Information Network

Lecture 3 : HTTP and the World Wide Web

Holger Thies

KYOTO UNIVERSITY

京都大学

Today’s lecture

- Web and HTTP overview
- HTTP in detail
- HTTP infrastructure

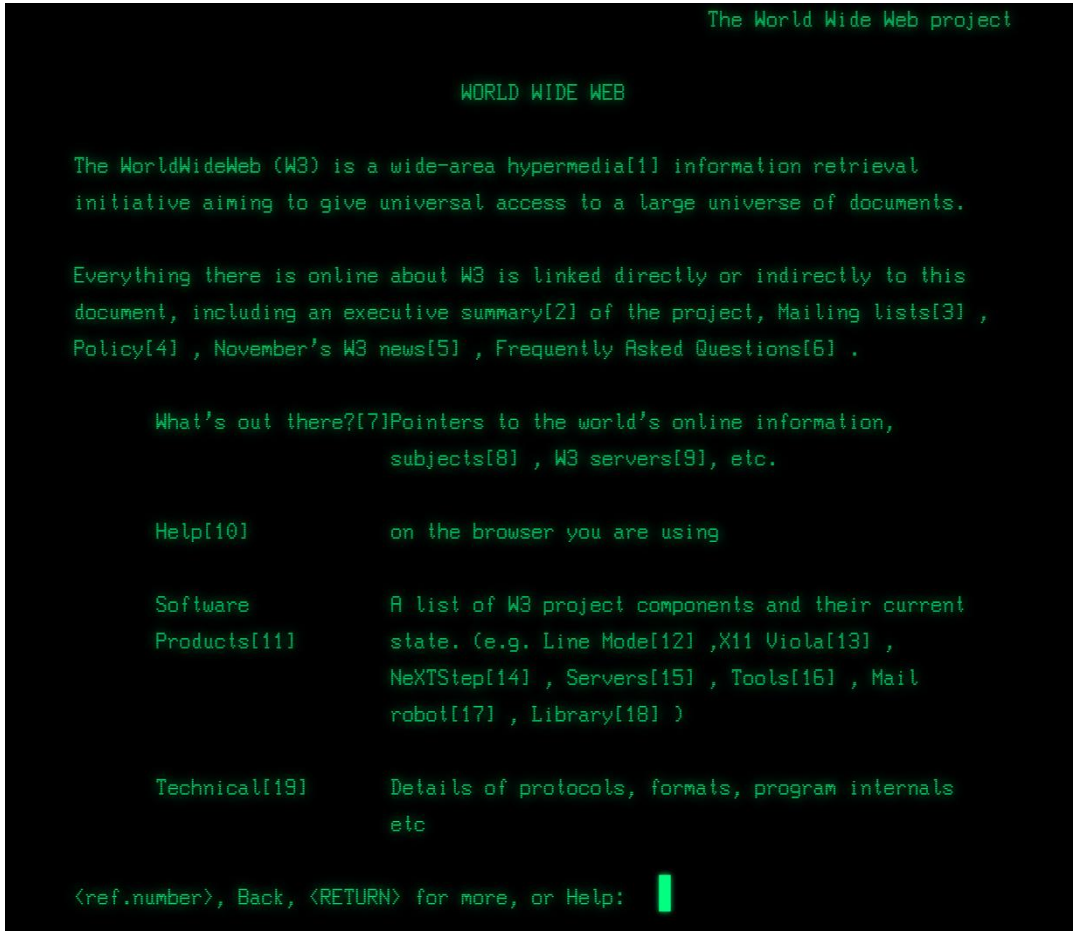
World Wide Web (WWW)

- The World Wide Web (WWW), or short the Web, is a global collection of websites that can be accessed through a web browser and that are interconnected through hyperlinks.
- The Web was developed by Tim Berners-Lee in 1989 at CERN in Switzerland.
- It was originally developed for automated information-sharing between scientists in universities and institutes around the world.
- The terms internet and world wide web are often used synonymously. However, the Web is just one specific service running on the internet.

京都大学

World Wide Web (WWW)

- The **World Wide Web** is a vast, interconnected system of web pages accessible over the internet.
- It consists of the **global collection of web pages** linked together via hyperlinks and hosted on web servers around the world.
- Each **web page** is made up of multiple **objects** (text, images, videos, scripts, etc.).
- These objects can be stored on **different web servers**, meaning a single webpage might retrieve data from multiple sources across the web.
- It uses a (distributed) client-server architecture.
- This architecture allows the distribution of resources and services globally, making it scalable and flexible.



京都大学

Main components of the world wide web

- Web Clients (Browsers)
  - Used to access and interact with web content.
- Web servers
  - Used for storing, processing and delivering web resources.
- Hyper Text Transfer Protocol (HTTP)
  - The protocol to exchange data between client and server.
- Hyper Text Markup Language (HTML)
  - The standard language to create websites.
- Uniform Resource Identifier (URI) / Uniform Resource Locator (URL)
  - Unique address to locate a resource on the web.

Web Browsers

- A web browser is a software that the client runs to initiate the communication with a web server and retrieve and display resources from the server.
- It is used to send requests to web servers and display the response in a readable format for the user.
- Google Chrome, Mozilla Firefox, Safari, Microsoft Edge, etc.

Web Browsers

京都大学

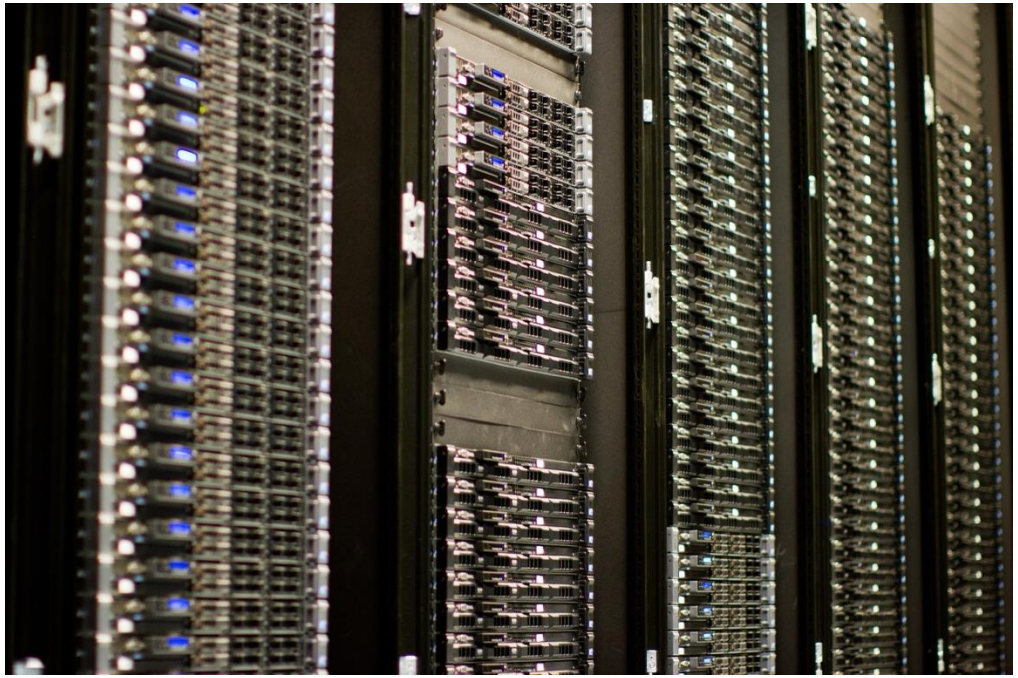
Main components of the world wide web

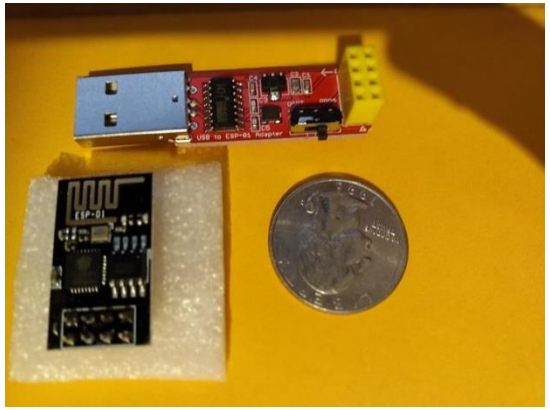
- Web Clients (Browsers)
  - Used to access and interact with web content.
- Web servers
  - Used for storing, processing and delivering web resources.
- Hyper Text Transfer Protocol (HTTP)
  - The protocol to exchange data between client and server.
- Hyper Text Markup Language (HTML)
  - The standard language to create websites.
- Uniform Resource Identifier (URI) / Uniform Resource Locator (URL)
  - Unique address to locate a resource on the web.

Web Server

- A web server stores, processes and delivers resources (HTML files, images, scripts, videos, etc.) to users over the internet.
- It receives requests from web browsers and provides the appropriate responses.
- Usually larger, more powerful computer.
- Can be static (only provide fixed content) or dynamic (generate content on the fly).
- Popular server software includes Apache, Nginx, Microsoft IIS, etc.

