Information Network

Lecture 2 : Application Layer

Holger Thies

KYOTO UNIVERSITY

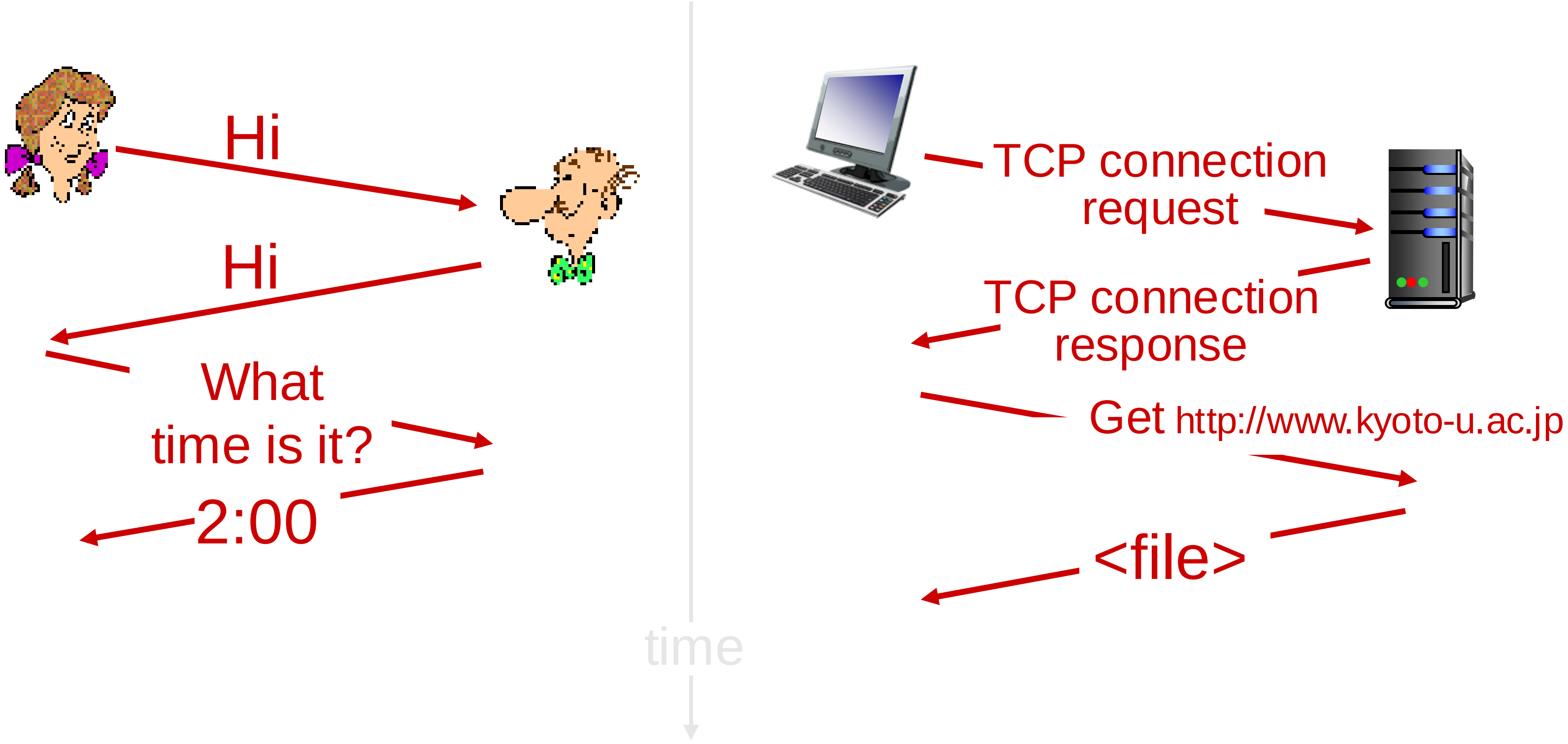
京都大学

Today's lecture

- The internet protocol stack
- Principles of the application layer
- Web and HTTP

What’ s a protocol?

a human protocol and a computer network protocol:



Protocols

human protocols:

- “what’ s the time?”
- “I have a question”
- introductions

… specific messages sent

… specific actions taken when messages received, or other events

network protocols:

- machines rather than humans
- all communication activity in Internet governed by protocols

protocols define format, order of messages sent and received among network entities, and actions taken on message transmission, receipt

Internet Protocol Stack

- To manage the complexity of the internet, a **layered approach** is used.
- Each layer in the stack serves a specific role and interacts only with the layer directly above or below it.
- This allows a modular design, easing maintenance and updating of system
- In this lecture, we focus on the **Internet Protocol Stack**, also known as the **TCP/IP model**.
- Another often-used layered model of transmitting data over the internet is the **OSI model.**
- The biggest difference between the OSI and the Internet Protocol stack is that the OSI model has seven layers instead of five.
- In this lecture we only use the Internet Protocol Stack.

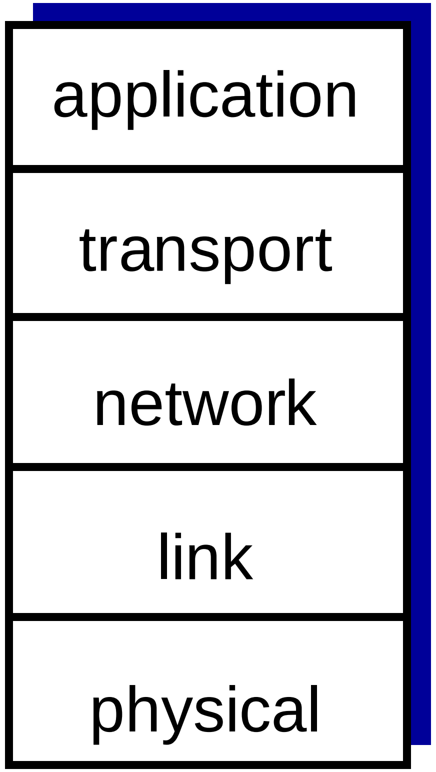
Internet protocol stack

Application Layer

- Supports network applications
- How to send and receive emails, display websites etc.
- Protocols: HTTP, SMTP, etc.

Transport Layer

- How to transfer data from one application to another
- Protocols: TCP, UDP



Internet protocol stack

Network Layer

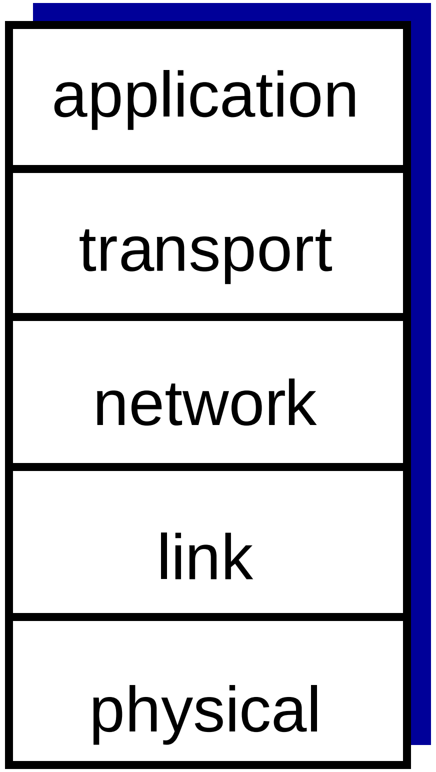
- How to transfer data through the network
- IP, routing protocols

Link Layer

- How to transfer data between neighboring network elements
- Protocols: Ethernet, Wifi, …

Physical Layer

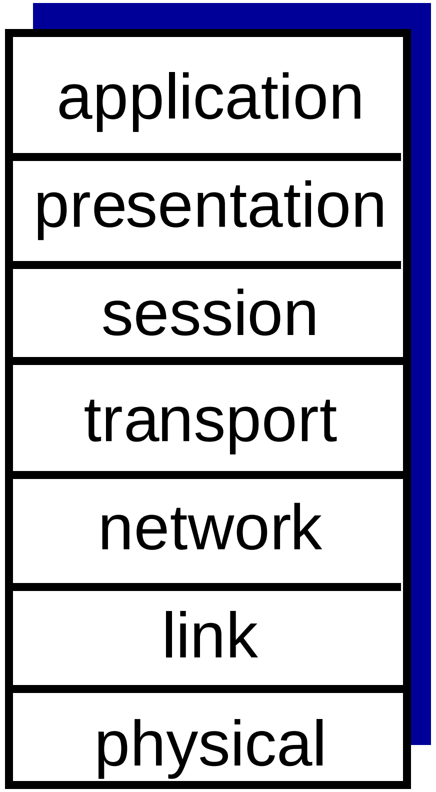
- Contains the physical devices that connect computers, how to send bits (ones and zeros)
- Specifications for cables, connectors, how to send signals, etc.



