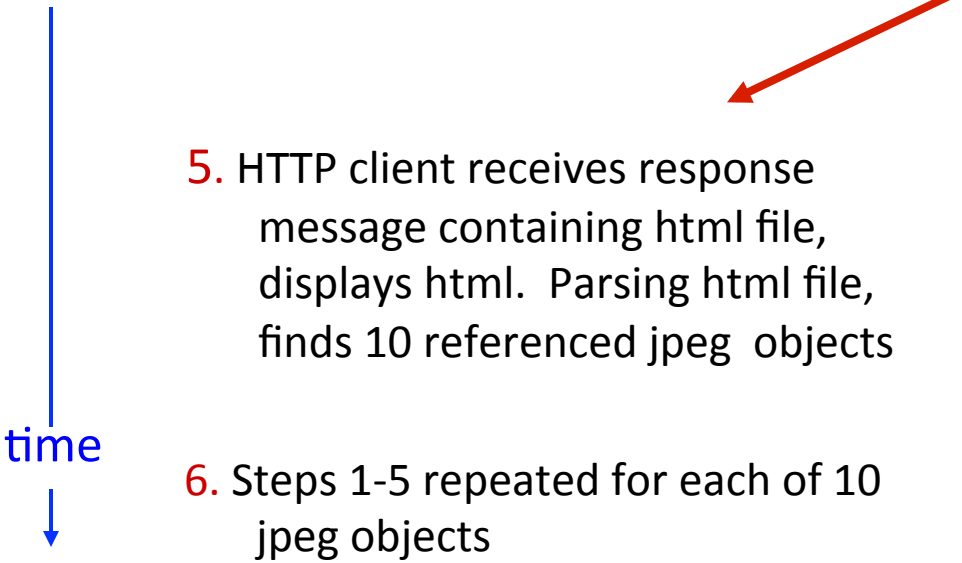
Non-persistent HTTP

User enters URL:
`www.someSchool.edu/someDepartment/home.index`

(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

Non-persistent HTTP: response time

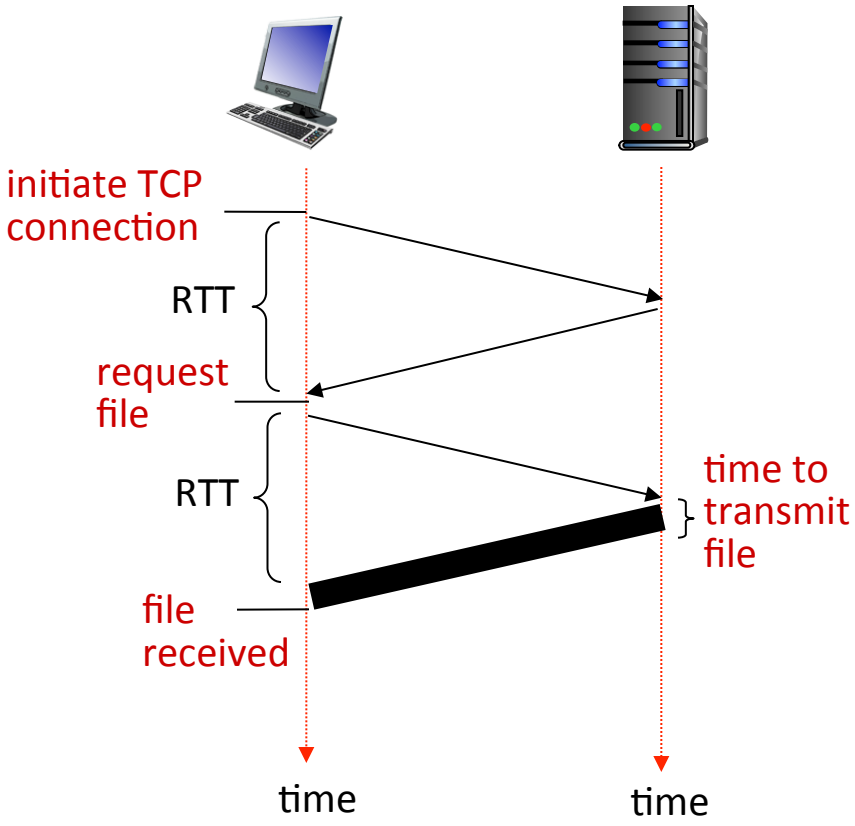
RTT:

Time for small packet to travel from client to server and back

HTTP response time:

- One RTT to initiate TCP connection
- One RTT for HTTP request and first few bytes of HTTP response to return
- File transmission time
- Non-persistent HTTP response time:

2RTT+ file transmission time



Persistent HTTP

Non-persistent 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

User-server state: cookies

Many Web sites use cookies

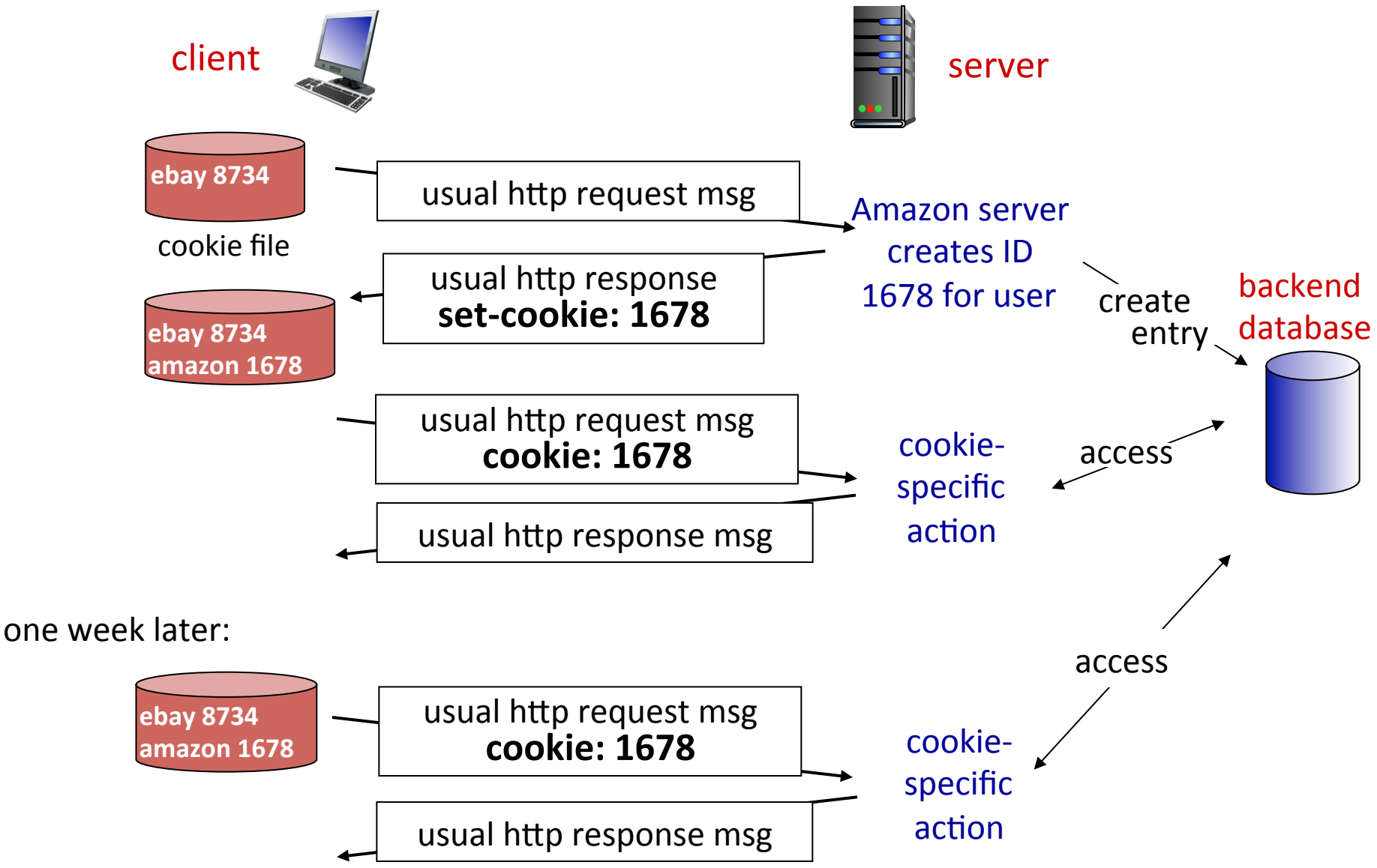
Four components:

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

Example:

- Susan always access 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



Cookies

What cookies can be used for:

- Authorization
- Shopping carts
- Recommendations
- User session state
  - e.g. web e-mail

aside

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":

