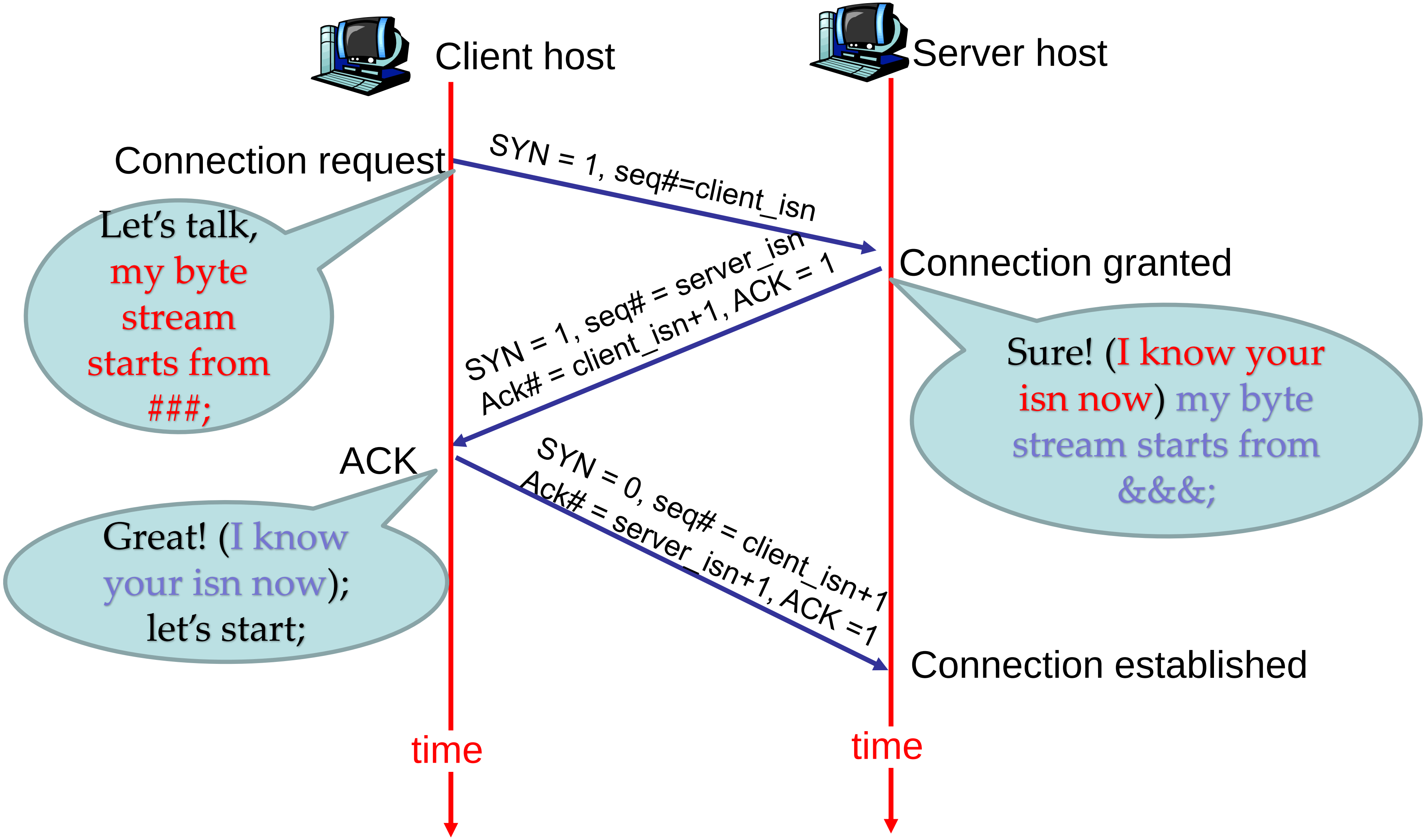
COSC264
Introduction to Computer Networks and the Internet

Introduction to the Web and HTTP

Dr. Barry Wu
Wireless Research Centre
University of Canterbury
barry.wu@canterbury.ac.nz

Why 3-way handshake? Not 2-way handshake?



Client host

Server host

Connection request

SYN = 1, seq#=client_isn

Let's talk, my byte stream starts from ###;

Connection granted

SYN = 1, seq# = server_isn
Ack# = client_isn+1, ACK = 1

Sure! (I know your isn now) my byte stream starts from &&&;

ACK

SYN = 0, seq# = client_isn+1
Ack# = server_isn+1, ACK =1

Great! (I know your isn now); let's start;

Connection established

time

time

Outline

- Network applications
  - Network apps vs app. protocols
  - Application structure
- The Web
- HTTP

Outline

- Network applications
  - Network apps vs app. protocols
  - Application structure
- The Web
- HTTP

The most popular network applications

- The Web
  - Search engine
  - E-commerce
  - Banking
  - News
  - Online video
  - Blog
- Email
- Instant messaging
  - Text/pic/sound
  - Video call
- Video conferencing/gaming

Other applications

- telnet
- File transfer (ftp)
- News group
- P2P file sharing

What is a network application?

- Programs that run on different end system and communicate with each other over the network;
  - Web browser program in the user's host;
  - Web server program in the web server host;

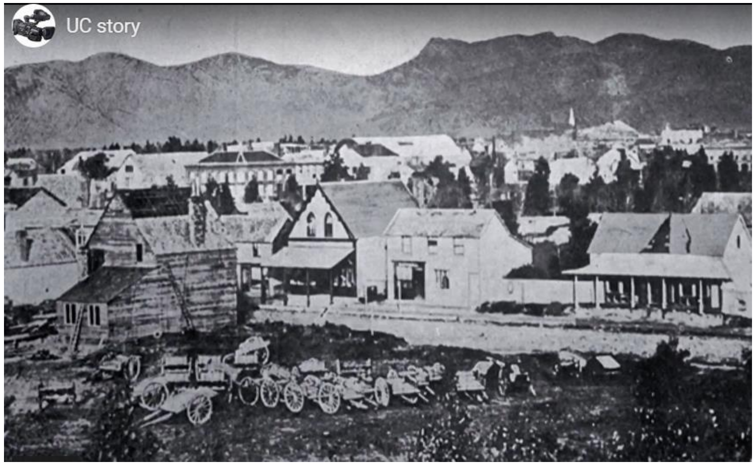
Network apps vs application protocols

- An application protocol is only one piece (a big one) of a network application.

Network app	Application protocol
Web (other pieces: HTML, web browsers, web servers)	HTTP (one piece of web, protocol)
Email	SMTP

Services an app needs

- Reliable data transfer
  - Email; instant messaging; file transfer, financial applications;
  - There are loss-tolerant applications though;
    - o Multimedia applications



Bandwidth

- Rate
  - Internet telephony application
  - Many multimedia apps are bandwidth-sensitive; (*adaptive coding technique*)
- Elastic apps
  - Not strict with bandwidth;
  - Email; file transfer;

Timing

- Tight timing constraints
  - Interactive real-time apps: Internet telephony, virtual environments (VR), teleconferencing, multiplayer games;
  - End-to-end delay: < 100s of ms;

Services provided by the Internet transport layer

TCP	**UDP**
Connected-oriented service	Connectionless service
Reliable transport service	*Unreliable data transfer service*
Congestion control (*NO guarantee of min transmission rate*)	*No congestion control*
Flow control	*No flow control*
No guarantee of delay	*No guarantee of timing*

The Internet has been hosting time-sensitive applications for many years!

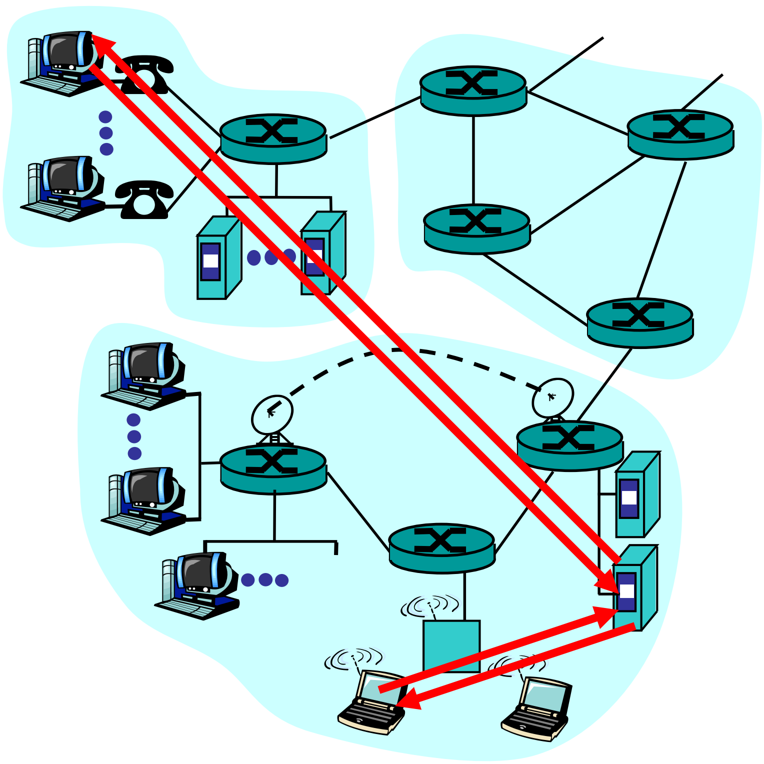
Outline

- Network applications
  - Network apps vs app. protocols
  - Application structure
- The Web
- HTTP

Application architectures

- Client-server
- Peer-to-peer (P2P)
- Hybrid of client-server and P2P

Client-server archicture



server:

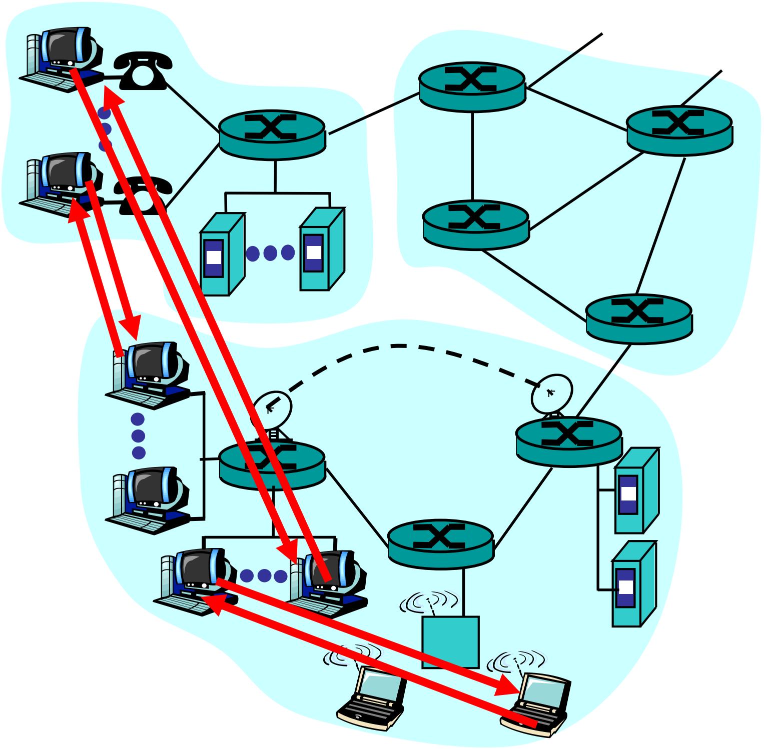
- always-on host
- permanent IP address
- server farms for scaling

clients:

- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

Pure P2P architecture

- no always on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses
- example: Gnutella



Highly scalable

But difficult to manage

Hybrid of client-server and P2P

Napster

- File transfer P2P
- File search centralized:
  - Peers register content at central server
  - Peers query same central server to locate content

Instant messaging

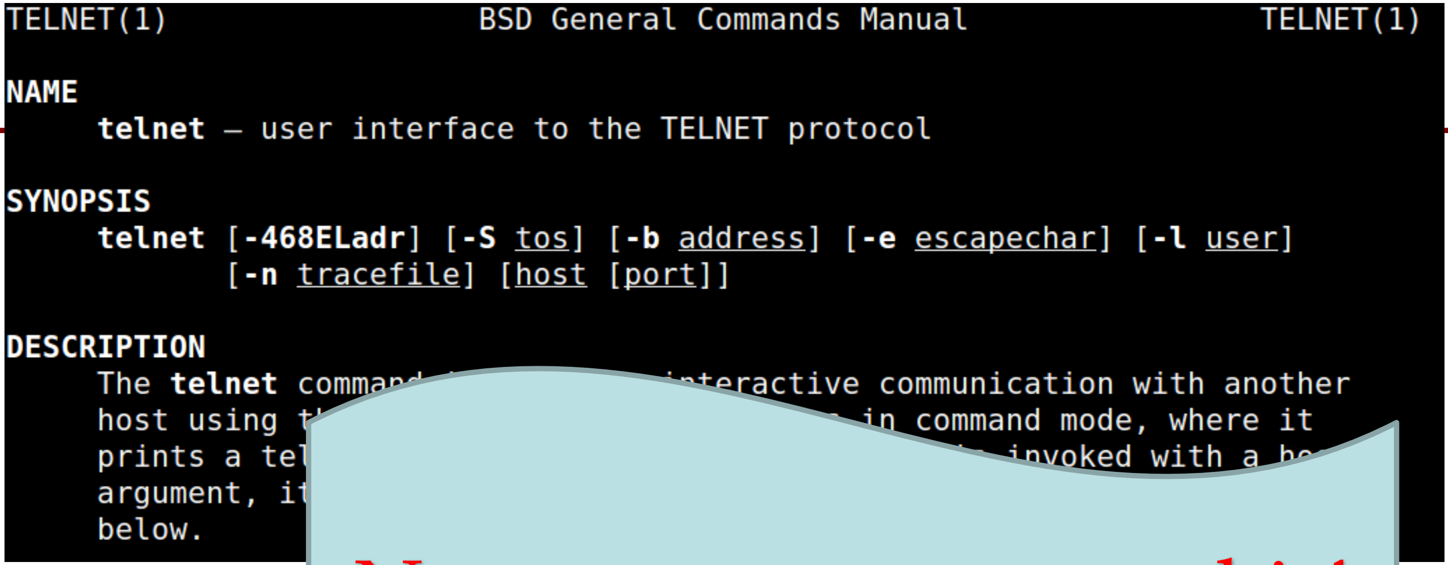
- Chatting between two users is P2P
- Presence detection/location centralized:
  - User registers its IP address with central server when it comes online
  - User contacts central server to find IP addresses of buddies

Outline

- Network applications
  - Network apps vs app. protocols
  - Application structure
- The Web
- HTTP

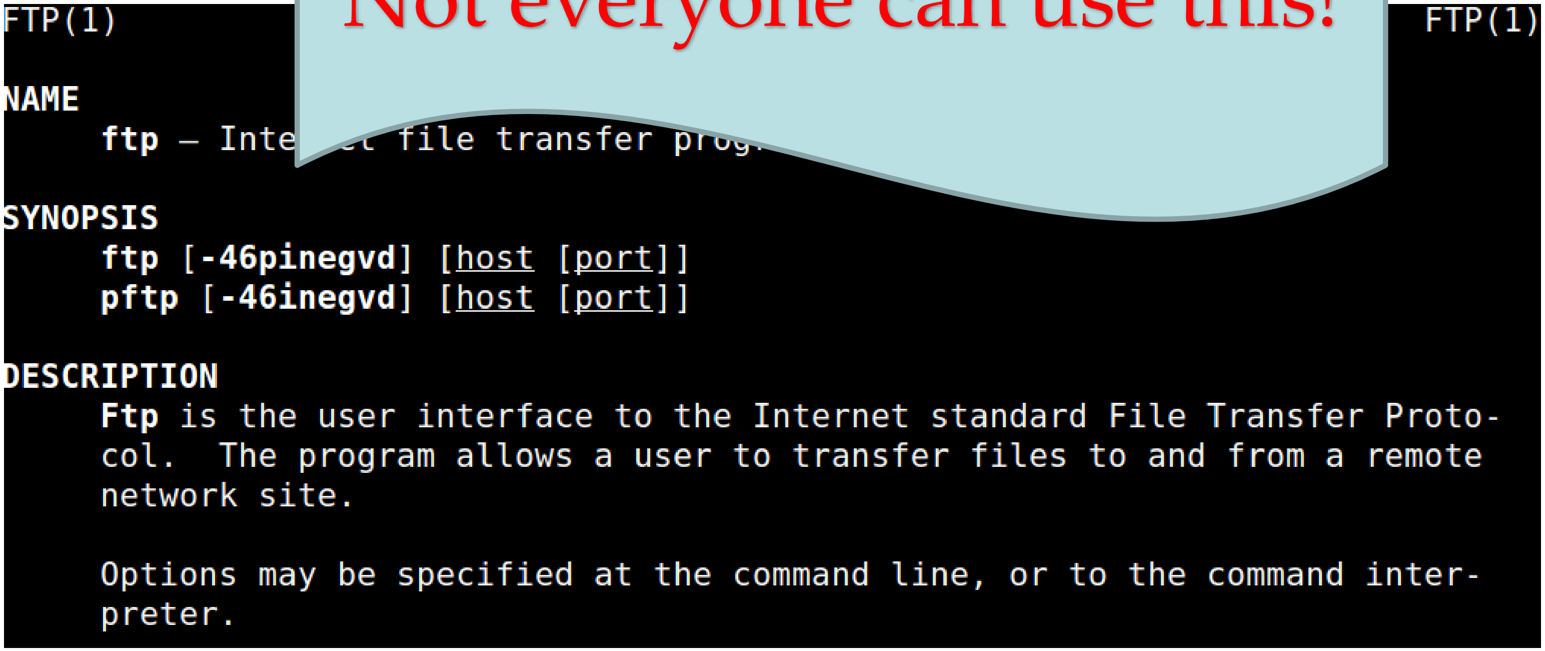
The Web

- Until 1990s the Internet was mainly used by researchers, academics and university students.



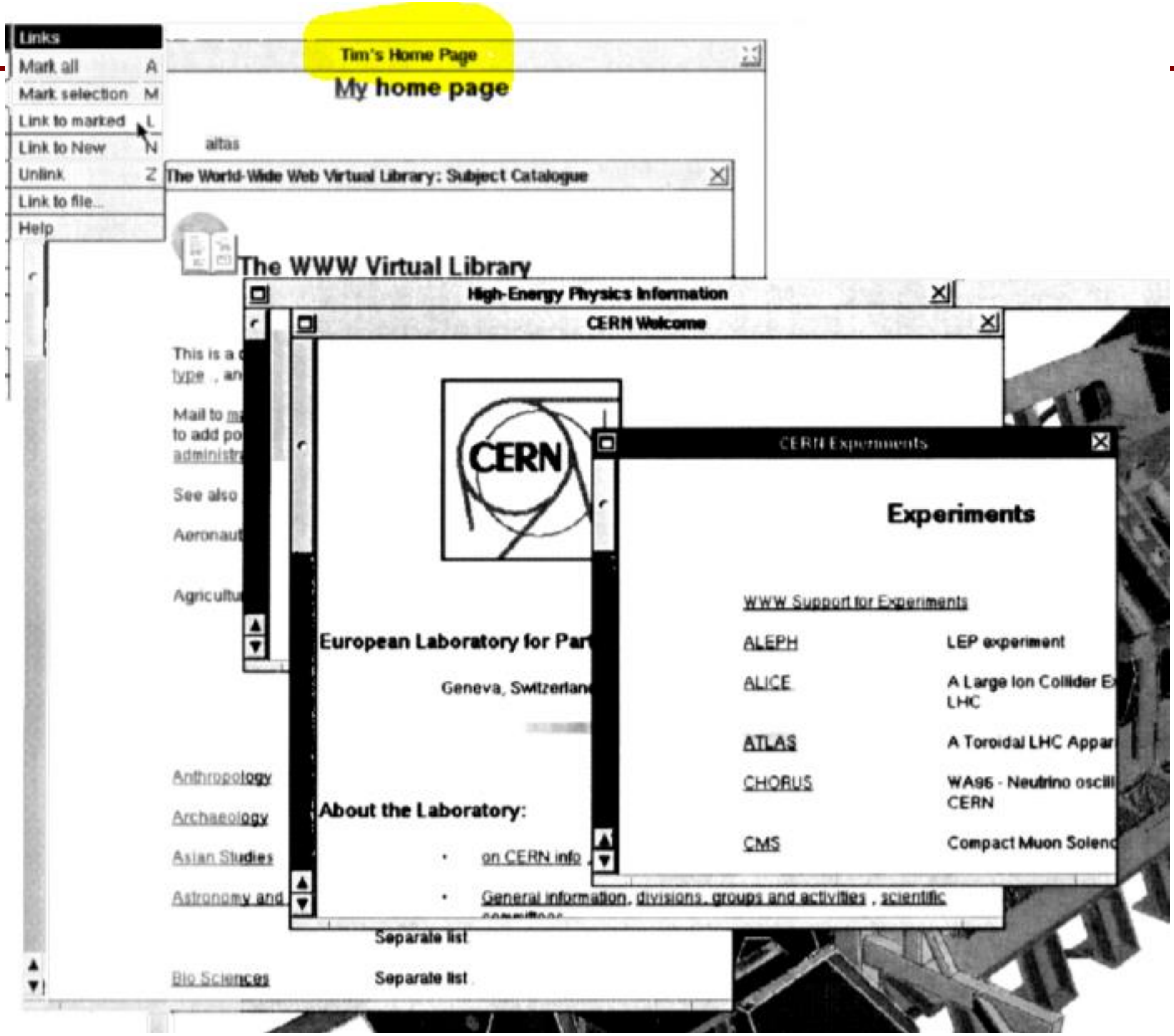
```
TELNET(1)               BSD General Commands Manual              TELNET(1)

NAME
     telnet — user interface to the TELNET protocol

SYNOPSIS
     telnet [-468ELadr] [-S tos] [-b address] [-e escapechar] [-l user]
            [-n tracefile] [host [port]]

DESCRIPTION
     The telnet command          interactive communication with another
     host using t                      in command mode, where it
     prints a tel                        invoked with a ho
     argument, i
     below.
```