ISO/OSI reference model

Two layers not found in Internet protocol stack!

- *presentation:* allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *session:* synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers!
  - these services, *if needed,* must be implemented in application
  - needed?



The seven layer OSI/ISO reference model

Today's lecture

- The internet protocol stack
- Principles of the application layer
- Web and HTTP

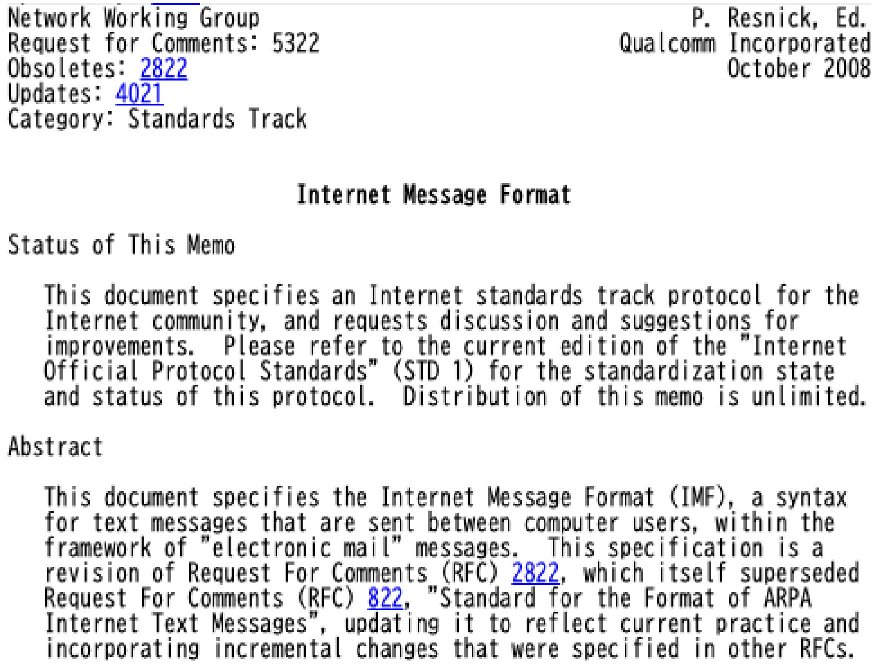
Application Layer

- The application layer is the highest layer in the internet protocol stack.
- It provides services to the user.
    - It is the only layer that provides services directly to the user.
- Protocols on the application layer do not provide services to other layers, but only receive services from the transport layer.
- Applications can communicate directly to each other using the transport layer services.
- Applications however must define a protocol to agree on the rules of communication.

Standardization in Internet Technologies

Several protocols have been standardized by standards-setting bodies for the Internet, most importantly the IETF (Internet Engineering Task Force).

- Enables different developers to create systems and products that interoperate seamlessly.
- Creates robust, reliable connections by standardizing protocols for data transmission.
- Guarantees that websites, applications, and devices work the same way across platforms.
- Establishes standardized security protocols to protect data and prevent unauthorized access.
- Standards for the Internet are documented in formal documents called RFCs (Request for Comments).



```
Network Working Group                                    P. Resnick, Ed.
Request for Comments: 5322                         Qualcomm Incorporated
Obsoletes: 2822                                             October 2008
Updates: 4021
Category: Standards Track


                         Internet Message Format

Status of This Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements.  Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol.  Distribution of this memo is unlimited.

Abstract

   This document specifies the Internet Message Format (IMF), a syntax
   for text messages that are sent between computer users, within the
   framework of "electronic mail" messages.  This specification is a
   revision of Request For Comments (RFC) 2822, which itself superseded
   Request For Comments (RFC) 822, "Standard for the Format of ARPA
   Internet Text Messages", updating it to reflect current practice and
   incorporating incremental changes that were specified in other RFCs.
```

Standard Application Layer Protocols

- HTTP/HTTPS (Hypertext Transfer Protocol / Secure)
  - Transfers web pages.
- SMTP (Simple Mail Transfer Protocol)
  - Sends and forwards email.
- POP/IMAP (Post Office Protocol/Internet Message Access Protocol)
  - Retrieves email from servers.
- FTP (File Transfer Protocol)
  - Transfers files between systems.
- DNS (Domain Name System)
  - Resolves domain names to IP addresses.
- There are also many applications that use their own non-standardized protocols (WhatsApp, Line, Dropbox, etc.).

Some network applications

- e-mail
- web
- text messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video (YouTube, Netflix)

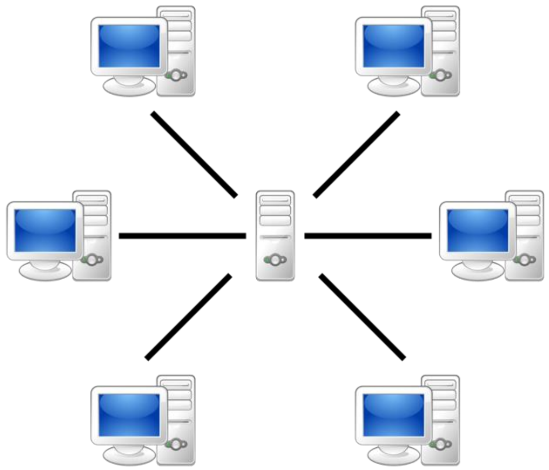
- voice over IP
- real-time video conferencing
- social networking
- search

IP Addresses

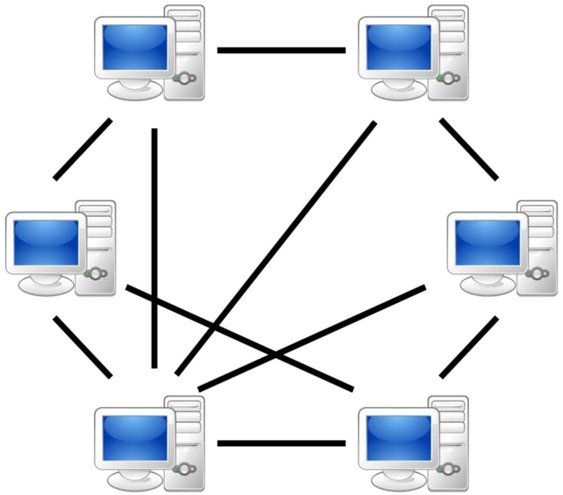
- To send a message from one device to another, each device needs a unique address.
- The internet uses IP addresses to identify devices and route data correctly.
- Each device has unique 32-bit (IPv4) or 128-bit (IPv6) IP address
  - 32-bit address, expressed as four decimal numbers separated by dots (e.g., 192.168.0.1).
  - 128-bit address, expressed in hexadecimal and separated by colons (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334).
- IP addresses are not permanently tied to a device, i.e., the IP address can change over time.
  - Some servers etc. have static IP addresses but most devices use dynamic IP addresses which are assigned by the network when they are connected and change over time.
- In later lectures, we will learn how IP addresses are assigned and how they are structured to enable efficient routing across global networks.

