Web

Master 1 IFI

Andrea G. B. Tettamanzi

Université de Nice Sophia Antipolis

Département Informatique

andrea.tettamanzi@unice.fr

Unit 1

The Web's Architecture and Protocols

Agenda

- Course Objectives and Structure
- What is the Web?
- Hypertexts
- The Hypertext Transfer Protocol
- The Programmable Web
- The Architecture of a Web Application

Course Objectives and Structure

- The Web, originally intended to be an open document-sharing platform, has evolved into a distributed platform for the deployment and execution of applications, to the point that it can now be viewed as a sort of global operating system (the programmable web).
- It has also become a "social machine" and a technological infrastructure for collective intelligence, which constitutes an interesting and complex subject of study.
- The objective of this course is to provide a comprehensive introduction to the architecture, standards, languages, and models that allow this huge distributed system to function, without forgetting its societal impact.

Course Structure

The course is organized in 8 units:

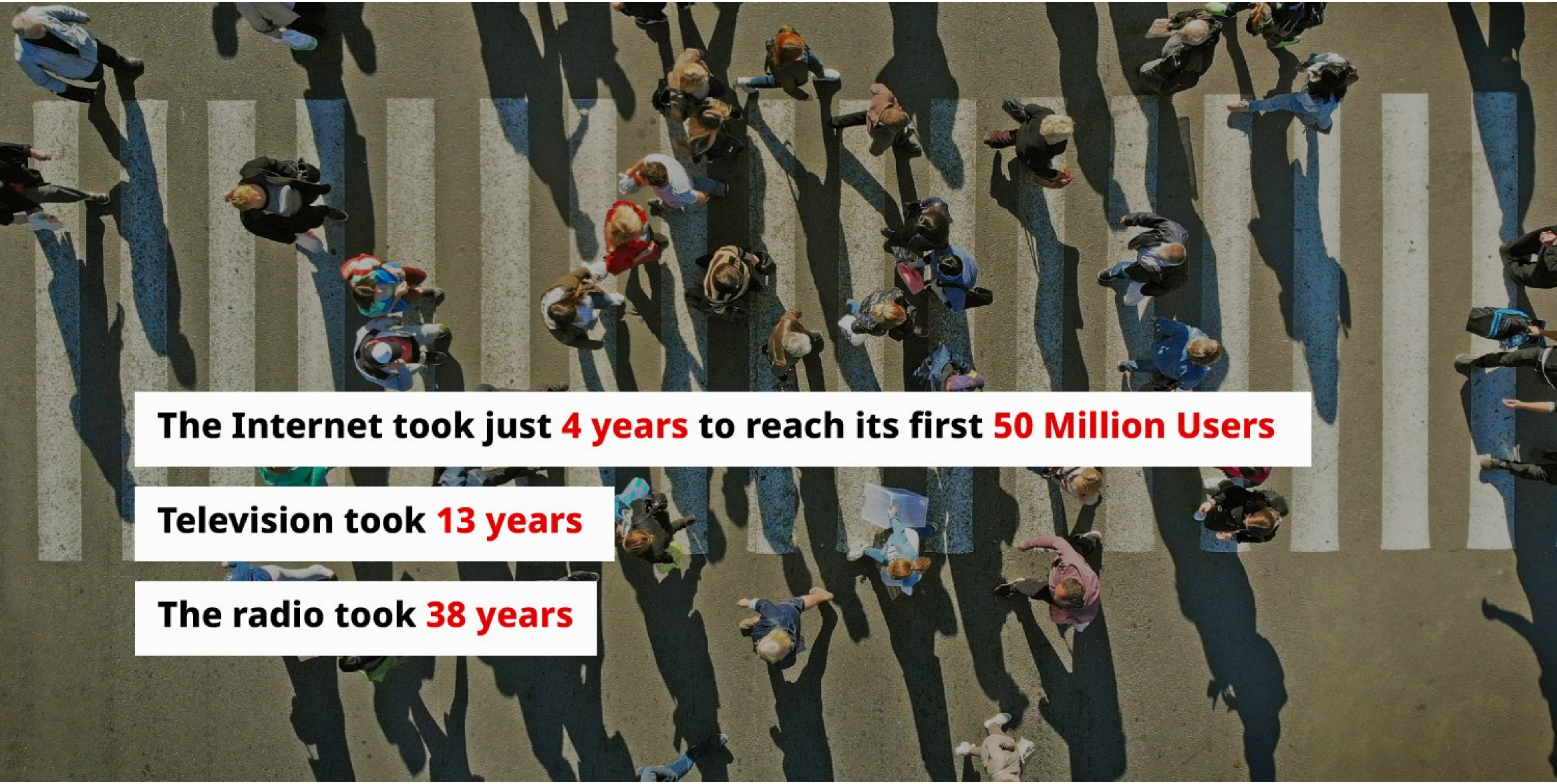
1) The Web's Architecture and Protocols (this unit)
2) A refresher on HTML, CSS, and the Document Object Model
3) The Common Gateway Interface and Server-side Programming
4) Client-side Programming (JavaScript and the HTML5 API)
5) Persistence, AJAX, and REST
6) An introduction to Web Services, UDDI, and SOAP
7) Ergonomy
8) Web Science: The Web as an object of study.