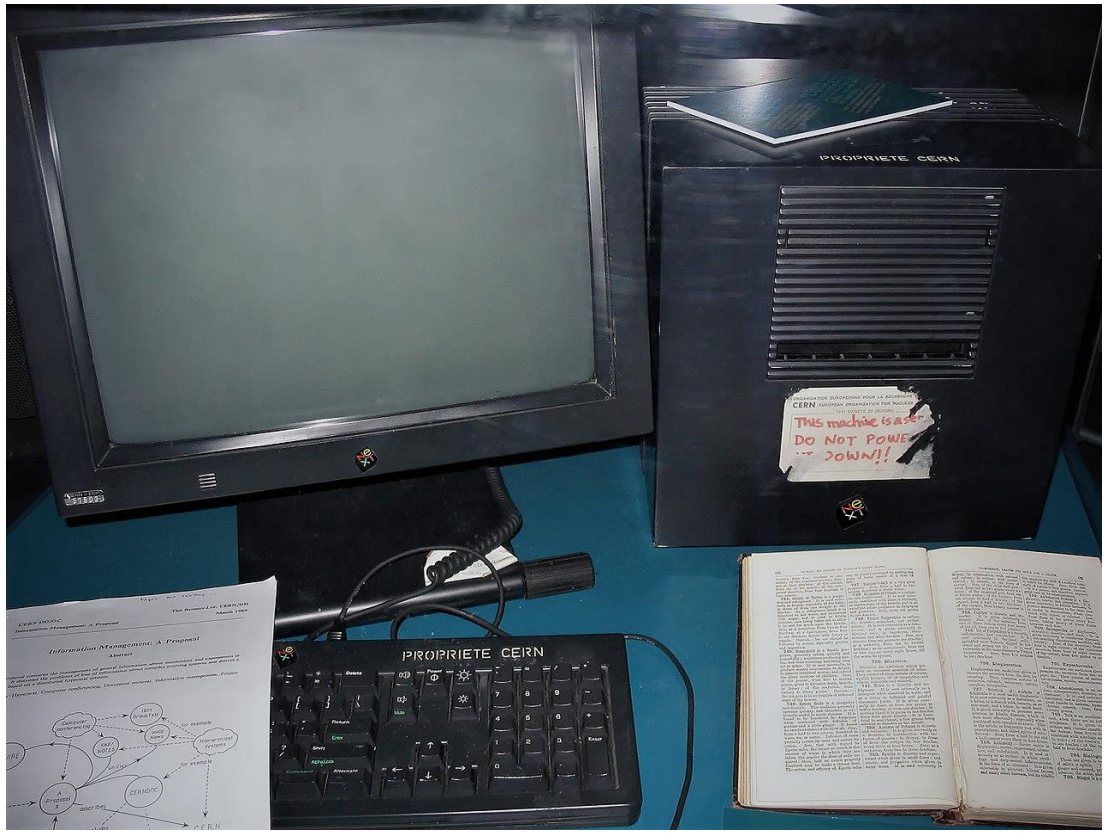
Web Server

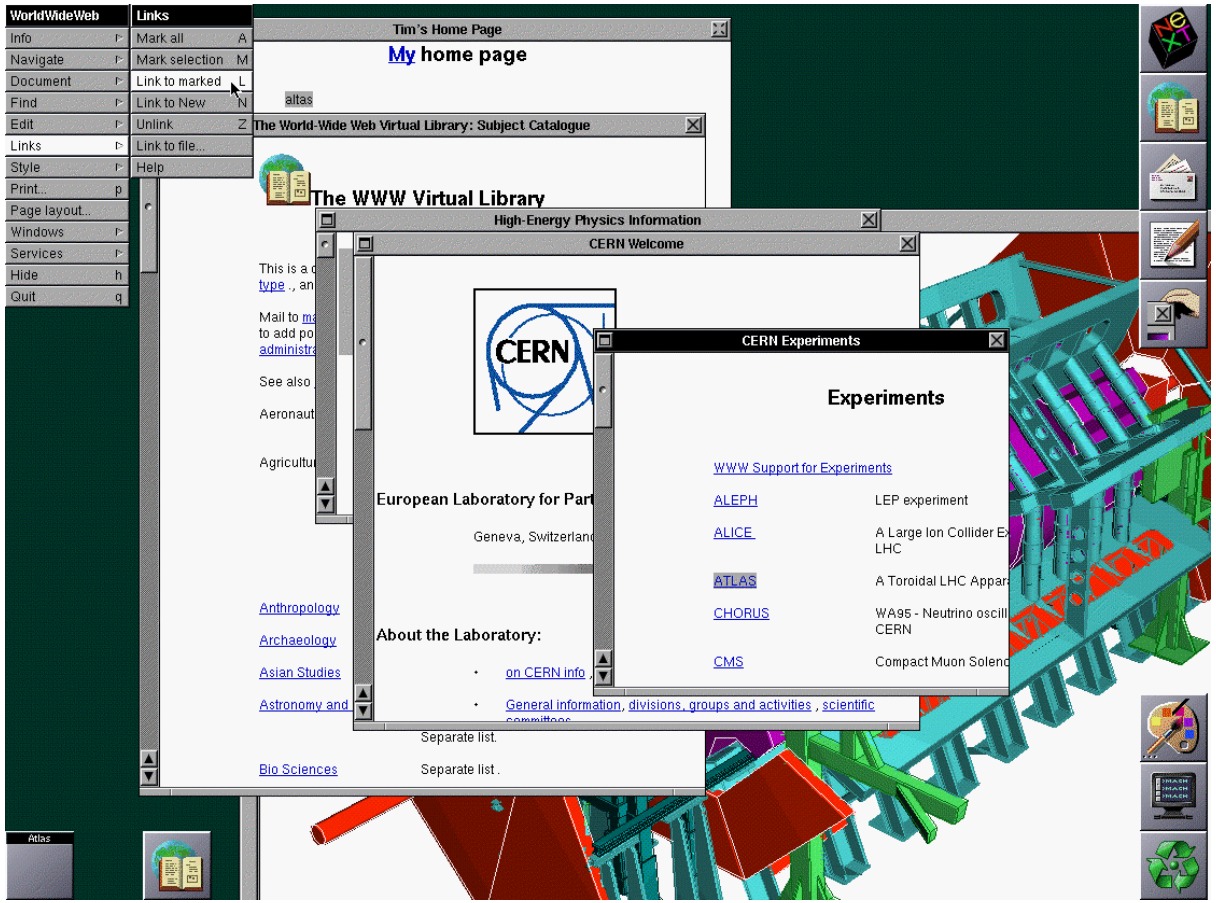




京都大学

First web server and browser





京都大学

Main components of the world wide web

- Web Clients (Browsers)
  - Used to access and interact with web content.
- Web servers
  - Used for storing, processing and delivering web resources.
- Hyper Text Transfer Protocol (HTTP)
  - The protocol to exchange data between client and server.
- Hyper Text Markup Language (HTML)
  - The standard language to create websites.
- Uniform Resource Identifier (URI) / Uniform Resource Locator (URL)
  - Unique address to locate a resource on the web.

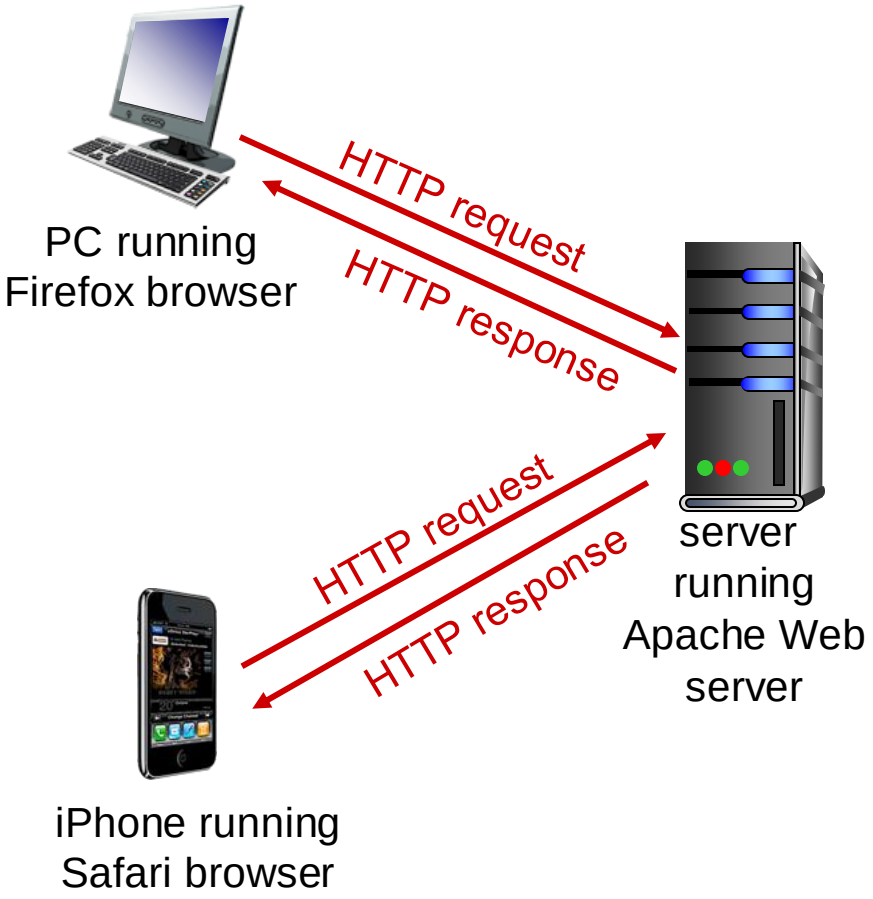
Hyper Text Transfer Protocol (HTTP)

- HTTP (Hypertext transfer protocol) is the application layer protocol for transferring various forms of data (plaintext, images, videos, sounds, etc.) between a server and a client over the web.
- Web browsers use HTTP to request resources.
- Web servers process the request and send back a HTTP response.
- HTTPS（Hypertext Transfer Protocol Secure) is an encrypted version of HTTP, used to make the communication secure.

HTTP overview

HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - *server:* Web server sends (using HTTP protocol) objects in response to requests



京都大学

Main components of the world wide web

- Web Clients (Browsers)
  - Used to access and interact with web content.
- Web servers
  - Used for storing, processing and delivering web resources.
- Hyper Text Transfer Protocol (HTTP)
  - The protocol to exchange data between client and server.
- Hyper Text Markup Language (HTML)
  - The standard language to create websites.
- Uniform Resource Identifier (URI) / Uniform Resource Locator (URL)
  - Unique address to locate a resource on the web.

Hyper Text Markup Language (HTML)

- Hypertext is text which contains links to other texts.
- HTML is a subset of Standardized General Markup Language (SGML) to generate hypertext documents.
- HTML contains embedded links to other documents and applications.
- Documents use elements to “mark up” or identify sections of text for different purposes or display characteristics,
- Mark up elements are not seen by the user when page is displayed.
- Documents are rendered by browsers.
- Not all documents in the Web are HTML.