Application architectures

Two main architectures for the application layer have been developed:

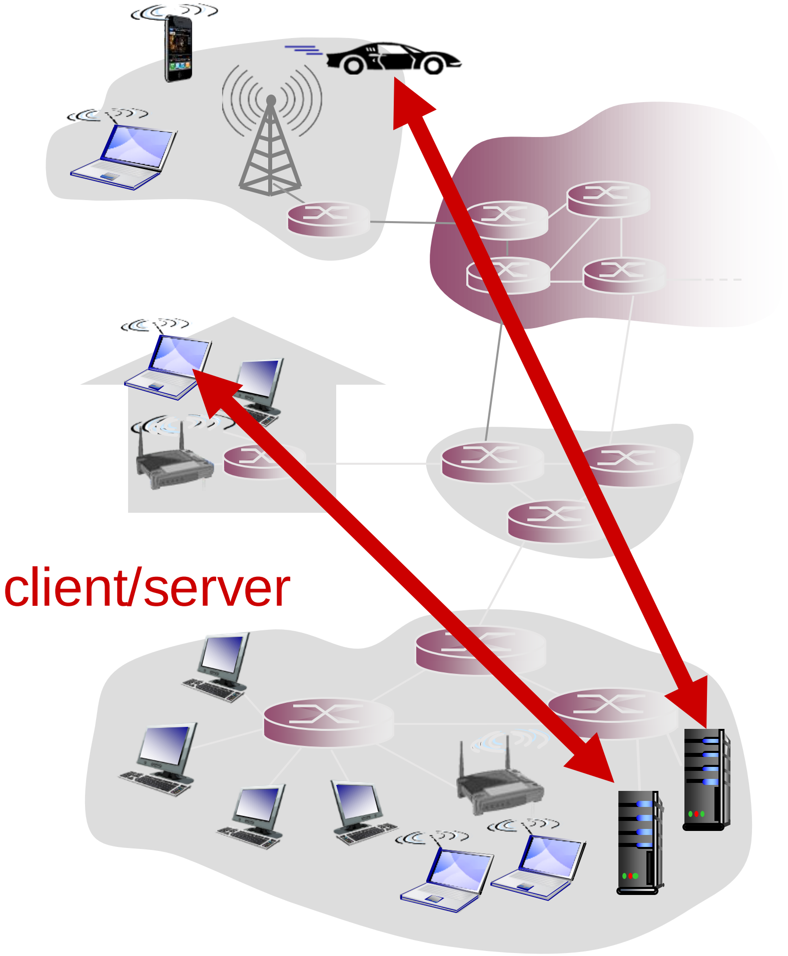


Client-server architecture



Peer-to-Peer (P2P) architecture

Client-server architecture



server:

- always-on host
- permanent IP address
- often in data centers for scaling

clients:

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

京都大学

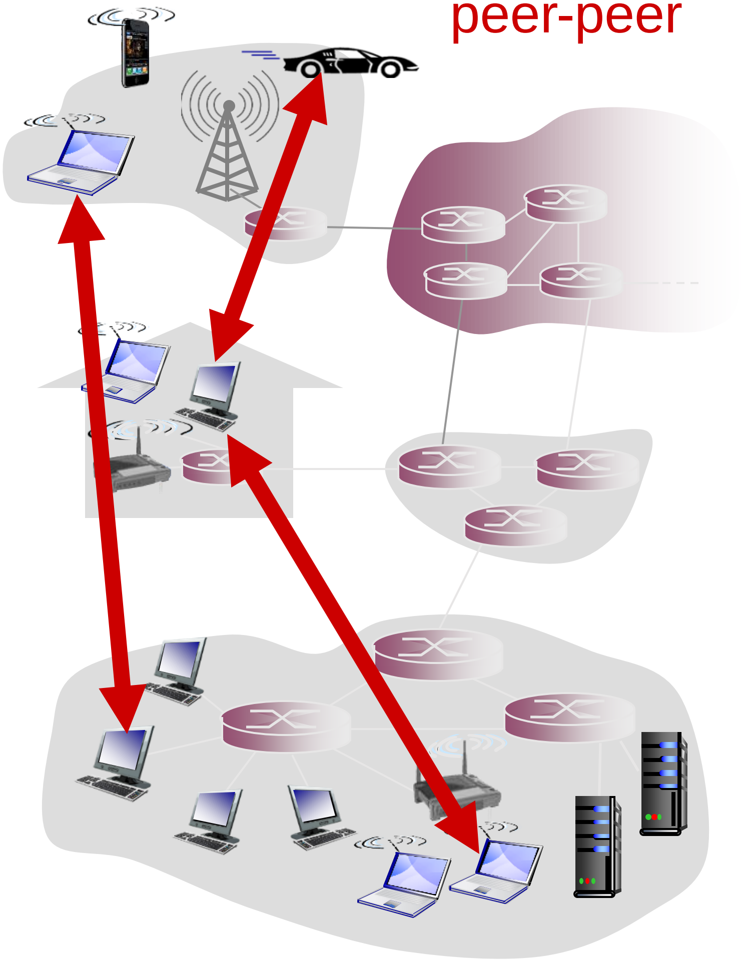
Client-server architecture

- Tasks are divided between servers that provide a resource/service and clients that request the resource/service.
- Usually only one or a few servers, but many clients.
- Servers must run all the time and have a fixed IP address where they can be reached.
- Servers wait for clients to connect to them.
- Examples: Web Browsing, Email, File transfer, etc.

P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, as well as new service demands
- peers are intermittently connected and change IP addresses
  - complex management
  - security challenges

peer-peer



P2P architecture

- No need for a server to be running all the time and waiting for clients to connect.
- No centralized authority, peers communicate directly and share resources with each other.
- Users can both provide and receive services at the same time.
- Capacity increases automatically with more peers joining.
- No single point of failure.
- However, requires more complex protocols to manage communication and for security.
- Examples: File-sharing (BitTorrent), Bitcoin, some messaging applications, etc.

Hybrid architecture

- Many applications do not follow one paradigm strictly but combine elements of both Client-Server and Peer-to-Peer architectures.
- Some functionality relies on a central server (e.g., authentication, discovery of peers, etc.) while others are decentralized (e.g. sharing of files, etc.)
- Examples: Skype uses central servers for login and user management, while using P2P connections for video calls for better performance.

Processes communicating

On the application layer, applications exchange messages directly with each other.

→All the communication is between applications.

More precisely, application <u>processes</u> communicate.

<u>Process: </u>An instance of a particular application.

Processes communicating

process: program running within a host

- within same host, two processes communicate using inter-process communication (defined by OS)
- processes in different hosts communicate by exchanging messages

clients, servers

client process: process that initiates communication

server process: process that waits to be contacted

- applications with P2P architectures have both client processes & server processes

京都大学

Client and Server process

In the context of a communication session between a pair of processes, the process that initiates the communication is labeled the **client**. The process that waits to be contacted to begin the session is the **server**.