Material

- Web page :
  - http://www.i3s.unice.fr/~tettaman/Classes/WebL3MIAGE/
- Official Standards :
  - http://www.w3.org/standards/webdesign/htmlcss

The Web and The Internet

- WWW = World-Wide Web
  - aka “The Web”
- A public **hypertext** system based on the **Internet**
- Created at CERN by **Tim Berners-Lee** in 1990
- Original idea: create a distributed hypertext system on the Internet to allow collaborators to share information within CERN
- On April 30, 1993, CERN puts in the public domain all the technologies developed around the WWW
- NCSA Mosaic: first « browser »
- The Web could not exist without standards
- To understand the Web is to understand its standards


The Internet took just **4 years** to reach its first **50 Million Users**

Television took **13 years**

The radio took **38 years**

Source: The Web Foundation, 2019

What's a Hypertext?

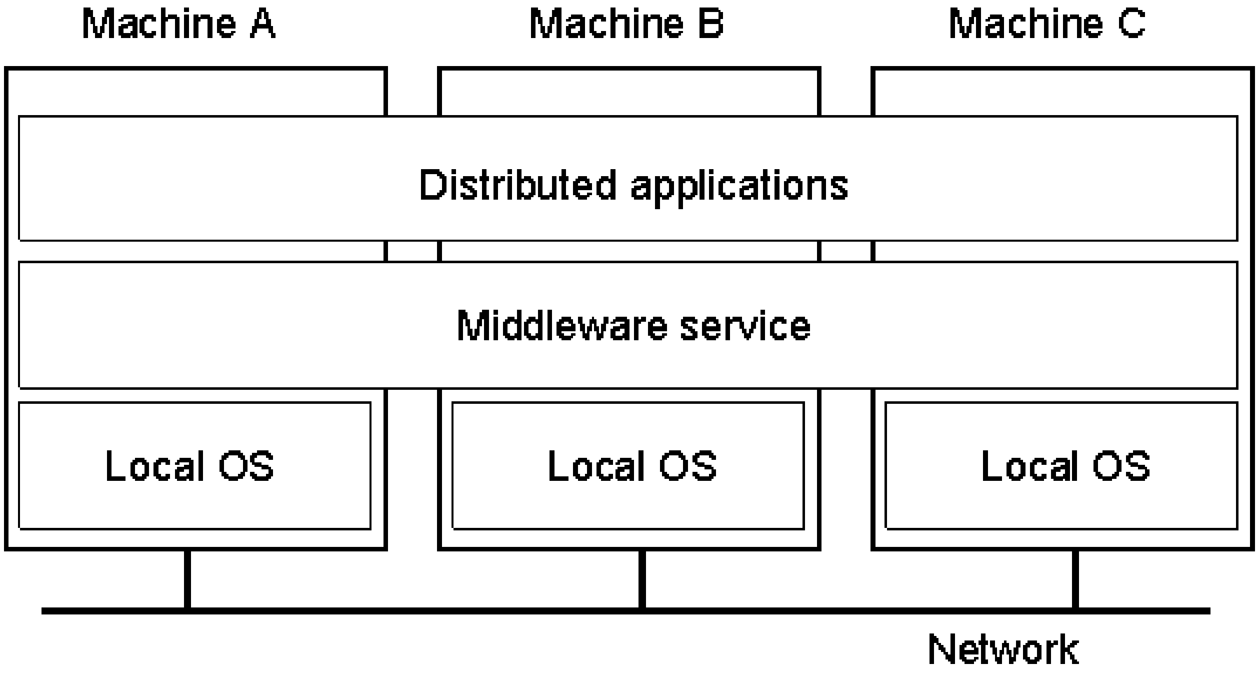
- Hypertext = a text equipped with links allowing to jump immediately from one point to the other
- In 1945 the American engineer and scientific consultant **Vannevar Bush** publishes on the Atlantic Monthly the article "*As We May Think*"
- Memex = memory extension
  - A photo-electro-mechanical device
  - Create and follow links between documents on microfilm
- In the 1960s:
  - Doug Engelbart creates the prototype NLS (oN-Line System), which makes it possible to edit and browse a hypertext
  - Ted Nelson invents the term "hypertext"

