
Aplicacions Distribuïdes

Silvia Llorente
silviall@ac.upc.edu

Distributed Multimedia Applications Group (DMAG)
Departament d'Arquitectura de Computadors (DAC)

Tema 1. Introducció

- Nivel Aplicación
- Modelo Cliente / Servidor (C/S)
- Invocación remota de operaciones y métodos: RPC, RMI, CORBA

Tema 1. Introducció

- Librerías de comunicaciones
 - Sockets: Nivel transporte TCP, UDP
 - Protocolo HTTP
- Formatos de información
 - MIME
 - HTML y HTML5
 - XML

1.1 Introducció

- Conceptos básicos a trabajar en la asignatura

Perspectiva hist3rica de las redes



Redes telef3nicas para transmisi3n de voz

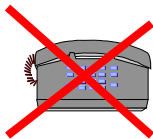
Años 60



Primeros ordenadores comerciales

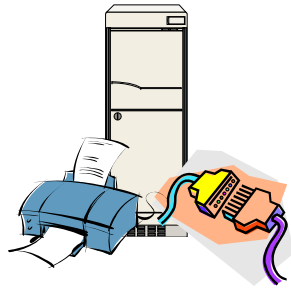


M3dems para transmisi3n de datos en redes telef3nicas



Las redes telef3nicas son demasiado caras para el env3o de datos, aparecen las redes de datos

Perspectiva histórica de las redes



Empresas de informática como IBM o DEC
Proporcionan todo lo necesario a sus clientes,
desde cables a S.O.

Sistemas cerrados

Años 70

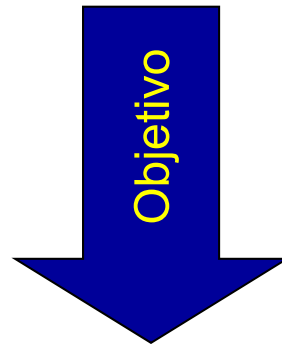
Protocolo Ethernet para LAN's
Aparición de UNIX
Protocolos TCP/IP

Sistemas abiertos

Perspectiva histórica de las redes

Aplicación
Presentación
Sesión
Transporte
Red
Enlace
Físico

ISO (*International Organization for Standardization*) y CCITT proponen la pila de protocolos OSI (*Open Systems Interconnection*)



Armonizar las distintas alternativas (propietarias, abiertas, ...) con un modelo organizado en niveles

Perspectiva hist3rica de las redes

Actualmente

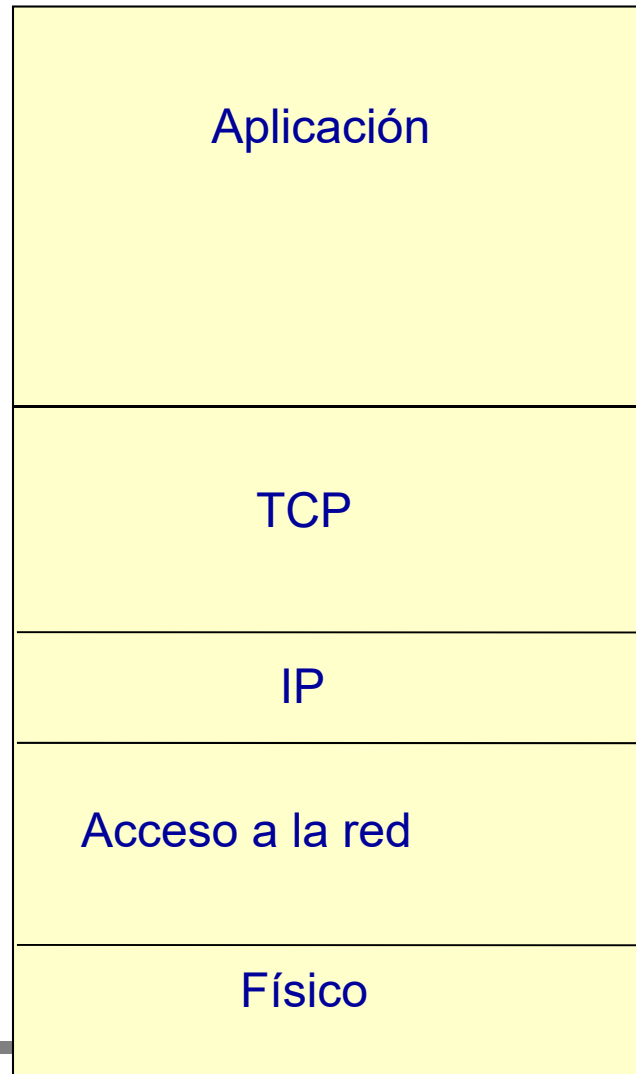
El modelo Internet basado en TCP/IP ha triunfado y aplicaciones de tipo Cliente/Servidor como WWW (*World Wide Web*) son las m1s utilizadas, aunque hay una evoluci3n hacia aplicaciones m3viles (tambi3n basadas en TCP/IP)

El uso del modelo OSI no se ha generalizado, pero se utiliza para explicar arquitecturas de protocolos

El modelo OSI



El modelo Internet

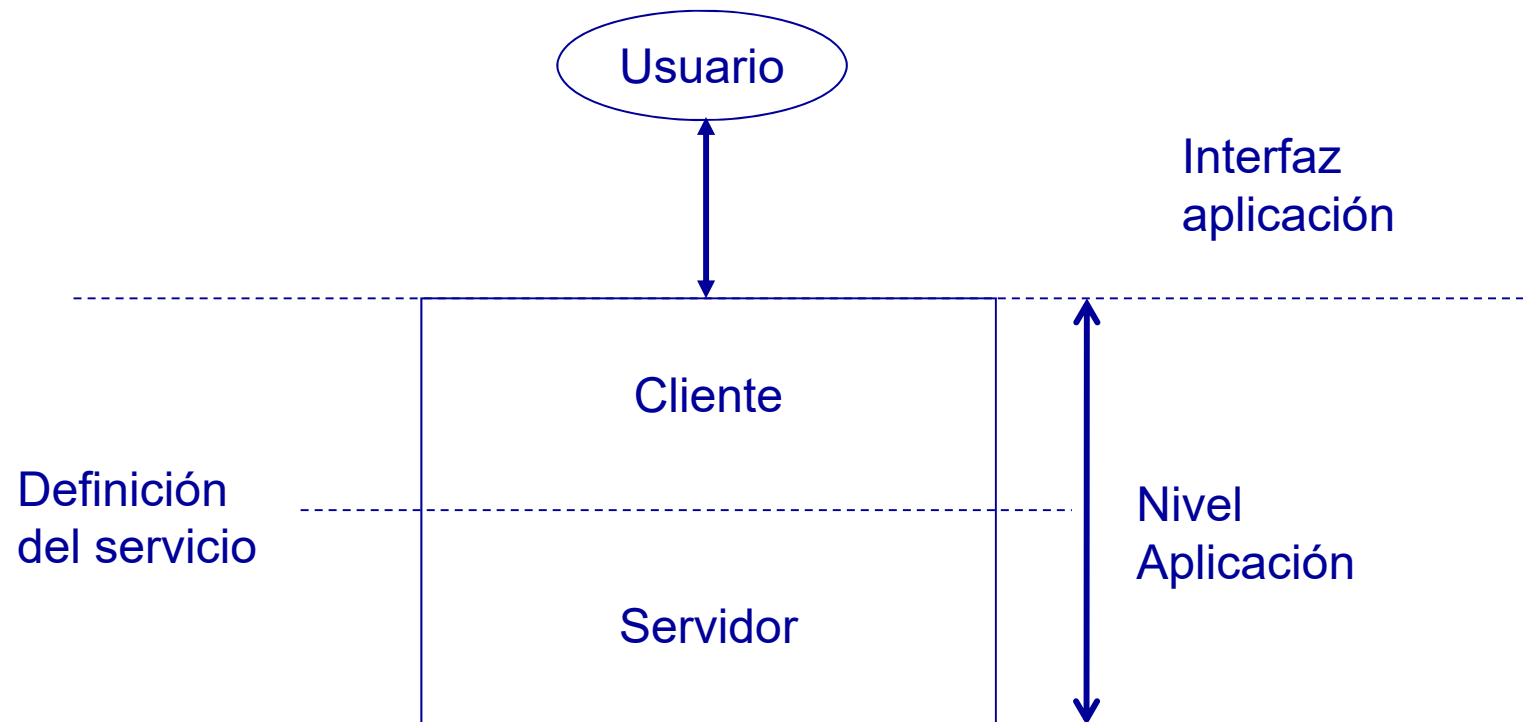


Modelo Internet vs. Modelo OSI

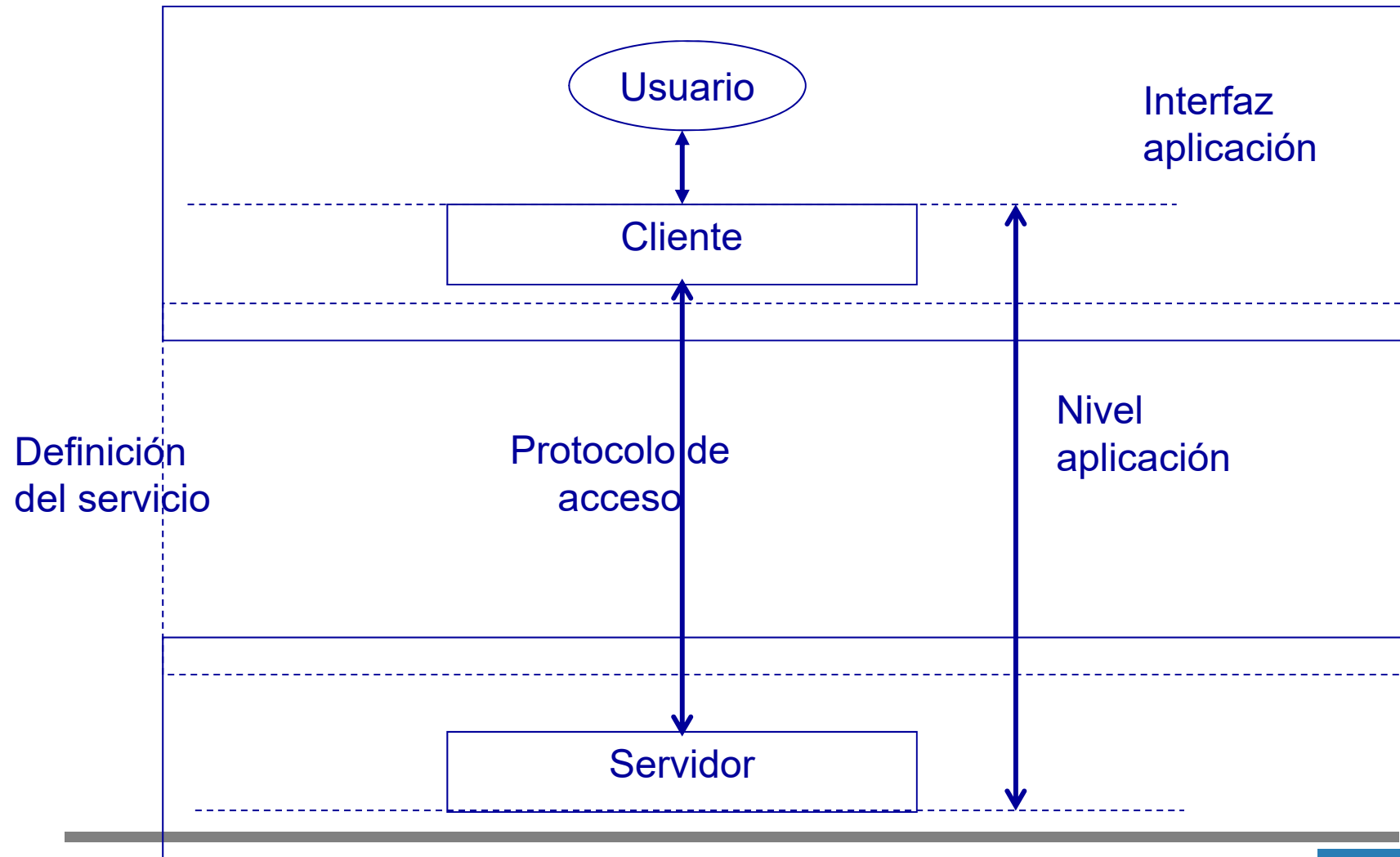
Aplicación	Aplicación
	Presentación
	Sesión
TCP	Transporte
IP	Red
Acceso a la red	Enlace
Físico	Físico