Examples

- Web: browser process initializes contact with a web server process
- P2P file sharing: Peer A (client) asks Peer B (server) to send a specific file

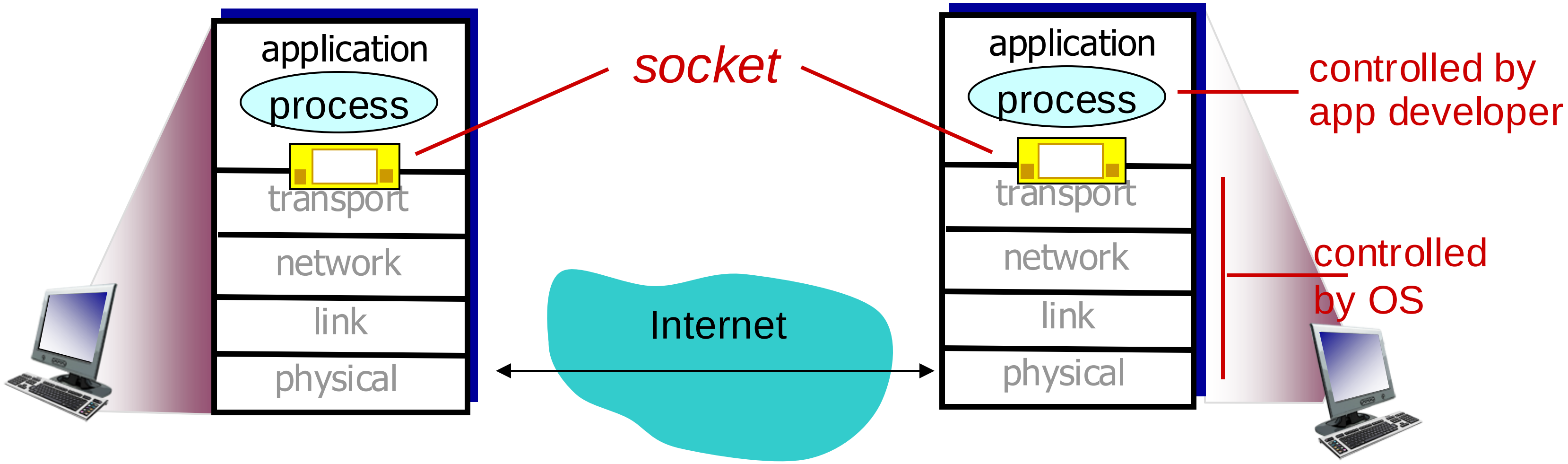
Addressing processes

- to send and receive messages, process must have *identifier*
- host device has unique 32-bit (IPv4) or 128 bit (IPv6) IP address
- *Q:* does IP address of host on which process runs suffice for identifying the process?
- *A:* no, *many* processes can be running on same host

- *identifier* includes both IP address and port numbers associated with process on host.
- example port numbers:
  - HTTP server: 80
  - mail server: 25
- to send HTTP message to web server:
  - IP address: 128.119.245.12
  - port number: 80

Sockets

- process sends/receives messages to/from its socket
- The port number is the number the identifies the socket.
- socket analogous to door
  - sending process pushes message out of door
  - sending process relies on transport infrastructure on other side of door to deliver message to socket at receiving process



App-layer protocol defines

- types of messages exchanged,
  - e.g., request, response
- message syntax:
  - what fields in messages & how fields are delineated
- message semantics
  - meaning of information in fields
- rules for when and how processes send & respond to messages

京都大学

Transport services

- To send messages through the network the application process needs to use services provided by the transport layer.
- Many networks provide more than one transport-layer protocol.
- Application developer must choose one of the available protocols.
- When discussing the application layer, it is important to understand which protocols are available and which services they provide.
- However, we do not have to understand how the services are implemented yet.
  - This will be covered in detail in later lectures.

What transport service does an app need?

reliable data transfer

- some apps (e.g., file transfer, web transactions) require 100% reliable data transfer
- other apps (e.g., audio) can tolerate some loss

timing

- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

throughput

- some apps (e.g., multimedia) require minimum amount of throughput to be "effective"
- other apps ("elastic apps") make use of whatever throughput they get

security

- encryption, data integrity, …

京都大学

Transport service requirements: common apps

application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100’s msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100’s msec
text messaging	no loss	elastic	yes and no

Internet transport protocols services

TCP (Transmission Control Protocol):

- *reliable transport* between sending and receiving process
- *flow control:* sender won't overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *connection-oriented:* setup required between client and server processes
- *does not provide:* timing, minimum throughput guarantee, security

UDP (User Datagram Protocol):

- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup

Internet apps: application, transport protocols

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

Application-Layer protocols

- We learned that network processes communicate by sending messages into sockets.
- How are these messages structured?
- An **application-layer protocol** defines how application processes running on different end-systems pass messages to each other
  - Types of messages exchanged
  - Syntax of various message types
  - Semantics of the fields
  - Rules to determine when and how a process sends a messages and responds to a message
- We are now ready to study some of the main internet applications.

Today’s lecture

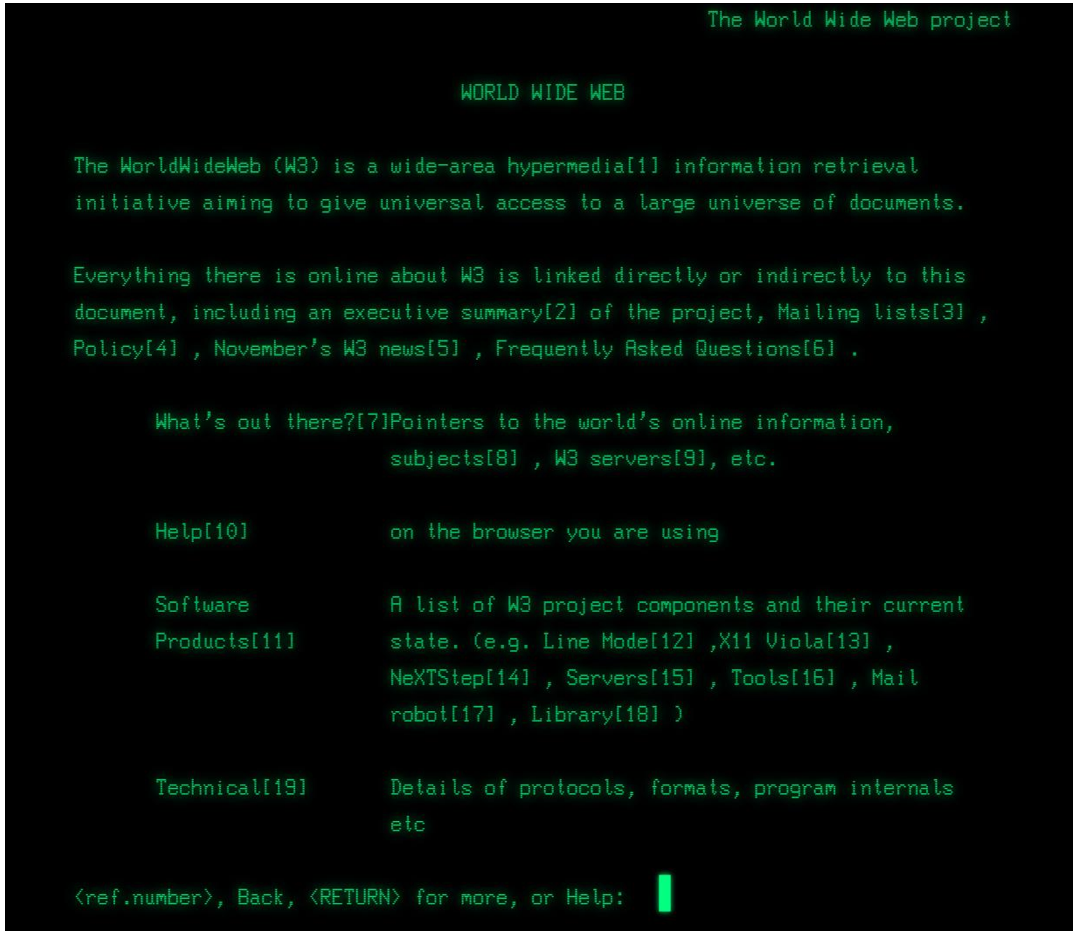
- The internet protocol stack
- Principles of the application layer
- Web and HTTP

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.
- More precisely, web resources are accessed using the HTTP or HTTPs application-layer Internet protocols which run on top of the Internet's transport protocols.