Hypertext System

- A set of nodes connected by hyperlinks,
  making it possible to jump automatically from one node to another
- Node = a minimal unit of information, a part of the text
- The links between these nodes are managed by the computer
  - Associative access to information
  - Non-linear, personalized traversal
- When nodes are also audio-visual, one can speak of a hypermedia system
- **Ted Nelson** : « *Let me introduce the word 'hypertext' to mean a body of written or pictorial material interconnected in such a complex way that it could not conveniently be presented or represented on paper* » (Proc. 20th ACM Nat'l Conf, 1965).

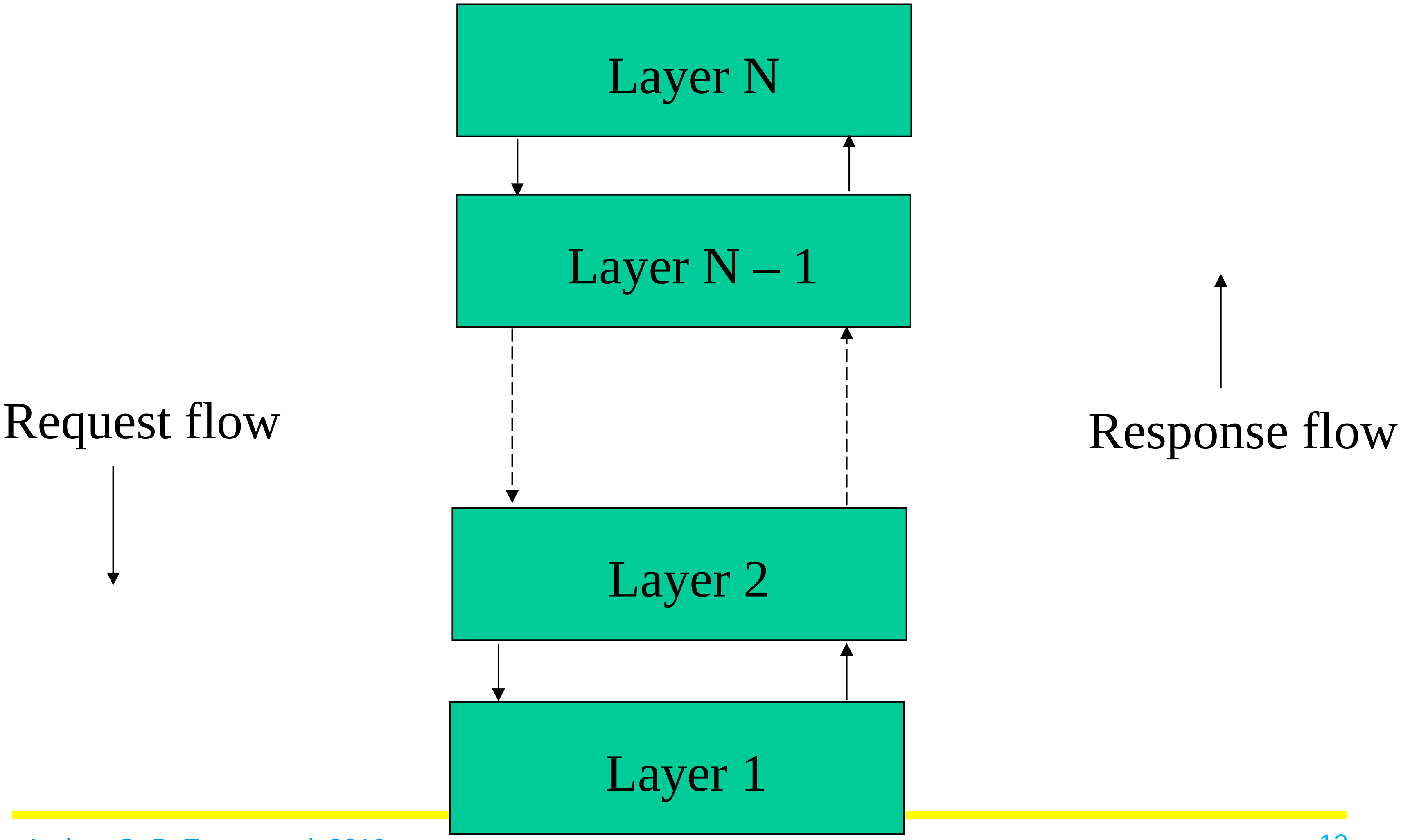
The Web ≠ The Internet

Distributed Systems

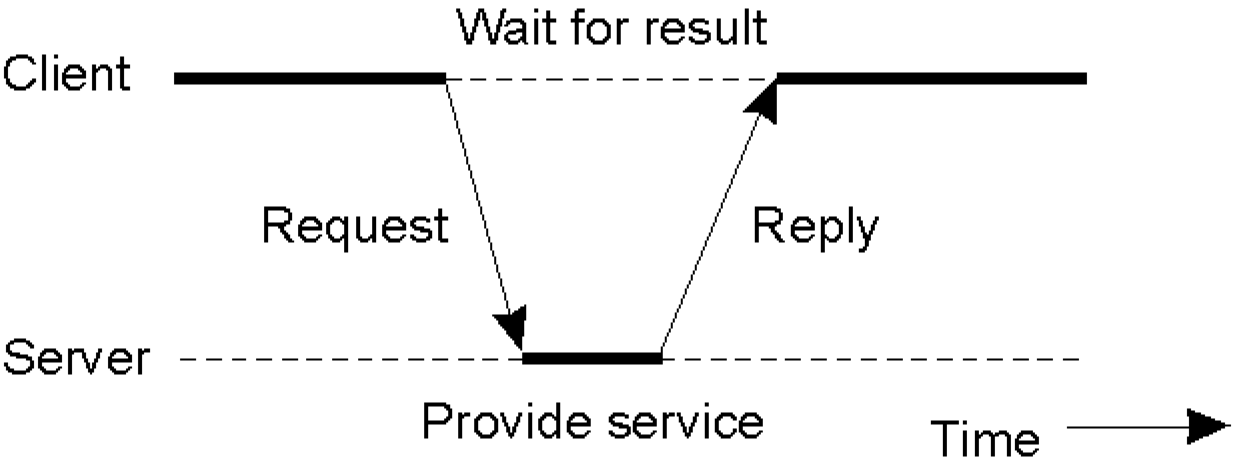


Definition: A collection of independent computers that appears to its users as a single coherent system.

