The Application Layer: DNS, HTTP

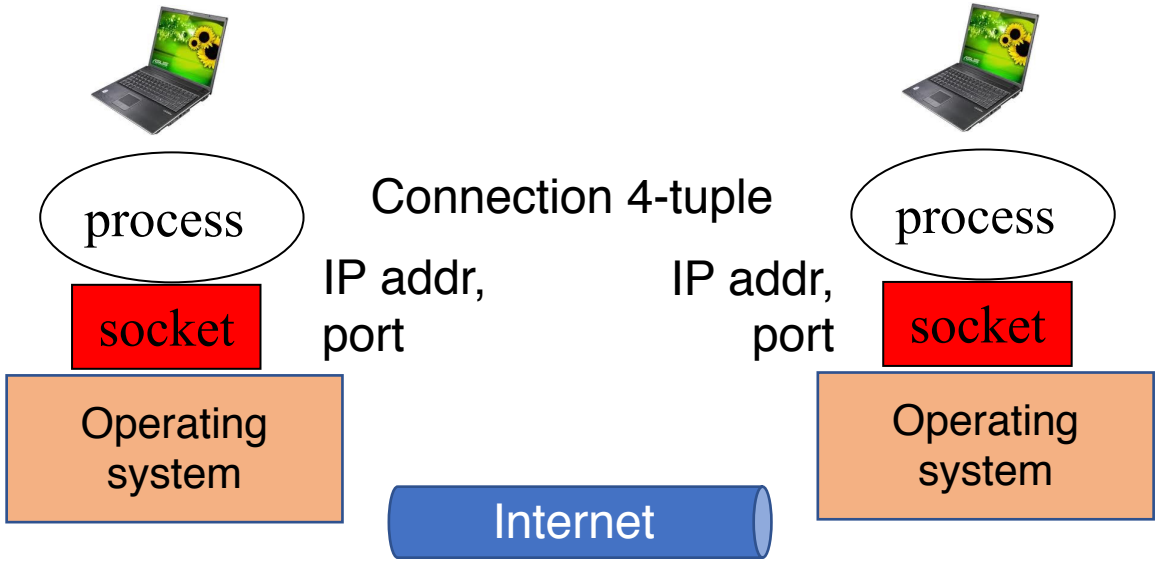
Lecture 4

http://www.cs.rutgers.edu/~sn624/352-S22

Srinivas Narayana

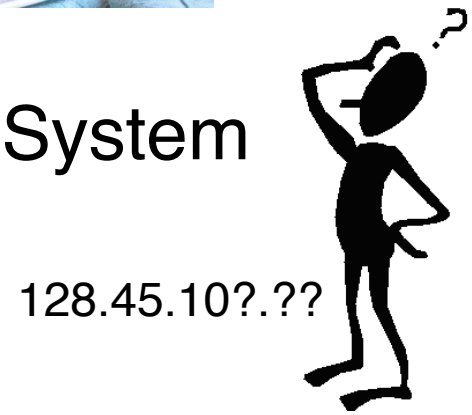
RUTGERS
UNIVERSITY | NEW BRUNSWICK

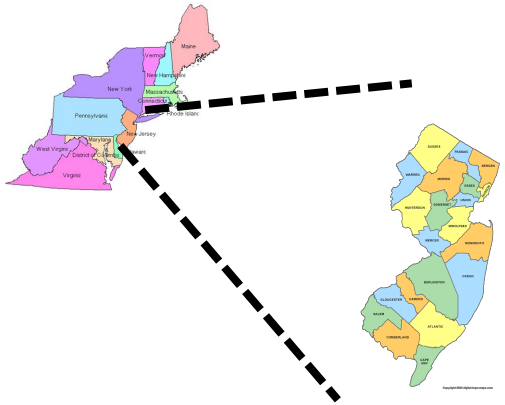
Quick recap of concepts

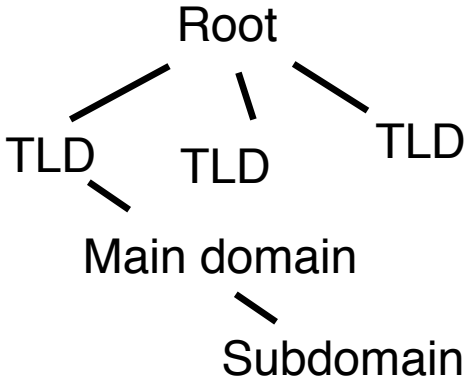


Domain Name System

Distributed database of name to IP addr mappings.