Fuente: Comunicaciones y redes de computadores, 6ª ed., William Stallings

El modelo Cliente / Servidor (C/S)



El modelo Cliente / Servidor (C/S)



Modelo C/S - Cliente

- Normalmente está en la máquina que utiliza el usuario
- Funciones principales:
 - Convertir llamadas locales a remotas
 - Recibir resultados y facilitárselos al usuario

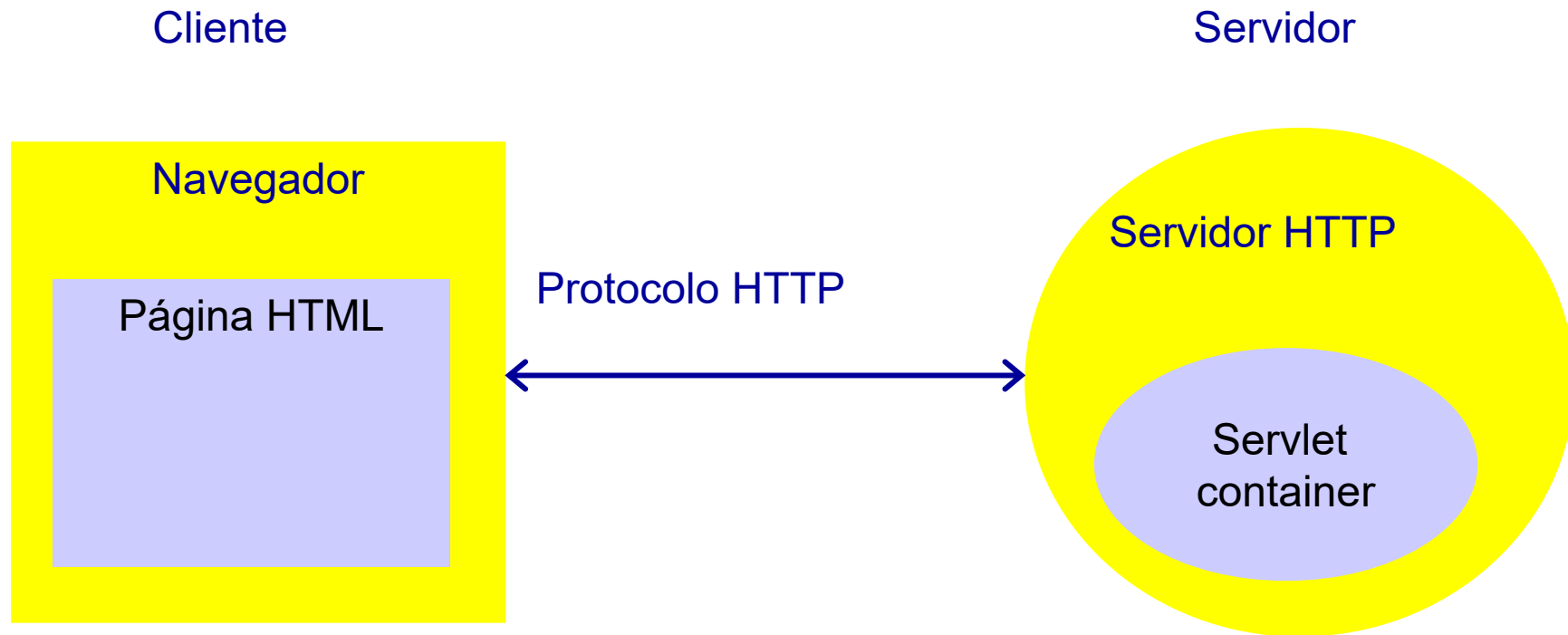
Modelo C/S - Servidor

- Suele residir en una máquina distinta al cliente, pero podrían estar en la misma
- Funciones principales:
 - Encargado de realizar las operaciones invocadas por el usuario
 - Devuelve los resultados de dichas operaciones

Ejemplo de aplicación distribuida

- Racó de la FIB
- Aplicación cliente / servidor
- Utiliza el protocolo HTTP para comunicarse

Arquitectura



Aplicación cliente

- Navegador
- Protocolo nivel aplicación: HTTP
- Interpreta y muestra ficheros en formato HTML
- Los ficheros HTML pueden contener formularios

Aplicación cliente

- Protocolo comunicación: HTTP
- Protocolo basado en comandos
- Permite el acceso a recursos alojados en el servidor
- Comandos GET y POST
 - Acceso a recursos, pero también envío de información al servidor

Aplicación servidor

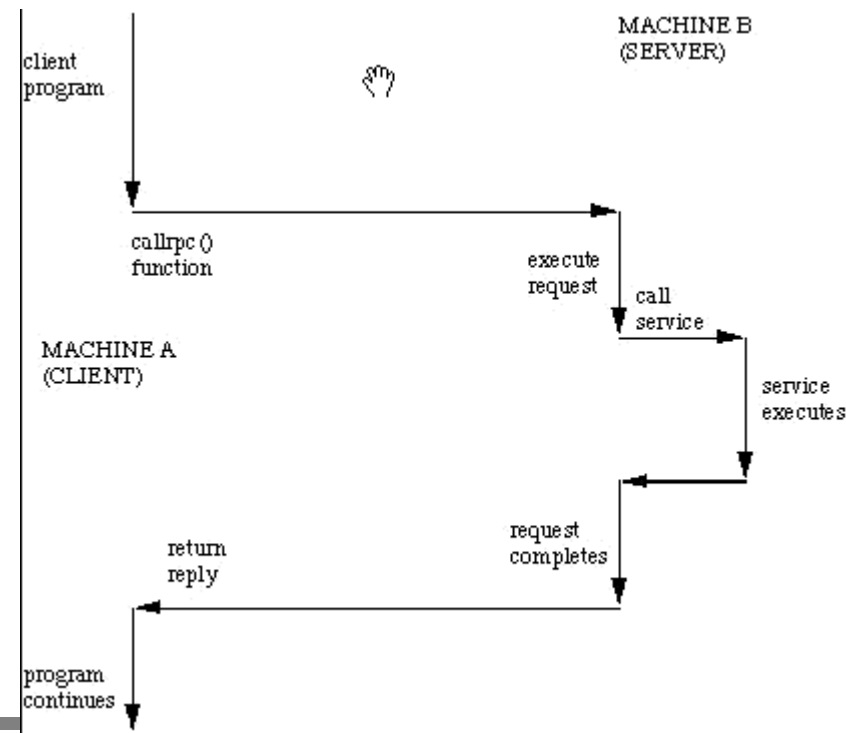
- Proceso que recibe comandos HTTP y decide qué hacer
- Si el recurso que hemos solicitado es un *servlet* → Entra en juego el *Servlet container* → *Tema 2*

Definición de Aplicación distribuida

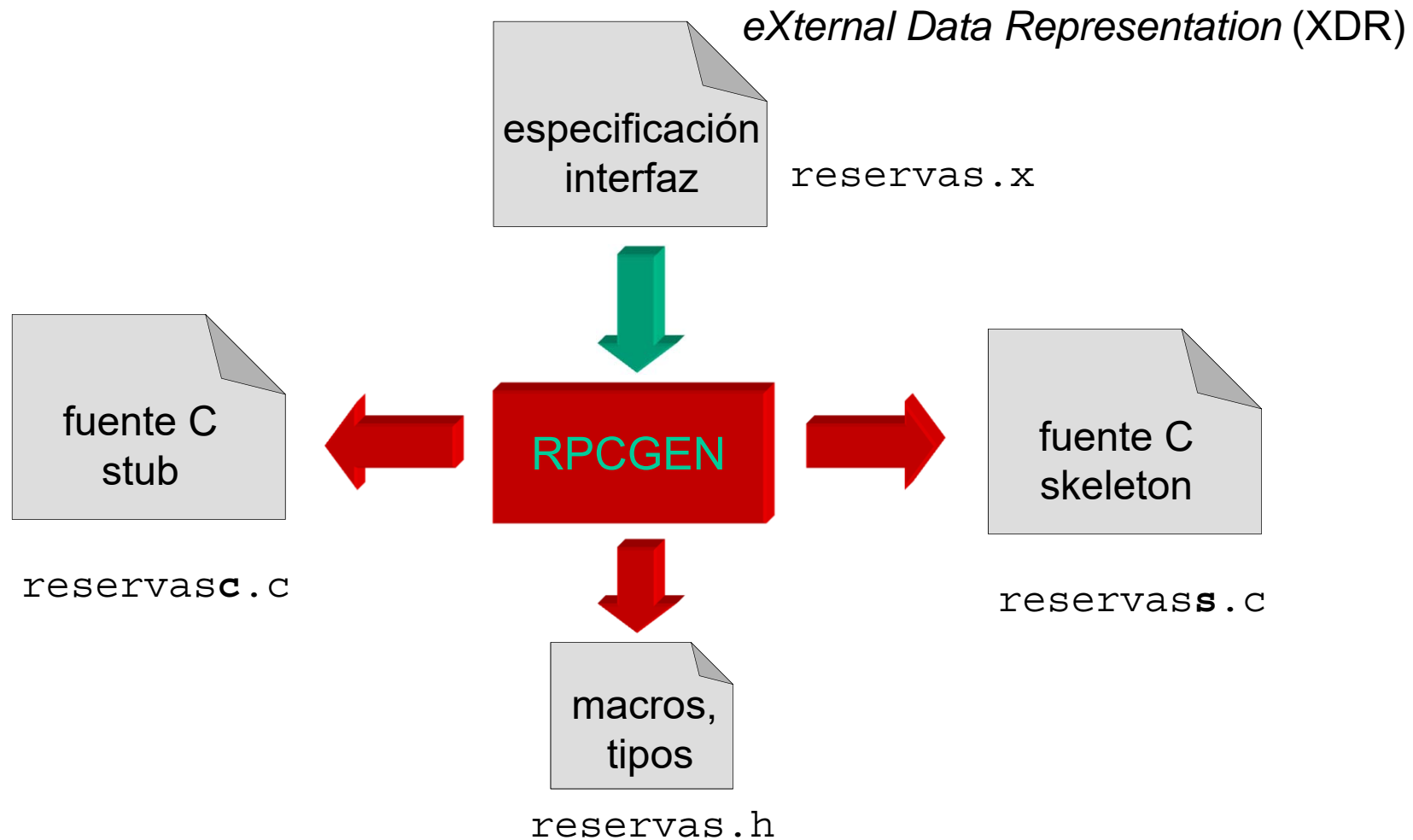
- An application made up of distinct components running in separate runtime environments, usually on different platforms connected via a network
- Typical distributed applications are two-tier (client-server), three-tier (client-middleware-server) and multitier (client-multiple middleware-multiple servers)

RPC, RMI, CORBA

- Remote Procedure Call (RPC)
 - La llamada a un procedimiento se realiza de forma remota.
 - Es necesario definir los tipos de datos e interfaces que utilizará el procedimiento remoto.
 - Lenguaje C – linux, unix