Layered Architectures

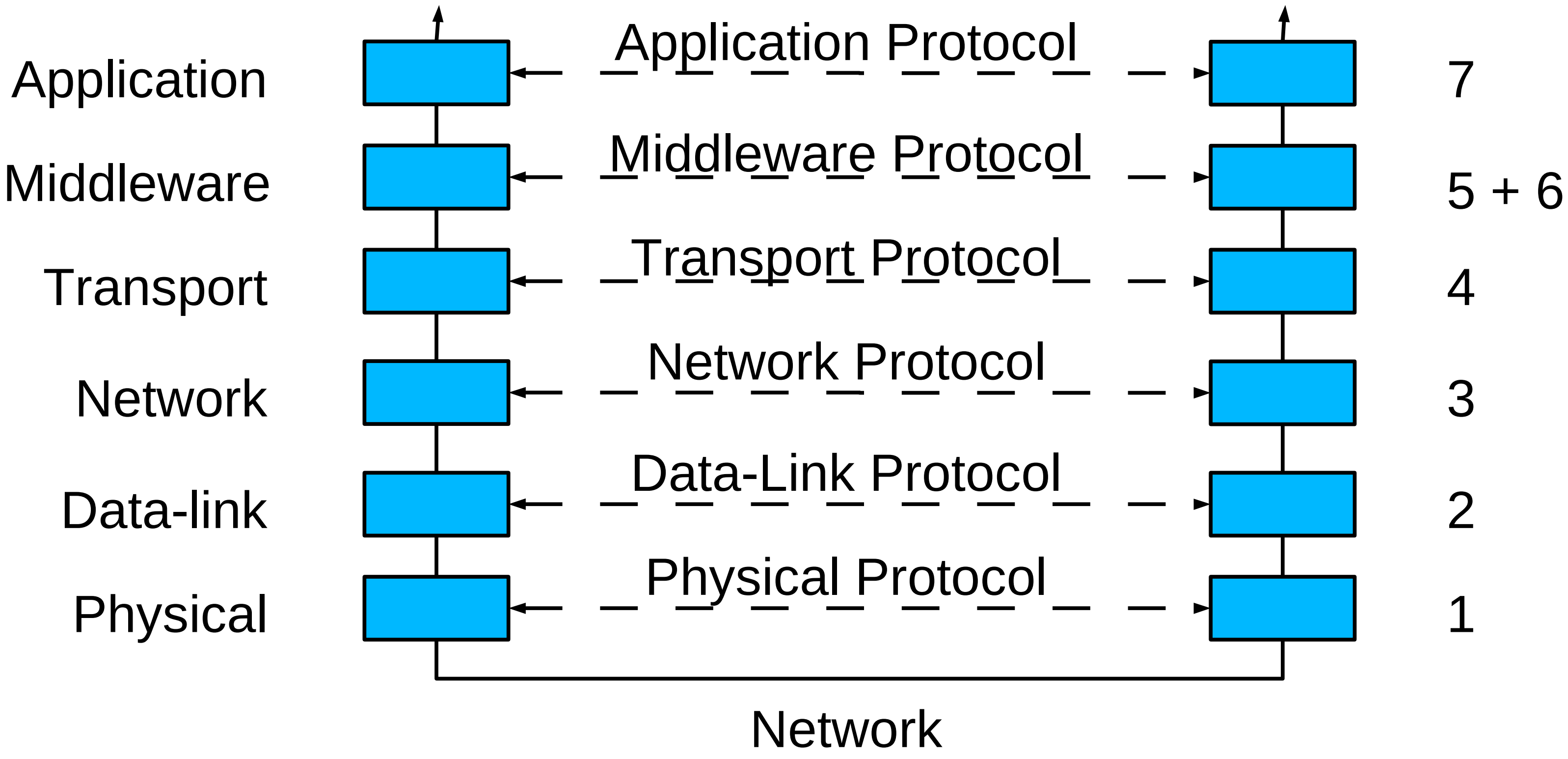


Clients and Servers



General interaction between a client and a server.