Not everyone can use this!



```
FTP(1)                                                             FTP(1)

NAME
     ftp – Inte  et file transfer prog

SYNOPSIS
     ftp [-46pinegvd] [host [port]]
     pftp [-46inegvd] [host [port]]

DESCRIPTION
     Ftp is the user interface to the Internet standard File Transfer Proto-
     col.  The program allows a user to transfer files to and from a remote
     network site.

     Options may be specified at the command line, or to the command inter-
     preter.
```

- Started by Tim Berners-Lee in 1989.
  - Telephone (1870s); radio (1920s)/TV (1930s);
  - Email and the Web;



Tim's Home Page

My home page

altas

The World-Wide Web Virtual Library: Subject Catalogue

The WWW Virtual Library

This is a d
type , an

Mail to m
to add po
administr

See also

Aeronaut

Agricultu

Anthropology

Archaeology

Asian Studies

Astronomy and Separate list

Bio Sciences Separate list

High-Energy Physics Information

CERN Welcome

CERN

European Laboratory for Par

Geneva, Switzerlan

About the Laboratory:

- on CERN info
- General information, divisions, groups and activities , scientific committees

CERN Experiments

Experiments

WWW Support for Experiments

ALEPH	LEP experiment
ALICE	A Large Ion Collider E LHC
ATLAS	A Toroidal LHC Appar
CHORUS	WA95 - Neutrino oscill CERN
CMS	Compact Muon Solen

- On-demand service!
- HTTP (HyperText Transfer Protocol) is at the heart of the Web.

Outline

- Network applications
  - Network apps vs app. protocols
  - Application structure
- The Web
- HTTP

Outline

- Network applications
  - Network apps vs app. protocols
  - Application structure
- The Web
- HTTP
  - Overview
  - Non-persistent HTTP and persistent HTTP
  - HTTP messages

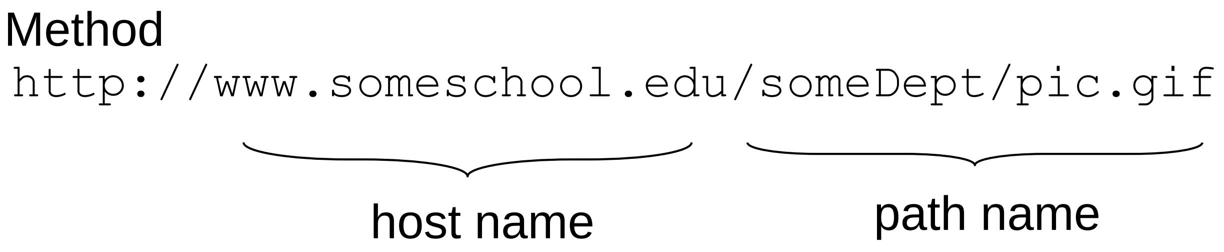
HTTP overview

HTTP: hypertext transfer protocol

- Web's application layer protocol
- HTTP /1.0: RFC 1945
- HTTP /1.1: RFC 2068/2616/7230
- HTTP /2: RFC 7540
- HTTP /3: Introduced in *26 September 2019*.

Web page

- Web page consists of objects
- Object can be HTML file, JPEG image, audio file,…
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL
- Example URL:



Method

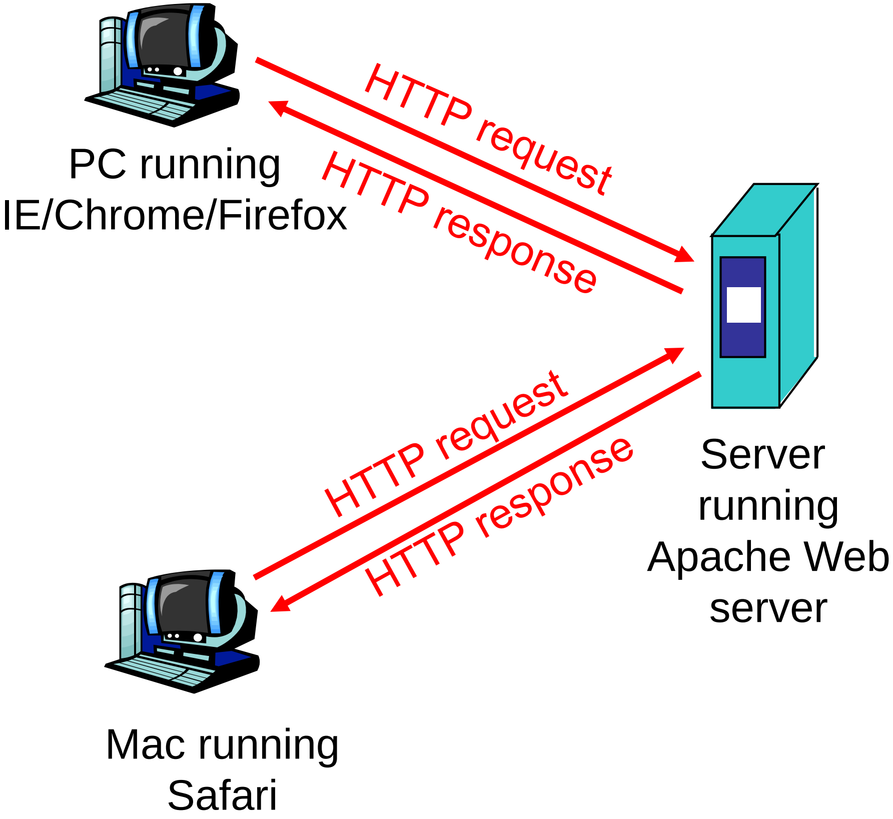
`http://www.someschool.edu/someDept/pic.gif`

host name

path name

client/server model

- *client:* browser that requests, receives, “displays” Web objects
- *server:* Web server sends objects in response to requests



HTTP overview (continued)

Uses TCP:

- 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 connections

Nonpersistent HTTP

- At most *one object* is sent over a TCP connection.
- HTTP/1.0 uses nonpersistent HTTP

Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server.
- HTTP/1.1 uses persistent connections in default mode

Nonpersistent HTTP

Suppose user enters URL

`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)

1. HTTP client initiates TCP connection to HTTP server (process) at www.someSchool.edu on port 80 (default port #)

2. HTTP client sends HTTP *request message* (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index

3. HTTP server receives request message, forms *response message* containing requested object, and sends message into its socket

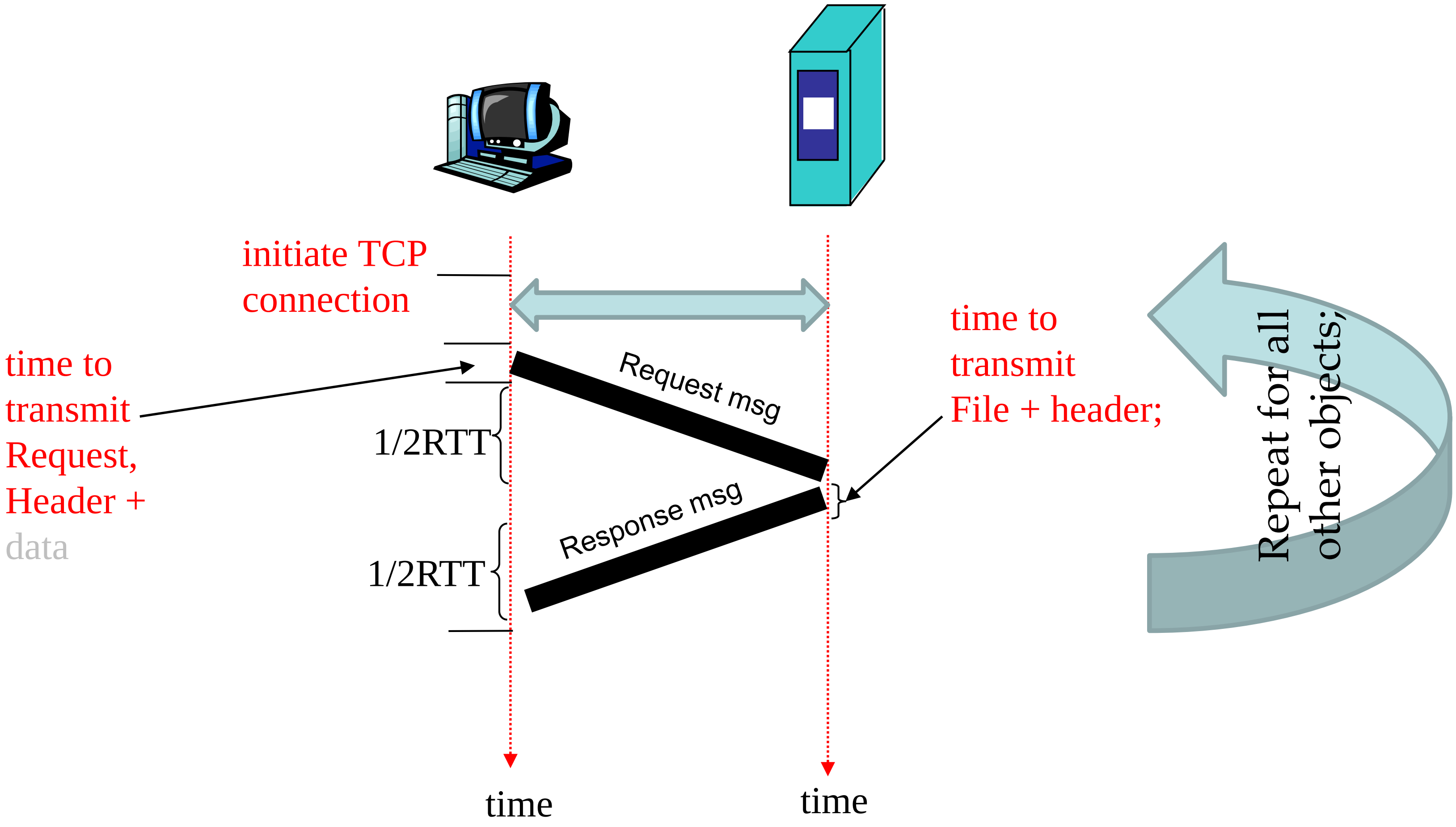
Nonpersistent HTTP (cont.)