Generación de stubs & skeletons en RPC



Reference: <http://www.cs.cf.ac.uk/Dave/C/node33.html>

Ejemplo especificación XDR

```
const MAX=10;
typedef int Cantidad;
typedef int Coste;
typedef int Bool;
```

Constantes y tipos como en C

```
struct ReserveArgs{
    char vuelo[MAX];
    Cantidad reserva;
};
```

RPC admite sólo 1 argumento

Ejemplo especificación XDR

```
struct Resultado{  
    Bool exito;  
    Coste precio;  
};
```

RPC tiene sólo un resultado

```
program RESERVASVUELOS{  
    version VERSION {  
        Resultado  
RESERVA(ReserveArgs)=1;  
    }=3;  
}=6786;
```

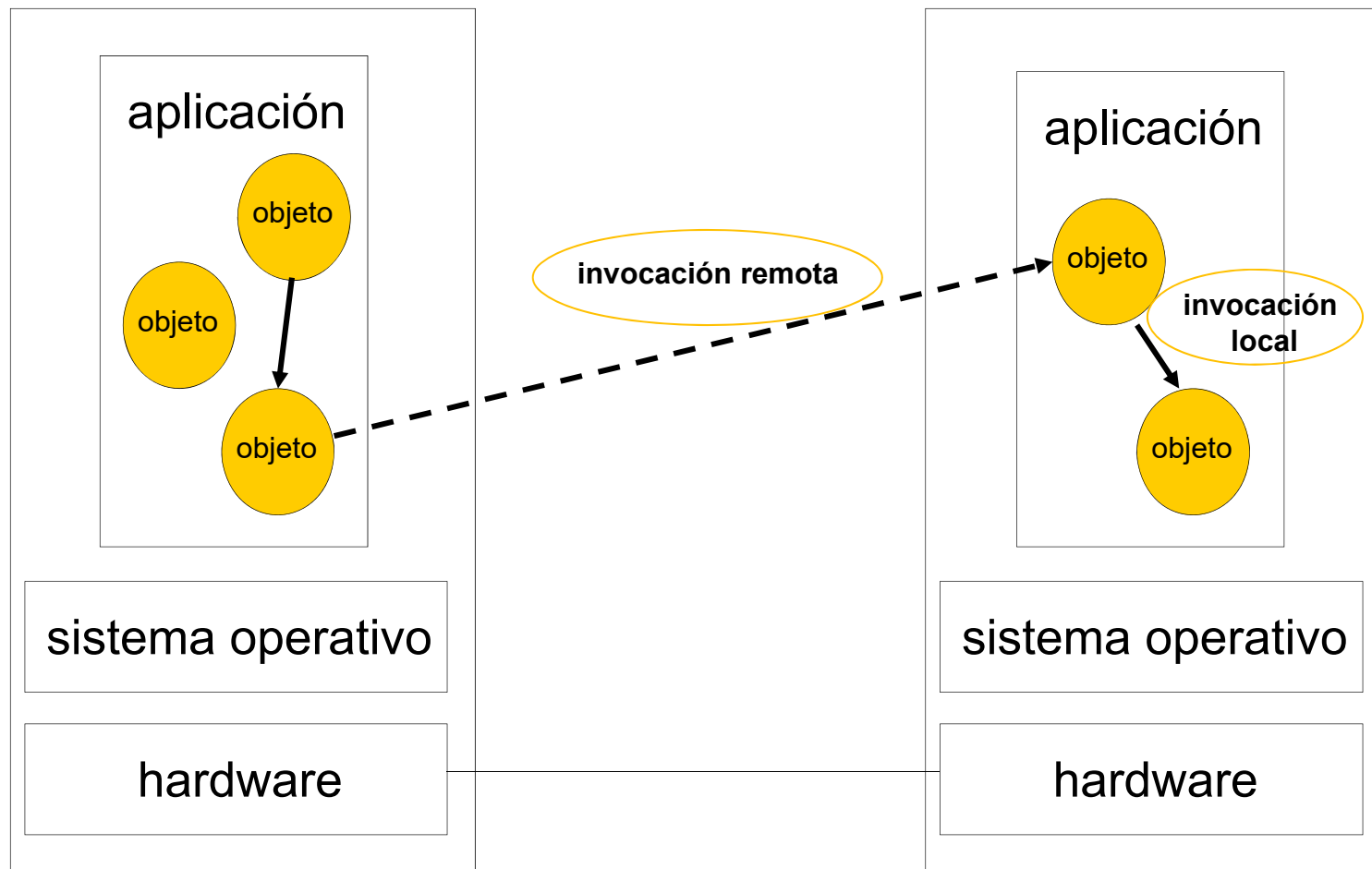
RPC da número a:

- programa
- versión
- procedimiento

RPC, RMI, CORBA

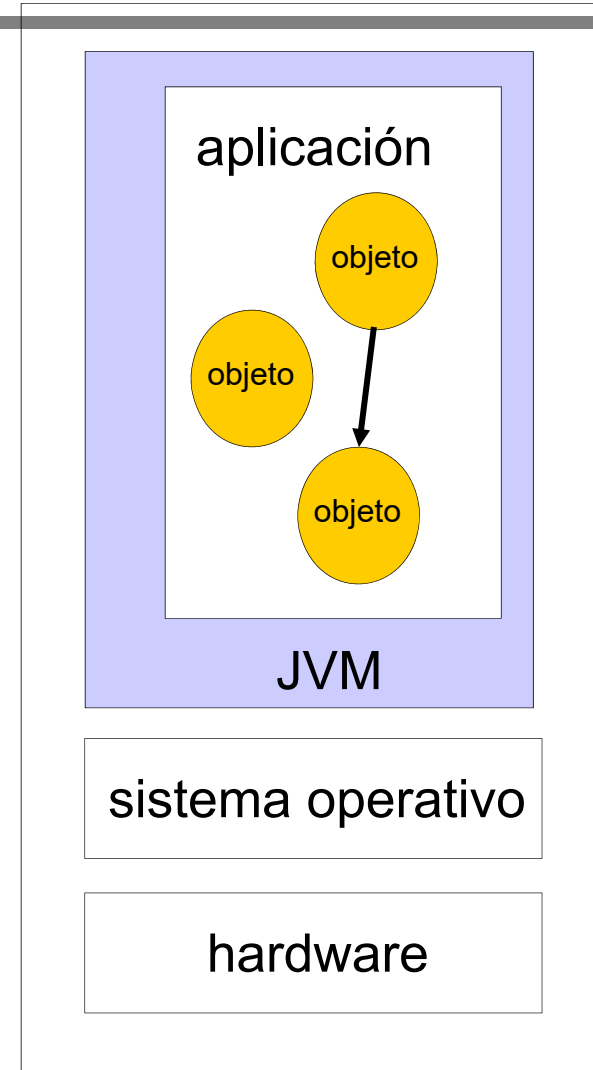
- Remote Method Invocation (RMI)
 - El concepto es similar al de RPC, pero utilizando Java, un lenguaje orientado a objetos.
 - Se invocan métodos de un objeto remoto.