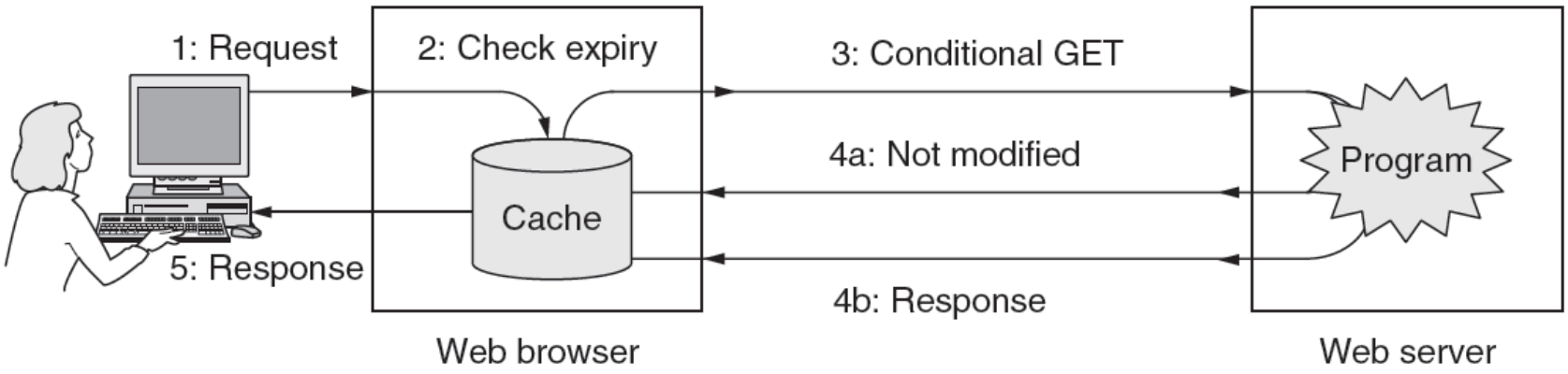
- Protocol endpoints: maintain state at sender/receiver over multiple transactions
- Cookies: HTTP messages carry state

Improving performance

- How do we make things faster?
- Minimize traffic/latency between client/server
  - Conditional requests
  - Caching
  - Compression
- Speed up server's response
  - Multiple servers
  - Geographically distributed servers
  - Content delivery networks

HTTP Caching

- Clients often cache documents
  - How and when should they check for changes?
- HTTP has cache related headers
  - HTTP/1.0: "Expires: <date>"; "Pragma: no-cache"
  - HTTP/1.1:
    - Cache-Control: No-Cache, Private, Max-age: <seconds>
    - E-tag: <some value>