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.



```
                                                    The World Wide Web project

                              WORLD WIDE WEB

The WorldWideWeb (W3) is a wide-area hypermedia[1] information retrieval
initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this
document, including an executive summary[2] of the project, Mailing lists[3] ,
Policy[4] , November's W3 news[5] , Frequently Asked Questions[6] .

      What's out there?[7]Pointers to the world's online information,
                          subjects[8] , W3 servers[9], etc.

      Help[10]            on the browser you are using

      Software            A list of W3 project components and their current
      Products[11]        state. (e.g. Line Mode[12] ,X11 Viola[13] ,
                          NeXTStep[14] , Servers[15] , Tools[16] , Mail
                          robot[17] , Library[18] )

      Technical[19]       Details of protocols, formats, program internals
                          etc

<ref.number>, Back, <RETURN> for more, or Help:
```

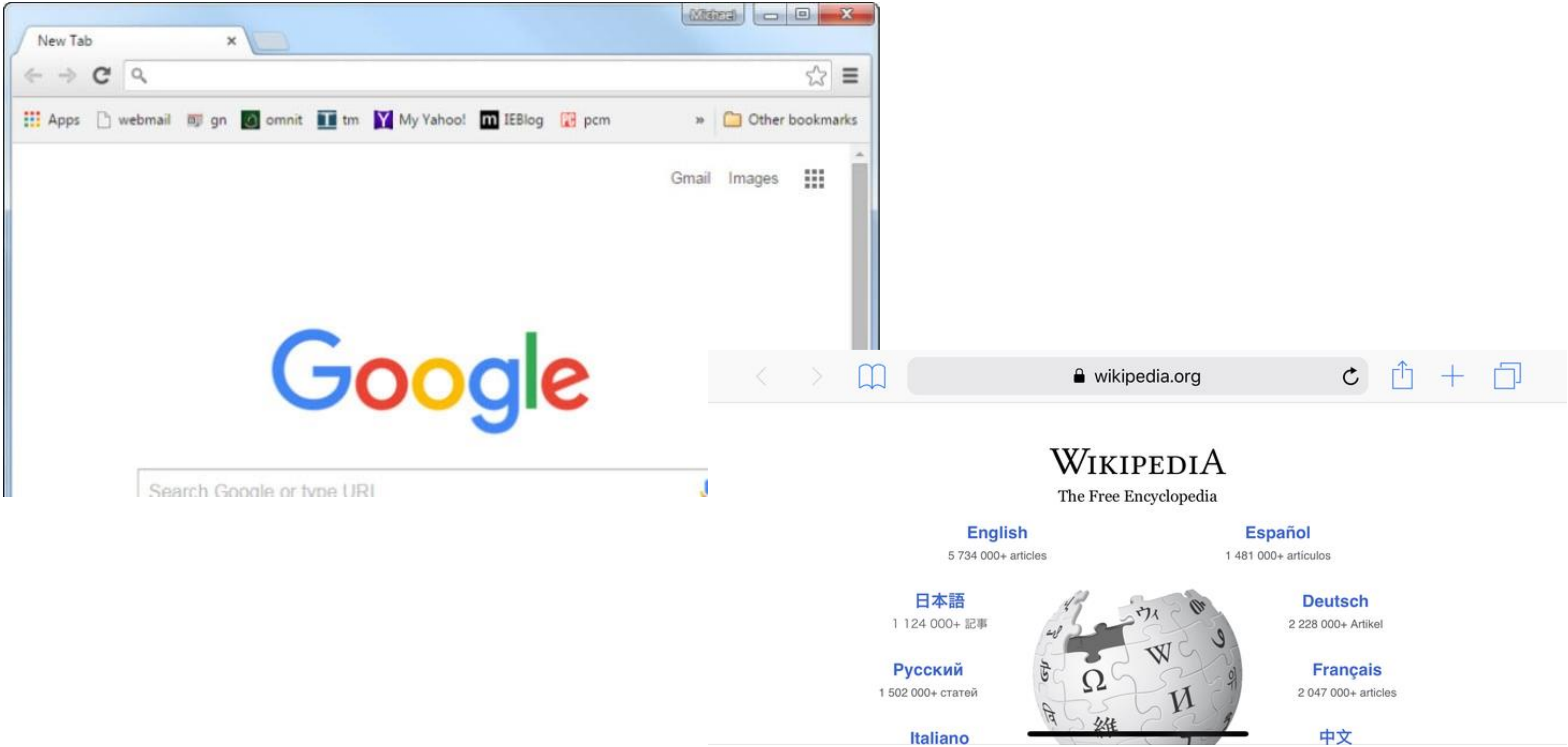
Main components of the world wide web

- 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 Browser

- 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