Aplicaciones distribuidas de objetos

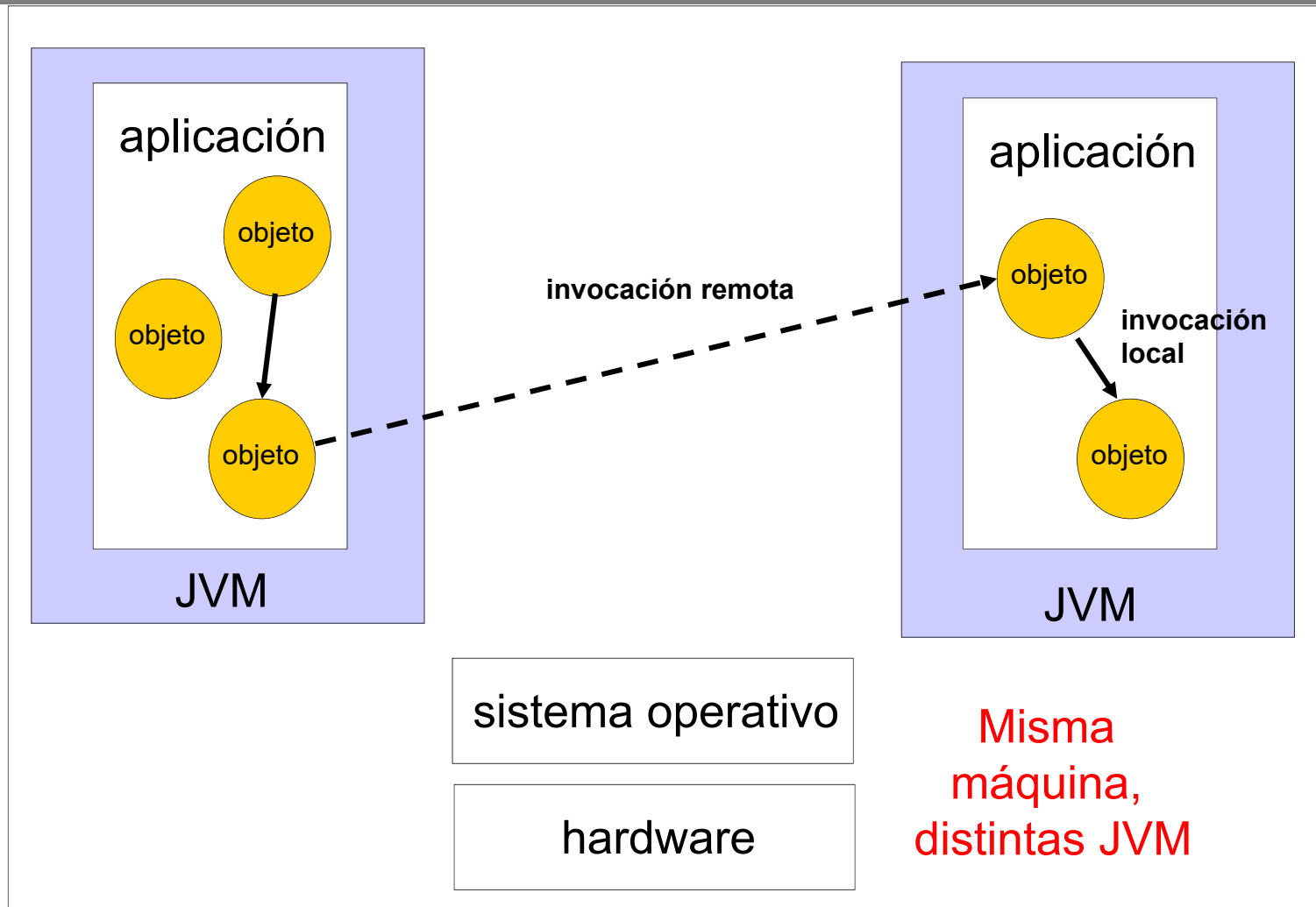


Aplicación Java local

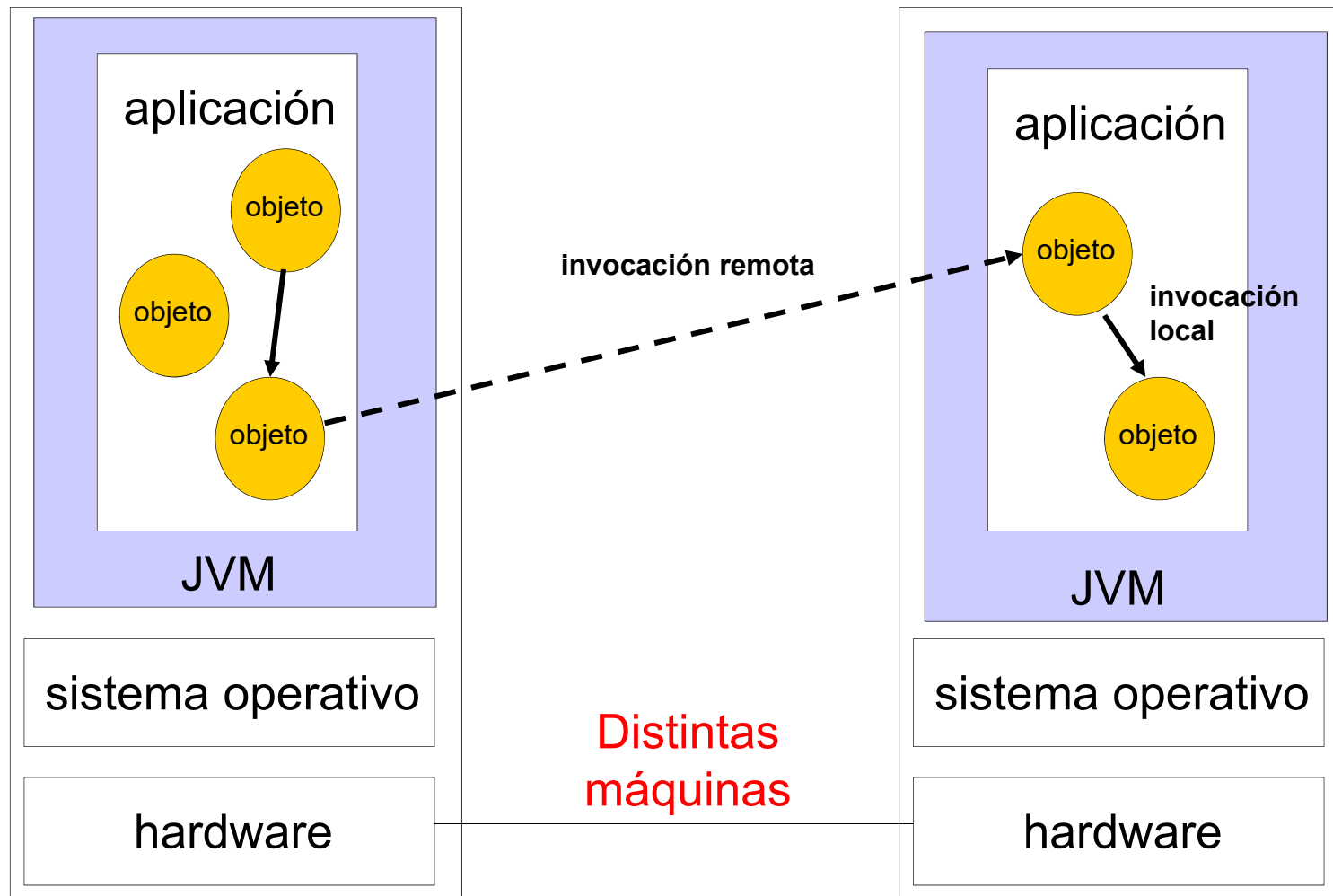
- Los objetos se encuentran en la misma JVM
- Referencias locales a los objetos



Aplicación Java remota



Aplicación Java remota



Aplicación Java remota

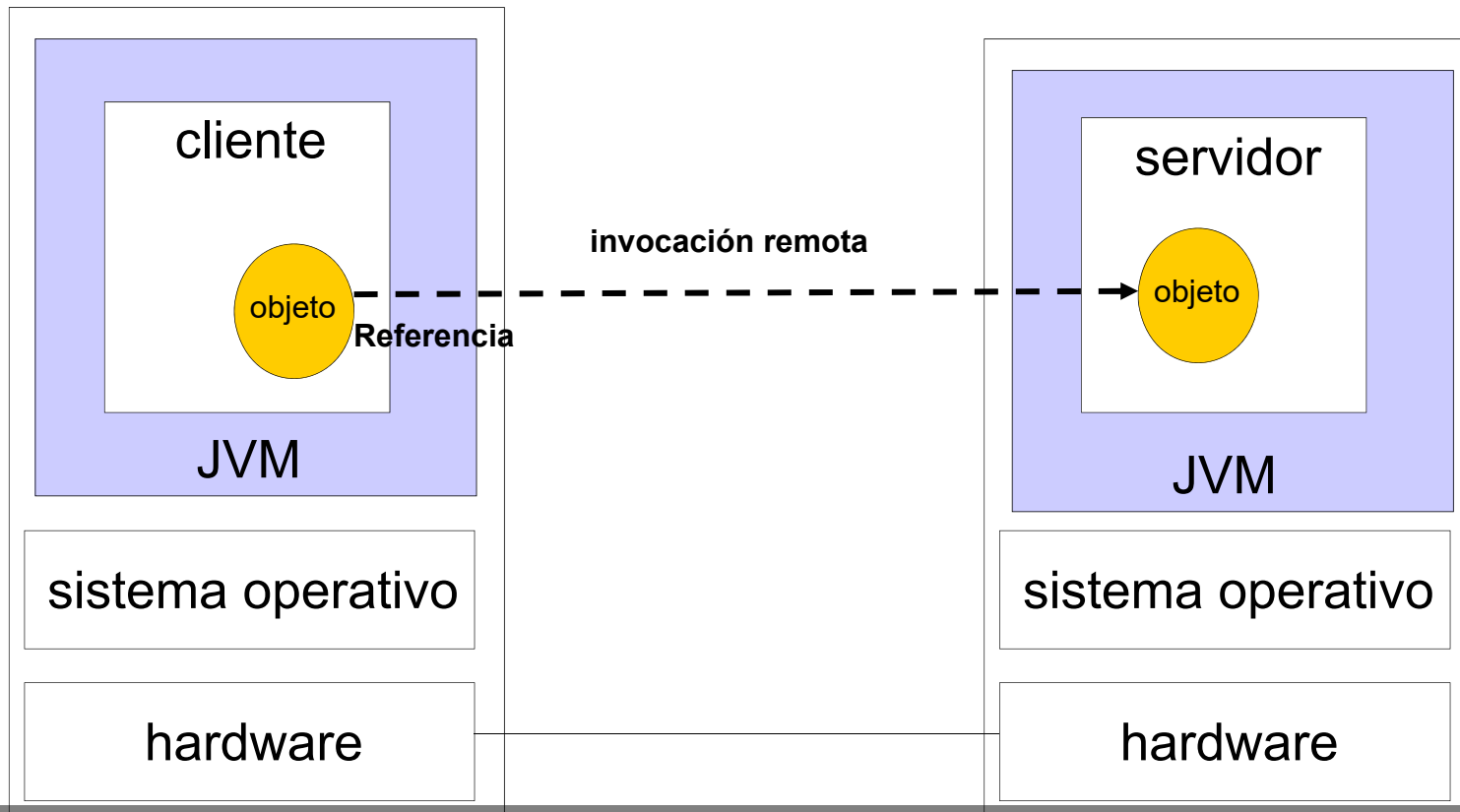
- Los objetos se encuentran en la distintas JVM (y posiblemente en distintas máquinas)
- ¿Cómo referenciar los objetos?

Paso de parámetros en RMI

- Tipo básico
 - Por valor
- Objeto que implementa interfaz *Remote*
 - Referencia al objeto que se trata como si fuera local
- Objeto que implementa interfaz *Serializable*
 - Se transforma en bytes y se envía

