The HTTP Protocol

RES, Lecture 6

Olivier Liechti

heig-vd

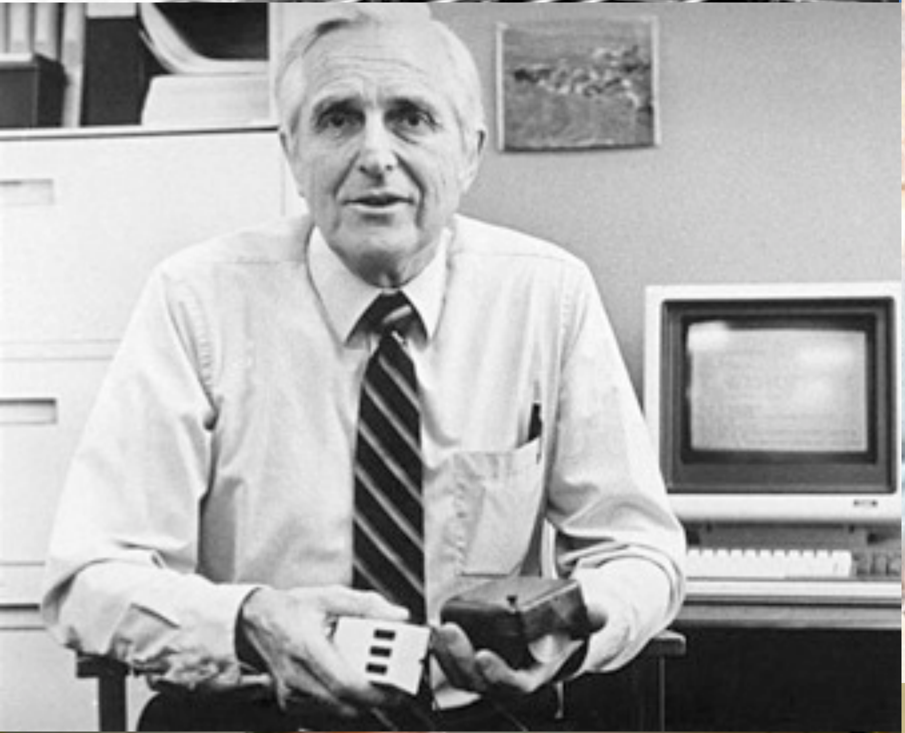
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

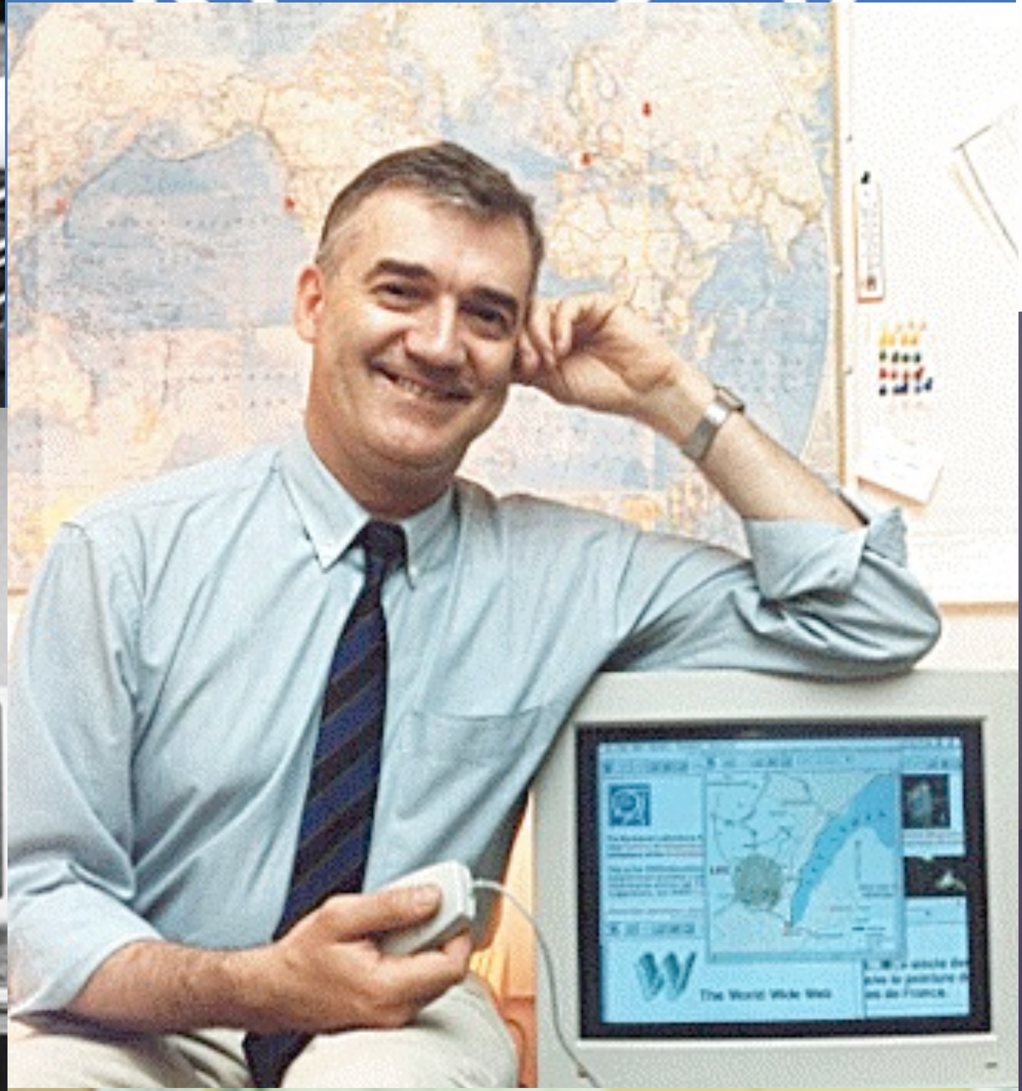
How Do We Use HTTP?