4. HTTP server process tells TCP to close the TCP connection. (It doesn’t actually close it until it knows for sure the client has received the response message.)

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

A simplified time modelling



initiate TCP connection

time to transmit Request, Header + data

time to transmit File + header;

Repeat for all other objects;

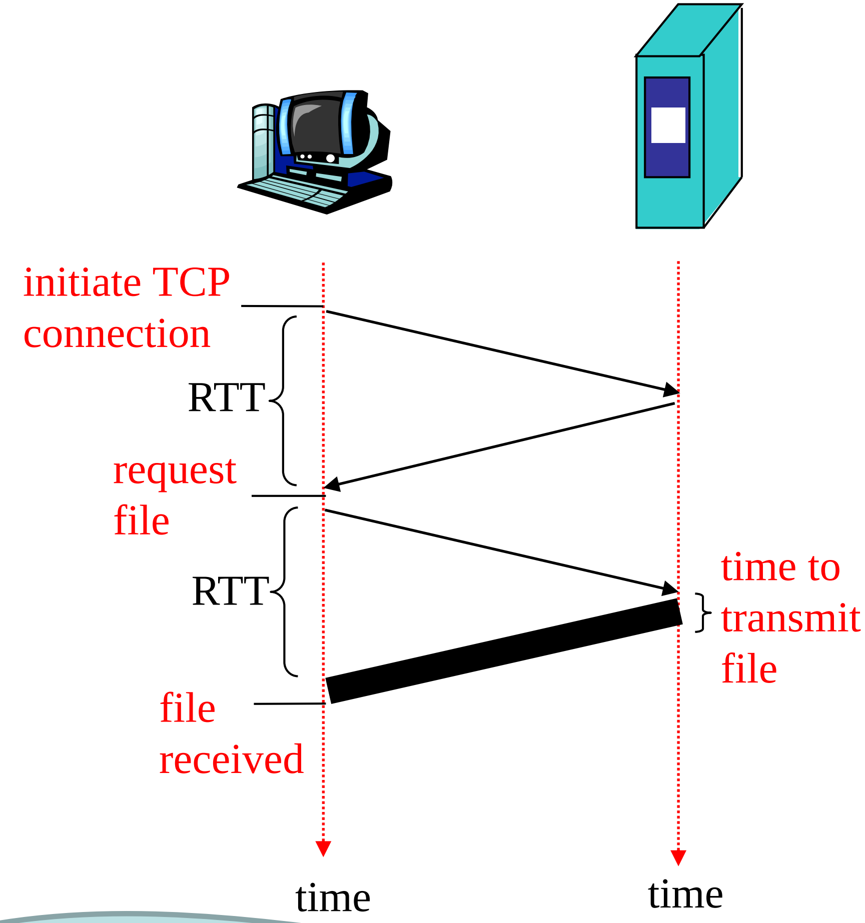
Another response time modeling

Definition of RRT: time to send a small packet to travel from client to server and back.

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

total = 2RTT+transmit time



Non-persistent HTTP requires reconnection for every objects.

Persistent HTTP

Nonpersistent HTTP issues:

- requires 2 RTTs per object
- OS must work and allocate host resources for each TCP connection
- *but 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 are sent over connection

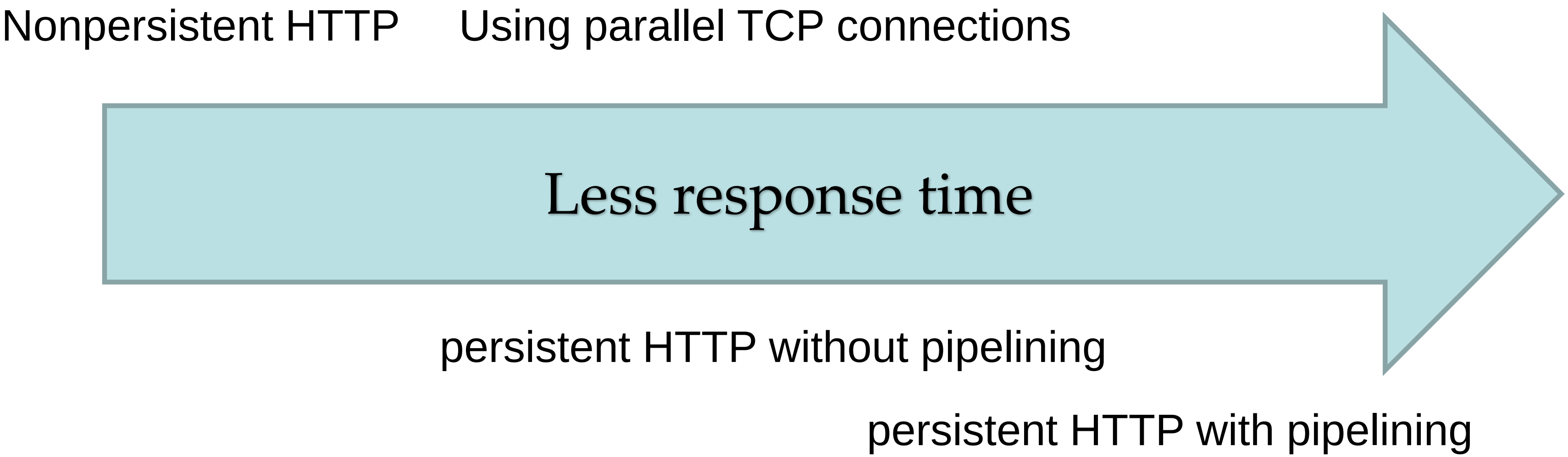
Persistent HTTP + pipelining

Persistent without pipelining:

- client issues new request only when previous response has been received
- one RTT for each referenced object

Persistent with pipelining:

- default in HTTP/1.1
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects



Nonpersistent HTTP

Using parallel TCP connections

Less response time

persistent HTTP without pipelining

persistent HTTP with pipelining

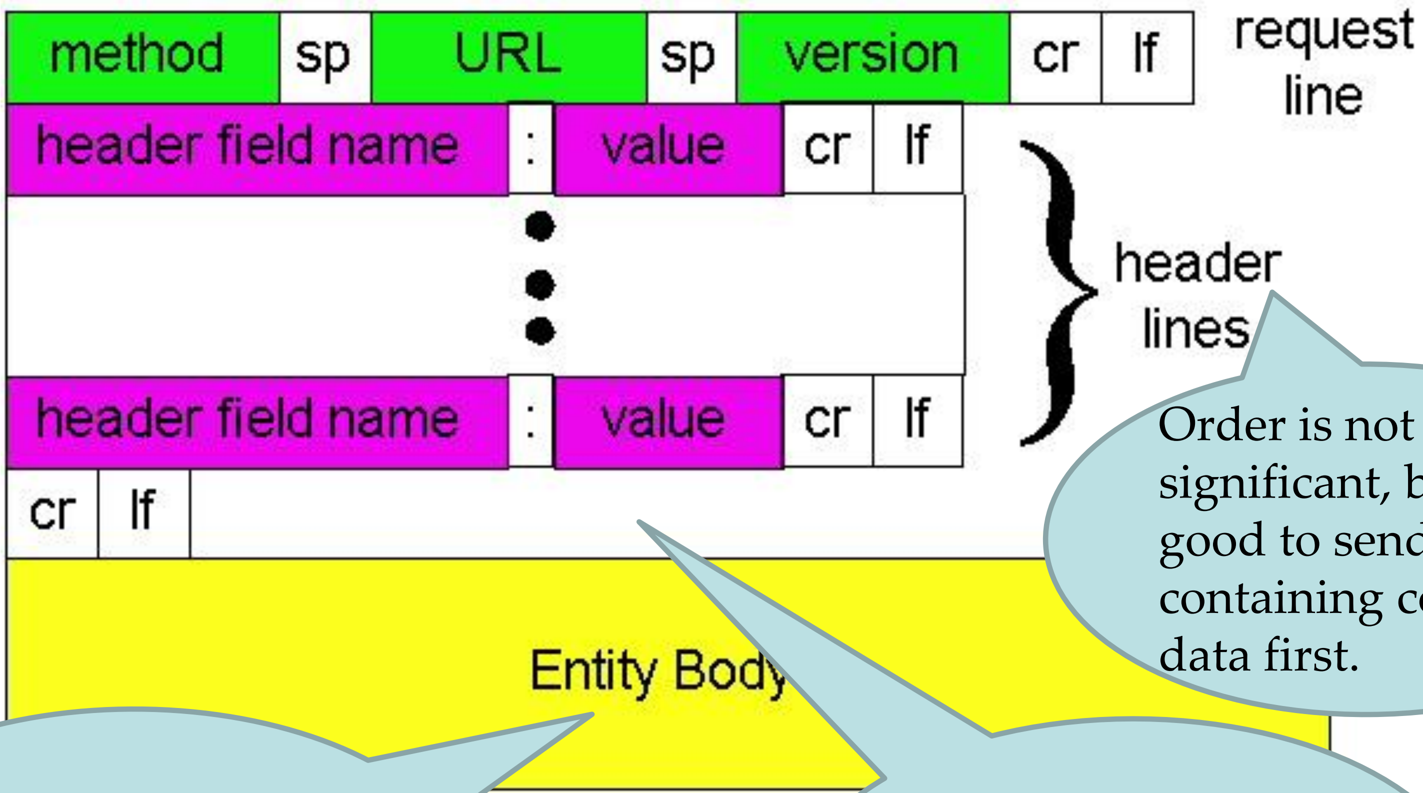
Outline

- Network applications
  - Network apps vs app. protocols
  - Application structure
- The Web
- HTTP
  - Overview
  - Non-persistent HTTP and persistent HTTP
  - HTTP messages