DNS Resource Records

DNS is a distributed database

- DNS stores resource records (RRs)
- (Incomplete) message format (headers):
  - Class, type, name, value, TTL
- You can read all the gory details of the message format at https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml

DNS records

Type=A

- **name** is hostname
- **value** is IP address

Type=AAAA

- **name** is hostname
- **value** is IPv6 address

- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain

Type=CNAME

- **name** is alias name for some "canonical" (the real) name

  e.g., www.ibm.com is really servereast.backup2.ibm.com
- **value** is canonical name

Type=MX

- **value** is name of mailserver associated with **name**

More complete info at https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml

DNS record example

RRs in response to query

NAME	Design.cs.rutgers.edu
TYPE	A
CLASS	IN
TTL	1 day(86400)
ADDRESS	192.26.92.30

records for authoritative servers Information about nameserver

NAME	Cs.rutgers.edu
TYPE	NS
CLASS	IN
TTL	1 day(86400)
NSDNAME	Ns-lcsr.rutgers.edu

DNS serves as a general repository of information for the Internet!

DNS record types

- `dig –t <type> <domain-name>`

DNS caching and updating records

- Once (any) name server learns a name to IP address mapping, it *caches* the mapping
  - Cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers
  - In practice, root name servers aren't visited often

Bootstrapping DNS

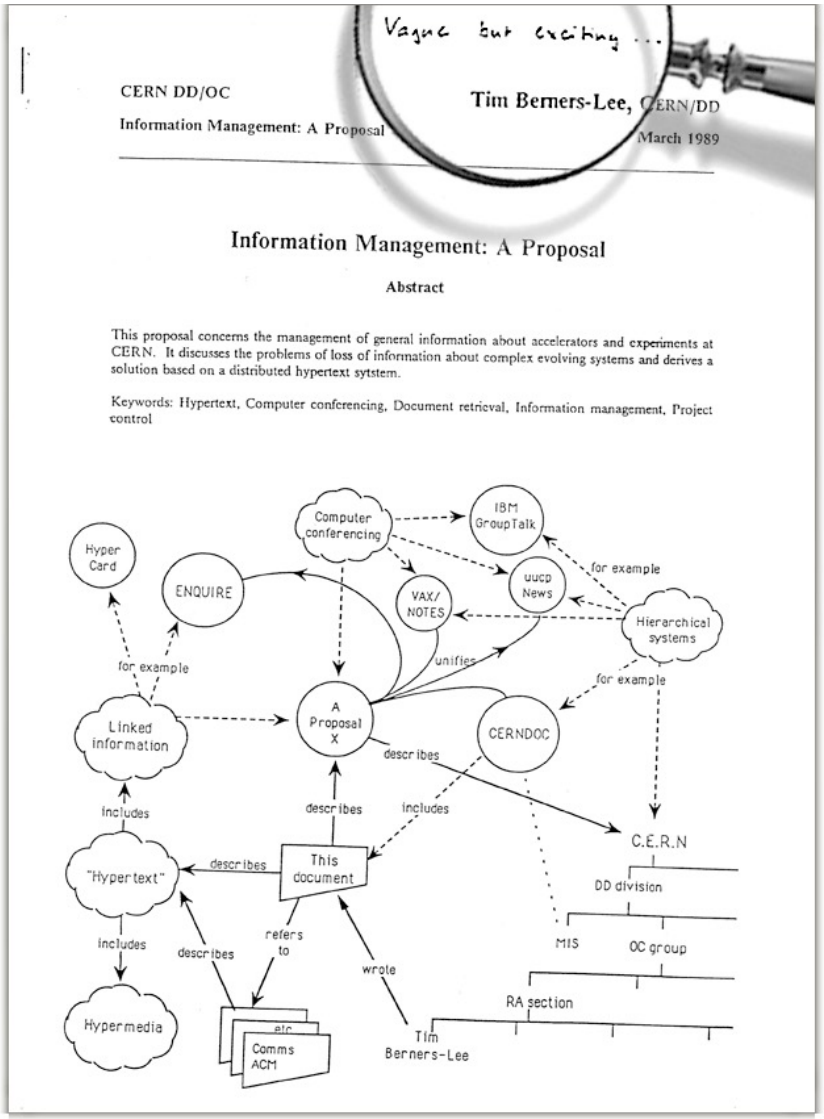
- How does a host contact the name server if all it has is the domain name and no (name server) IP address?
- IP address of at least 1 nameserver (usually, a local resolver) must be known a priori
- The name server may be bootstrapped “statically”, e.g.,
  - File `/etc/resolv.conf` in unix
  - `Start -> settings-> control panel-> network ->TCP/IP -> properties` in windows
- … or with another protocol!
  - DHCP: Dynamic Host Configuration Protocol (more on this later)