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

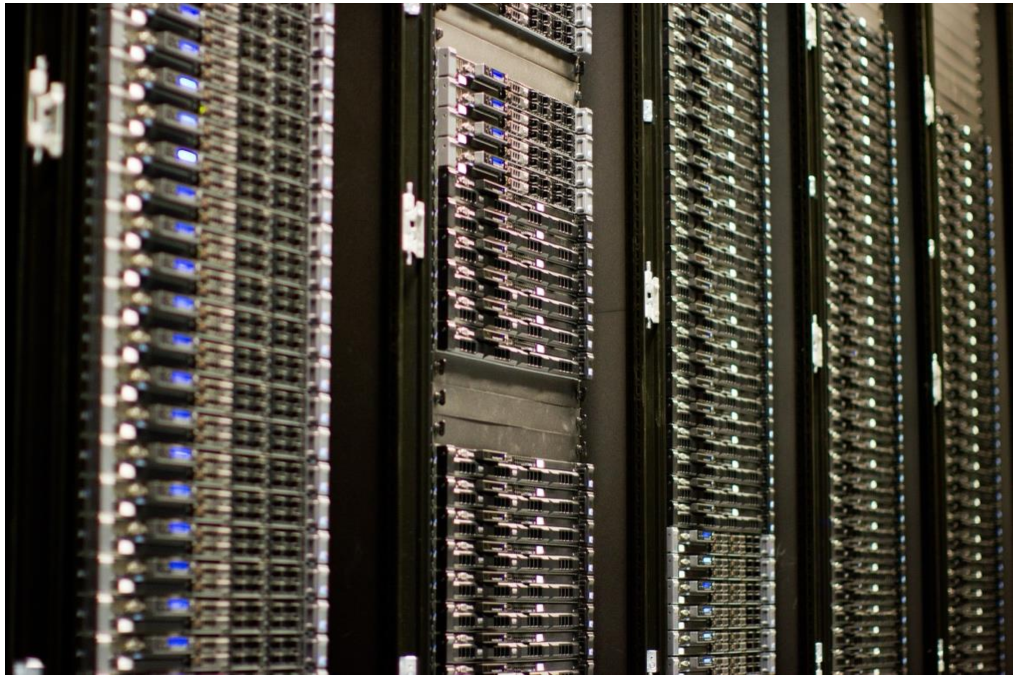
THE APACHE® SOFTWARE FOUNDATION

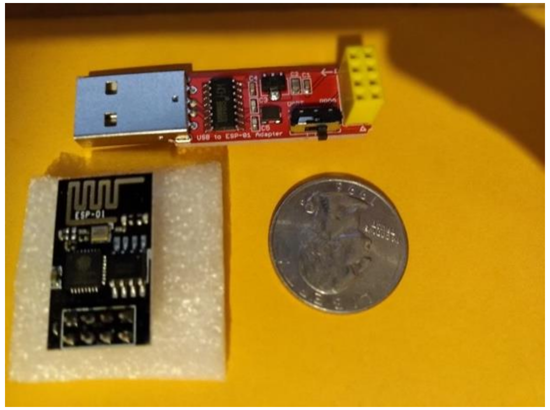
NGINX®

Microsoft IIS

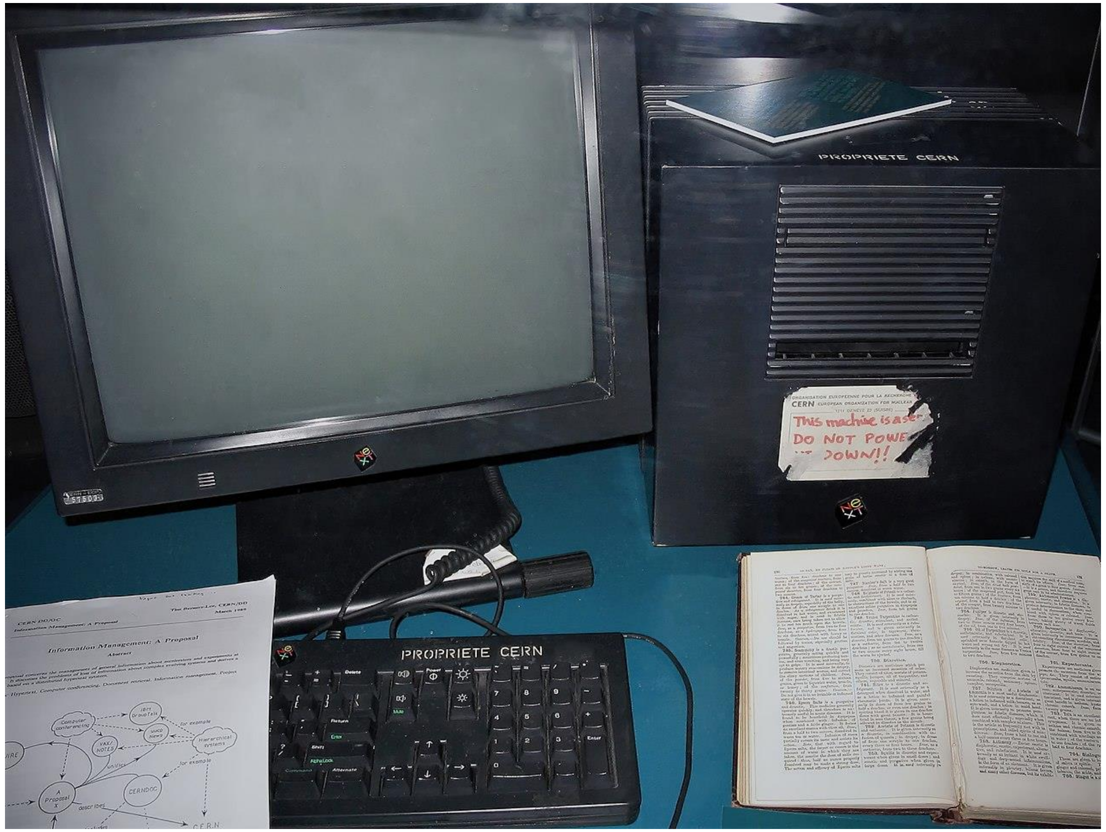
- A web server stores, processes and delivers resources (HTML files, images, scripts, videos, etc.) to users over the internet.
- It receives requests from web browsers and provides the appropriate responses.
- Usually larger, more powerful computer.
- Can be static (only provide fixed content) or dynamic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

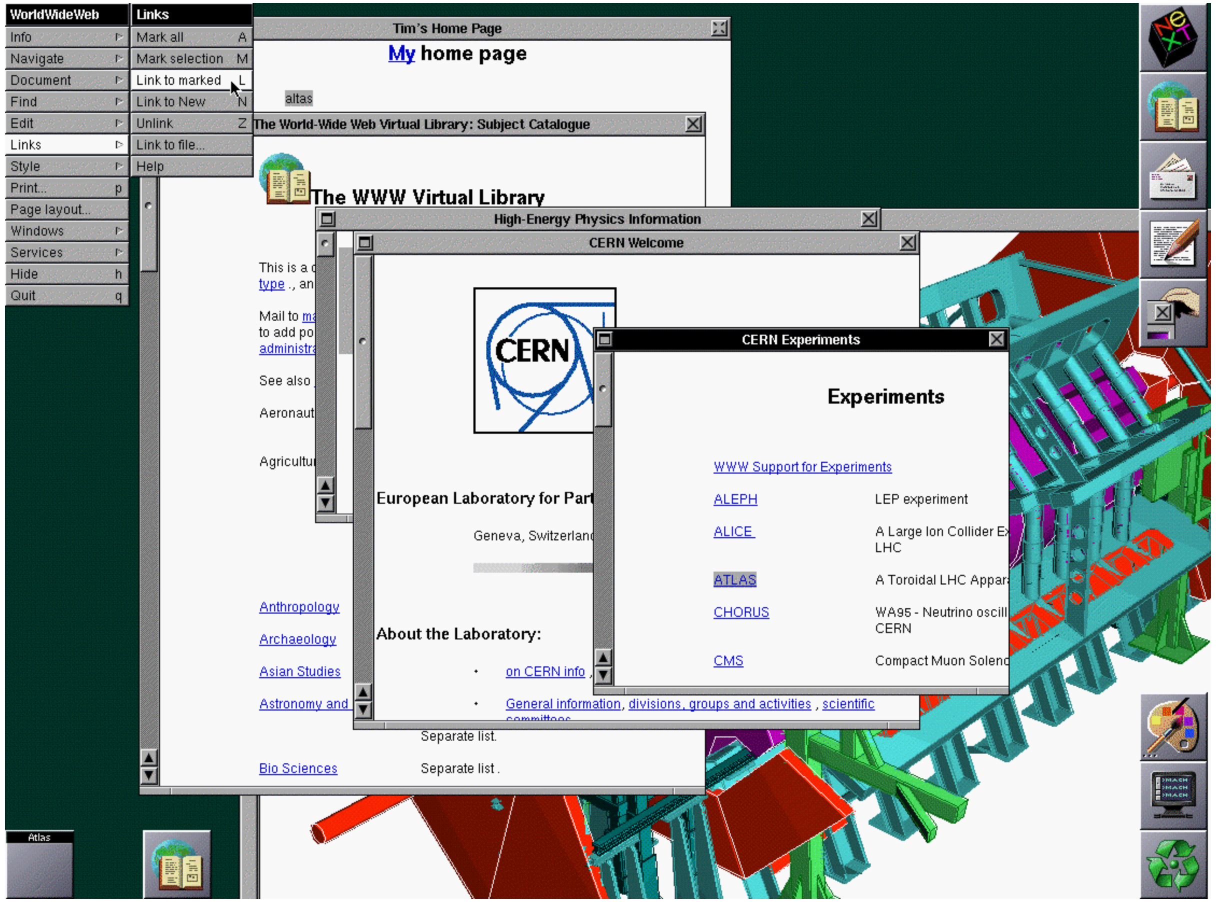
Web Server





First web server and browser


PROPRIETE CERN
This machine is a server. DO NOT POWER IT DOWN!!
PROPRIETE CERN
Information Management: A Proposal



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 ma
to add po
administra
See also
Aeronaut
Agricultur
Anthropology
Archaeology
Asian Studies
Astronomy and
Bio Sciences Separate list .

High-Energy Physics Information

CERN Welcome

CERN

European Laboratory for Part

Geneva, Switzerlanc

About the Laboratory:

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

Separate list.

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 oscil CERN
CMS	Compact Muon Solen

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.