HTML Example

```
<HTML>
<HEAD>
<TITLE> My Homepage</TITLE>
</HEAD>
<BODY>
<IMG SRC = “picture.gif” />
<P><CENTER><H1>Welcome to my Page</H1></CENTER>
Main text…
<A HREF = “http:/www.kyoto-u.ac.jp”> Kyoto University</A>
</BODY>
</HTML>
```

京都大学

Main components of the world wide web

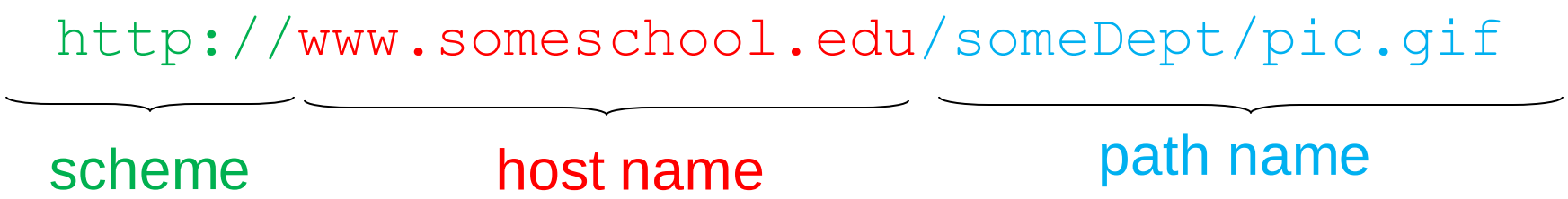
- Web Clients (Browsers)
  - Used to access and interact with web content.
- Web servers
  - Used for storing, processing and delivering web resources.
- Hyper Text Transfer Protocol (HTTP)
  - The protocol to exchange data between client and server.
- Hyper Text Markup Language (HTML)
  - The standard language to create websites.
- Uniform Resource Identifier (URI) / Uniform Resource Locator (URL)
  - Unique address to locate a resource on the web.

Uniform Resource Locator (URL)

- Each object needs a unique identifier, so that clients can point out which resource they want.
- Such an identifier is called Uniform Resource Identifier (URI).
- The web uses a specific type of URI called Uniform Resource Locator (URL).
  - An URL identifies a resource and provides its location on the web.
- URLs tell you how to fetch a resource from a precise, fixed location.
- Clients include the URL of the resource in the HTTP request message.

URLs

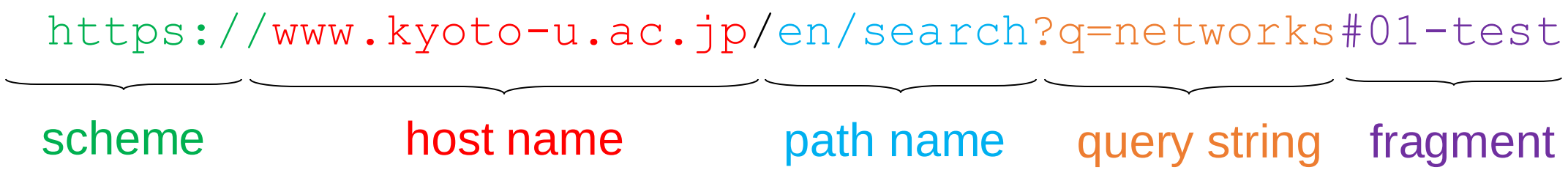
- Most URLs follow a standardized format of three main parts.



- The scheme describes the protocol used to access the resource (usually http:// or https://).
- The host name gives the server Internet address.
- The rest gives the path to a resource on the web server.

Query String and Fragment

URLs can further provide additional parameters:



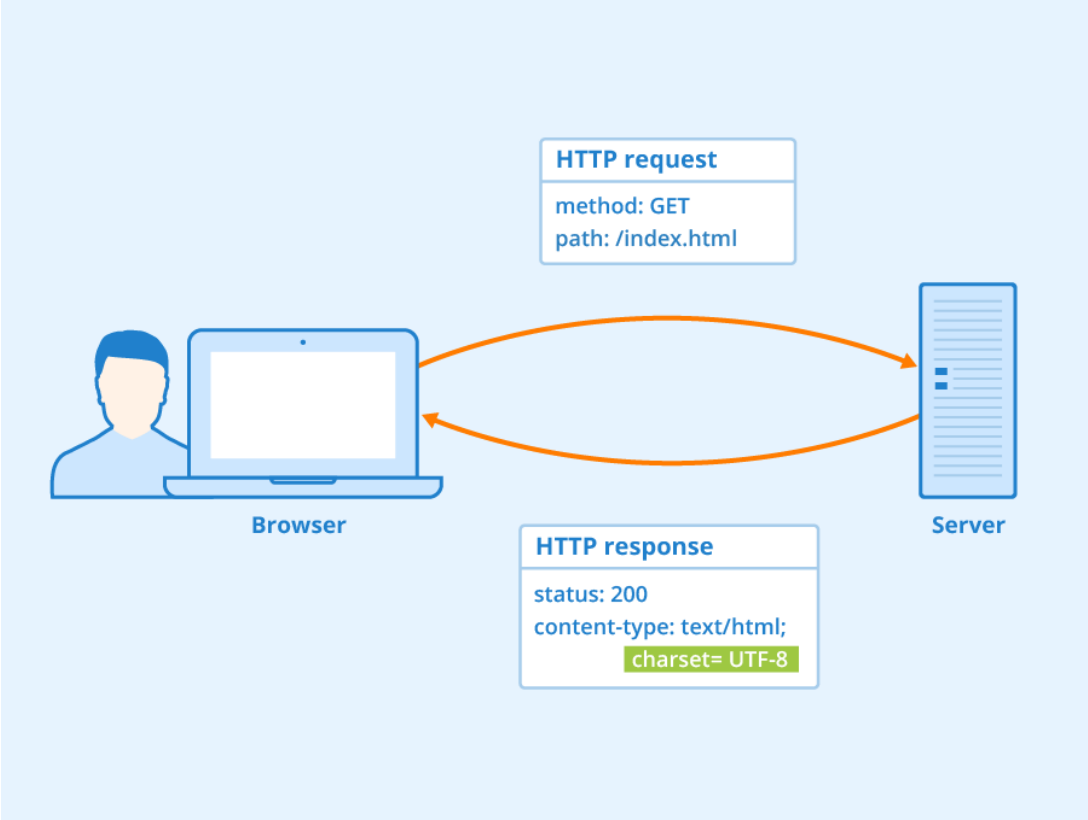
- The query string provides additional parameters for the request used in dynamic pages.
- The fragment can point to a specific location in a web site

Today's lecture

- Web and HTTP overview
- HTTP in detail
- HTTP infrastructure

HTTP overview

- HTTP stands for hypertext transfer protocol.
- It is used to exchange data on the world wide web.
- HTTP is a simple request-response protocol.
- A client (browser) sends a HTTP request to a web server.
- The web server receives and processes the request.
- The server replies with a HTTP response message.
- The browser receives the message and displays the result.



京都大学

HTTP in Detail

- What type of messages are there and how do they differ?
- What is contained in the messages and how are they formatted?
- What transport layer protocol does HTTP use, and why?
- How are HTTP connections opened and closed?
- What are some additional services added on top of HTTP and how are they implemented?

Main Features of HTTP

- Client-Server Architecture:
  - Clients send requests, and servers provide responses.
- Stateless protocol
  - Each request from a client to a server is independent.
  - The server does not remember information from previous requests.
- Text-Based Communication:
  - Uses plain text for communication between client and server.
  - Headers, body, and status codes are in human-readable format.

Message types

- HTTP has two types of messages request messages and response messages.
  There are no other type of messages.
- Request messages are messages sent from web clients to web servers, response messages are sent from web servers to web clients.
- HTTP request and response messages are in human readable format.
- Both request and response message contain additional header information to specify the type of request/response and provide further details.