HTTP Message Format

two types of HTTP messages: *request*, *response*

HTTP request message: general format



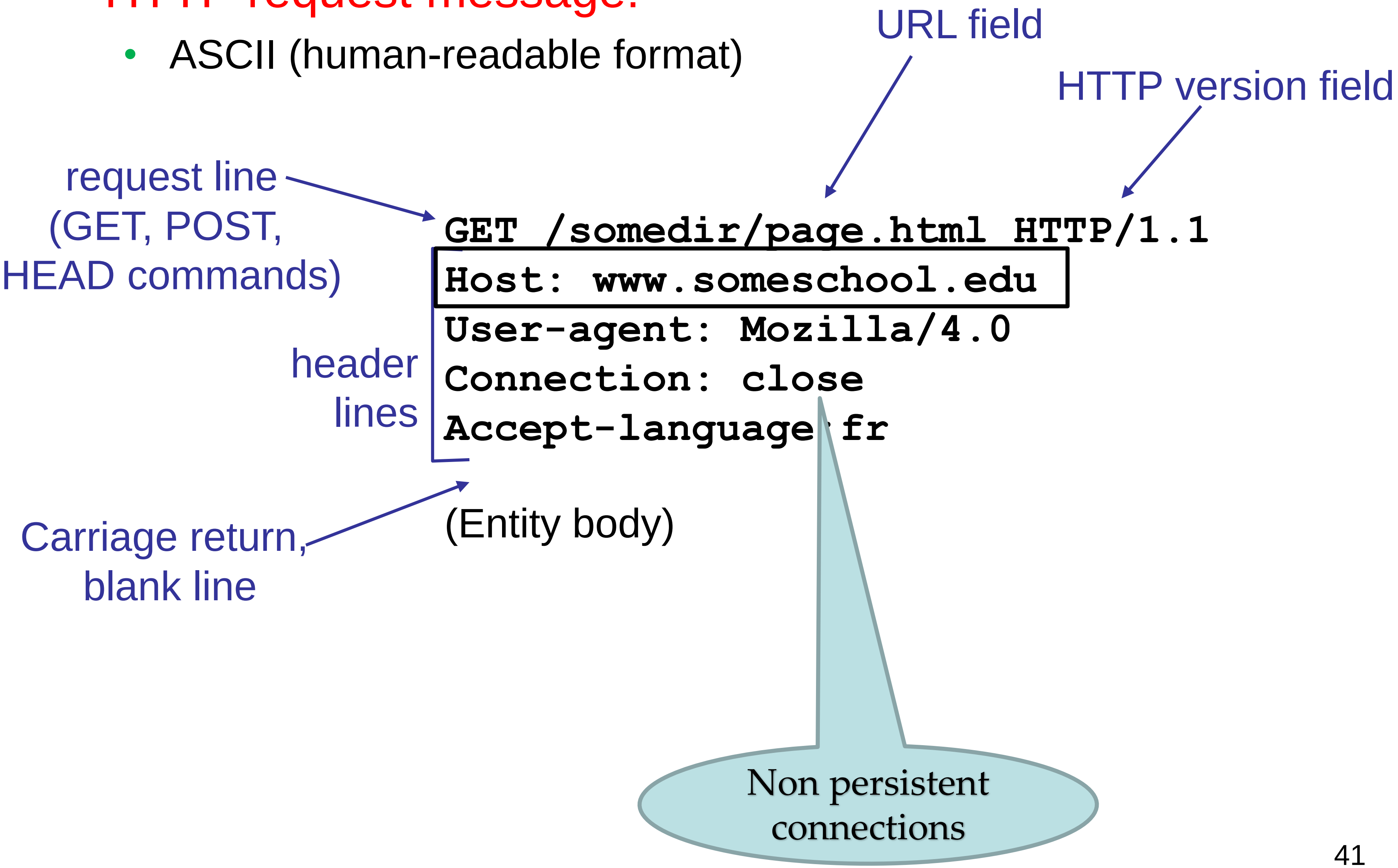
Order is not significant, but it is good to send fields containing control data first.

Empty for GET but it is used with POST (filling a form);

An empty line to indicate the end of the header section

HTTP request message

- HTTP request message:
  - ASCII (human-readable format)



```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr

(Entity body)
```

URL field
HTTP version field
request line (GET, POST, HEAD commands)
header lines
Carriage return, blank line
Non persistent connections

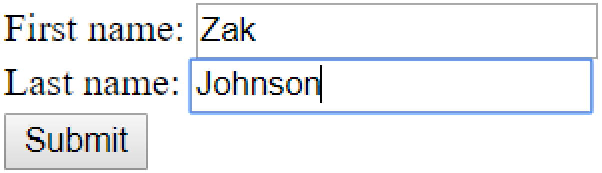
Each language-range MAY be given an associated quality value which represents an estimate of the user's preference for the languages specified by that range. The quality value defaults to "q=1".

For example,

Accept-Language: da, en-gb;q=0.8, en;q=0.7

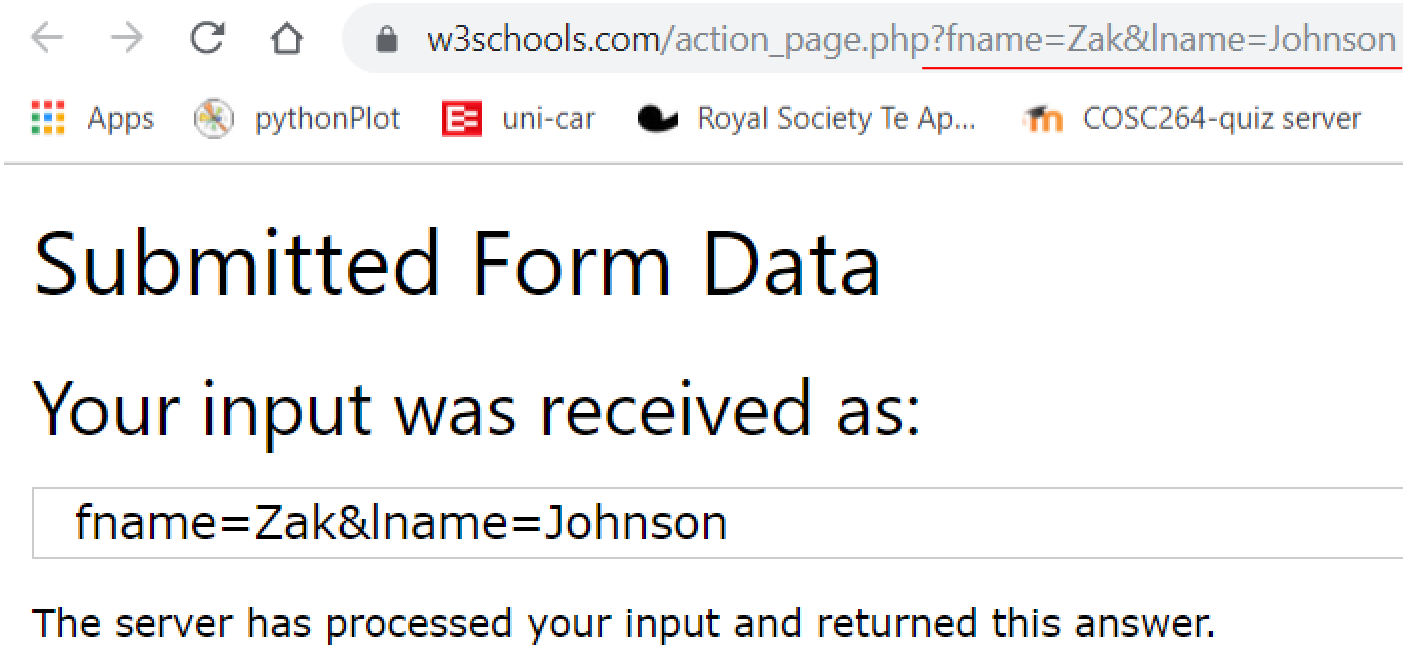
would mean: "*I prefer Danish, but will accept British English and other types of English*."

Form submission can be done with GET as well; “www.somesite.com/animalsearch?monkeys&bananas”



First name: Zak

Last name: Johnson



w3schools.com/action_page.php?fname=Zak&lname=Johnson

Submitted Form Data

Your input was received as:

```
fname=Zak&lname=Johnson
```

The server has processed your input and returned this answer.