Summary of DNS

- Hostname to IP address translation via a global network of servers
- Embodies several scaling principles
  - Partitioning through a hierarchy
  - Load distribution through replication at each level of hierarchy
  - Caching
- Once you have a reliable DB, can implement many useful things on top!
- Example 1: Scaling large web services, e.g., google search, by redirecting different clients to different servers (IP addresses)
  - Reliability, load balancing, performance optimization
- Example 2: Associating certificates, keys (security info) with domain names
  - https://www.rfc-editor.org/rfc/rfc8162.html
  - https://datatracker.ietf.org/doc/draft-ietf-dnsop-svcb-https/00/

The Web: HTTP

The Web: Humble origins



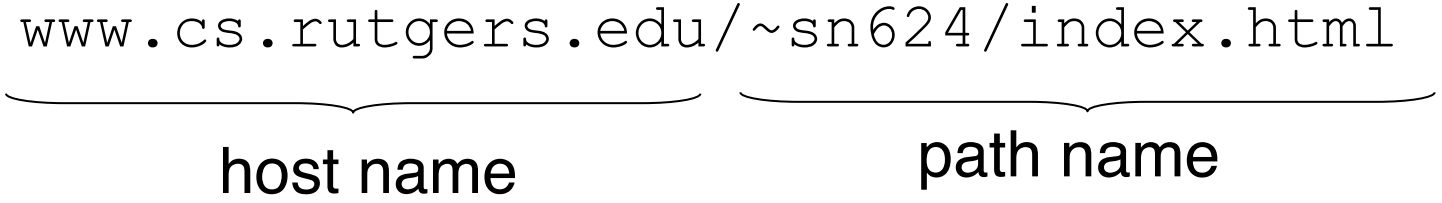
Tim Berners-Lee: a way to manage and access documents at CERN research lab

Info containing links to other info, accessible remotely, through a standardized mechanism.

His boss is said to have written on his proposal:
“vague, but exciting”

Web and HTTP: Some terms

- HTTP stands for “HyperText Transfer Protocol”
- A web page consists of many objects
- Object can be HTML file, JPEG image, video stream chunk, audio file,…
- Web page consists of base HTML-file which includes several referenced objects.
- Each object is addressable by a uniform resource locator (URL)
  - sometimes also referred to as uniform resource identifier (URI)
- Example URL:

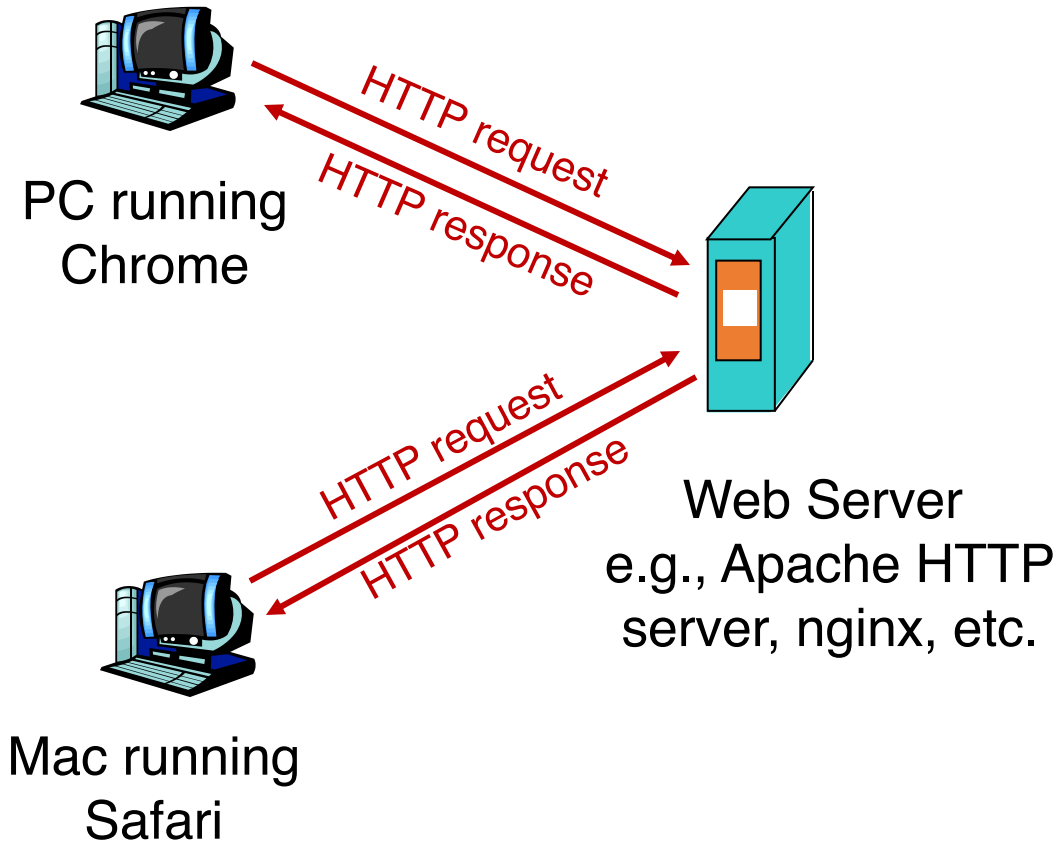


HTTP Protocol Overview

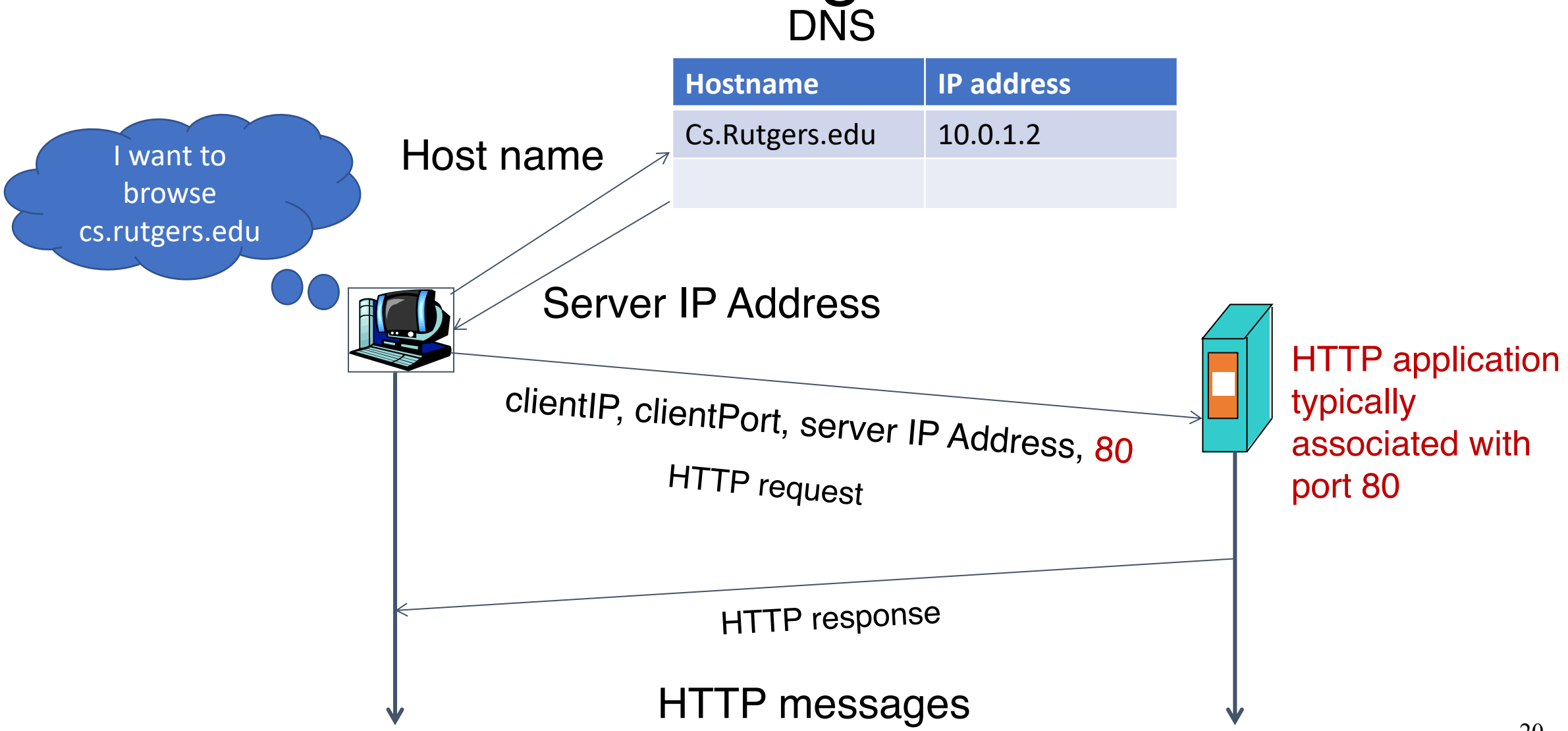
HTTP overview

HTTP: hypertext transfer protocol

- Client/server model
  - *Client:* browser that requests, receives, “displays” Web objects
  - *Server:* Web server sends objects in response to requests
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