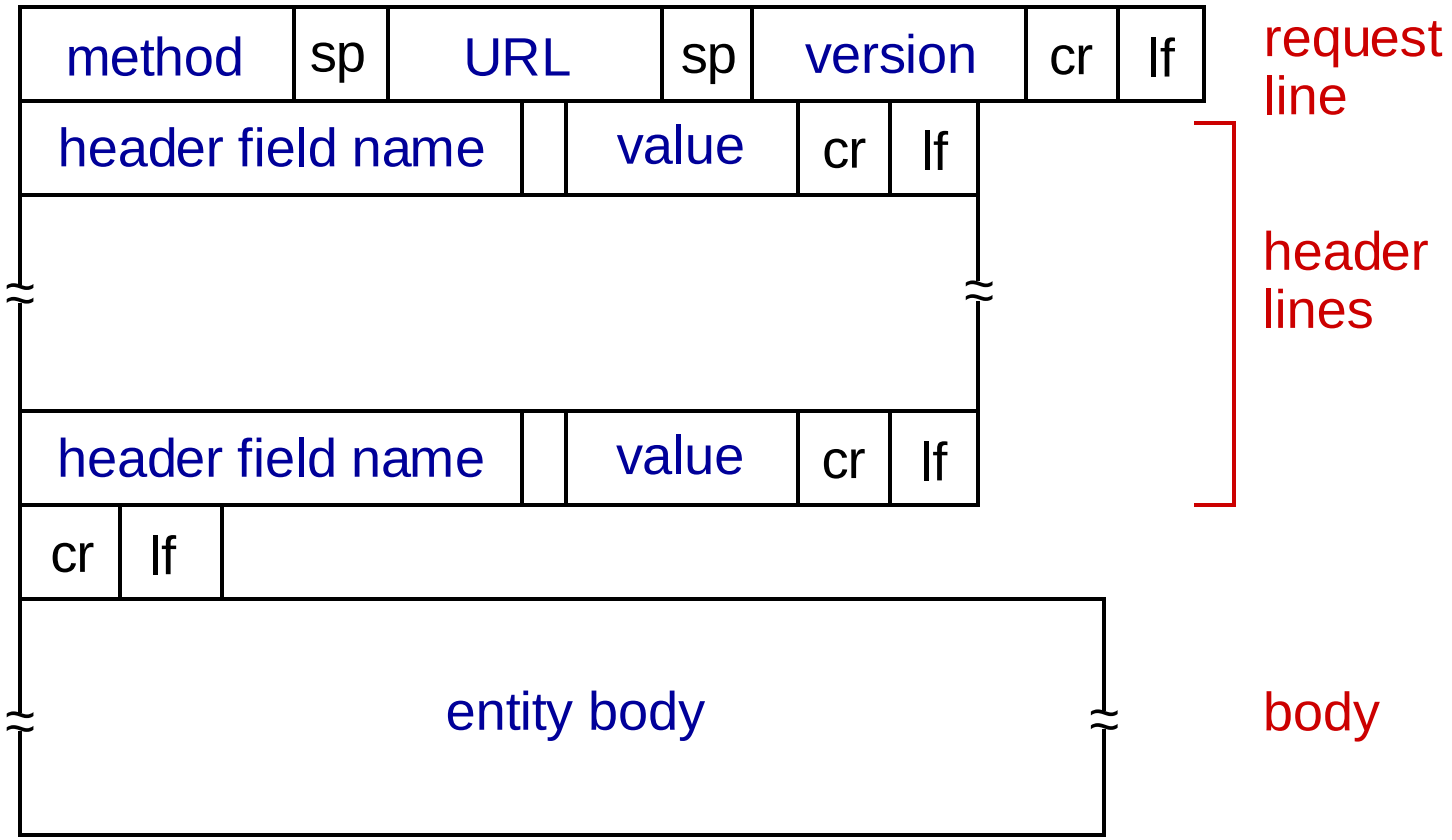
HTTP message format

- Request and response messages have almost the same format.
- HTTP messages are line-oriented sequences of characters, written in plain-text and readable by humans.
- Lines are separated by special characters: a carriage return (cr) and line-feed (lf) character (“\r\n”).
- HTTP messages consist of three parts
  - Start line
  - Header Fields
  - Message Body

HTTP message format

- HTTP messages consist of three parts
  - Start line: The first line of the message indicates what to do for the request or what kind of response is received. For request messages the start line is also called request line and for response messages it is called status line.
  - Header fields: The start line is followed by zero or more lines for header fields that can provide some additional information about the request, the client/server, etc. Each header field consists of a name and a value, separated by a colon (:).
    The end of the header lines is indicated by a blank line, i.e., a line only containing a carriage return and line-feed character.
  - Body: The rest of the message is called the message body and can contain any kind of data. In request messages the body is used to send data to the web server and in response messages the body usually contains the data that is requested from the server.

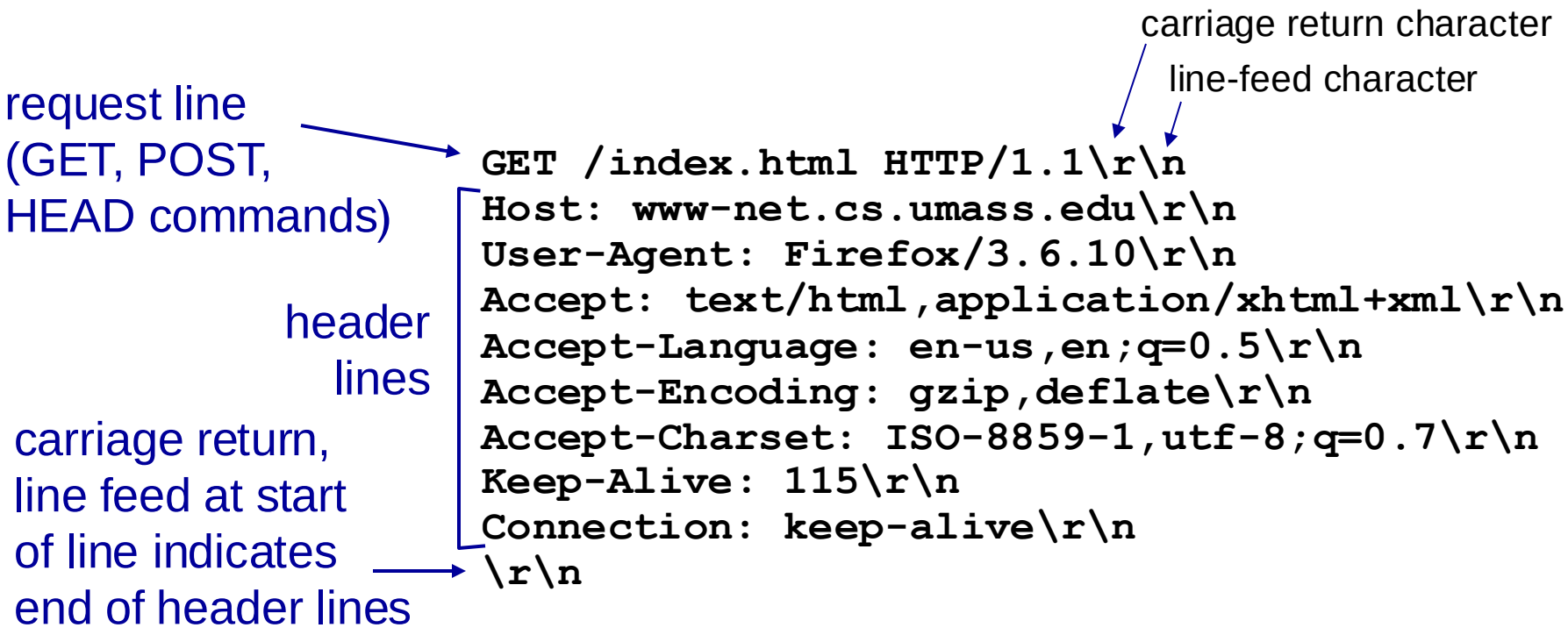
HTTP request message: general format



京都大学

HTTP request message

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



Method types

- GET
  - Retrieve a resource/data from the server
- POST
  - Send data to a resource and process it
- HEAD
  - Only get information about a resource, without sending the actual resource
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field

GET and POST method

Get method:

- Used for retrieving data from the server
- Should not have side-effects on the server
  - mostly idempotent, multiple requests should return the same result
- Data is sent in the URL:

```
https://www.google.com/search?q=kyoto+university
```

Post method:

- Used to send data to be processed by the specified resource
- Can have side-effects on the server, non-idempotent
- Data is sent in the request body

Uploading form input

POST method:

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

URL method:

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

```
www.somesite.com/animalsearch?monkeys&banana
```

HTTP response message

status line (protocol status code status phrase) → `HTTP/1.1 200 OK`

HTTP response status codes

- status code appears in 1st line in server-to-client response message.
- Important status codes:

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

- Not allowed to access the document

404 Not Found

- requested document not found on this server

500 Internal server error

HTTP response status codes

- status code appears in 1st line in server-to-client response message.
- Important status codes:

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

- Not allowed to access the document

404 Not Found

- requested document not found on this server

500 Internal server error

josh@blackbox: ~

File Edit View Terminal Tabs Help

```
josh@blackbox:~$ telnet en.wikipedia.org 80
Trying 208.80.152.2...
Connected to rr.pmtpa.wikimedia.org.
Escape character is '^]'.
GET /wiki/Main_Page http/1.1
Host: en.wikipedia.org

HTTP/1.0 200 OK
Date: Thu, 03 Jul 2008 11:12:06 GMT
Server: Apache
X-Powered-By: PHP/5.2.5
Cache-Control: private, s-maxage=0, max-age=0, must-revalidate
Content-Language: en
Vary: Accept-Encoding,Cookie
X-Vary-Options: Accept-Encoding;list-contains=gzip,Cookie;string-contains=enwikiToken;string-contains=enwikiLoggedOut;string-contains=enwiki_session;
string-contains=centralauth_Token;string-contains=centralauth_Session;string-contains=centralauth_LoggedOut
Last-Modified: Thu, 03 Jul 2008 10:44:34 GMT
Content-Length: 54218
Content-Type: text/html; charset=utf-8
X-Cache: HIT from sq39.wikimedia.org
X-Cache-Lookup: HIT from sq39.wikimedia.org:3128
Age: 3
X-Cache: HIT from sq38.wikimedia.org
X-Cache-Lookup: HIT from sq38.wikimedia.org:80
Via: 1.0 sq39.wikimedia.org:3128 (squid/2.6.STABLE18), 1.0 sq38.wikimedia.org:80 (squid/2.6.STABLE18)
Connection: close

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" dir="ltr">
        <head>
                <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
                                <meta name="keywords" content="Main Page,1778,1844,1863,1938,1980 Summer Olympics,2008,2008 Guizhou riot,2008 Jerusal
...
''' This content has been removed to save space
'''
"Non-profit organization">nonprofit</a> <a href="http://en.wikipedia.org/wiki/Charitable_organization" title="Charitable organization">charity</a>.<b
r /></li>
                                <li id="privacy"><a href="http://wikimediafoundation.org/wiki/Privacy_policy" title="wikimedia:Privacy policy">Privac
y policy</a></li>
                                <li id="about"><a href="/wiki/Wikipedia:About" title="Wikipedia:About">About Wikipedia</a></li>
                                <li id="disclaimer"><a href="/wiki/Wikipedia:General_disclaimer" title="Wikipedia:General disclaimer">Disclaimers</a>
</li>
                        </ul>
                </div>
</div>

                <script type="text/javascript">if (window.runOnloadHook) runOnloadHook();</script>
<!-- Served by srv93 in 0.050 secs. --></body></html>
Connection closed by foreign host.
josh@blackbox:~$
```

Request

Response headers

Response body

京都大学

HTTP versions and history

- **1991:** HTTP/0.9
  - Simple, one-line requests (method + path)
    - GET /mypage.html
  - Response is just the requested file
  - no header support etc.
  - Only HTML files can be transmitted
  - Connection is terminated after the file is transferred
- **1996:** HTTP/1.0
  - Introduced headers, making the protocol more flexible and extensible
  - Allows to transfer other types of content than just plain HTML files
  - Added different method types
  - Status code line added to the beginning of the response, telling the browser if the request was a success or failure
  - No standard yet: Servers and browsers often experimentally added new features, no interoperability
  - Connection between client and server is closed after each request

HTTP versions and history

- **1999:** HTTP/1.1
  - First standardized version
  - Connection can be kept alive (persistent HTTP) and other performance optimizations
  - Content negotiation, allowing different languages, encodings, etc.
  - Introduced new method types
- **2015:** HTTP/2
  - Mostly performance improvements
  - Allows sending multiple requests concurrently
  - Allows server to push data to the client
- **2020:** HTTP/3
  - Uses a new transport protocol (QUIC) based on UDP

Nowadays most larger websites use HTTP/2 but HTTP/1.1 is still used and HTTP/3 is becoming more popular.

