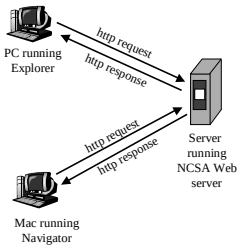
Lecture 20 (April 8, 2004)

Outline
- HTTP
- Caching
- CDNs

The Web: the http protocol

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
- http1.0: RFC 1945
- http1.1: RFC 2068



The http protocol: more

http: TCP transport service:
- 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 maintains no information about past client requests

aside

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

http example

Suppose user enters URL www.someSchool.edu/someDepartment/home.index (contains text, references to 10 jpeg images)

1a. http client initiates TCP connection to http server (process) at www.someSchool.edu. Port 80 is default for http server.

1b. http server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

2. http client sends http *request message* (containing URL) into TCP connection socket

3. http server receives request message, forms *response message* containing requested object (someDepartment/home.index), sends message into socket

time

http example (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

time

Non-persistent, persistent connections

Non-persistent
- http/1.0: server parses request, responds, closes TCP connection
- 2 RTTs to fetch object
  - TCP connection
  - object request/transfer
- each transfer suffers from TCP's initially slow sending rate
- many browsers open multiple parallel connections

Persistent
- default for htp/1.1
- on same TCP connection: server, parses request, responds, parses new request,..
- client sends requests for all referenced objects as soon as it receives base HTML.
- fewer RTTs, less slow start.

http message format: request

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

request line (GET, POST, HEAD commands)

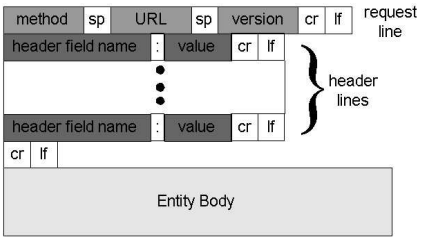
```
GET /somedir/page.html HTTP/1.0
User-agent: Mozilla/4.0
Accept: text/html, image/gif,image/jpeg
Accept-language:fr
```

header lines

(extra carriage return, line feed)

Carriage return, line feed indicates end of message

http request message: general format



http message format: response

status line (protocol status code status phrase)

```
HTTP/1.0 200 OK
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 ….…
Content-Length: 6821
Content-Type: text/html
```

header lines

```
data data data data data ...
```

data, e.g., requested html file

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:)

`400 Bad Request`

- request message not understood by server

`404 Not Found`

- requested document not found on this server

`505 HTTP Version Not Supported`

Trying out http (client side) for yourself

1. Telnet to your favorite Web server:

```
telnet www.eurecom.fr 80
```

Opens TCP connection to port 80 (default http server port) at www.eurecom.fr. Anything typed in sent to port 80 at www.eurecom.fr

2. Type in a GET http request:

```
GET /~ross/index.html HTTP/1.0
```

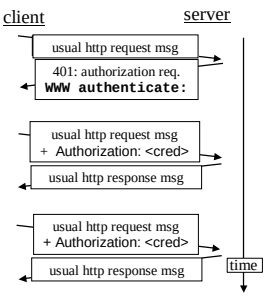
By typing this in (hit carriage return twice), you send this minimal (but complete) GET request to http server

3. Look at response message sent by http server!

User-server interaction: authentication

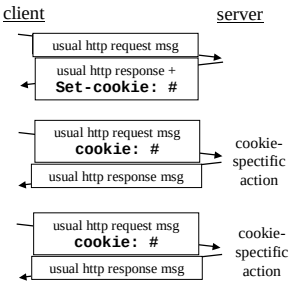
Authentication : control access to server content

- authorization credentials: typically name, password
- stateless: client must present authorization in *each* request
  - authorization: header line in each request
  - if no authorization: header, server refuses access, sends **`WWW authenticate:`** header line in response



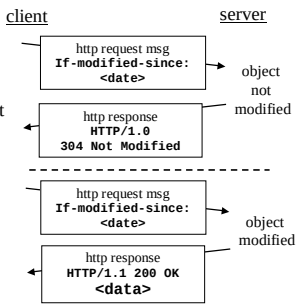
Cookies: keeping "state"

- server-generated # , server-remembered #, later used for:
  - authentication
  - remembering user preferences, previous choices
- server sends "cookie" to client in response msg
  `Set-cookie: 1678453`
- client presents cookie in later requests
  `cookie: 1678453`



Conditional GET: client-side caching

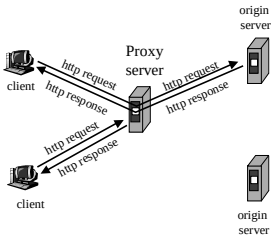
- Goal: don't send object if client has up-to-date cached version
- client: specify date of cached copy in http request
  `If-modified-since: <date>`
- server: response contains no object if cached copy is up-to-date:
  `HTTP/1.0 304 Not Modified`



Web Caches (proxy server)

Goal: satisfy client request without involving origin server

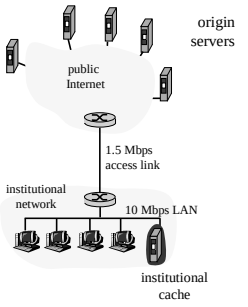
- user sets browser: Web accesses via web cache
- client sends all http requests to web cache
  - object in web cache: web cache returns object
  - else web cache requests object from origin server, then returns object to client



Why Web Caching?

Assume: cache is "close" to client (e.g., in same network)

- smaller response time: cache "closer" to client
- decrease traffic to distant servers
  - link out of institutional/local ISP network often bottleneck
- Hierarchy of caches
  - e.g., Harvest, Squid



Content Delivery Networks

- e.g. Akamai, DigitalIsland, etc.
- Has its own network of servers that replicates some content of the content provider (e.g. cnn.com)
  - e.g. all images
  - In the index.html file all references of:

    www.cnn.com/images/sports.gif is re-mapped to www.akamai.com/www.cnn.com/images/sports.gif
    - Server domain name: www.akamai.com
    - File: www.cnn.com/images/sports.gif

Content Delivery Networks

- When client downloads www.cnn.com/index.html
- Next tries to resolve www.akamai.com/…/sports.gif
- When local nameserver of client tries to resolve www.akamai.com
  - DNS server of Akamai will identify one of its server that is closest to the local nameserver of client
  - Expectation is that the client is close to its local nameserver
- A good approximation of nearest server to client
- Read "King" paper from class website