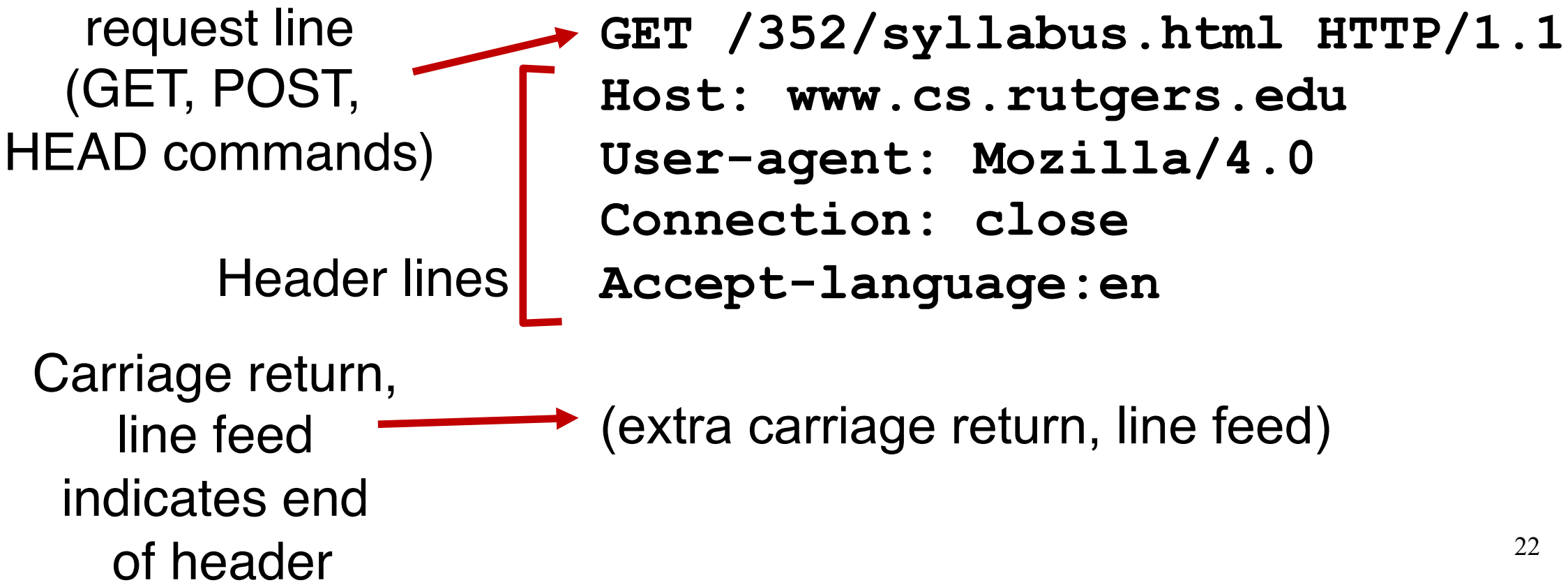
Client server exchanges



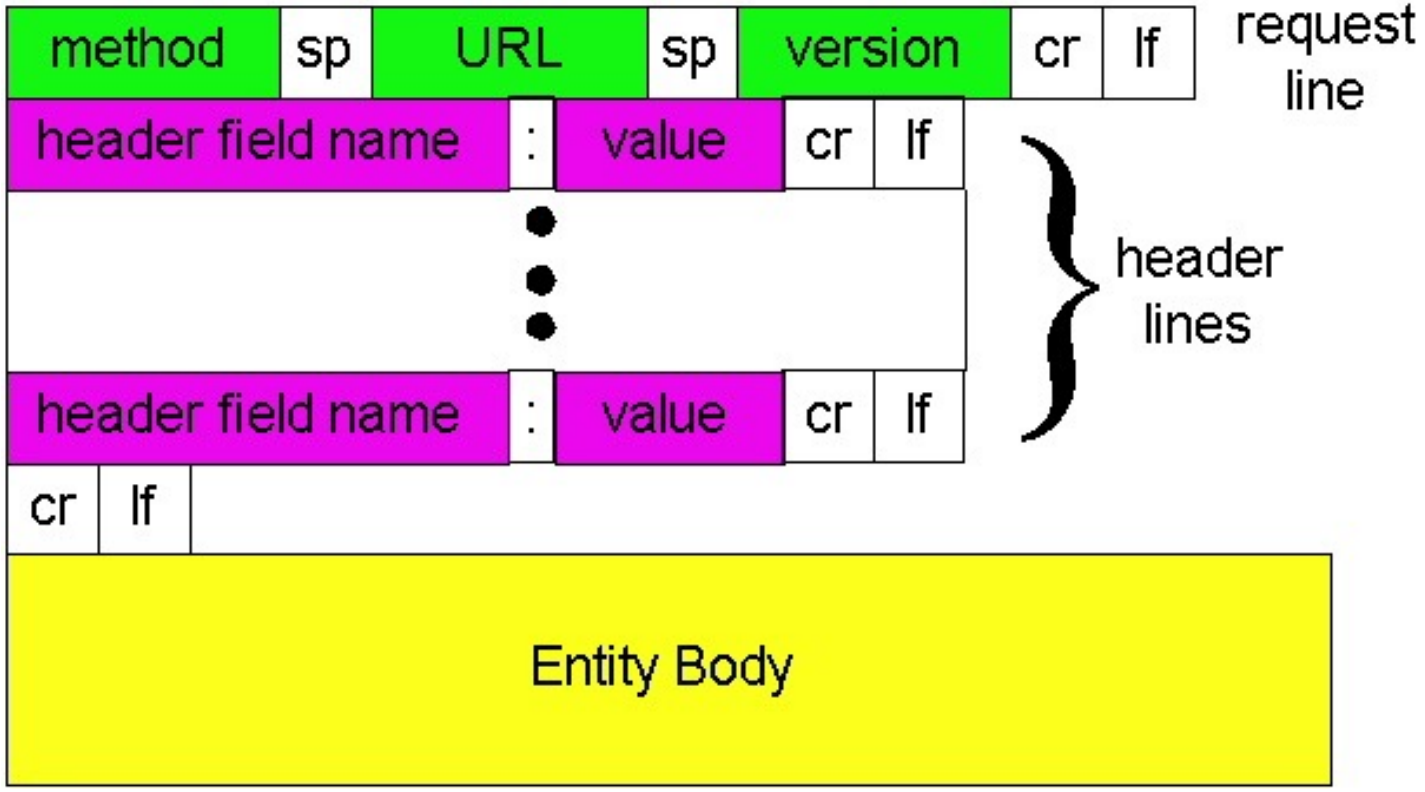
HTTP Messages

HTTP messages: request message

- ASCII (human-readable format)



HTTP Request: General format



http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14

Sidebar: Protocol headers and formats in the exam?

In general, you don't need to memorize protocol message formats in this course (they are easily looked up)

However, you may be asked to interpret protocol headers based on a general (conceptual) understanding of the protocol.