Connections

- HTTP is an application layer protocol and thus does not need to worry about the details of the network communication.
- These details are left to the transport-layer protocol.
- We have seen that there are two main choices for protocols UDP and TCP.
  - UDP does not provide any guarantees.
  - TCP provides error-free data transportation and in-order delivery of messages.
- HTTP uses the TCP protocol to transport messages between client and server.
- Once a TCP connection is established, messages between client and server will never be lost, damaged or received out of order.
- Thus, the HTTP protocol can assume that messages are always delivered reliably.

HTTP connection overview

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

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

Persistent and Non-Persistent HTTP

- There is two types of HTTP connections: persistent and non-persistent.
- Persistent means that the connection is kept open for multiple requests/responses.
- Non-persistent is simpler and was used in the early times of HTTP.
- Nowadays, mostly persistent HTTP is used.
- Non-persistent might still be used in some simple applications.

Non-persistent HTTP

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 on port 80

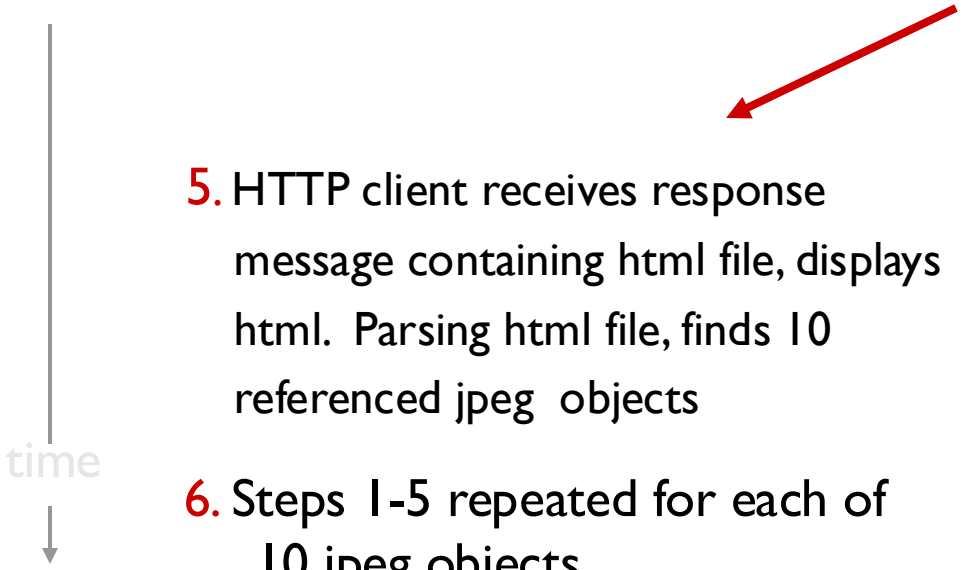
1b. HTTP server at host www.someSchool.edu waiting for TCP connection at port 80. "accepts" connection, notifying client

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

time

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

HTTP connections

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

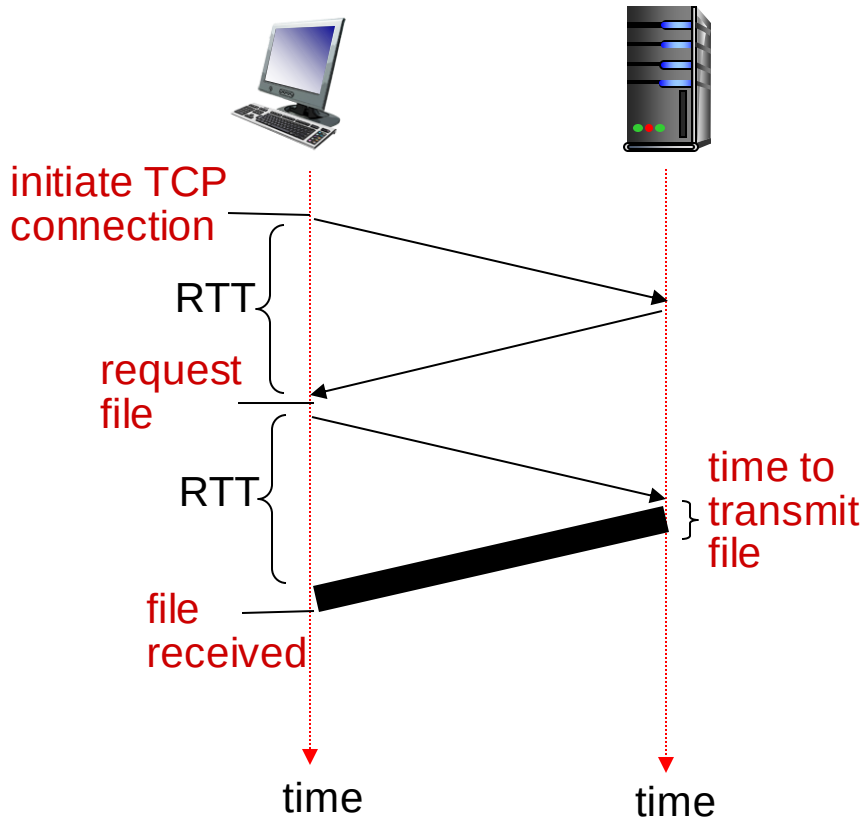
Non-persistent HTTP: response time

RTT (definition): round-trip time
time for a 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



京都大学

Persistent HTTP

non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for *each* TCP connection

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

Today's lecture

- Web and HTTP overview
- HTTP in detail
- HTTP infrastructure
  - Cookies
  - Web caches

HTTP infrastructure

- Several technologies exist to enhance HTTP and to facilitate communication over the internet.
- Cookies are used to introduce state, e.g., to keep track of users etc.
- Web caches or proxy servers are used to store copies of frequently used content, reducing the load on servers and improving

Cookies

- HTTP is stateless.
  - Simplifies server design which is necessary to support thousands of simultaneous TCP connections.
- However, it is often desirable for a web site to identify users.
- HTTP uses **cookies** to allow sites to keep track of users.

What is a cookie?

- Cookies are used by web sites to remember information about the user.
- A cookie is a small piece of data that the web server sends to the user's web browser.
- The browser stores the data and sends it back to the server each time when the user accesses the same domain.
- Two types of cookies
  - Session Cookies: These types of browser cookies delete once the session ends.
  - Permanent Cookies: These types of browser cookies remain on the system and communicate with the server every time the website opens.

User-server state: cookies

many Web sites use cookies

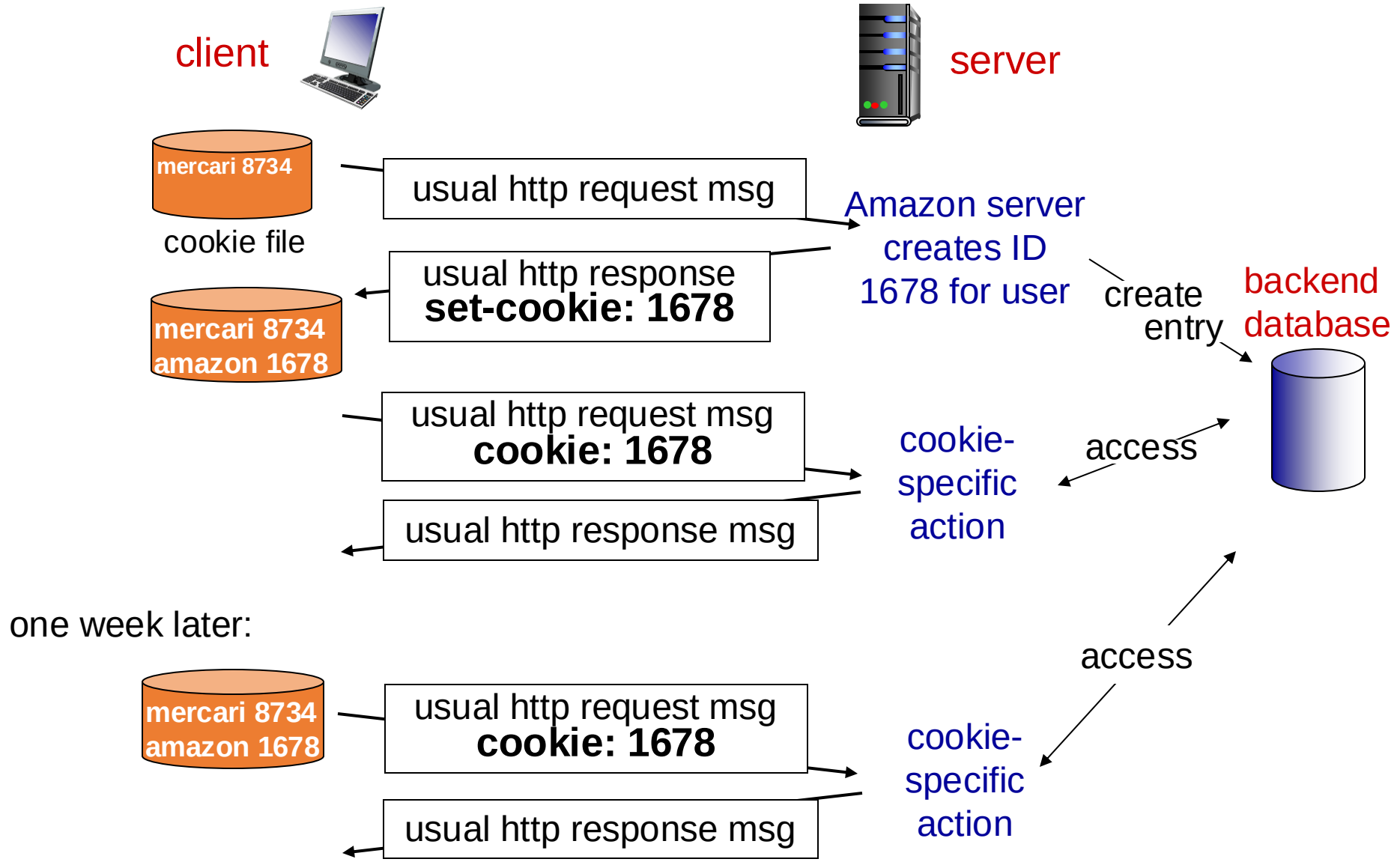
four components:

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:

- User 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" (cont.)