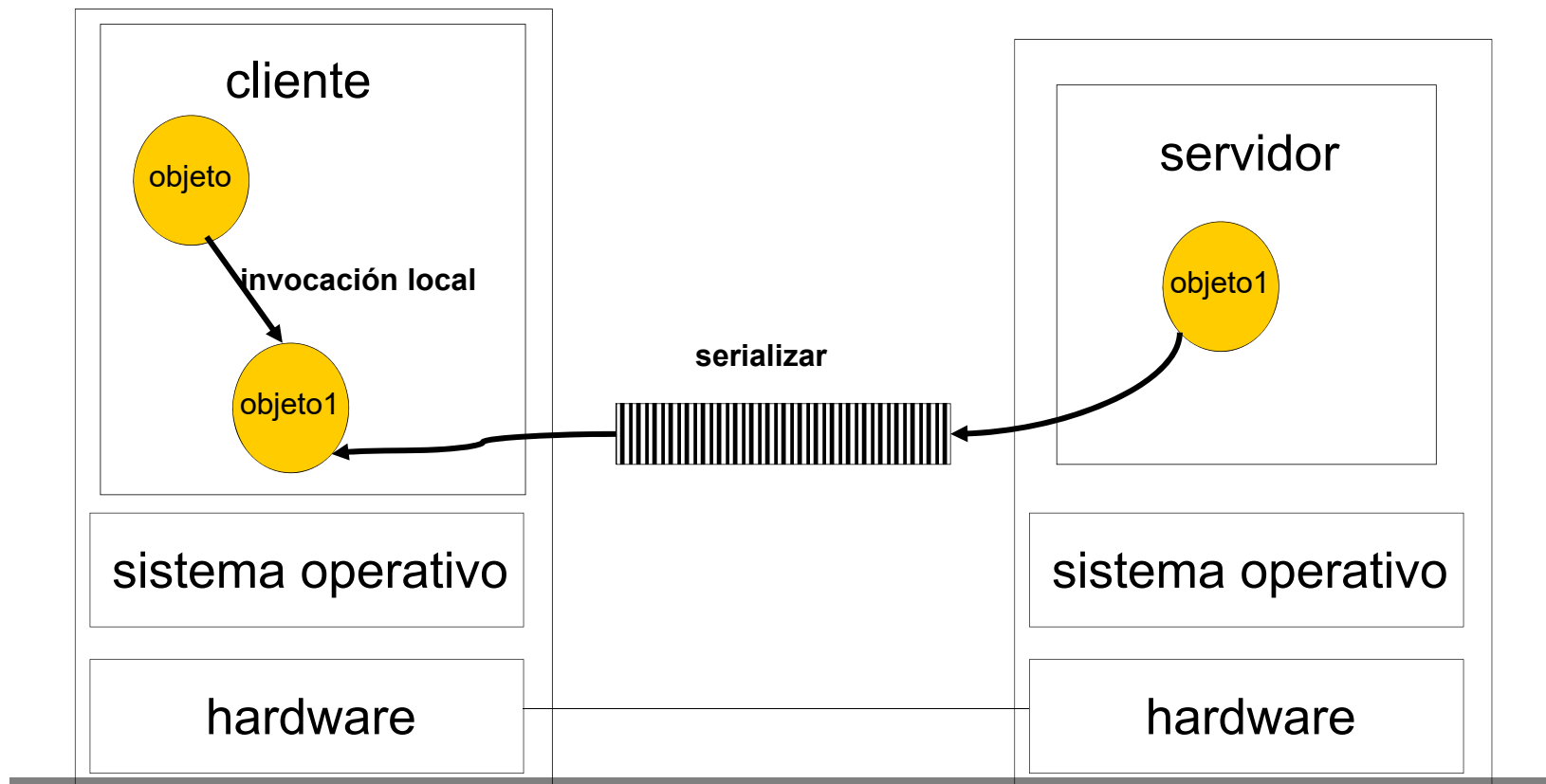
Remote

- Ejecución en el servidor, pero se invoca desde el cliente



Serializable

- El objeto viaja hasta el cliente y la ejecución es local



Problemas específicos de los objetos

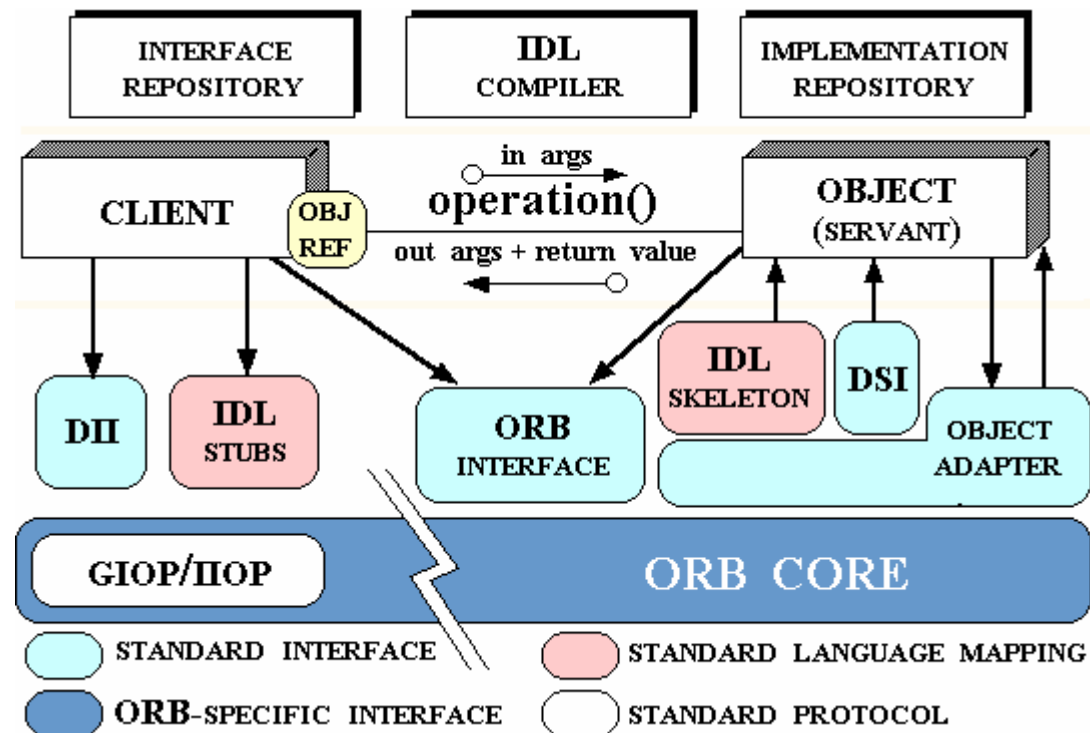
- Referenciar los objetos remotos
- Creación de objetos remotos
- Recolección de memoria (*garbage collection*)
- Excepciones

CORBA

- Common Object Request Broker Architecture (CORBA)
 - Estándar definido por el Object Management Group (OMG) en el año 1995
 - Define una arquitectura que permita conectar objetos escritos en distintos lenguajes de programación
 - La comunicación entre componentes es con un formato binario
 - Ejemplo de uso: Cliente C++ conecta con servidor en Java
- Referencia:
- <http://www.cs.wustl.edu/~schmidt/corba-overview.html>

CORBA

- CORBA
 - Arquitectura y componentes



Sockets, HTTP

- Sockets
 - Mecanismo que permite establecer conexiones remotas a nivel TPC y UDP.
 - Dos tipos de socket
 - Stream para TCP
 - Datagram para UDP
 - El nivel más bajo al que podemos programar la comunicación.
 - Es necesario definir el protocolo, los formatos de peticiones y respuestas, etc.
 - Operaciones para abrir / cerrar conexión y leer / escribir en el socket.

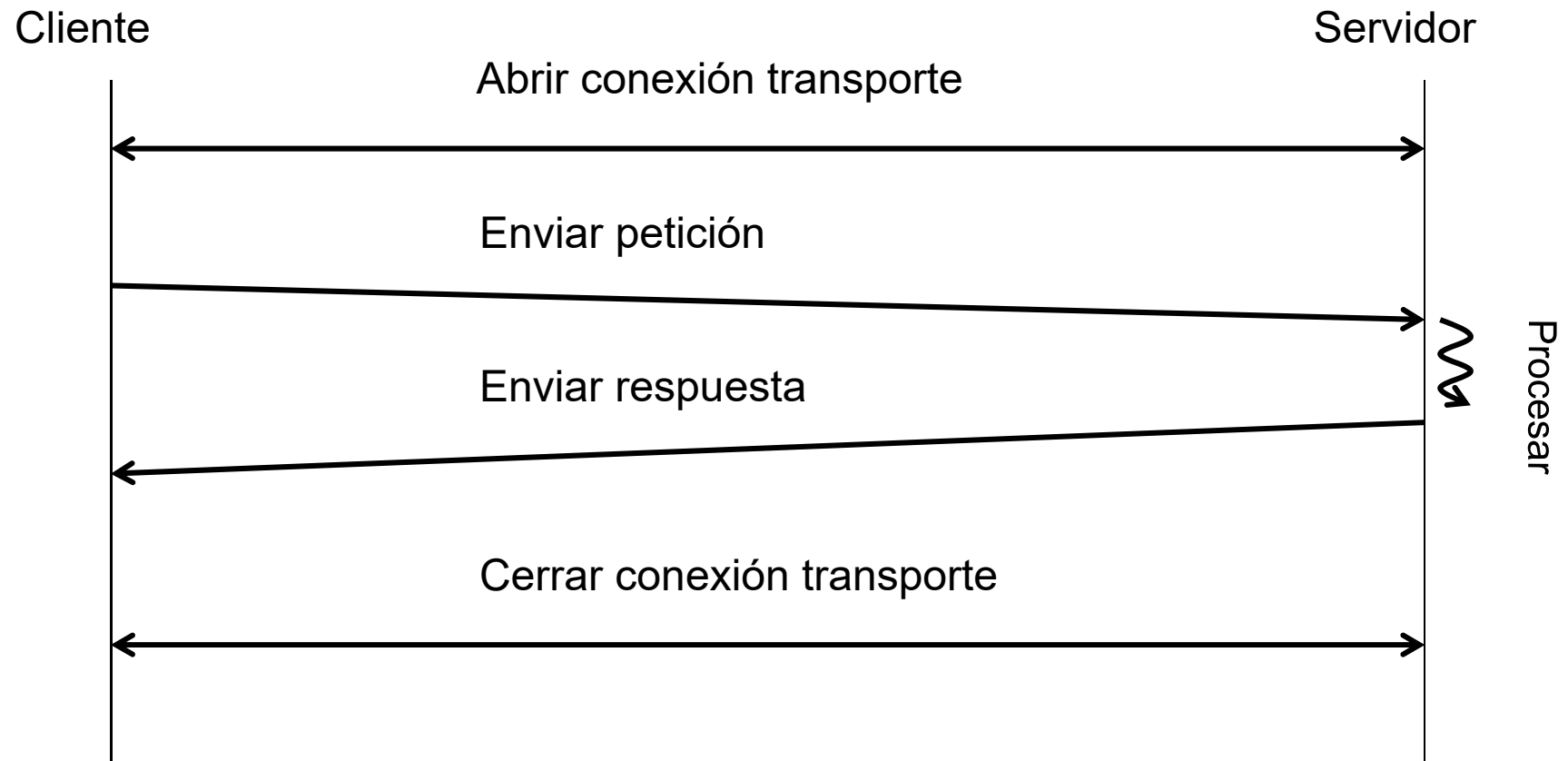
HTTP

- HyperText Transfer Protocol
- Protocolo nivel de aplicación para sistemas de información distribuidos
- Soporte a diversos tipos de información
- Versión actual: 1.1 (rfc7230 – rfc7237, 2014)
 - Existe la versión HTTP 2 (rfc7540, Mayo 2015)
 - https://en.wikipedia.org/wiki/Comparison_of_web_browsers#Protocol_support

Características de HTTP

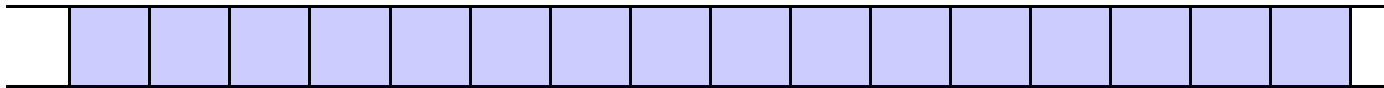
- Protocolo sin estado
- Tipo pregunta/respuesta
- Mecanismos “externos” para control de estado: Cookies, sesiones
- Funciona sobre TCP
- Puerto 80 por defecto