Method types

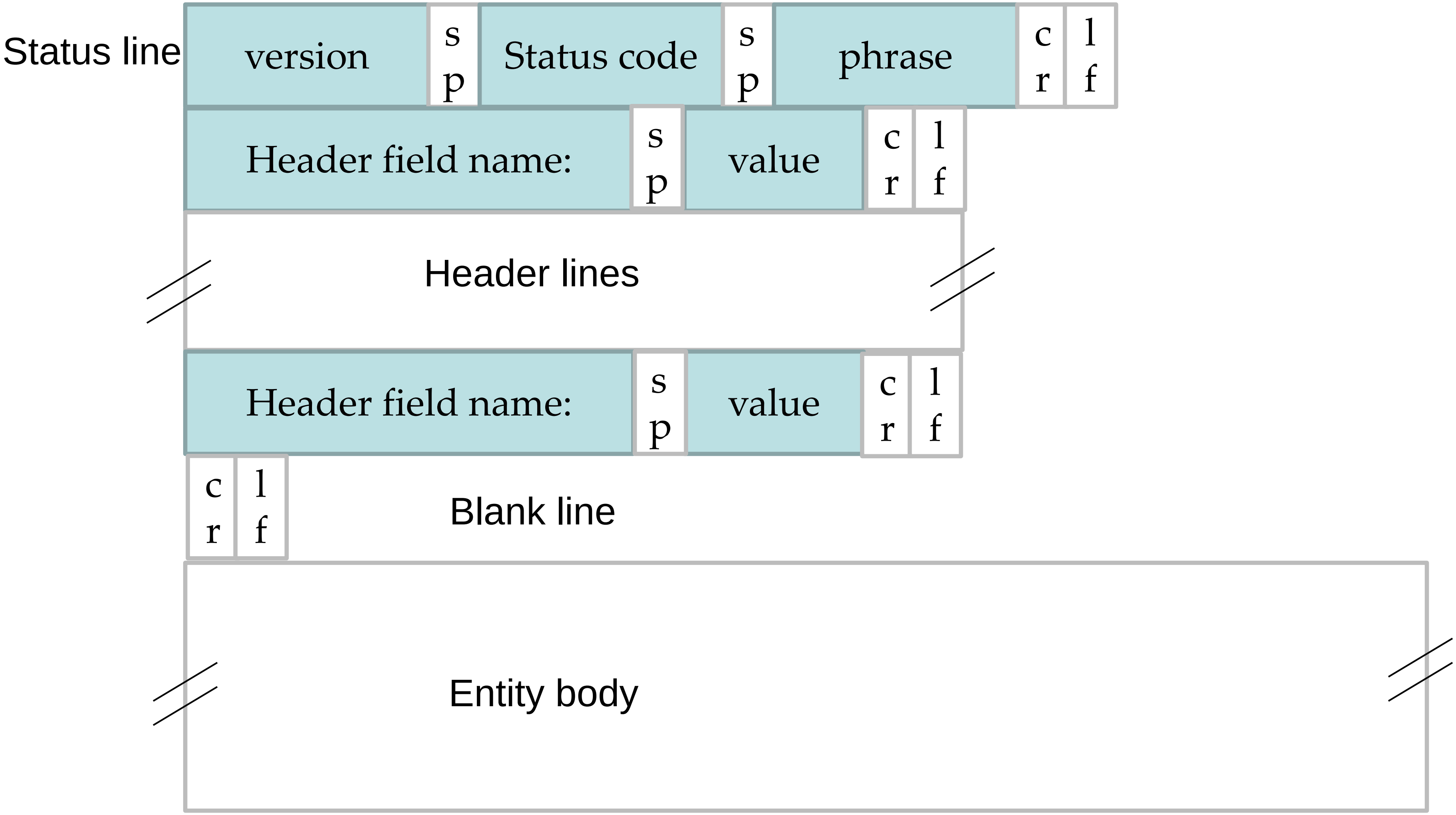
HTTP/1.0

- GET
- POST – filling a form
- HEAD
  - asks server to leave requested object out of response
  - Can be used for debugging

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

General format of a response message



Status line

version | sp | Status code | sp | phrase | cr lf

Header field name: | sp | value | cr lf

Header lines

Header field name: | sp | value | cr lf

cr lf

Blank line

Entity body

HTTP response status 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`

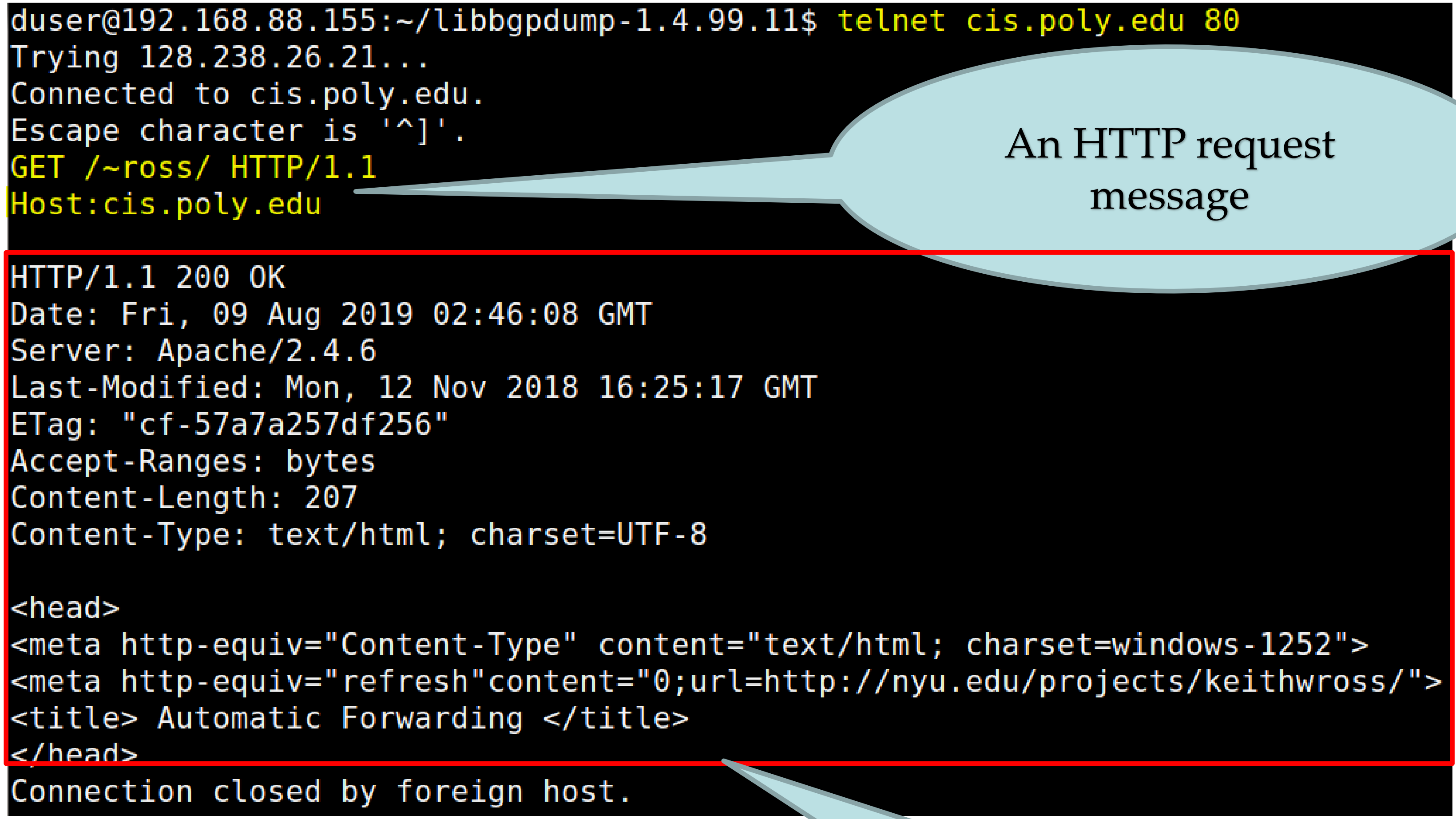
HTTP response message

status line (protocol ver, status code, status msg.)

```
HTTP/1.1 200 OK
Connection close
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

data data data data data ...
```