京都大学

Cookies (continued)

what cookies can be used for:

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

cookies and privacy:

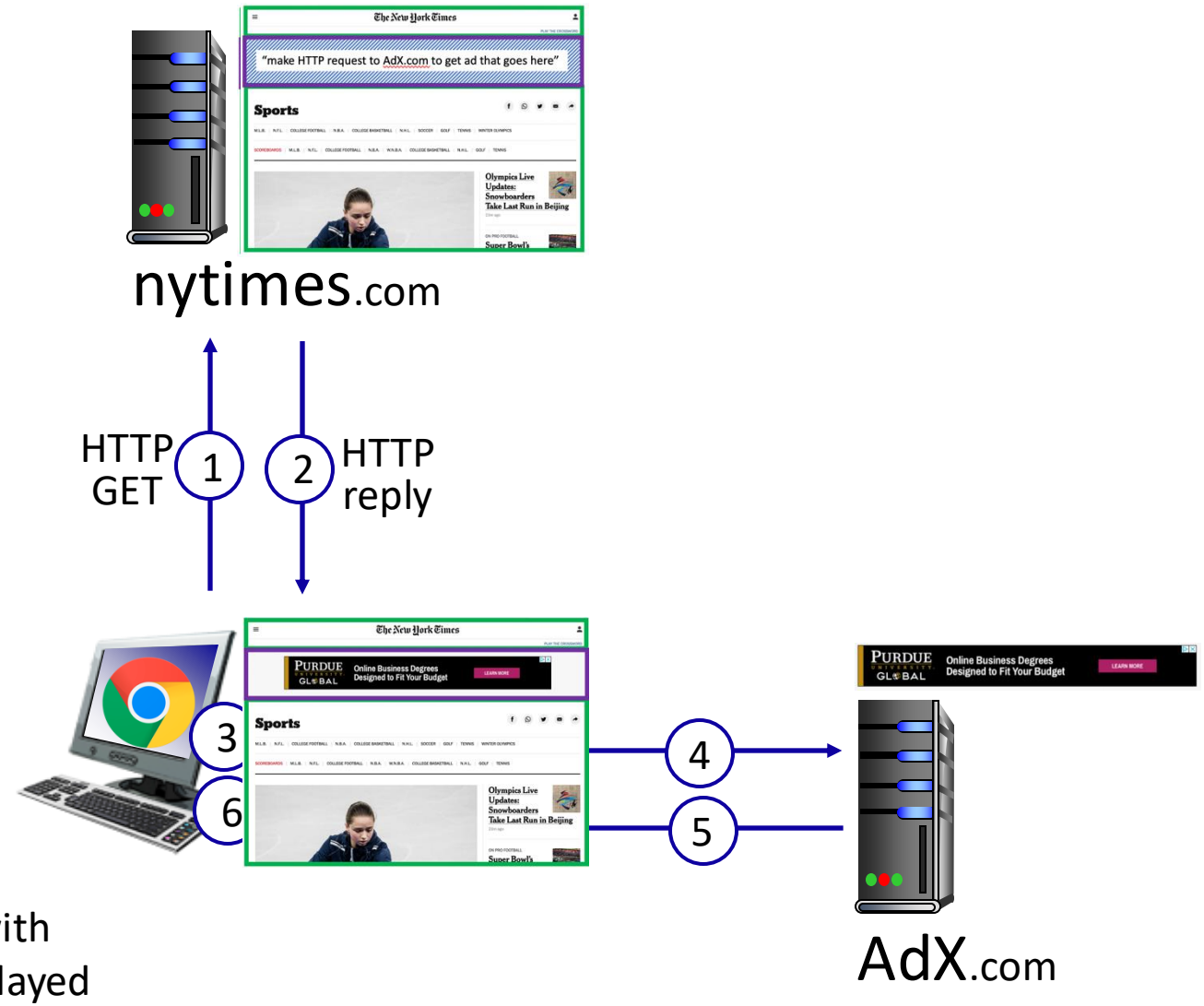
- Cookies are integral to web functionality, but they also raise privacy concerns.
- Cookies permit sites to learn a lot about you.
- They can track user behavior and collect personal information which is sometimes shared with third parties.
- They are used for profiling and targeted advertising.
- Regulations like GDPR have been introduced.

Example: displaying a NY Times web page

1 2 GET base html file from nytimes.com

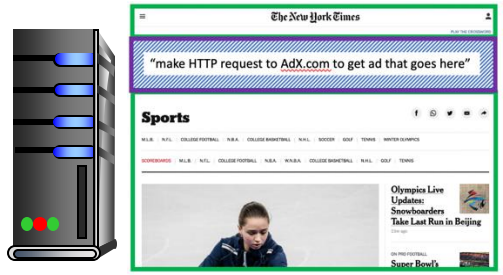
4 5 fetch ad from AdX.com

7 display composed page



NY times page with embedded ad displayed

Cookies: tracking a user's browsing behavior



nytimes.com (sports)

1634: sports, 2/15/22

"first party" cookie – from website you chose to visit (provides base html file)

HTTP GET

HTTP reply
Set cookie: 1634

"third party" cookie – from website you did not choose to visit

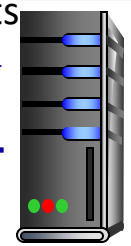
NY Times: 1634
AdX: 7493

HTTP GET
Referrer: NY Times Sports

4

5

HTTP reply
Set cookie: 7493

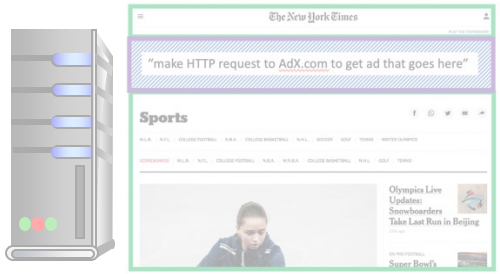


AdX.com

7493: NY Times sports, 2/15/22

Cookies: tracking a user's browsing behavior

socks.com



nytimes.com

HTTP reply

2

HTTP GET

1

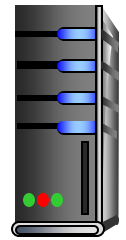
NY Times: 1634
AdX: 7493

HTTP GET
Referrer: socks.com, cookie: 7493

4

5

HTTP reply
Set cookie: 7493

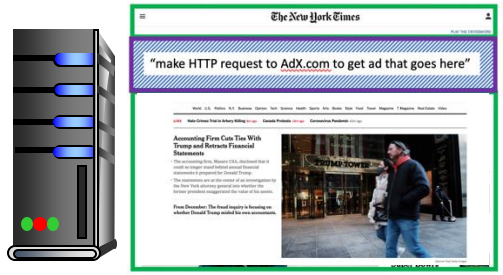


AdX.com

AdX:

- *tracks my web browsing* over sites with AdX ads
- can return targeted ads based on browsing history

Cookies: tracking a user's browsing behavior (one day later)



1634: sports, 2/15/22
1634: arts, 2/17/22

HTTP GET
cookie: 1634

HTTP reply
Set cookie: 1634

HTTP GET
Referrer: nytimes.com, cookie: 7493

4

5

HTTP reply
Set cookie: 7493
Returned ad for socks!

NY Times: 1634
AdX: 7493

AdX.com

7493: NY Times sports, 2/15/22
7493: socks.com, 2/16/22
7493: NY Times arts, 2/15/22

Cookies: tracking a user's browsing behavior

Cookies can be used to:

- track user behavior on a given website (first party cookies)
- track user behavior across multiple websites (third party cookies) without user ever choosing to visit tracker site (!)
- tracking may be *invisible* to user:
  - rather than displayed ad triggering HTTP GET to tracker, could be an invisible link

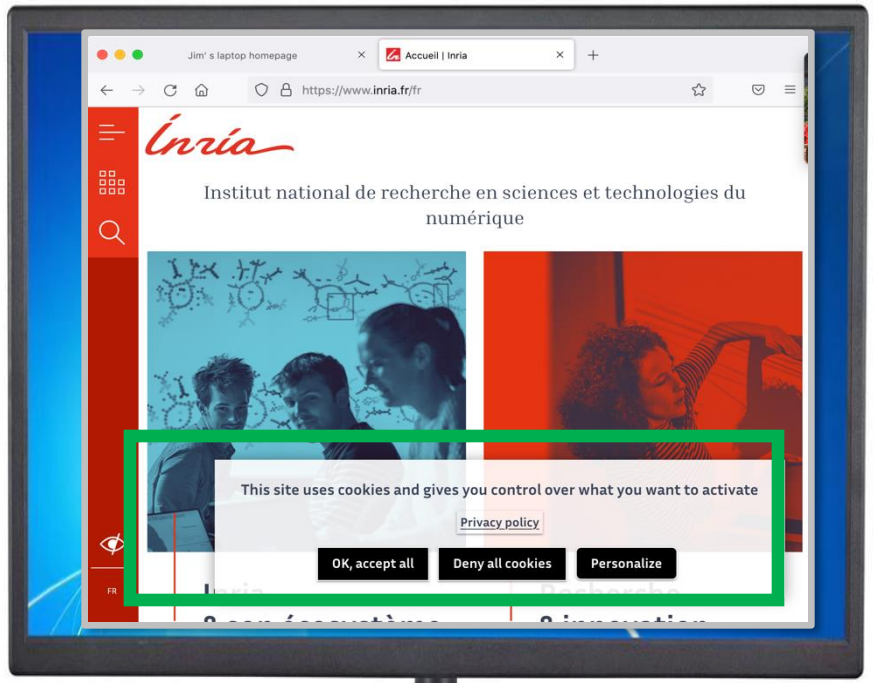
GDPR (EU General Data Protection Regulation) and cookies

"Natural persons may be associated with online identifiers [⋯] such as internet protocol addresses, cookie identifiers or other identifiers [⋯].