Descripción de HTTP

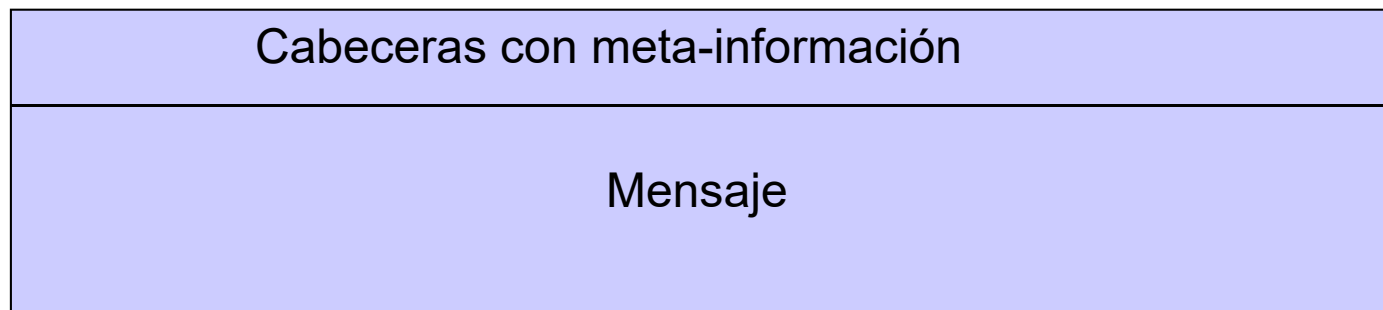


Evolución HTTP

- Versión 0.9: Tiras de *bytes* sin formato

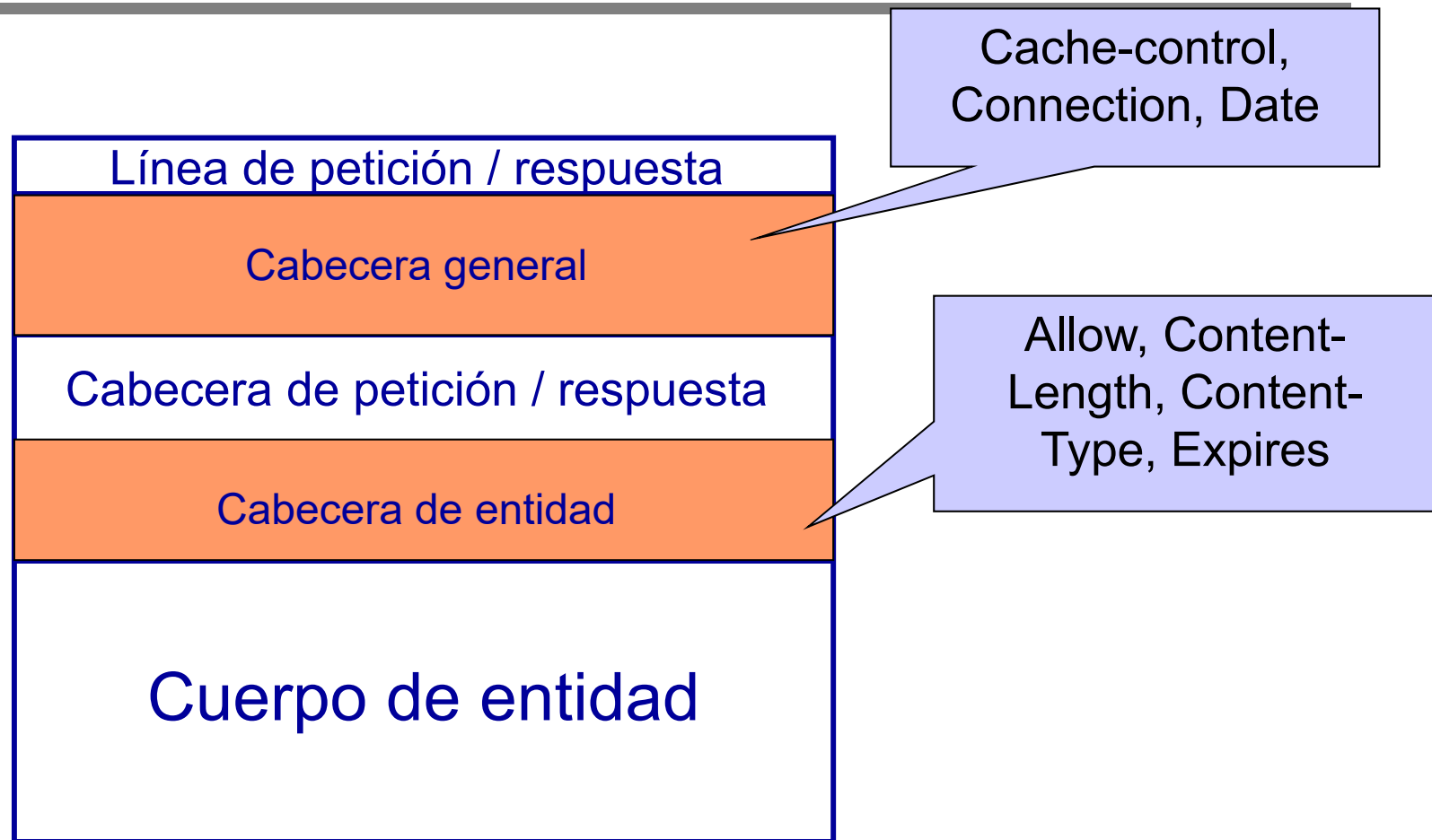


- Versión 1.0: Mensajes con formato

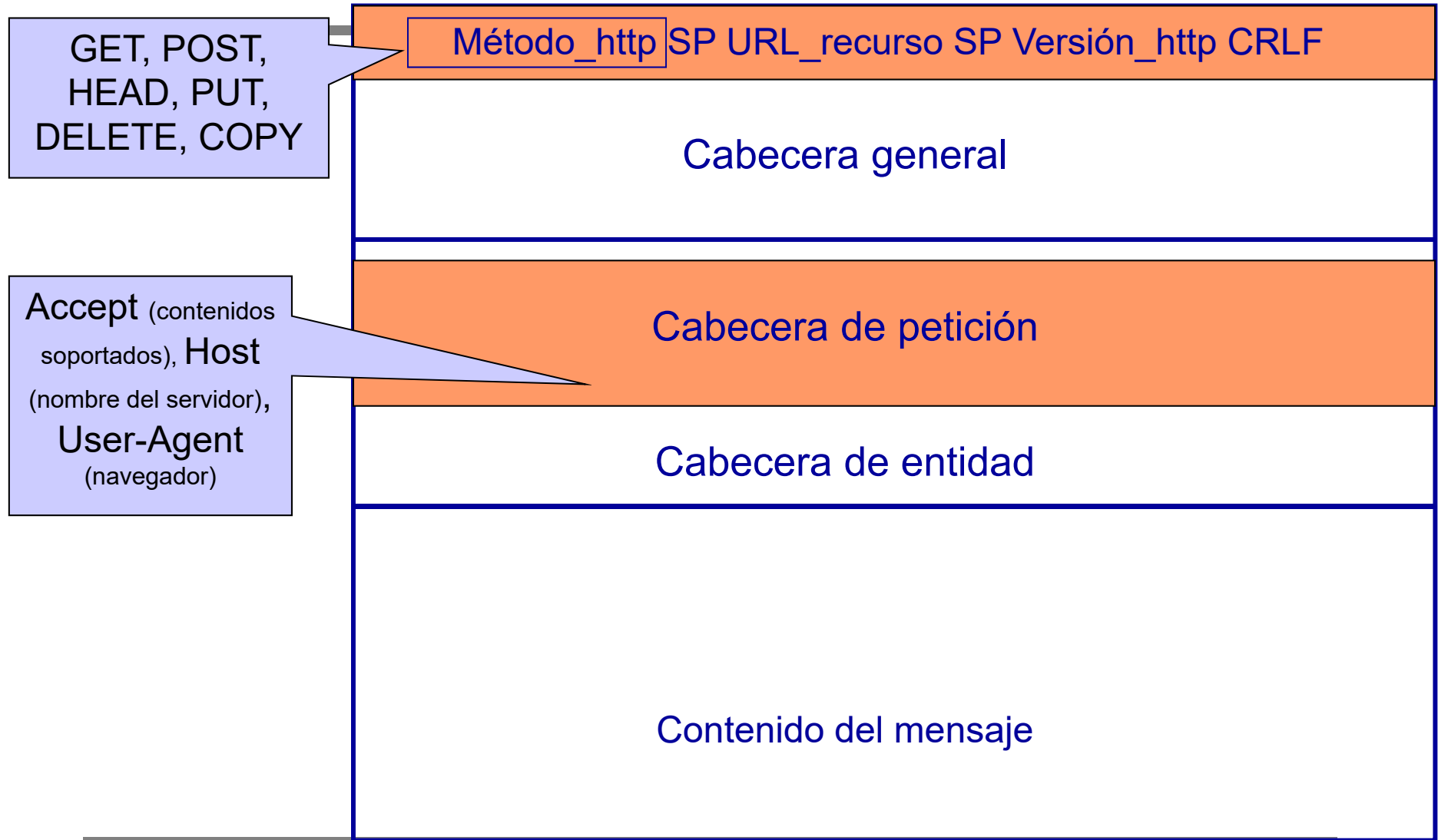


- Versión 1.1: Nuevas funcionalidades (Persistencia de conexiones, proxys, cache de los datos, etc)

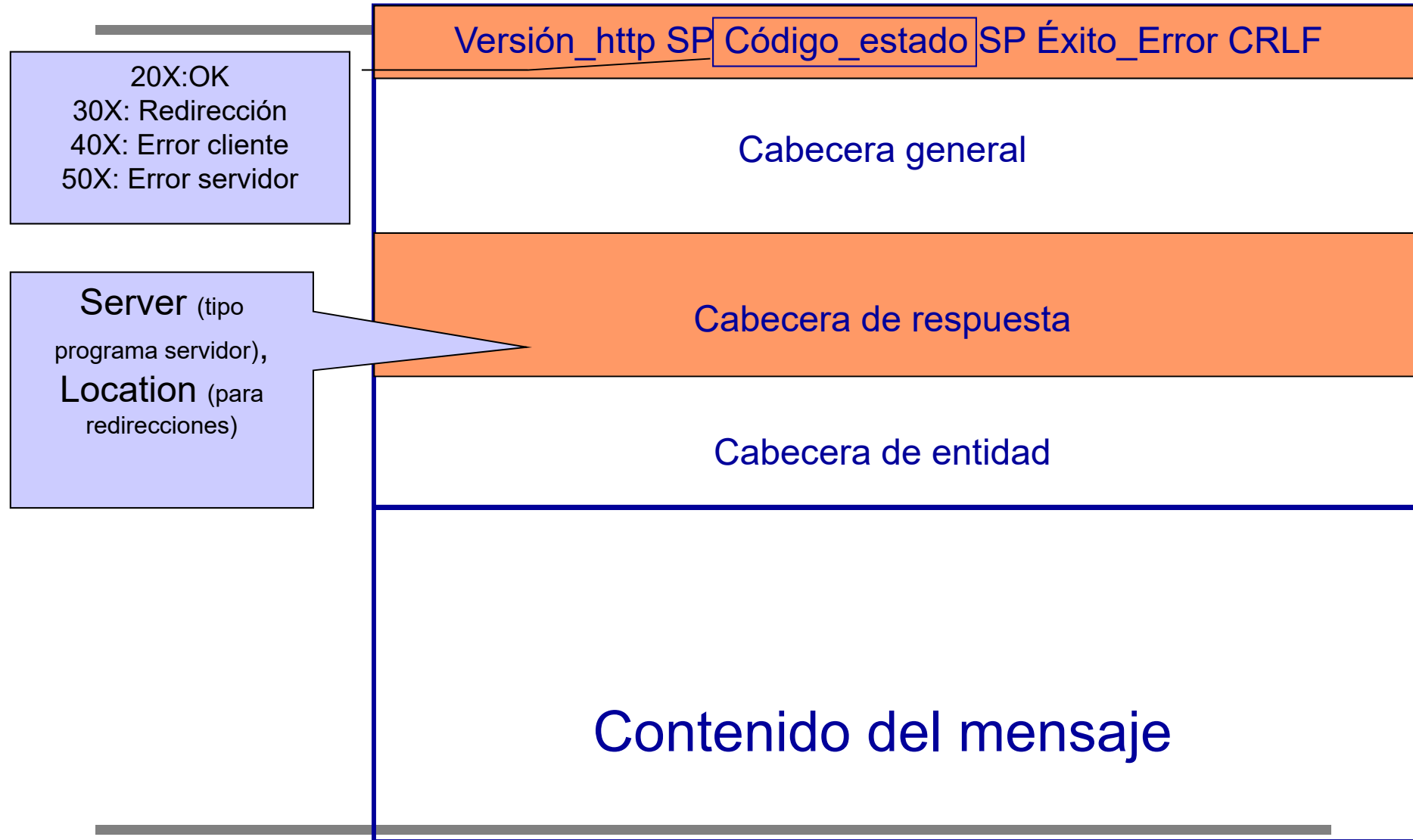
Mensajes HTTP



Mensajes HTTP: Petición



Mensajes HTTP: Respuesta



Ref: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

HTTP methods (“No modification”)

- **GET.** Requests the specified resource.
Should only retrieve data. No other effect.
- **HEAD.** Response identical to GET without the body.
- **TRACE.** Echoes back the received request.
- **OPTIONS.** Returns the HTTP methods that the server supports for the specified URL.

HTTP methods (“Modification”)

- **POST.** Submits data to be processed → update, creation.
Examples: HTML form, annotation, message, item to add to a database, ...
- **PUT.** Uploads the specified resource.
- **DELETE.** Deletes the specified resource.
- **PATCH.** Applies partial modifications to the resource.

HTTP GET Request example

GET /search?q=myBook HTTP/1.1

Host: www.google.com

User-Agent: Mozilla/5.0 ...

MIME types

Accept: text/xml,application/xml,text/html,
text/plain,image/png, ...

Accept-Language: da,en-us, ...

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8 ...

Keep-Alive: 300 Time out (in s.)

Connection: keep-alive Persistence

Referer: http://www.google.com/

HTTP GET Response example

HTTP/1.1 200 OK

Date: Fri, 17 Sep 2009 07:59:01 GMT

Server: Apache/2.0.50 (Unix) ...

Last-Modified: Tue, 24 Feb 2009 08:32:26 GMT

ETag: "ec002-afa-fd67ba80" Entity Tag

Accept-Ranges: bytes Accept range req.

Content-Length: 2810

Content-Type: text/html

... body content ...