header lines

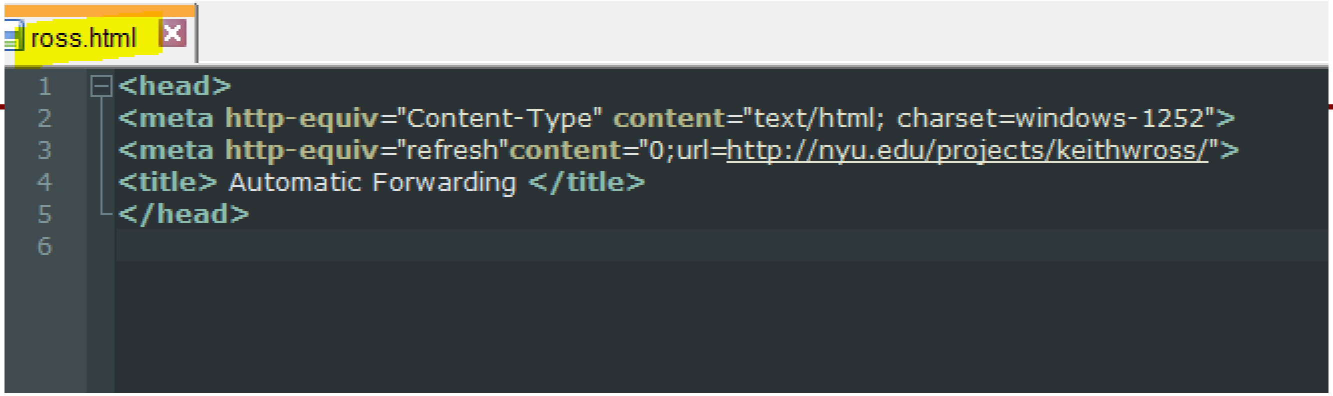
data, e.g., requested HTML file



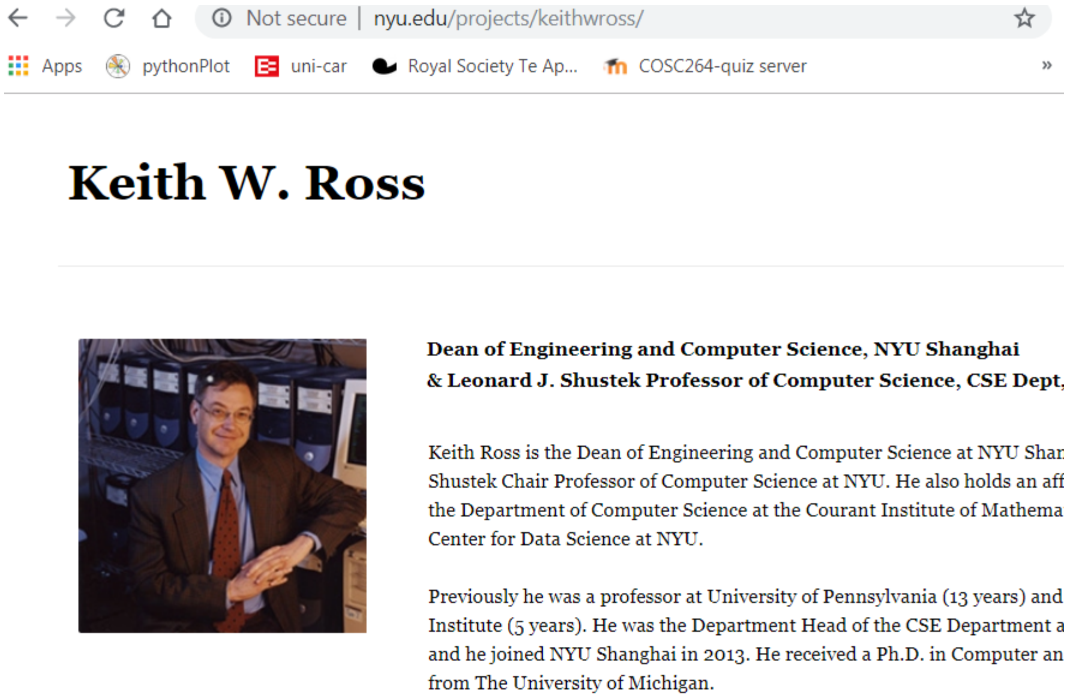
```
duser@192.168.88.155:~/libbgpdump-1.4.99.11$ telnet cis.poly.edu 80
Trying 128.238.26.21...
Connected to cis.poly.edu.
Escape character is '^]'.
GET /~ross/ HTTP/1.1
Host:cis.poly.edu

HTTP/1.1 200 OK
Date: Fri, 09 Aug 2019 02:46:08 GMT
Server: Apache/2.4.6
Last-Modified: Mon, 12 Nov 2018 16:25:17 GMT
ETag: "cf-57a7a257df256"
Accept-Ranges: bytes
Content-Length: 207
Content-Type: text/html; charset=UTF-8

<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta http-equiv="refresh"content="0;url=http://nyu.edu/projects/keithwross/">
<title> Automatic Forwarding </title>
</head>
Connection closed by foreign host.
```

An HTTP request message

An HTTP response message, sent by the server.



```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<meta http-equiv="refresh"content="0;url=http://nyu.edu/projects/keithwross/">
<title> Automatic Forwarding </title>
</head>
```



Keith W. Ross

Dean of Engineering and Computer Science, NYU Shanghai
& Leonard J. Shustek Professor of Computer Science, CSE Dept,

Keith Ross is the Dean of Engineering and Computer Science at NYU Shar
Shustek Chair Professor of Computer Science at NYU. He also holds an aff
the Department of Computer Science at the Courant Institute of Mathema
Center for Data Science at NYU.

Previously he was a professor at University of Pennsylvania (13 years) and
Institute (5 years). He was the Department Head of the CSE Department a
and he joined NYU Shanghai in 2013. He received a Ph.D. in Computer an
from The University of Michigan.

ross.html

User-server state: cookies

Many major Web sites use cookies to *identify users*.

Four components:

1) cookie header line in the HTTP response message

2) cookie header line in HTTP request message

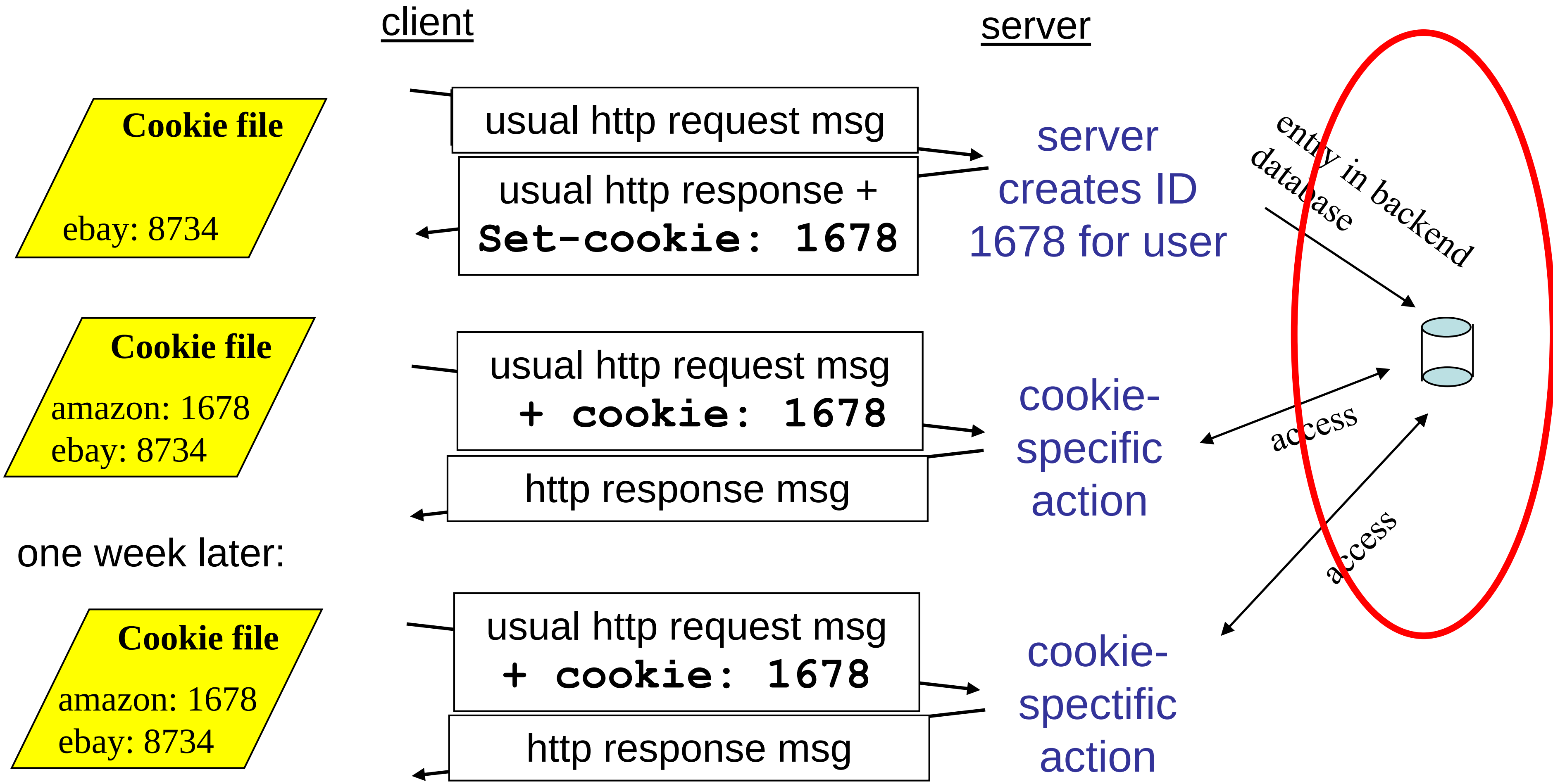
3) cookie file kept on user's host and managed by user's browser

4) back-end database at Web site

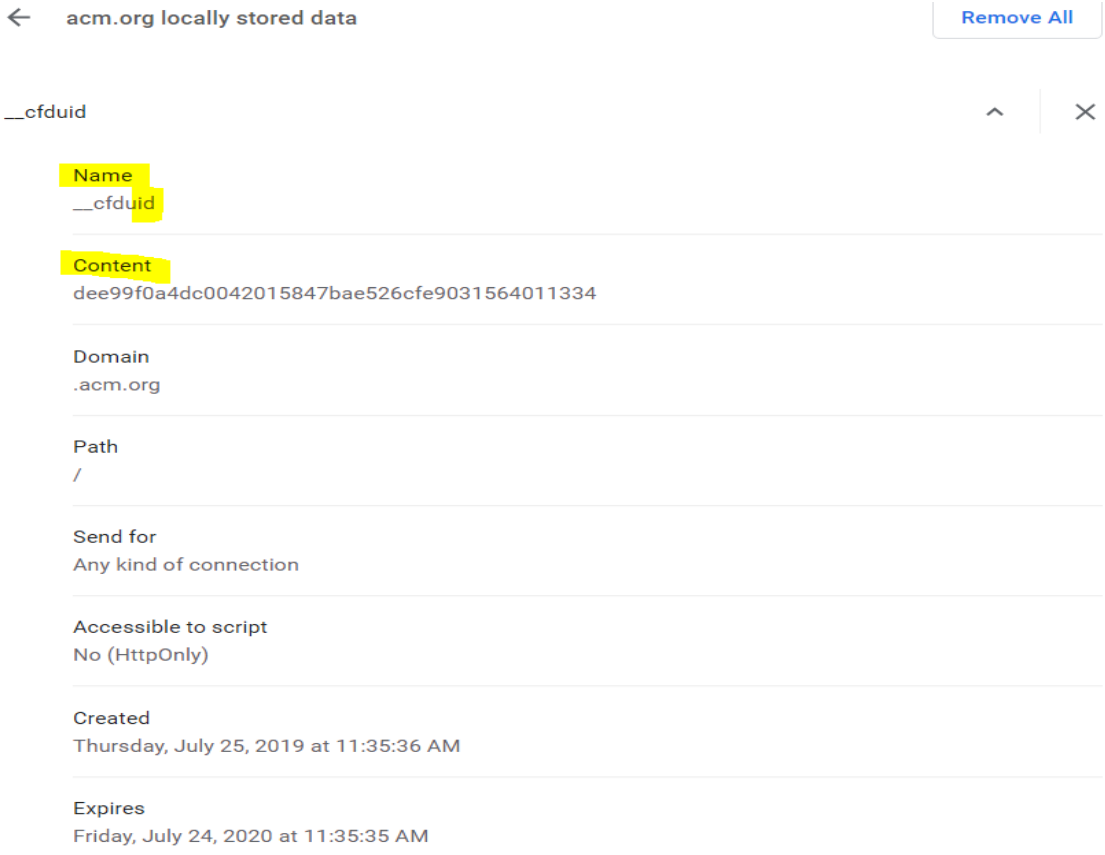
Example:

- Susan access Internet always from same PC
- She visits a specific e-commerce site for first time
- When initial HTTP requests arrives at site, site creates a unique ID and creates an entry in backend database for ID

Cookies: keeping "state" (cont.)