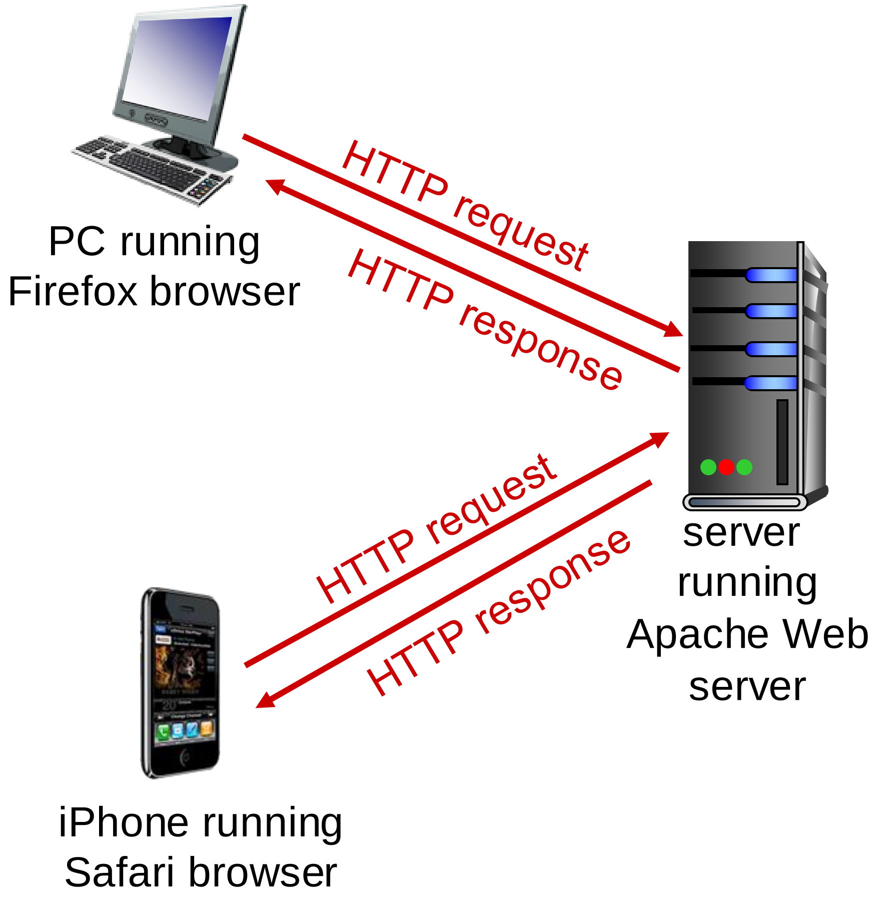
Hyper Text Transfer Protocol (HTTP)

- 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


PC running Firefox browser
HTTP request
HTTP response
server running Apache Web server
HTTP request
HTTP response
iPhone running Safari browser

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>
```

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

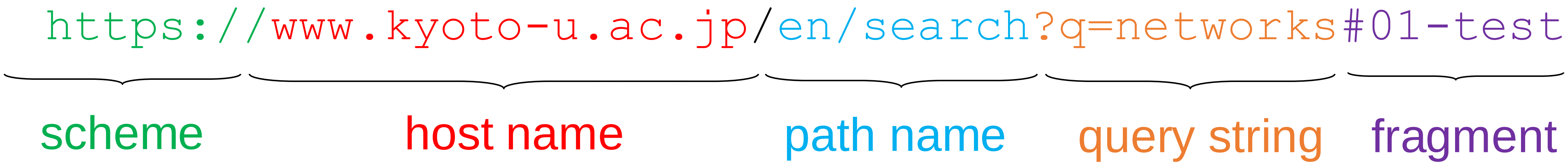
- Most URLs follow a standardized format of three main parts.

$$\underbrace{\texttt{http://}}_{\text{scheme}}\underbrace{\texttt{www.someschool.edu}}_{\text{host name}}\underbrace{\texttt{/someDept/pic.gif}}_{\text{path name}}$$

- 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:



https://www.kyoto-u.ac.jp/en/search?q=networks#01-test

- 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

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?

Messages

- 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.

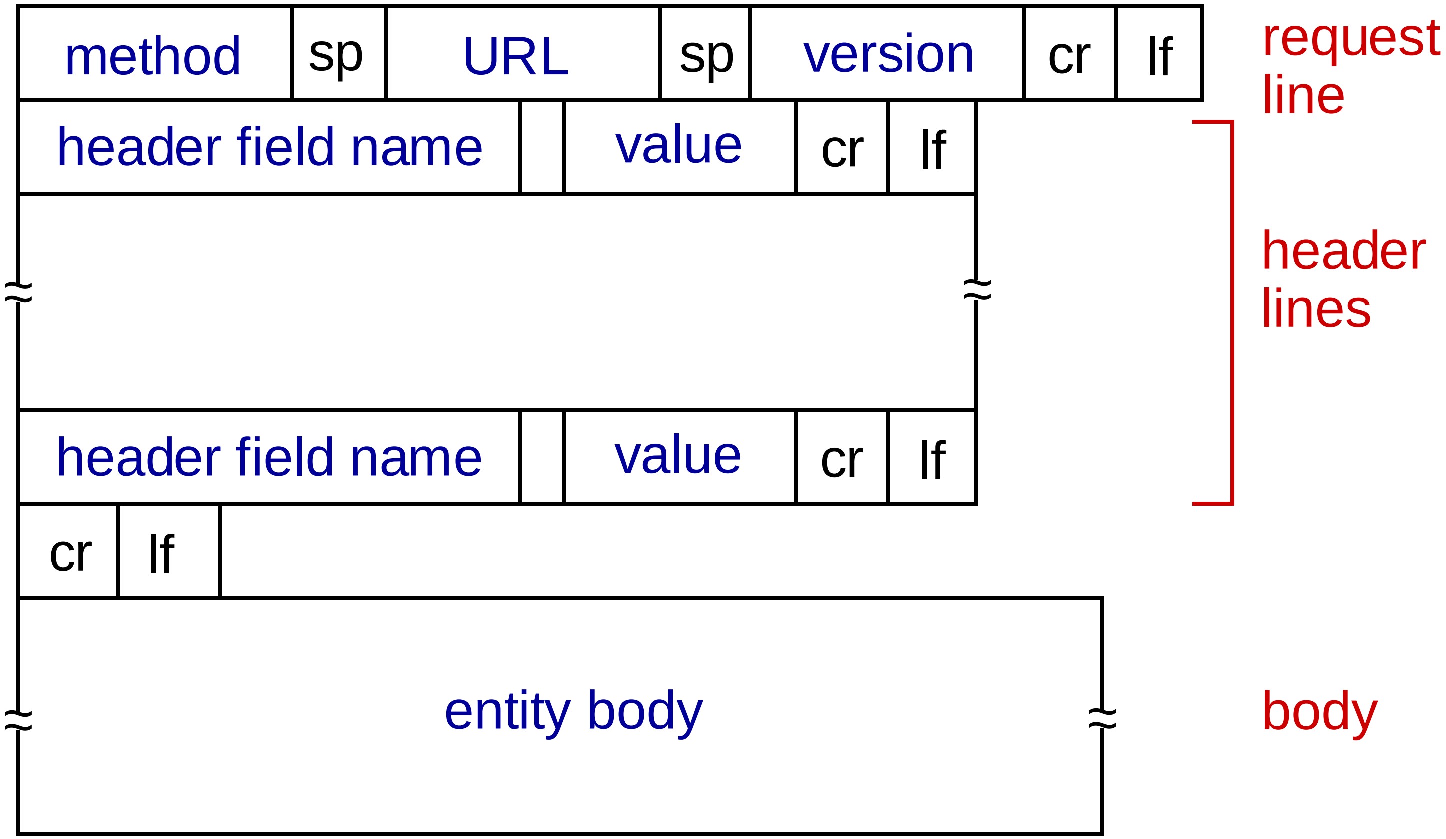
HTTP message format

- 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



method sp URL sp version cr lf

header field name value cr lf

header field name value cr lf

cr lf

entity body

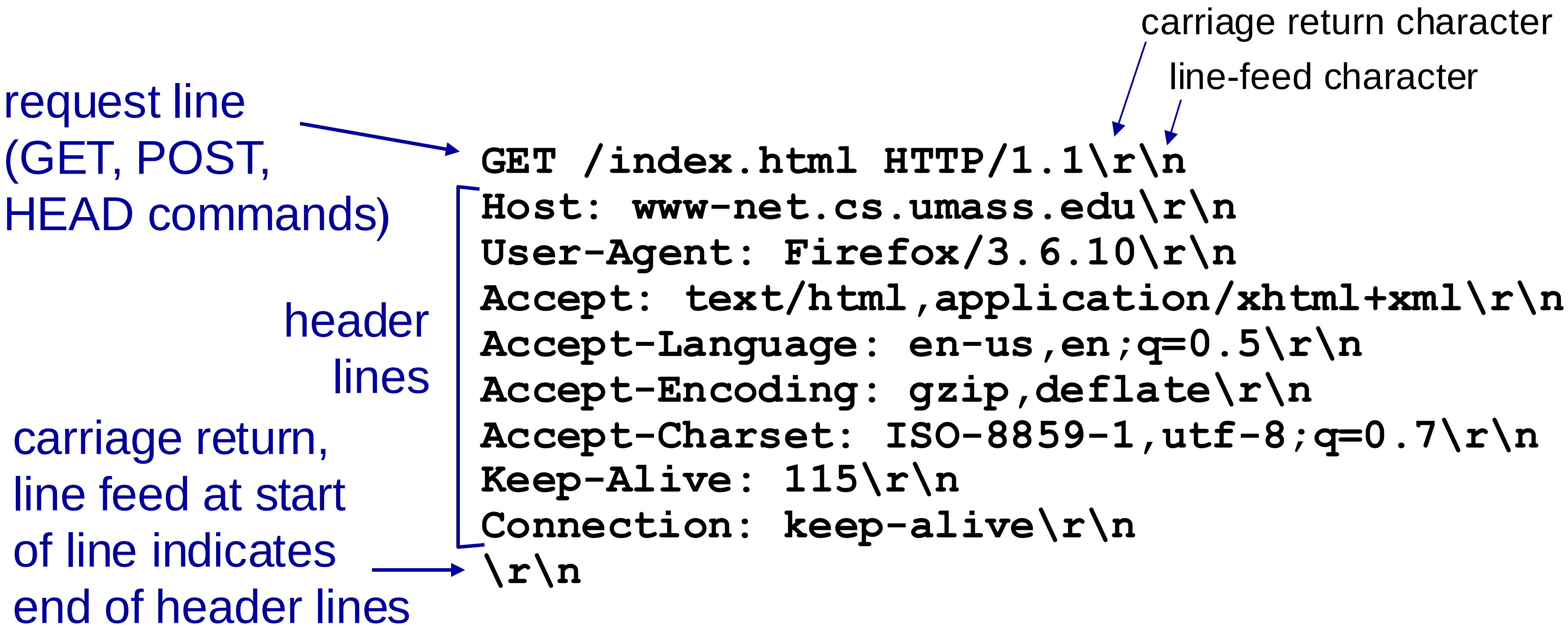
request line

header lines

body

HTTP request message

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



carriage return character

line-feed character

request line (GET, POST, HEAD commands)

header lines

carriage return, line feed at start of line indicates end of header lines

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

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
```