The URL

- Universal Resource Locator: a way to name objects on server
- But can also name an application process on the server!
- Examples:
  - Data storage from data entered in web forms
  - Login pages
  - Web carts
- Providing almost any service requires data handling by running code at the server
  - Not just rendering “static” resources

HTTP method types

- GET
  - Get the resource specified in the requested URL (could be a process)
- POST
  - Send entities (specified in the entity body) to a data-handling process at the requested URL
- HEAD
  - Asks server to leave requested object out of response, but send the rest of the response
  - Useful for debugging
- PUT
  - Update a resource at the requested URL with the new entity specified in the entity body
- DELETE
  - Deletes file specified in the URL
- and other methods

Uploading form input: GET and POST

POST method:

- Web page often includes form input
- Input is uploaded to server in entity body

- Posted content not visible in the URL
  - Free form content (ex: images) can be posted since entity body interpreted as data bytes

GET method:

- Entity body is empty
- Input is uploaded in URL field of request line

- Example:
  - http://site.com/form?first=jane&last=austen

Difference between POST and PUT

- POST: the URL of the request identifies the resource that processes the entity body
- PUT: the URL of the request identifies the resource that is contained in the entity body

https://tools.ietf.org/html/rfc2616

Difference between HEAD and GET

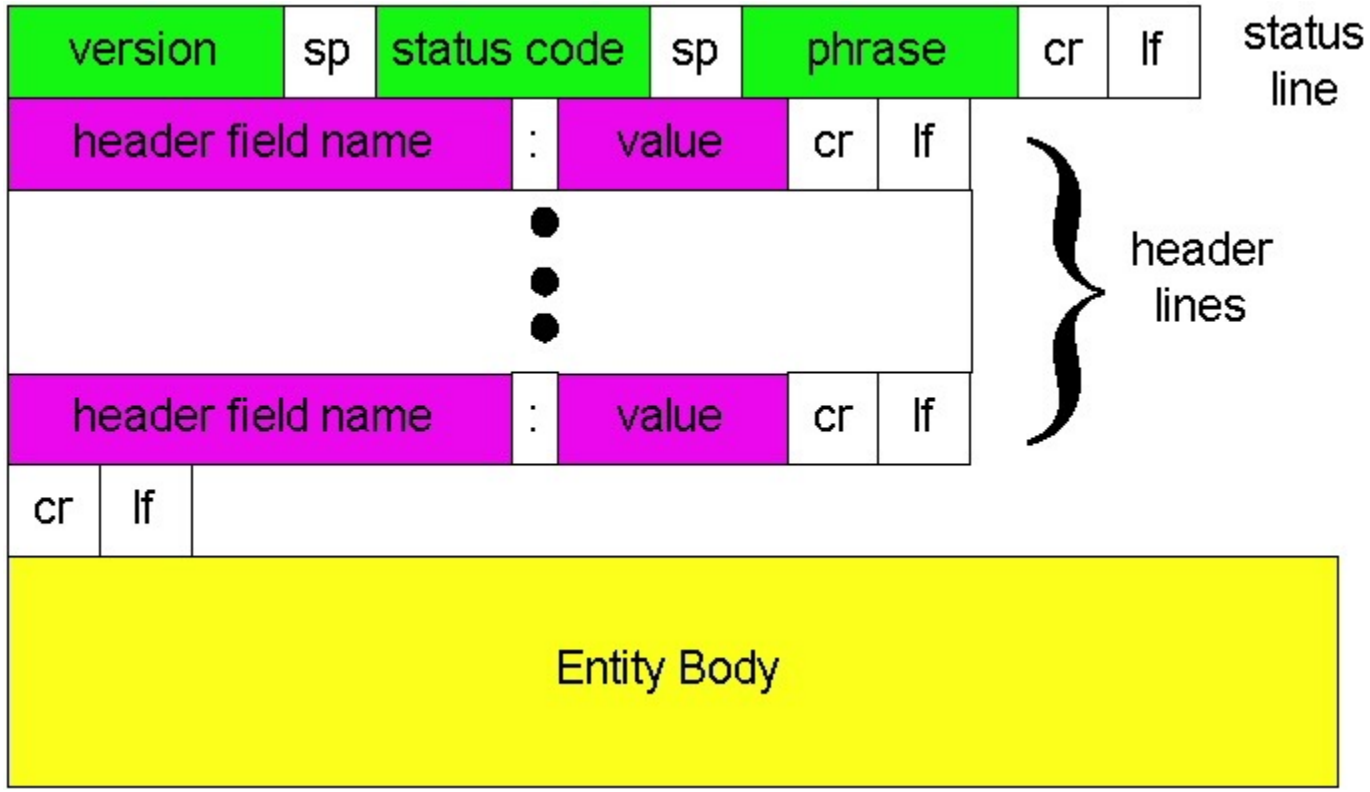
- GET: return the requested resource in the entity body of the response along with response headers (we'll see these shortly)
- HEAD: return all the response headers in the GET response, but without the resource in the entity body