A cookie sample



acm.org locally stored data

Remove All

__cfduid

Name
__cfduid

Content
dee99f0a4dc0042015847bae526cfe9031564011334

Domain
.acm.org

Path
/

Send for
Any kind of connection

Accessible to script
No (HttpOnly)

Created
Thursday, July 25, 2019 at 11:35:36 AM

Expires
Friday, July 24, 2020 at 11:35:35 AM

Cookies (continued)

What cookies can bring:

- shopping carts
- recommendations
- user session state (Web e-mail)

aside

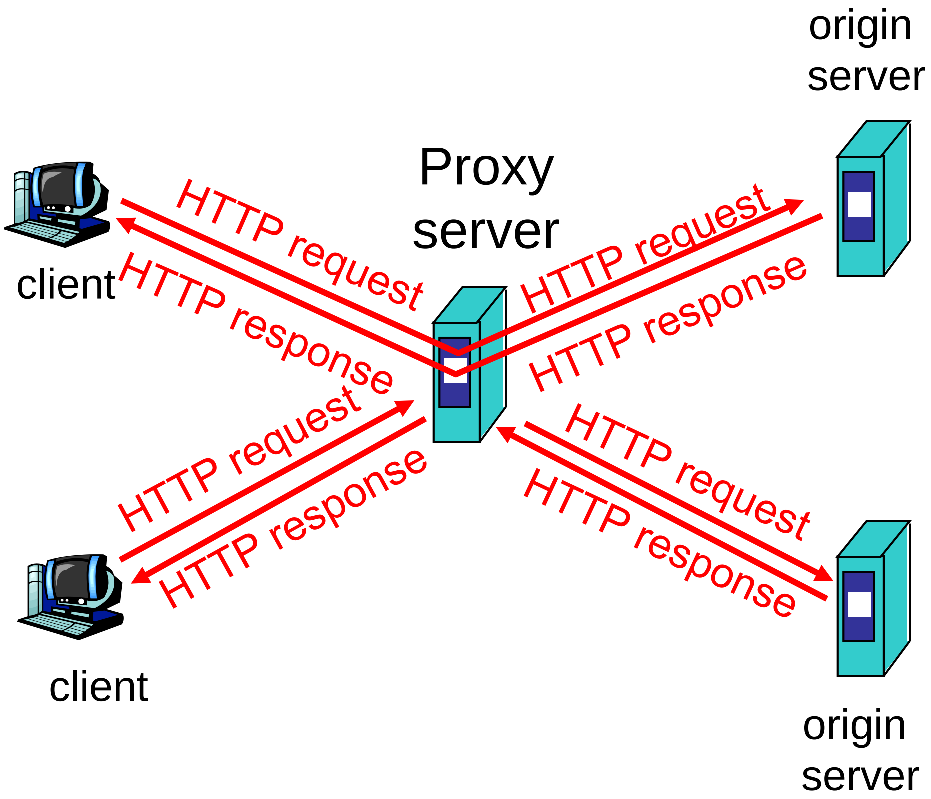
Cookies and privacy:

1. cookies permit sites to learn a lot about you
2. you may supply name and e-mail to sites
3. search engines use redirection & cookies to learn yet more
4. advertising companies obtain info across sites

Web caches (proxy server)

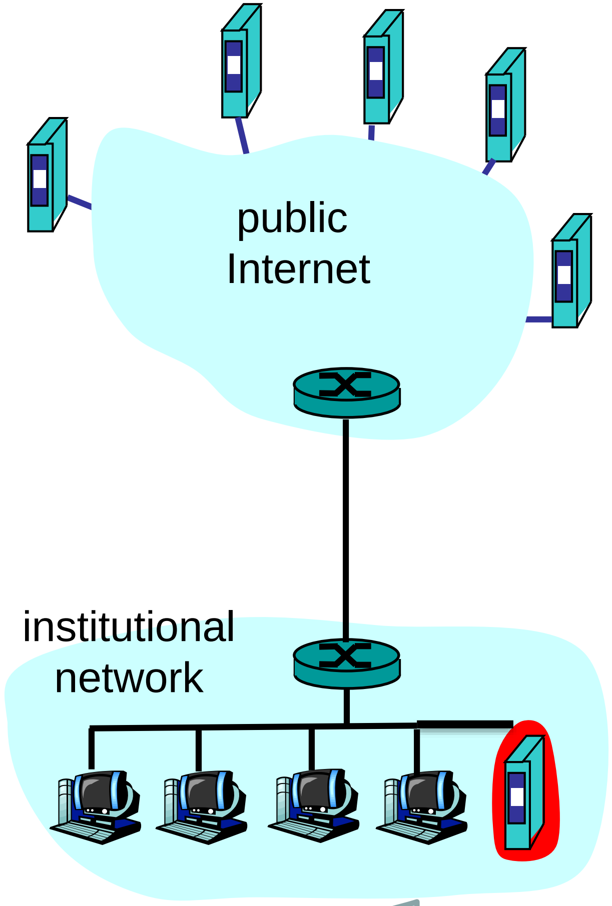
Goal: *satisfy client request without involving origin server*

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



More about Web caching

- Cache acts as both client and server
- Typically cache is installed by ISP (university, company, residential ISP)



Why Web caching?

- Reduce response time for client request.
- Reduce traffic on an institution’s access link.

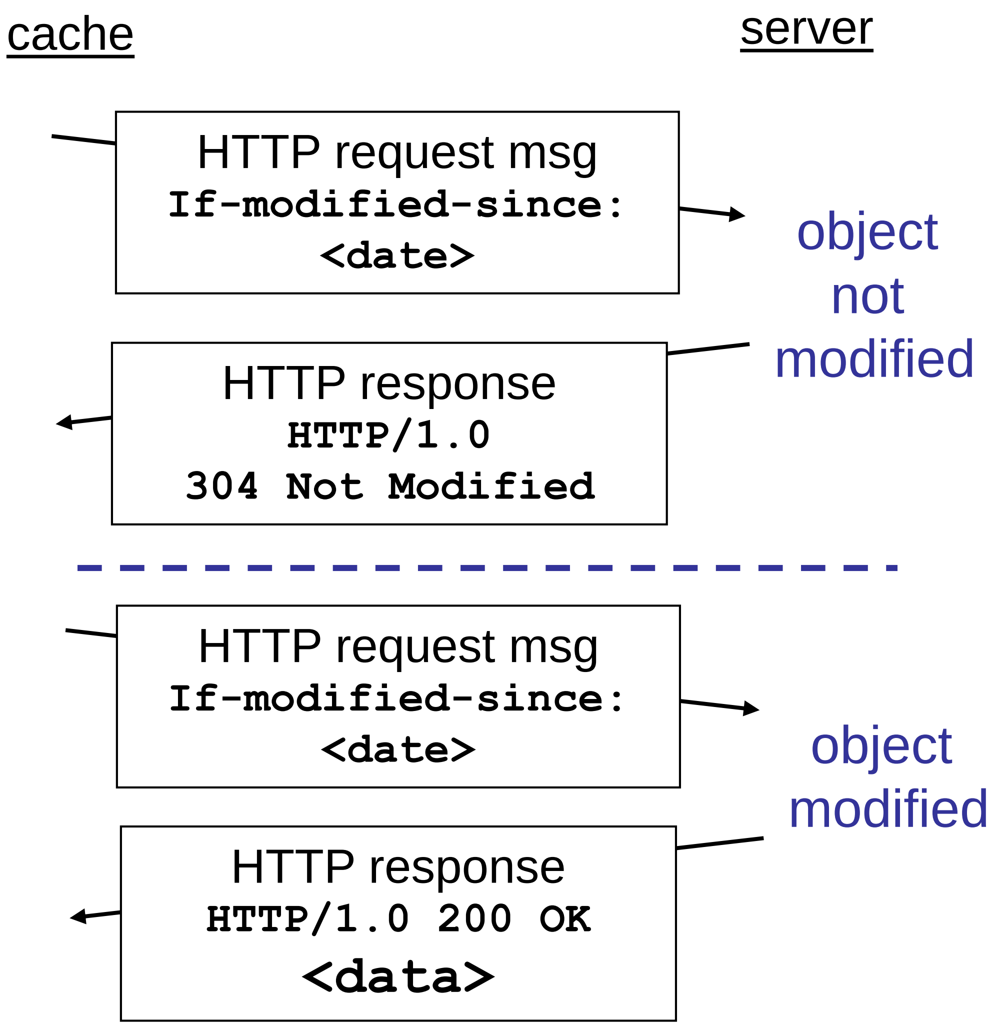
There is always new problem!
The copy of an object in the cache may be stale.

Conditional GET

- Goal: don’t send object if cache has up-to-date cached version
- cache: 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`



cache

server

HTTP request msg
`If-modified-since: <date>`

HTTP response
`HTTP/1.0 304 Not Modified`

object not modified

HTTP request msg
`If-modified-since: <date>`

HTTP response
`HTTP/1.0 200 OK`
`<data>`

object modified

Summary

- Network applications
  - Network apps vs app. protocols
  - Application structure
- The Web
- HTTP

References

- [KR3] James F. Kurose, Keith W. Ross, *Computer networking: a top-down approach featuring the Internet*, 3rd edition.
- [PD5] Larry L. Peterson, Bruce S. Davie, *Computer networks: a systems approach*, 5th edition
- [TW5] Andrew S. Tanenbaum, David J. Wetherall, *Computer network*, 5th edition
- [LHBi]Y-D. Lin, R-H. Hwang, F. Baker, *Computer network: an open source approach*, International edition

Acknowledgements

- All slides are developed based on slides from the following two sources:
  - Dr DongSeong Kim’s slides for COSC264, University of Canterbury;
  - Prof Aleksandar Kuzmanovic’s lecture notes for CS340,Northwestern University, https://users.cs.northwestern.edu/~akuzma/classes/CS340-w05/lecture_notes.htm