HTTP 2 – Algunos conceptos

- Basado en SPDY (speedy)
- Métodos, códigos de estado y semántica son los mismos que en HTTP/1.1
- ¿Qué cambia?
 - La manera de enviar los datos entre cliente y servidor
 - Aparece el concepto de “Frame”
 - Los datos se envían en formato binario, no textual como hasta ahora

HTTP 2 – Algunos conceptos

- Se persigue una mejora del rendimiento:
 - Percepción de la latencia por parte del usuario final
 - Uso de la red y los recursos del servidor
 - Multiplexar varias peticiones a un servidor web sobre una única conexión TCP
- Permite que el servidor envíe respuestas al cliente (Server push) sin necesidad de petición previa
 - En HTTP 1.1 las peticiones siempre vienen del cliente

HTTP 2 - Resumen

1. One TCP connection

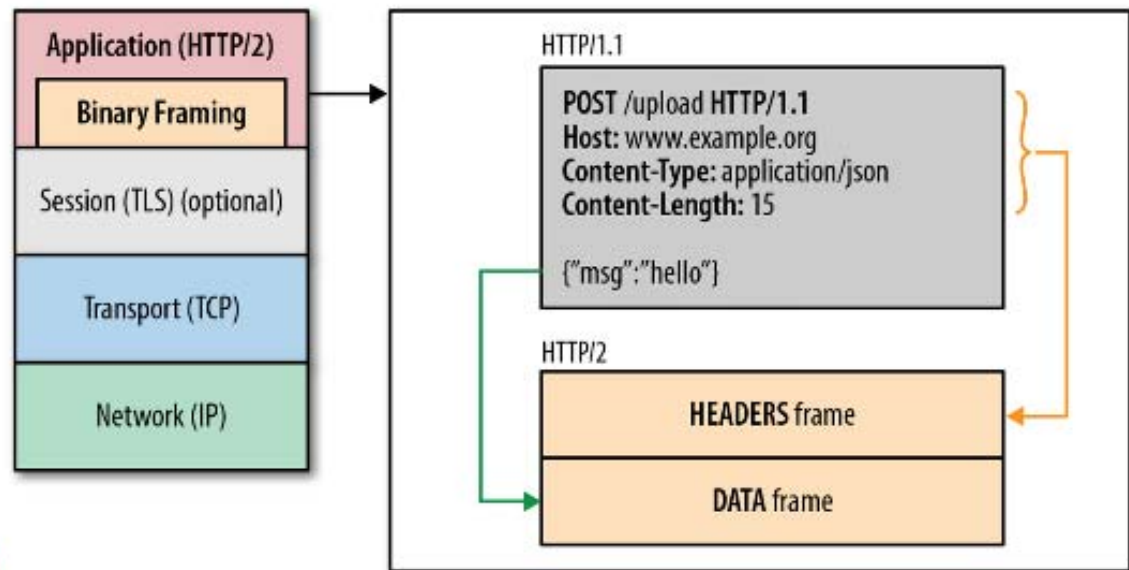
2. Request → Stream

- Streams are multiplexed
- Streams are prioritized

3. Binary framing layer

- Prioritization
- Flow control
- Server push

4. Header compression (HPACK)



HTTP 2 – Algunas referencias

- RFC HTTP/2
 - <https://tools.ietf.org/html/rfc7540>
- Artículos sobre funcionamiento HTTP/2
 - <http://resources.infosecinstitute.com/http2-faster-and-safer-web-enforcing-strong-encryption-as-the-de-facto-standard/#gref>
 - <https://blog.newrelic.com/2016/02/09/http2-best-practices-web-performance/>
 - <https://hpbn.co/http2/>
 - <https://www.smashingmagazine.com/2017/04/guide-http2-server-push/>
 - <https://bagder.gitbooks.io/http2-explained/en/part6.html>

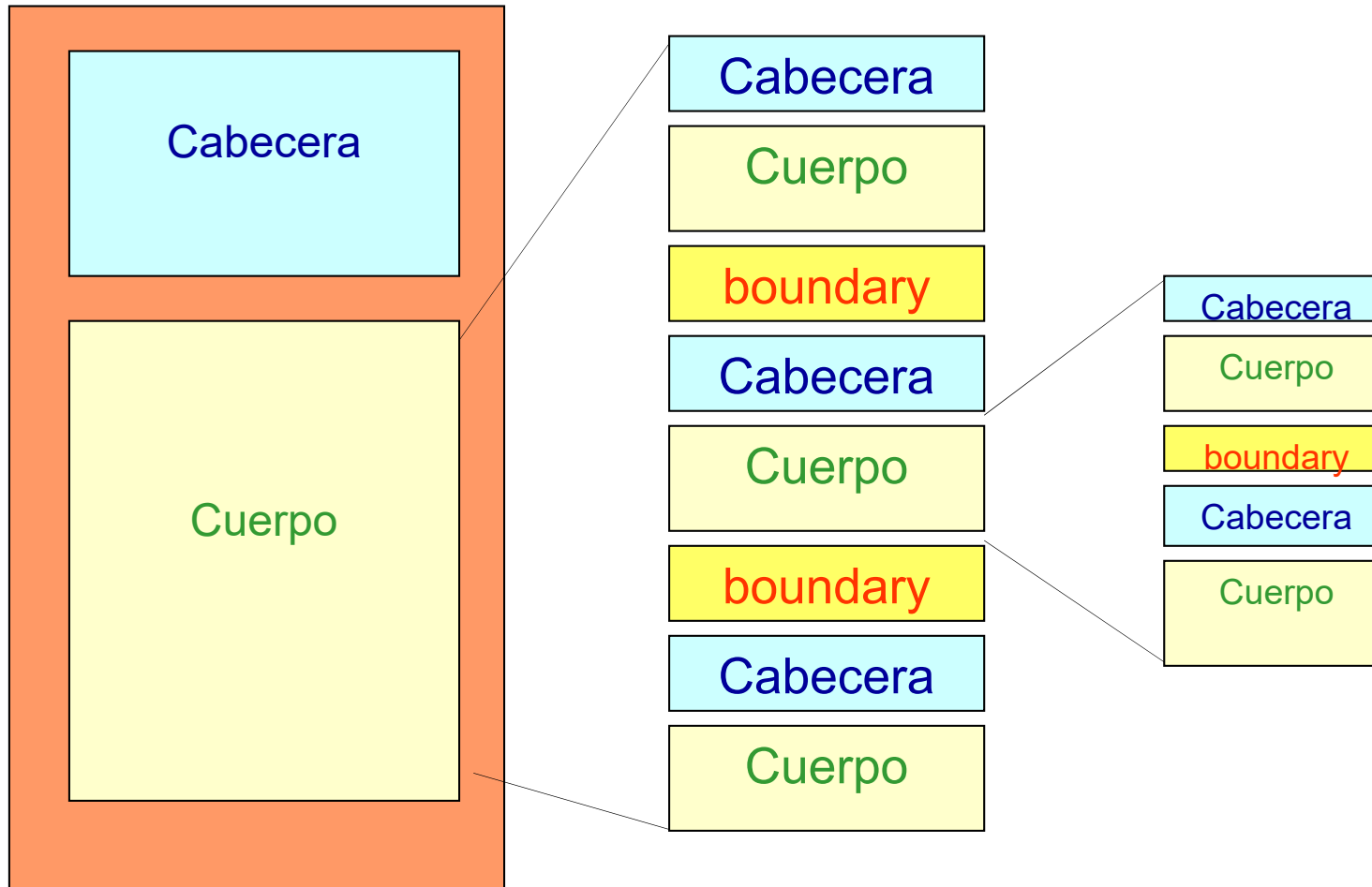
MIME, HTML, HTML5

- Formatos de información utilizados en aplicaciones de correo electrónico y web.
- Permiten incluir / enlazar distintos tipos de información.
 - Ejemplos:
 - Imágenes o documentos dentro de un mensaje MIME.
 - Páginas web que enlazan a imágenes otros documentos, vídeos, etc.
- Tutorial “vintage” de MIME:
 - <http://www2.rad.com/networks/1995/mime/mime.htm>
- RFC 1521 - MIME
 - <https://www.ietf.org/rfc/rfc1521.txt>

MIME

- MIME (Multipurpose Internet **Mail** Extensions)
 - Permite enviar contenidos con múltiples formatos
- Su uso se ha extendido a otros contextos de aplicación: Web, S.O.,
- Generalización del correo electrónico
 - Uso de mensajes para enviar distintos contenidos
 - Normalizar la codificación de información no-ASCII

Formato MIME



Tipos de contenidos MIME

- Campo Content-Type
 - Tiene dos partes: tipo / subtipo
- Tipos:
 - application, audio, image, message, multipart, text, video
- Ejemplos:
 - image/gif, image/jpeg, image/png, ...
 - text/plain, text/html, message/rfc822,
 - application/postscript, application/msword, ...

Ejemplo tipos de contenidos

- multipart/mixed

From: John Doe <example@example.com>

MIME-Version: 1.0

Content-Type: multipart/mixed; boundary="XXXXboundary text"

This is a multipart message in MIME format.

--XXXXboundary text

Content-Type: text/plain

this is the body text

--XXXXboundary text

Content-Type: text/plain;

Content-Disposition: attachment; filename="test.txt"

this is the attachment text

--XXXXboundary text--

Ejemplo tipos de contenidos

- multipart/mixed

From: John Doe <example@example.com>
MIME-Version: 1.0

Cabecera correo
Quién envía, versión de MIME

Content-Type: multipart/mixed; boundary="XXXXboundary text"
This is a multipart message in MIME format.

Organización del contenido
dentro del mensaje

--XXXXboundary text
Content-Type: text/plain
this is the body text

Primera parte, tipo texto dentro del
mensaje

--XXXXboundary text
Content-Type: text/plain;
Content-Disposition: attachment; filename="test.txt"
this is the attachment text

Segunda parte, fichero adjunto

--XXXXboundary text--

Fin contenido mensaje

From: "Senders Name" <sender@sendersdomain.com>
To: "Recipient Name" <somerecipient@recipientdomain.com>
Message-ID: <5bec11c119194c14999e592feb46e3cf@sendersdomain.com>
Date: Sat, 24 Sep 2005 15:06:49 -0400
Subject: Sample Multi-Part
MIME-Version: 1.0
Content-Type: multipart/alternative;
boundary="-----=_NextPart_DC7E1BB5_1105_4DB3_BAE3_2A6208EB099D"

-----=_NextPart_DC7E1BB5_1105_4DB3_BAE3_2A6208EB099D
Content-type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: quoted-printable

Sample Text Content
-----=_NextPart_DC7E1BB5_1105_4DB3_BAE3_2A6208EB099D
Content-type: text/html; charset=iso-8859-1
Content-Transfer-Encoding: quoted-printable

<html>
<head>
</head>
<body> <div style=3D"FONT-SIZE: 10pt; FONT-FAMILY: Arial">Sample HTML = Content</div></body>
</html>

-----=_NextPart_DC7E1BB5_1105_4DB3_BAE3_2A6208EB099D--

Cabeceras correo

From: "Senders Name" <sender@sendersdomain.com>
To: "Recipient Name" <somerecipient@recipientdomain.com>
Message-ID: <5bec11c119194c14999e592feb46e3cf@sendersdomain.com>
Date: Sat, 24 Sep 2005 15:06:49 -0400
Subject: Sample Multi-Part
MIME-Version: 1.0

Content-Type: multipart/alternative;
boundary="-----=_NextPart_DC7E1BB5_1105_4DB3_BAE3_2A6208EB099D"

Tipo de mensaje y
separación

-----=_NextPart_DC7E1BB5_1105_4DB3_BAE3_2A6208EB099D

Content-type: text/plain; charset=iso-8859-1
Content-Transfer-Encoding: quoted-printable

Sample Text Content

Primer elemento, texto

-----=_NextPart_DC7E1BB5_1105_4DB3_BAE3_2A6208EB099D

Content-type: text/html; charset=iso-8859-1
Content-Transfer-Encoding: quoted-printable

Segundo elemento, html

<html>
<head>
</head>
<body> <div style=3D"FONT-SIZE: 10pt; FONT-FAMILY: Arial">Sample HTML = Content</div></body>
</html>

-----=_NextPart_DC7E1BB5_1105_4DB3_BAE3_2A6208EB099D--

Ejemplo tipos de contenidos

- multipart/form-data: Formato de envío de formularios
 - boundary
 - Content-Type: multipart/form-data, boundary=AaB03x
 - Content-Disposition
 - AaB03x
 - Content-Disposition: form-data; name="fichero";
filename="file1.html"
 - Content-Type: text/html
 - ... Contenido del fichero file1.html ...
 - AaB03x--

Algunas cabeceras MIME

- MIME-Version
- Content-Type
- Content-Transfer-Encoding
- Content-ID
- Content-Disposition
- Content-Description
- Content-Language

- Algunos ejemplos más:
 - <http://www.oreilly.com/openbook/mh/mulmes.htm>

HTML

- Definido por W3C para la creación de páginas web
- Basado en etiquetas (tag): `<tag> ... </tag>` ó `<tag/>`
- Definen la estructura lógica del documento
 - No la presentación, que la decide el navegador
- Cada vez más complejo y estructurado
- Ejercicio
 - ¿Cómo se utilizan las etiquetas HTML?
 - ¿Y los atributos?