This may leave traces which, in particular when combined with unique identifiers and other information received by the servers, may be used to create profiles of the natural persons and identify them."

GDPR, recital 30 (May 2018)

when cookies can identify an individual, cookies are considered personal data, subject to GDPR personal data regulations



User has explicit control over whether or not cookies are allowed

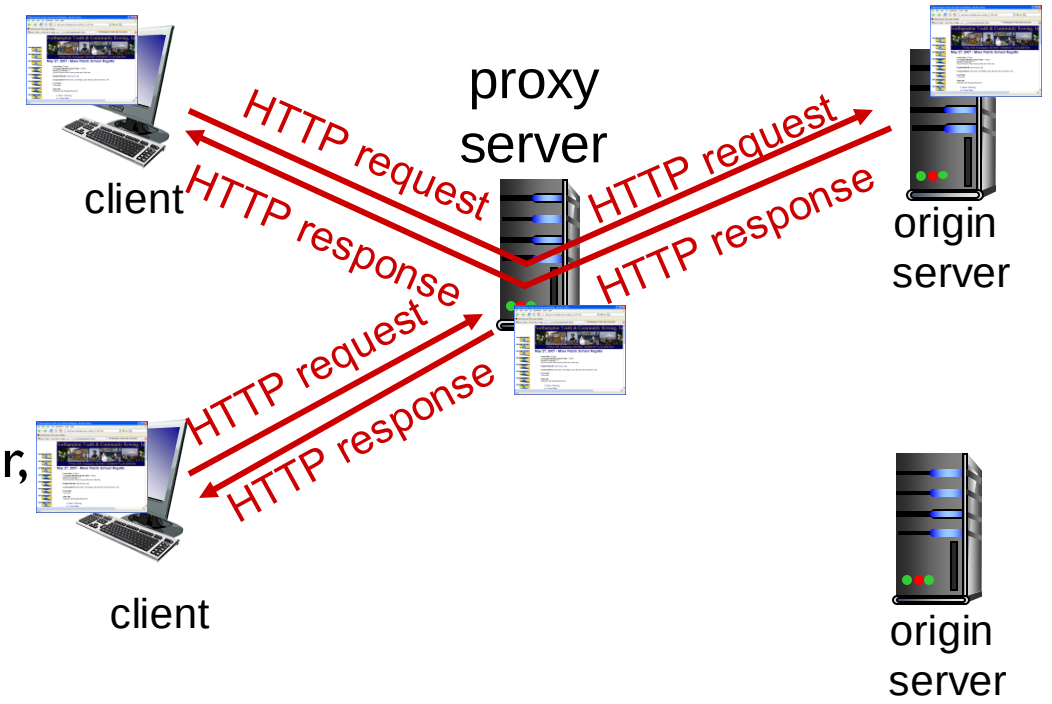
Today's lecture

- Web and HTTP overview
- HTTP in detail
- HTTP infrastructure
  - Cookies
  - Web caches

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



京都大学

Web cache example

Suppose browser is requesting http://www.somepage.jp/image.gif

1. The browser establishes a TCP connection to the Web cache.
2. The browser sends a HTTP request for the object to the web cache.
3. The web cache checks to see if a copy of the object is stored locally.
4. If the web cache does have the object, it returns the object in a HTTP response message.
5. If the web cache does not have the object, it opens a TCP connection to the origin server http://www.somepage.jp.
6. The web cache receives the object from the origin server.
7. The web cache stores a copy of the object in its local storage and sends a copy of the response message to the client.

More about Web caching

- cache acts as both client and server
  - server for original requesting client
  - client to origin server
- typically cache is installed by ISP (university, company, residential ISP)

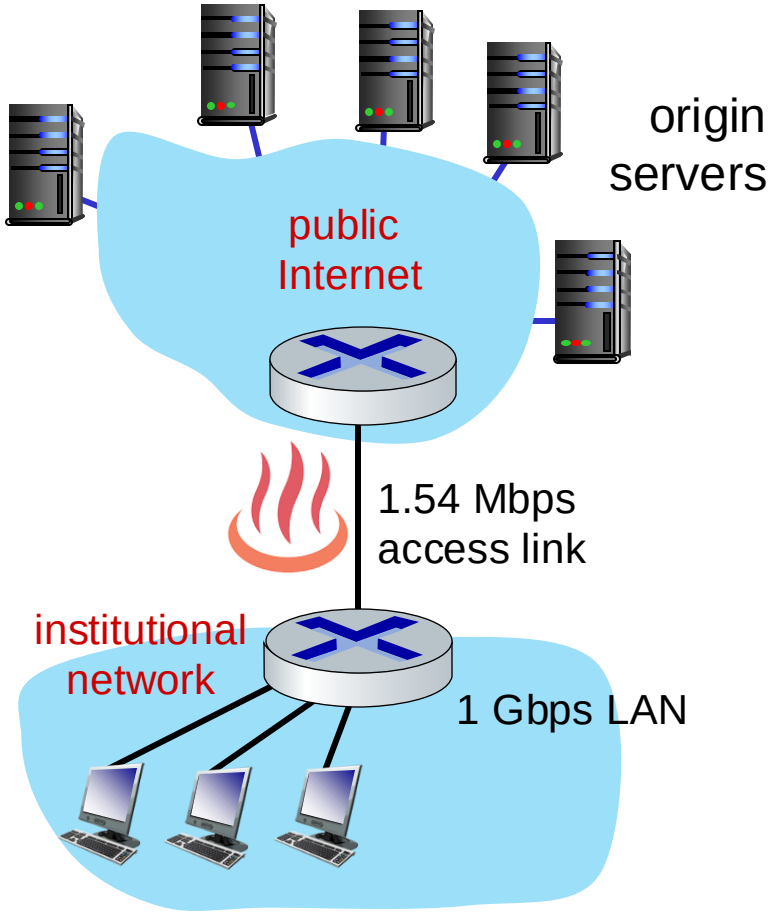
why Web caching?

- reduce response time for client request
- reduce traffic on an institution's access link
- Internet dense with caches: enables "poor" content providers to effectively deliver content

Caching example

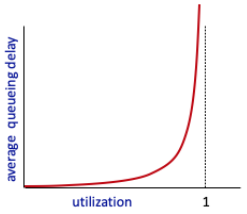
Scenario:

- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- web object size: 100K bits
- average request rate from browsers to origin servers: 15/sec
  - avg data rate to browsers: 1.50 Mbps



Performance:

- access link utilization = 0.97
- LAN utilization: 0.0015
- end-end delay = Internet delay + access link delay + LAN delay
  = 2 sec + minutes + usecs



problem: large queueing delays at high utilization!

京都大学

Option 1: buy a faster access link

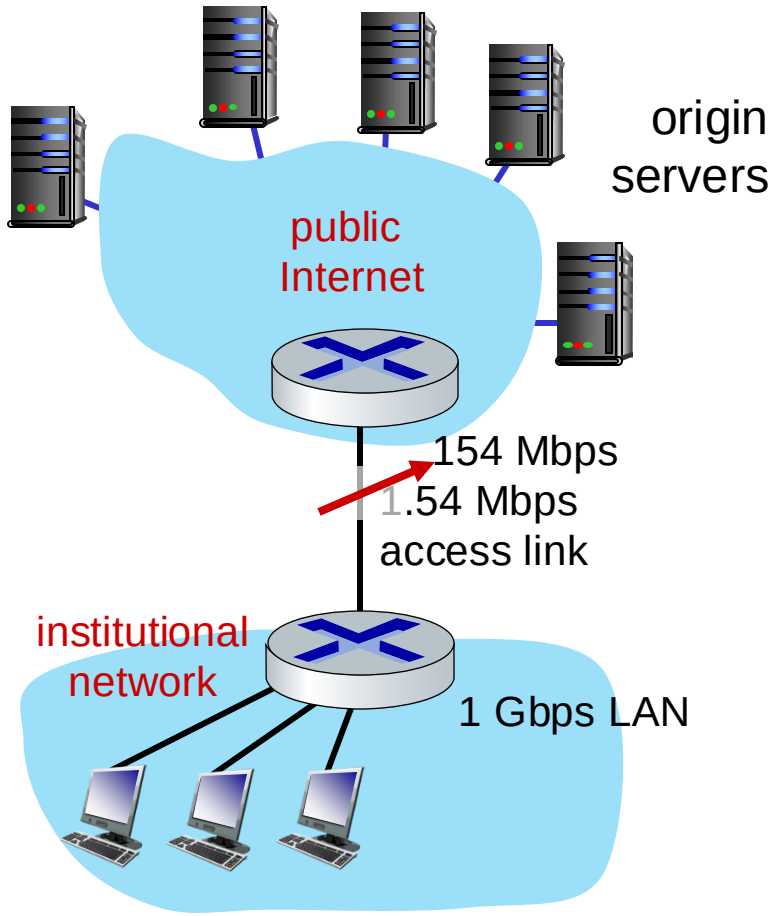
Scenario:

- access link rate: 1.54 Mbps → 154 Mbps
- RTT from institutional router to server: 2 sec
- web object size: 100K bits
- average request rate from browsers to origin servers: 15/sec
  - avg data rate to browsers: 1.50 Mbps

Performance:

- access link utilization = .97 → .0097
- LAN utilization: .0015
- end-end delay = Internet delay + access link delay + LAN delay
  = 2 sec + minutes + usecs → msecs

Cost: faster access link (expensive!)