京都大学

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`**

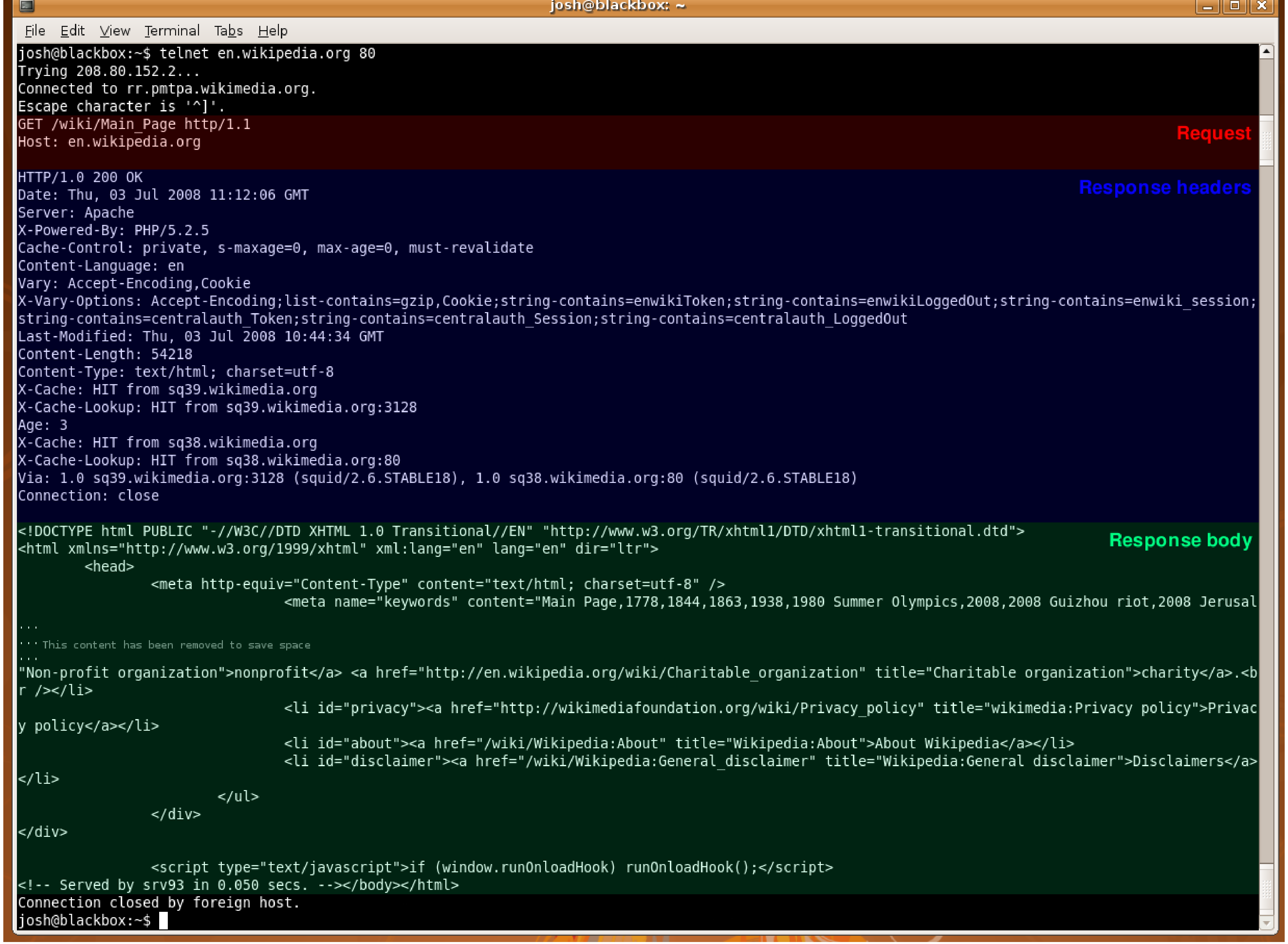
- Not allowed to access the document

`404 Not Found`

- requested document not found on this server

`500 Internal server error`

京都大学



```
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:~$
```

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.