Middleware Protocols: An adaptation of the ISO/OSI Stack



Hypertext Transfer Protocol (HTTP)

- Works on top of TCP and IP
- HTTP allows Web servers to send Web content to clients
- To make it simple:
  - Server: a host who can serve Web content
  - Client: a browser
- An HTTP server is implemented by a computer program (e.g., httpd) running on a network host
- A browser is a computer program (e.g., Firefox) running on a user device (PC, laptop, tablet, smartphone,...)
- Web content consists of “documents” (aka Web pages)

Servers and State

Stateless servers

- Never keep accurate information about the status of a client after having handled a request:
- Don’t record whether a file has been opened (simply close it again after access)
- Don’t promise to invalidate a client’s cache
- Don’t keep track of your clients

Consequences

- Clients and servers are completely independent
- State inconsistencies due to client or server crashes are reduced
- Possible loss of performance because, e.g., a server cannot anticipate client behavior (think of prefetching file blocks)

Servers and State

Stateful servers: Keep track of the status of their clients:

- Record that a file has been opened, so that prefetching can be done
- Know which data a client has cached, and allow clients to keep local copies of shared data

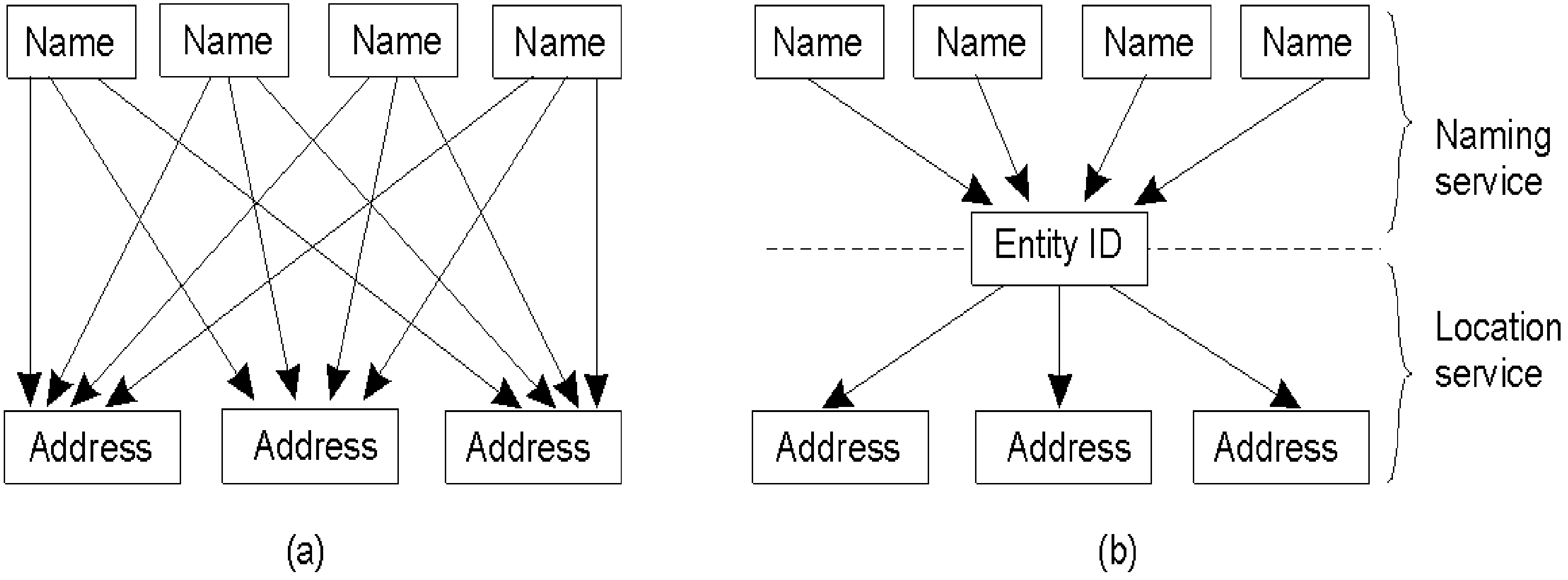
Observation

- The performance of stateful servers can be extremely high, provided clients are allowed to keep local copies. As it turns out, reliability is not a major problem.

Naming in Distributed Systems

- Name
  - Bit or character string → resource
  - Resource:
    - Host, printer, disk, file, Web page,
    - Process, User, Mailbox, window, etc.
- Address
  - Every resource has one or more **access points**
  - Address = name of an access point
  - Access points are not fixed
- Identifier: a special kind of name
  - Resource ↔ Identifier

Naming versus Locating Entities



a) Direct, single level mapping between names and addresses.
b) Three-level mapping using identities.