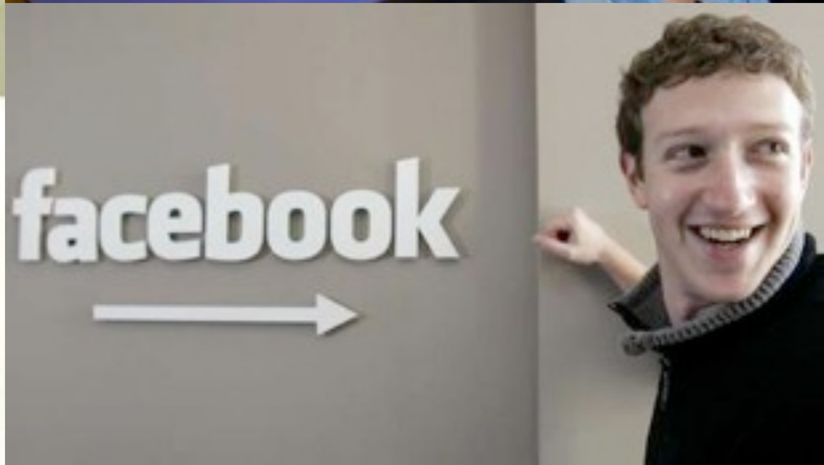
CORVETTE
5.7L

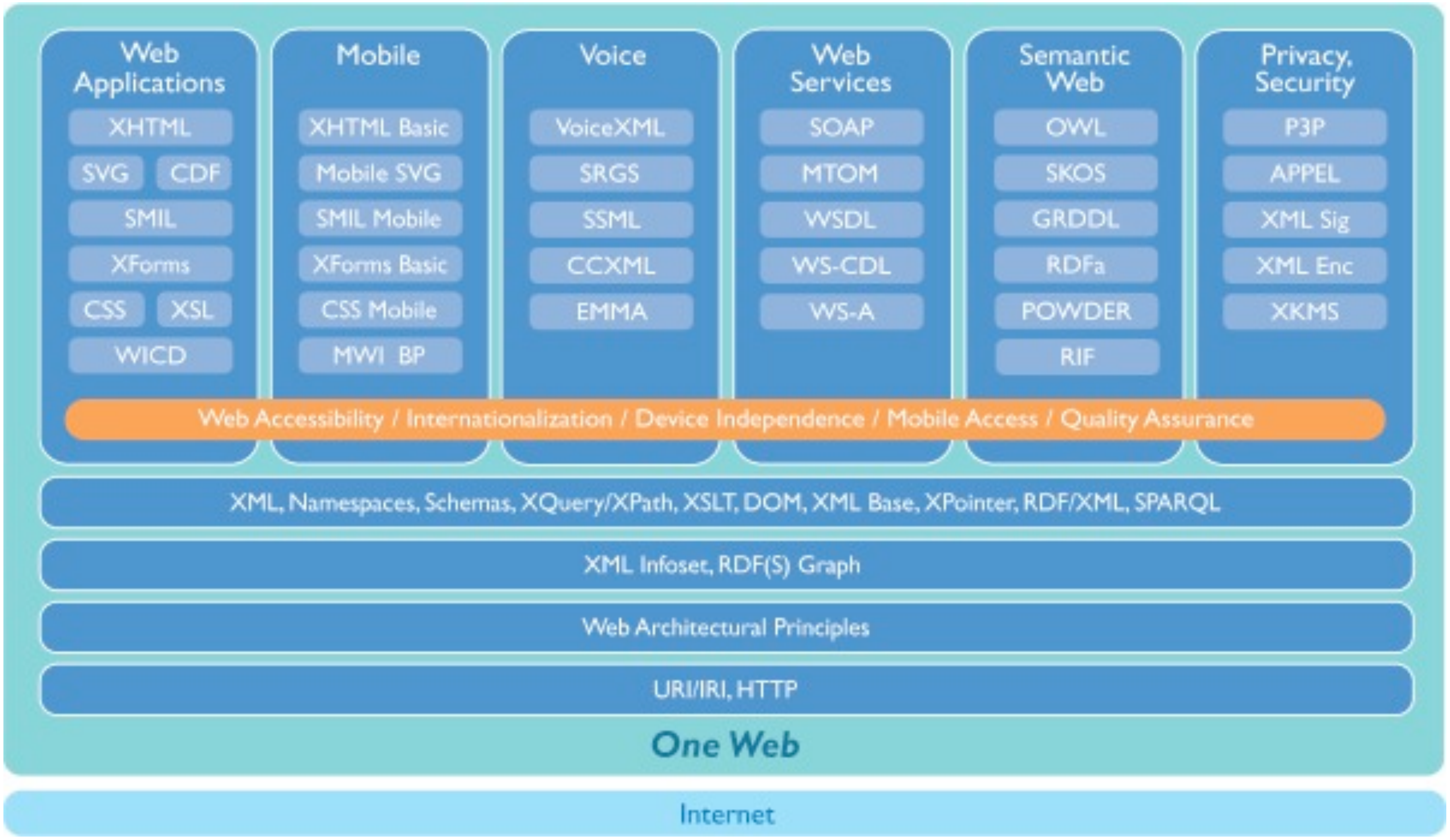
PROPRIETE CERN
This machine is a server. DO NOT POWER IT DOWN!!
PROPRIETE CERN
Information Management: A Proposal

http://en.wikipedia.org/wiki/Image:First_Web_Server.jpg




The World Wide Web

amazon

facebook



http://www.w3.org/Consortium/technology

Looking at a Conversation...

WIRESHARK

Firebug

```
GET / HTTP/1.1 CRLF
Host: www.nodejs.org CRLF
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:28.0) Gecko/20100101 Firefox/28.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 CRLF
Accept-Language: en-us,en;q=0.8,fr;q=0.5,fr-fr;q=0.3 CRLF
Accept-Encoding: gzip, deflate CRLF
Cookie: __utma=212211339.431073283.1392993818.1395308748.1395311696.27;
__utmz=212211339.1395311696.27.19.utmcsr=stackoverflow.com|utmccn=(referral)|utmcmd=referral
|utmcct=/questions/7776452/retrieving-a-list-of-network-interfaces-in-node-js-ioctl-siocgifconf
Connection: keep-alive CRLF
CRLF
```

```
HTTP/1.1 200 OK CRLF
Server: nginx CRLF
Date: Sat, 05 Apr 2014 11:45:48 GMT CRLF
Content-Type: text/html CRLF
Content-Length: 6368 CRLF
Last-Modified: Tue, 18 Mar 2014 02:18:40 GMT CRLF
Connection: keep-alive CRLF
Accept-Ranges: bytes CRLF
CRLF
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <link type="image/x-icon" rel="icon" href="favicon.ico">
    <link type="image/x-icon" rel="shortcut icon" href="favicon.ico">
    <link rel="stylesheet" href="pipe.css">
...
```