https://tools.ietf.org/html/rfc2616

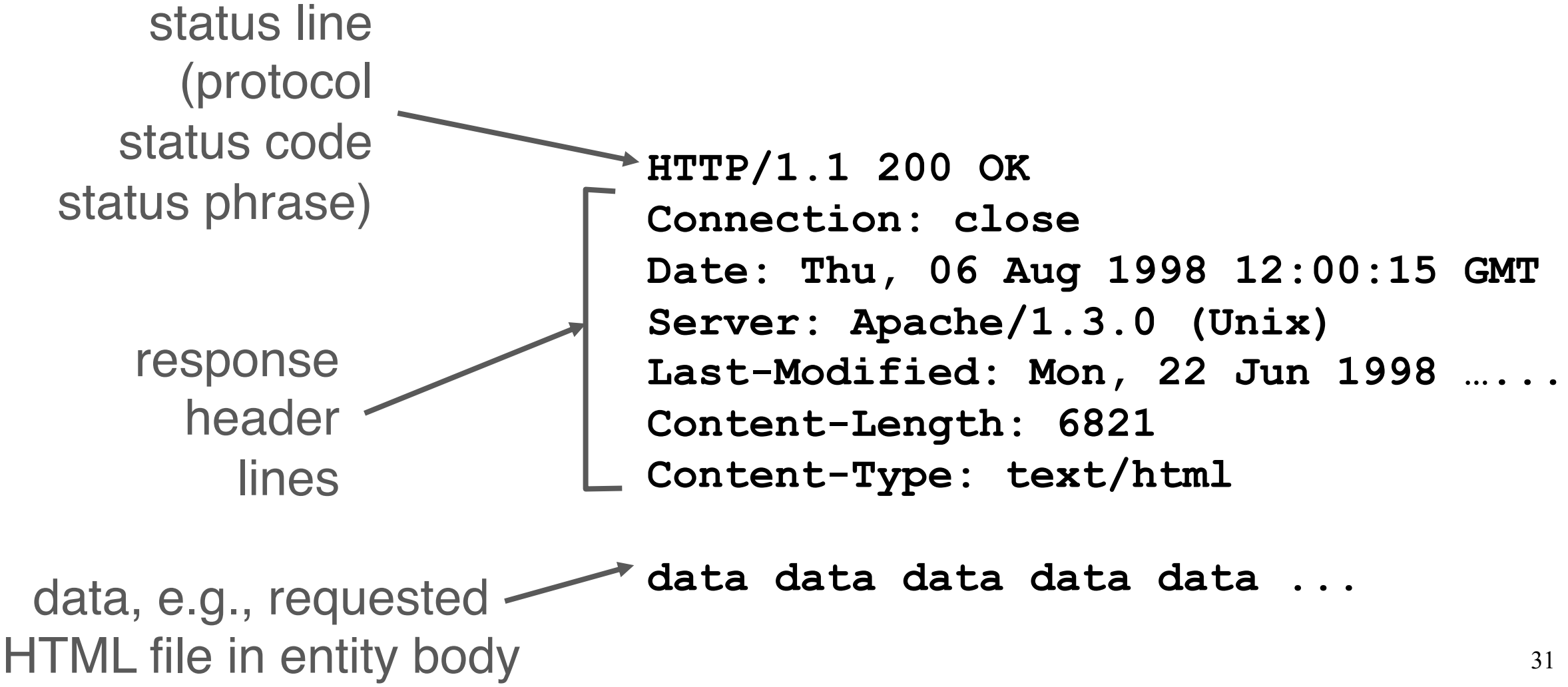
Observing HTTP GET and POST

HTTP Response: General format

Unlike HTTP request, No method name



HTTP message: 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:)

403 Forbidden

- Insufficient permissions to access the resource

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

Observing response codes

- `wget google.com`

- `telnet web.mit.edu 80`
  - `GET / HTTP/1.1`
  - `Host: web.mit.edu`