Naming Resources on the Web: URIs, IRIs, and URLs

- URI = Uniform Resource Identifier (→ IRI = Internationalized …)
- URL = Uniform Resource Locator, if used to locate a resource
- A URL identifies
  - Where a resource is stored (its address)
  - How it can be accessed (a protocol)
- A resource may be stored in different places: it will thus possess one or more URLs
- A resource may be a directory, a document, a fragment of a document, an image, a multimedia file, an executable file, etc.

Structure of an URL

Protocol://	Host	[:Port]	Path	Name	[#Anchor]	[?Parameters]

- Example 1 : a static resource (an HTML document)

http://	www.i3s.unice.fr		/~tettaman/	index.html	#classes	

- Example 2 : a dynamically generated resource

http://	iihm.imag.fr		/cgi-bin/Vitesse2/	vitesse2.bat		?Keywords=unsa&SearchEngine=Google&Kind=Search&InfoSpace=&MaxInfoNumber=100&VitesseMode=Win

URL Encoding

- The components of an URL are alphanumerical strings, plus the two characters – (dash) and _ (underscore)
- The syntax builds on top of IP addresses and Posix paths
- A blank space is sometimes replaced by a +
- Escape sequences are used to represent special characters:
  - %xx, where xx is the hex ASCII code of the character
  - %20 = space
  - %7E = ~
  - %2B = +
  - %25 = %
  - Etc.

HTTP

- HTTP is the most used protocol on the Internet since 1990.
- Developed initiated by Tim Berners-Lee at CERN in 1989
- Development of HTTP standards coordinated by IETF and W3C
- HTTP/1.1 is the version of HTTP in common use
- RFC 2068 (1997) < RFC 2616 (1999) < RFCs 7230 (2014)
- A request–response protocol in the client–server architecture
- HTTP uses plain-text ASCII messages
- The client submits an HTTP **request** message to the server
- The server returns a **response** message to the client
  - Header: completion status information, MIME type of content
  - Body: the requested content (if available/applicable)
- HTTP/1.1 can reuse the same TCP connection: HTTP session

Request Message

- A request message consists of
  - A request line (e.g., GET /images/logo.png HTTP/1.1)
  - Request header fields (e.g., Accept-Language: en)
  - An empty line
  - An optional message body
- Lines are terminated by CR LF
- The request line defines the request **method** (e.g., GET)
  - GET, POST, HEAD (HTTP/1.0)
  - OPTIONS, PUT, DELETE, TRACE, CONNECT (HTTP/1.1)
  - Additional methods can be defined, e.g. PATCH
  - Method names are case-sensitive
- The “Host” header field indicates the target host

HTTP Methods

- GET: requests a representation of the specified resource
- HEAD: same as GET, but response header only
- POST: sends data/items to be added to the specified resource
- PUT: (over)write the resource at the specified URI
- DELETE: delete the specified resource
- TRACE: echoes the received requests
- OPTIONS: requests the supported methods for the specified URI
- CONNECT: open a transparent TCP/IP tunnel
- PATCH: applies partial modifications to a resource

HEAD, GET, OPTIONS and TRACE are **safe**, i.e. they do not change the state of the server