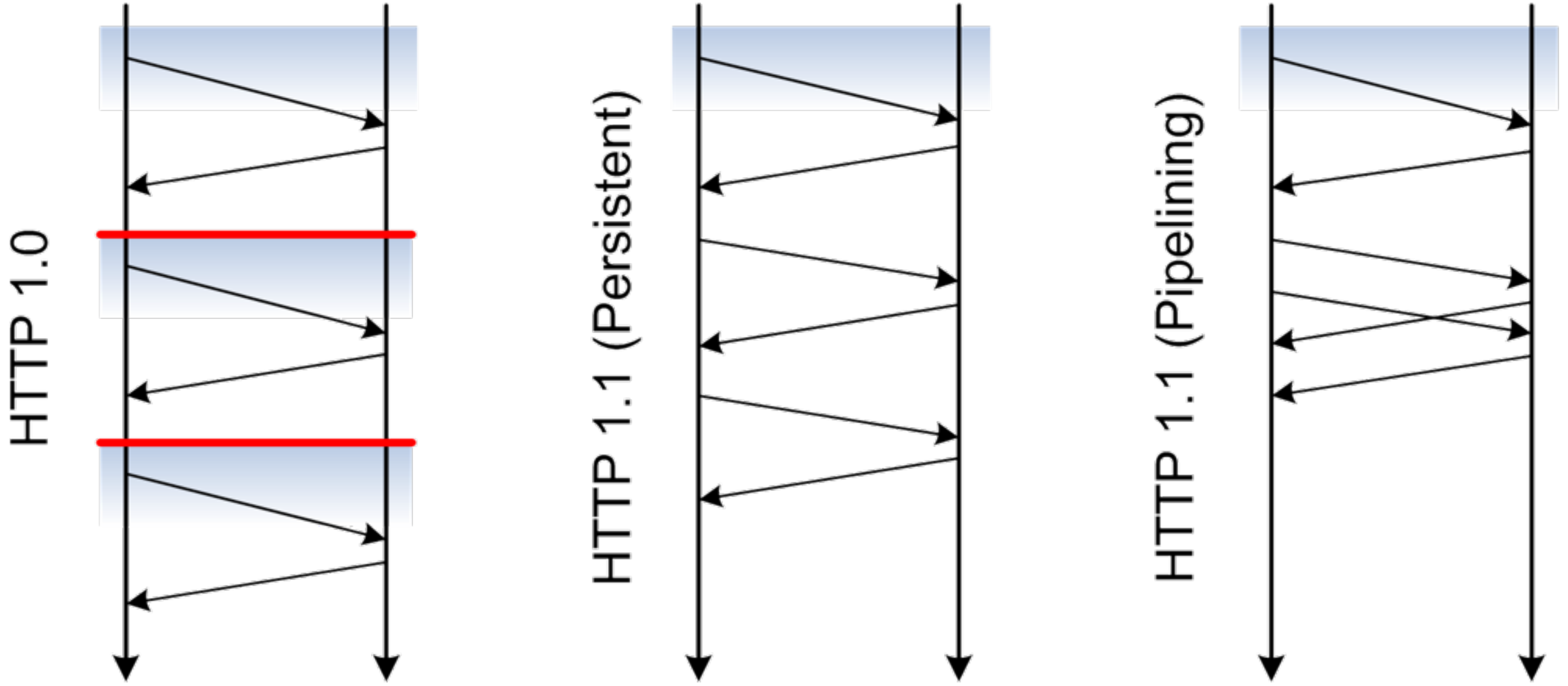
HTTP is a Stateless Request-Reply Transfer Protocol

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, **hypermedia information systems**.

It is a **generic**, **stateless**, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through **extension** of its request methods, error codes and headers [47].

A feature of HTTP is the typing and **negotiation of data representation**, allowing systems to be built independently of the data being transferred.

HTTP & TCP Connections



http://dret.net/lectures/web-fall07/foundations#(20)
http://www.apacheweek.com/features/http11

Stateless Protocol... but Stateful Applications!

Managing State on Top of HTTP

- **Approach 1: moving the state back-and-forth**
  - One way to do it is to used **hidden fields** in HTML forms
- **Approach 2: maintaing state on the backend, transfer session IDs**
  - One way to do it is to use parameters in the **query string** (security...)
  - One way is to use **cookies**

Passing State Back and Forth

```
C: Hello, I am new here. My name is Bob.

S: Welcome, let's have a chat [You told me that "My name is Bob"].

C: [My name is Bob]. What's the time?

S: Hi again Bob. It's 10:45 AM. [You told me that "My name is
Bob". You asked me what is the time]
```

Passing Session ID Back and Forth

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

```
C: Hello, I am new here. My name is Bob.
S: Welcome Bob, let's have a chat. Your session id is 42.
C: My session id is 42. What's the time?
S: -- checking my notes… hum… ok, I found what I remember about
session 42…
S: Hi again Bob. It's 10:45 AM.
C: My session id is 42. How do you do?
S: -- checking my notes… hum… ok, I found what I remember about
session 42…
S: I am fine, Bob, thank you.
C: My session id is 42. How do you do?
S: -- checking my notes… hum… ok, I found what I remember about
session 42…
S: I told you I am fine… are you stupid or what?
C: My session id is 42. If you take it like that, I am gone.
Forever.
S: -- checking my notes… hum… ok, I found what I remember about
session 42…
S: -- putting 42 file into trash…
S: Bye Bob.
```

Resources, Resource Representations & Content Negotiation

Resource vs Resource Representation

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

- **The notion of resource is very generic and can represent anything...**
  - An online document
  - A list of online documents
  - A stock quote updated in realtime
  - A vending machine
- **What is transferred is not the resource, but a representation of the resource**
  - HTML representation, JSON representation, PNG representation
  - french representation, english representation, japanese representation
  - etc.