京都大学

Option 2: install a web cache

Scenario:

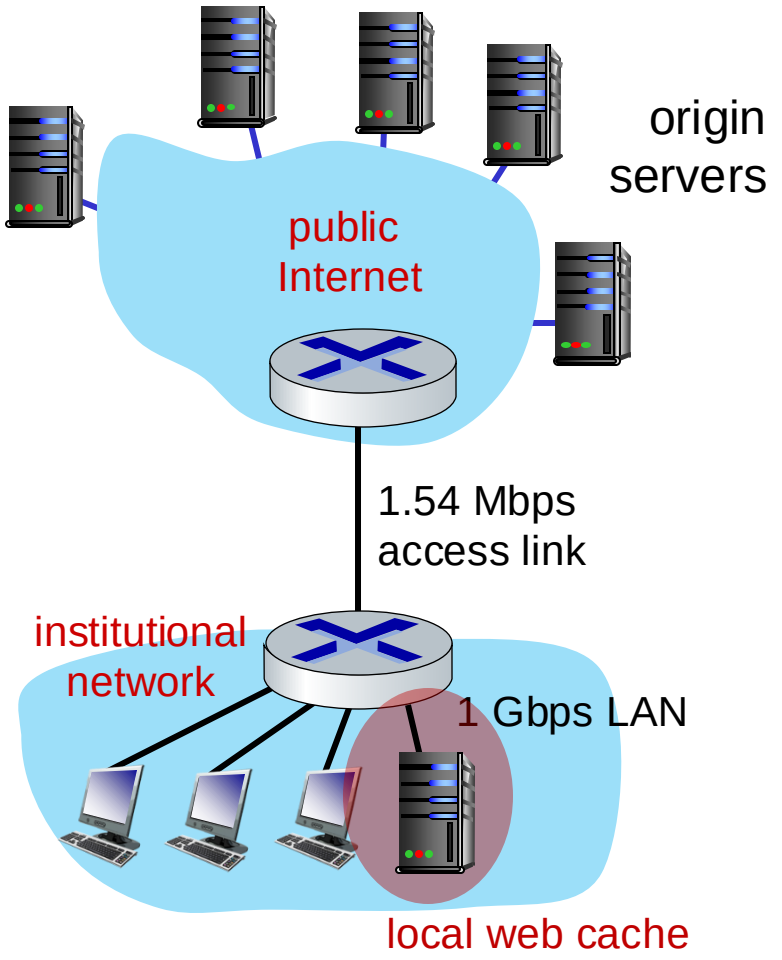
- access link rate: 1.54 Mbps
- RTT from institutional router to server: 2 sec
- web object size: 100K bits
- average request rate from browsers to origin servers: 15/sec
  - avg data rate to browsers: 1.50 Mbps

Cost: web cache (cheap!)

Performance:

- LAN utilization: .?
- access link utilization = ?
- average end-end delay = ?

How to compute link utilization, delay?



京都大学

Calculating access link utilization, end-end delay with cache:

suppose cache hit rate is 0.4:

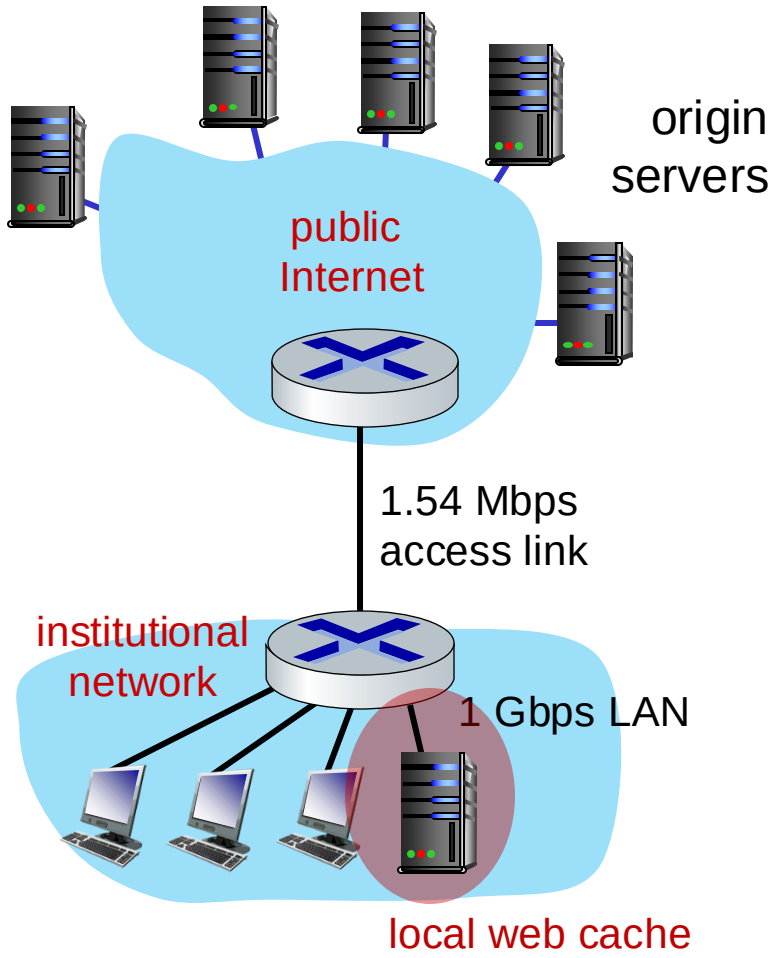
- 40% requests served by cache, with low (msec) delay
- 60% requests satisfied at origin
  - rate to browsers over access link

    = 0.6 * 1.50 Mbps = .9 Mbps
  - access link utilization = 0.9/1.54 = .58 means low (msec) queueing delay at access link
- average end-end delay:

  = 0.6 * (delay from origin servers)

  + 0.4 * (delay when satisfied at cache)

  = 0.6 (2.01) + 0.4 (~msecs) = ~ 1.2 secs



lower average end-end delay than with 154 Mbps link (and cheaper too!)

京都大学

Browser caching: Conditional GET

Goal: don't send object if browser has up-to-date cached version

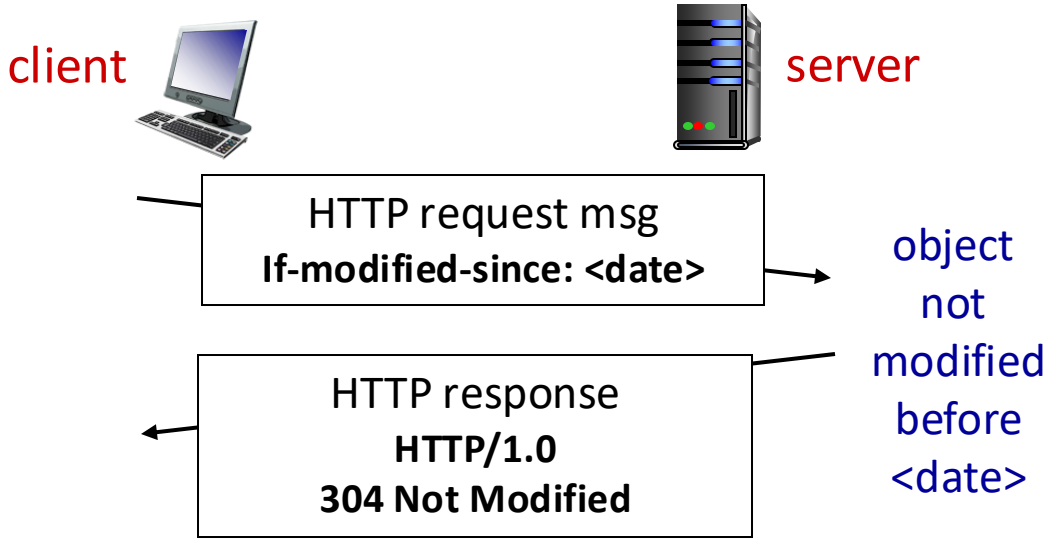
- no object transmission delay (or use of network resources)

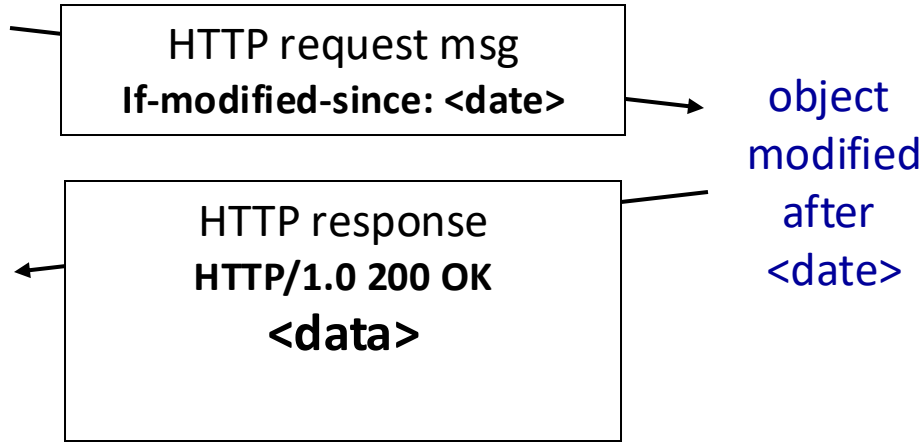
- *client:* specify date of browser-cached copy in HTTP request

  If-modified-since: <date>

- *server:* response contains no object if browser-cached copy is up-to-date:

  HTTP/1.0 304 Not Modified





Homework 1

- Homework 1 is online (PandA)
- Submission Deadline Oct 29
  - One day before the lecture on Oct 30