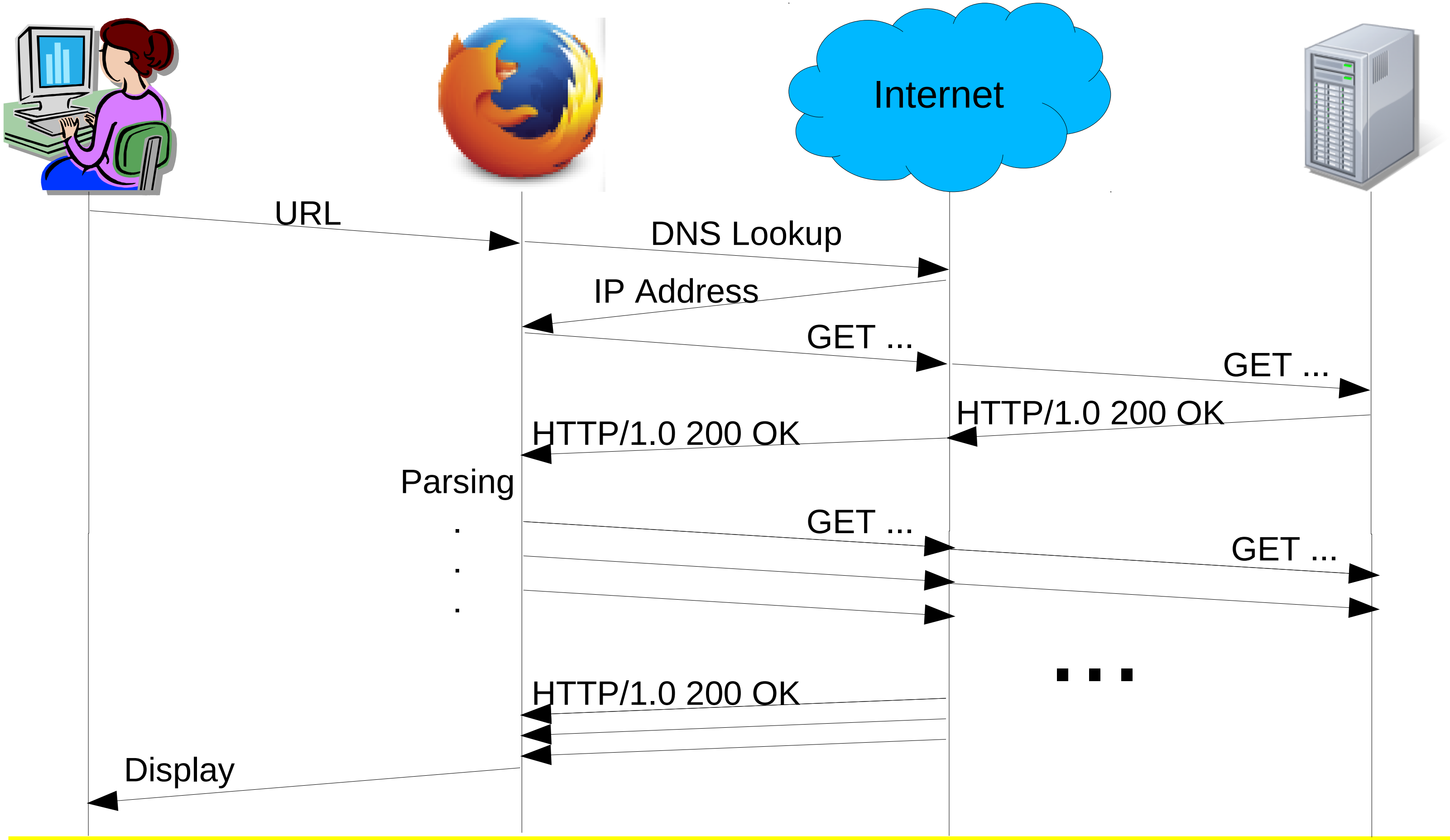
Response Message

- A response message consists of
  - A status line (e.g., HTTP/1.1 200 OK)
  - Response header fields (e.g., Content-Type: text/html)
  - An empty line
  - An optional message body
- Lines are terminated by CR LF
- The status line includes
  - a machine-readable numerical code (e.g., 404)
  - a human-readable textual reason phrase (e.g., “Not Found”)
- Reason phrases are only recommendations
- The first digit of the numerical code defines its general class

Response Classes

- Informational 1XX
- Successful 2XX
  - 200 OK
- Redirection 3XX
  - 301 Moved Permanently
  - 302 Found
  - 303 See Other
- Client Error 4XX
  - 403 Forbidden
  - 404 Not Found
  - 451 Unavailable For Legal Reasons
- Server Error 5XX

In Summary


Internet
URL
DNS Lookup
IP Address
GET ...
GET ...
HTTP/1.0 200 OK
HTTP/1.0 200 OK
Parsing
.
.
.
GET ...
GET ...
...
HTTP/1.0 200 OK
Display