Content Negotiation

- **When making a request, the client specifies its abilities and preferences**
  - media type: image, text, structured text?
  - media format: JSON, XML, etc.?
  - language: english, french, etc.
  - character encoding: UTF-8, ASCII, etc.
- **When answering the request, the server tries to do its best and indicates what it has been able to do**
- **Special headers are used to support this process**
  - **Request:** `Accept, Accept-Charset, Accept-Language`
  - **Response:** `Content-Type, Content-Language`

Protocol Syntax

HTTP Methods

GET

POST

PUT

DELETE

(PATCH)

URI

http://www.heig-vd.ch

HTTP Requests

```
Full-Request   = Request-Line              ; Section 5.1
                 *( General-Header         ; Section 4.3
                  | Request-Header         ; Section 5.2
                  | Entity-Header )        ; Section 7.1
                 CRLF
                 [ Entity-Body ]           ; Section 7.2

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Request-Header = Authorization             ; Section 10.2
               | From                      ; Section 10.8
               | If-Modified-Since         ; Section 10.9
               | Referer                   ; Section 10.13
               | User-Agent                ; Section 10.15

Entity-Header  = Allow                     ; Section 10.1
               | Content-Encoding          ; Section 10.3
               | Content-Length            ; Section 10.4
               | Content-Type              ; Section 10.5
               | Expires                   ; Section 10.7
               | Last-Modified             ; Section 10.10
               | extension-header
```

HTTP Responses

```
Full-Response   = Status-Line                  ; Section 6.1
                  *( General-Header            ; Section 4.3
                   | Response-Header           ; Section 6.2
                   | Entity-Header )           ; Section 7.1
                   CRLF
                   [ Entity-Body ]             ; Section 7.2

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

Response-Header = Location                     ; Section 10.11
                | Server                       ; Section 10.14
                | WWW-Authenticate             ; Section 10.16


Entity-Header  = Allow                         ; Section 10.1
               | Content-Encoding              ; Section 10.3
               | Content-Length                ; Section 10.4
               | Content-Type                  ; Section 10.5
               | Expires                       ; Section 10.7
               | Last-Modified                 ; Section 10.10
               | extension-header
```

Status Codes

```
The first digit of the Status-Code defines the class of response. The
last two digits do not have any categorization role. There are 5
values for the first digit:

   o 1xx: Informational - Not used, but reserved for future use

   o 2xx: Success - The action was successfully received,
          understood, and accepted.

   o 3xx: Redirection - Further action must be taken in order to
          complete the request

   o 4xx: Client Error - The request contains bad syntax or cannot
          be fulfilled

   o 5xx: Server Error - The server failed to fulfill an apparently
          valid request
```

Status Codes

```
Status-Code     = "200"    ; OK
                | "201"    ; Created
                | "202"    ; Accepted
                | "204"    ; No Content
                | "301"    ; Moved Permanently
                | "302"    ; Moved Temporarily
                | "304"    ; Not Modified
                | "400"    ; Bad Request
                | "401"    ; Unauthorized
                | "403"    ; Forbidden
                | "404"    ; Not Found
                | "500"    ; Internal Server Error
                | "501"    ; Not Implemented
                | "502"    ; Bad Gateway
                | "503"    ; Service Unavailable
                | extension-code

extension-code = 3DIGIT
Reason-Phrase  = *<TEXT, excluding CR, LF>
```

Parsing HTTP Messages

Process for Parsing HTTP Messages

- **Do not read characters, read bytes**
  - At the beginning, you want to parse line by line
  - When consuming the body, you may be dealing with binary content
- **HTTP 1.0**
  - On the client side, read until the connection is closed (end of stream reached).
  - On the server side, use the **`Content-Length`** header (for `POST` requests)
- **HTTP 1.1**
  - Static content: use the **`Content-Length`** header
  - Dynamic content: use the **`chunked`** transfer encoding

Recommendations

- **Implement your own LineByLineInputStream**
  - Remember the lecture about IOs & decorators?
  - You would like to have a `readLine()` method... but this one is available only in `Reader` classes
  - Implement your subclass of FilterInputStream and detect `\r\n` sequences
- **Add functionality incrementally, starting with a client**
  - Start with HTTP 1.0 (read until close of connection)
  - Deal with `Content-Length` header
  - Deal with `chunked` transfer encoding