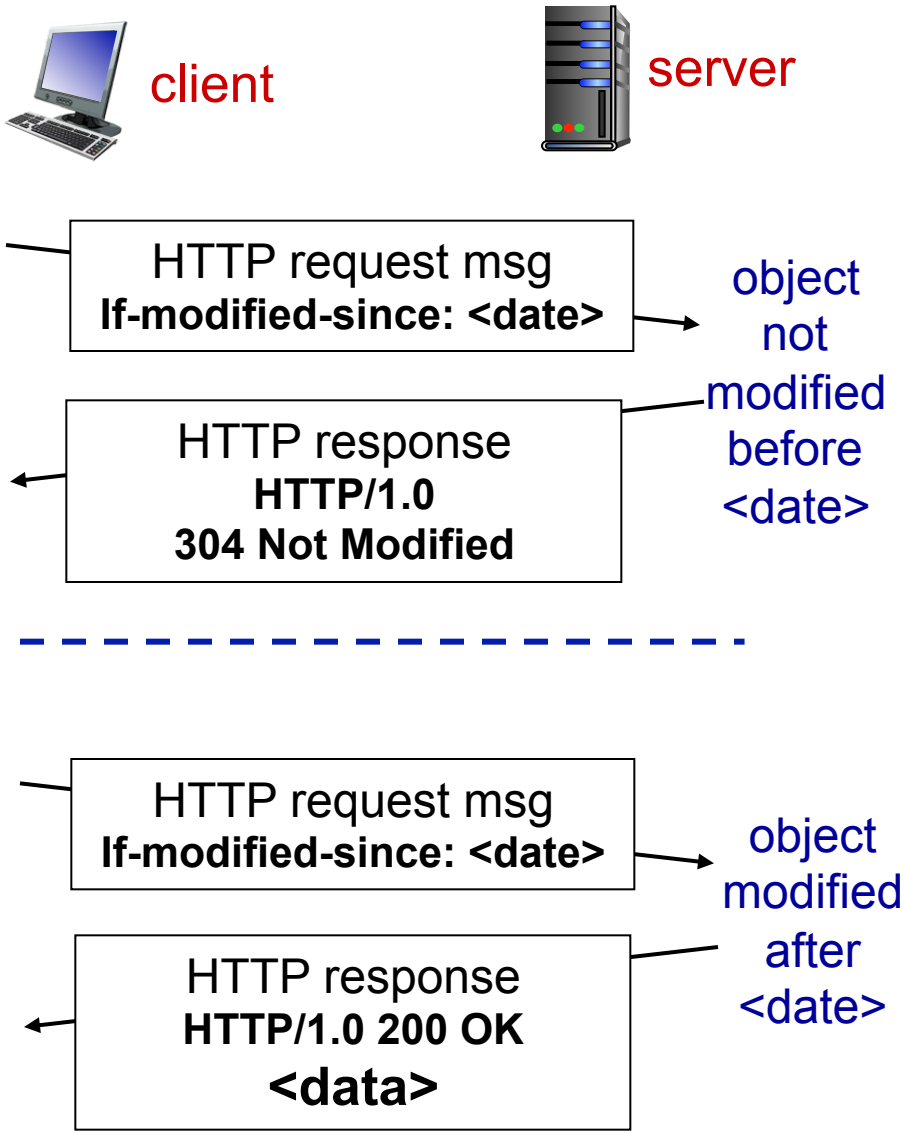
Conditional GETs

- Conditional GET
  - Fetch resource only if it has changed
  - Server avoids wasting resources to send again
  - Client sets "if-modified-since" header field
    - Server inspects "last modified" time of object
    - Returns "304 Not Modified" if unchanged, otherwise "200 OK" and new version.
  - Client sets "if-no-match" using previous received ETag for the desired object
    - Server compares with current "hash" of object

HTTP conditional GET

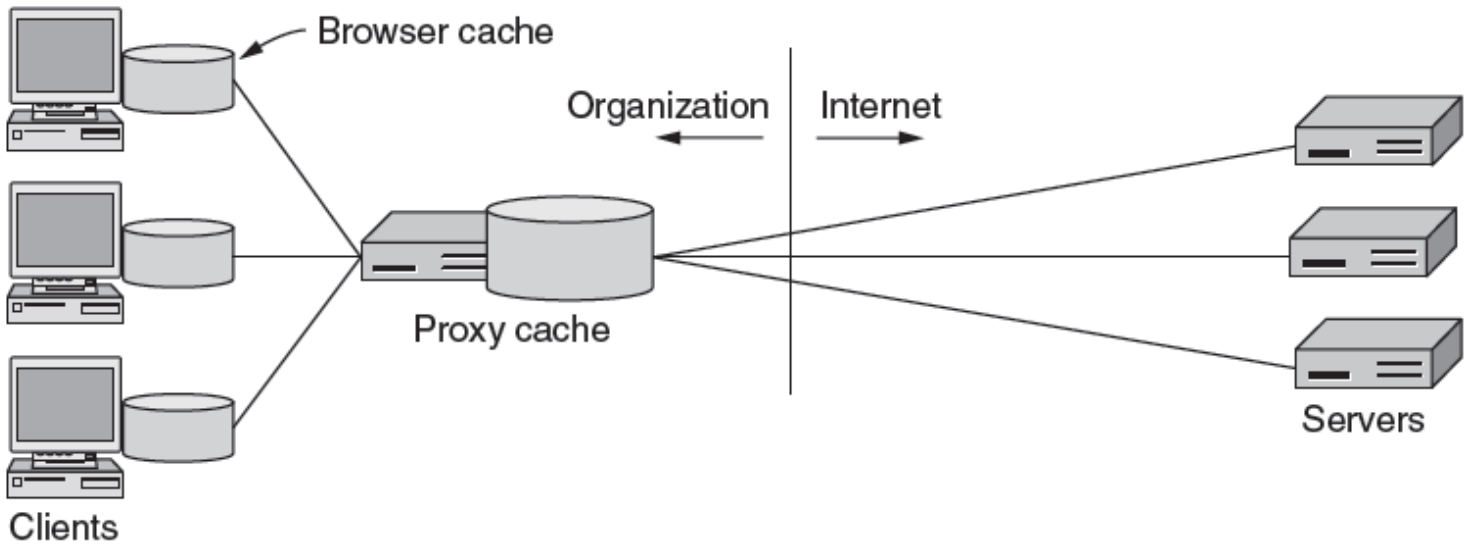
GET / HTTP/1.1
Host: katie.mtech.edu
Connection: keep-alive
User-Agent: Mozilla/5.0
Accept: text/html, application/xhtml+xml
Accept-Encoding: gzip,deflate,sdch
If-None-Match: "c-221-4ace9c0b09cc0"
If-Modified-Since: Wed, 14 Sep 2011 17:04:27 GMT

HTTP/1.1 304 Not Modified
Date: Thu, 17 Nov 2011 16:57:53 GMT
Server: Apache/2.2.16 (Debian)
Keep-Alive: timeout-15, max=100
Connection: Keep-Alive
ETag: " c-221-4ace9c0b09cc0"



Levels of caching

- Caching can occur at many levels:
  - In the client's browser
  - Client configures browser to use web proxy
  - Proxy at the ISP
  - "voluntary proxy" versus intercepting/forced/ transparent



Summary

- HTTP protocol
  - Statelessness
    - Helps keep things simple and scale
  - Non-persistent vs. persistent connections
    - Avoid 2RTT for each object retrieved
  - Cookies
    - Adding state to web interactions
  - Caching
    - Improve performance
    - Reduce bandwidth requirements