HTML

- Solución
 - Etiquetas
 - El contenido va delimitado por un inicio y un final de etiqueta.
 - `<etiqueta> Contenido </etiqueta>`
 - Atributos
 - Sirven para añadir características a una etiqueta (semántica)
 - `<etiqueta atrib = "valor"> Contenido </etiqueta>`

HTML

- Estructura de un documento HTML
 - `<html>` `</html>` Inicio / Final de documento
 - `<head>` `</head>` Elementos de cabecera del documento, que no se muestran en el navegador. Incluyen cosas como el título o metadatos sobre la página (hora de consulta, si se puede hacer caché, etc.).
 - `<body>` `</body>` Cuerpo del documento HTML, lo que se muestra en el navegador.
- Referencia:
 - <http://www.w3schools.com/html/default.asp>

HTML

- Javascript
 - Añadir scripting en el cliente (navegador)
 - Validación datos en navegador
 - Minimizar conexiones con servidor
 - Ventanas de avisos
 - Cada vez más complejo y completo, permite trasladar parte de la aplicación web al cliente
 - Otros ejemplos de uso de javascript: node.js, angularJs
 - Referencias:
 - <http://nodejs.org/>
 - <http://angularjs.org/> (Google)

HTML

- Cascade Style Sheets (CSS)
 - Hojas de estilo
 - Se utilizan para dar formato a los elementos HTML
 - Permiten separar claramente la estructura del documento, del estilo (gráfico)
 - Aplican la información en cascada, de ahí su nombre
 - Referencia:
 - <http://www.w3schools.com/css/default.asp>

HTML5

- Nueva versión HTML definida por W3C y Web Hypertext Application Technology Working Group (WHATWG)
- Todavía en proceso de creación (no está finalizado)
- Los navegadores soportan algunas de sus nuevas características
- Reglas básicas
 - Basado en HTML, CSS, DOM, Javascript
 - Reducir la necesidad de plug-ins externos
 - Mejorar la gestión de errores
 - Más markup para reducir scripting
 - Independiente de dispositivo

HTML5

- Nuevos elementos:

- <canvas> para dibujar elementos 2D
- <video> y <audio> para gestión de archivos multimedia
- Soporte para almacenamiento local
- Elementos específicos del contenido, como <article>, <footer>, <header>, <nav>, <section>
- Nuevos controles de formulario, como calendar, date, time, email, url, search
- Ejemplo

```
<video width="320" height="240" controls="controls">
```

```
  <source src="movie.mp4" type="video/mp4" />
```

```
  <source src="movie.ogv" type="video/ogg" />
```

```
  Your browser does not support the video tag.
```

```
</video>
```

HTML5

- Referencias:
 - <http://diveintohtml5.info/introduction.html>
 - <https://html.spec.whatwg.org/multipage/>

XML

- XML: eXtensible Markup Language
- Designed to transport and store data
 - HTML used to display data
 - HW&SW-independent
- XML
 - To carry data. To process it automatically.
 - **Users** must define their own **tags**.
 - **Users**: Private users and SDO (*Standards Developing Organizations*).

XML structure & syntax

- XML:
 - Tree structure.
 - Elements, attributes & text.
 - Example:

```
<book category="COOKING">  
    <title lang="en">Everyday Italian</title>  
    <author>Giada De Laurentiis</author>  
    ...  
</book>
```

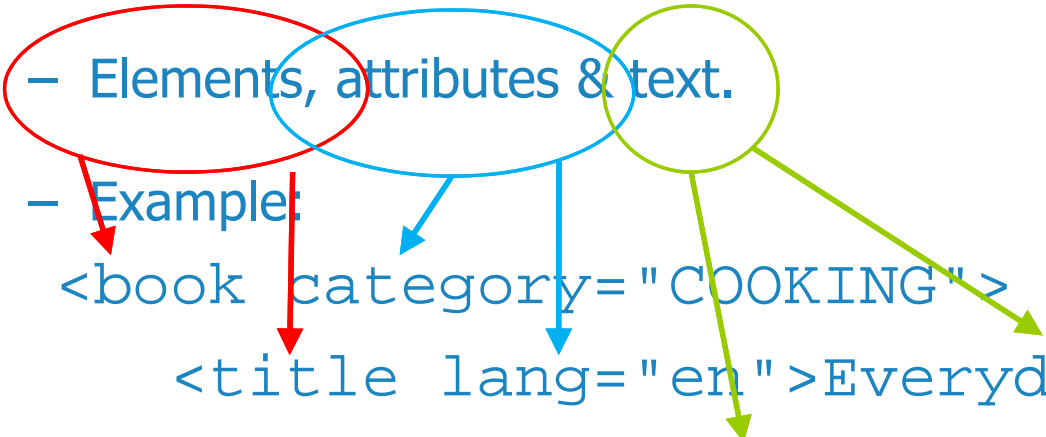
XML structure & syntax

- XML:

- Tree structure

- Elements, attributes & text.

- Example:



```
<book category="COOKING">  
  <title lang="en">Everyday Italian</title>  
  <author>Giada De Laurentiis</author>  
  ...  
</book>
```

XML structure & syntax

- First line (example):
`<?xml version="1.0" encoding="ISO-8859-1"?>`
- XML simple syntax:
 - Closing tag mandatory.
 - Tags are case sensitive.
 - Elements could be nested:
`<a> ... <c>...</c> `
(a parent, b , c childs , b , c siblings).
 - Root element needed (Is unique and defines the document type).

XML structure & syntax

- XML simple syntax (cont.):

- ...

- Attribute values must be quoted.

- Entity references.

- Examples:

- predefined entity

- `"` ;

- `&` ;

- represented character

- `"`

- `&`

- Comments:

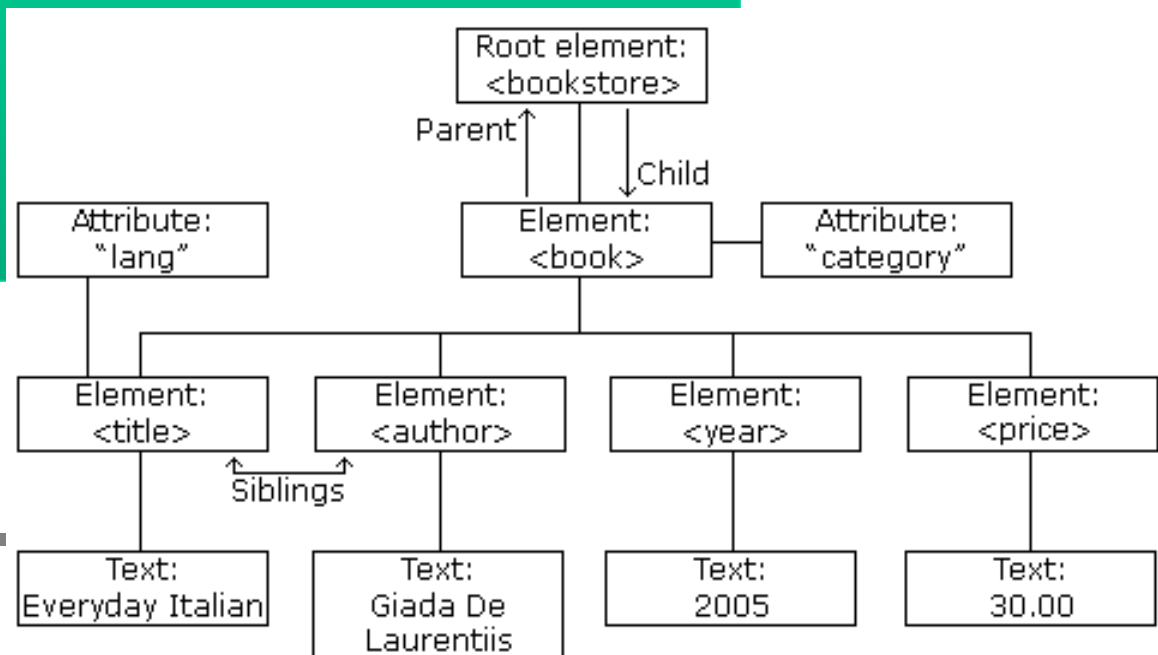
- `<!-- ... -->`

XML other issues

- Attributes vs. Elements: *Design decision*
- Name conflicts:
 - *Namespaces*
 - Allow differentiating element names defined by different developers/standards.
 - xmlns attribute (in the start tag of an element):
`xmlns:prefix="URI"`
 - URLs often used as an easy way to define *unique* namespaces
- How to define tags and structure: Schemas
 - Examples ...

XML simple example

```
<bookstore>
  <book category="COOKING">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="CHILDREN">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  ...
</bookstore>
```



XML: Idea of Schema

- XML Schema Definition, **XSD**

- Content of the file “**note.xsd**”:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

<http://www.w3schools.com/xml/>

namespace where the **schema** is defined,
the namespace should be prefixed xs.

root element

complexType: contains other elements

sequence: child elements must
appear in the same order

- Reference to the XSD defined in “**note.xsd**”:

```
<?xml version="1.0"?>
<note xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="note.xsd">
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

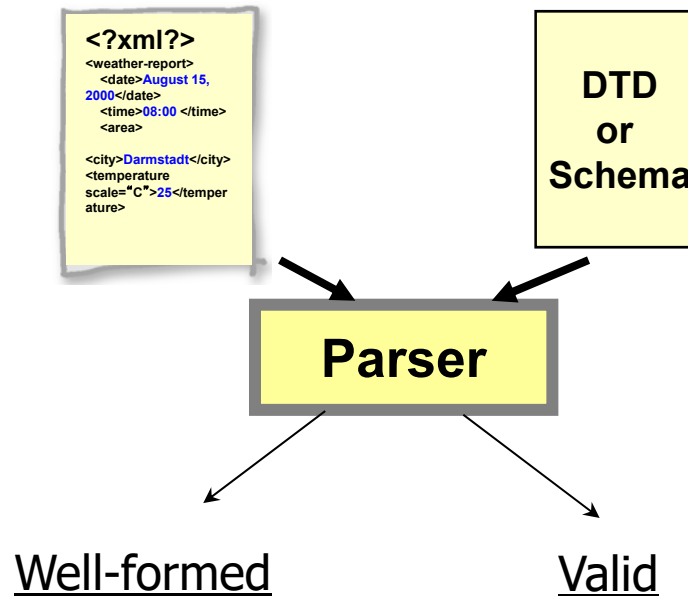
XSD schema defined in
location “note.xsd”

XML: validity

- Document XML “well-formed”:
 - A document that **satisfies syntax rules** of XML

Document XML “valid”:

- A well-formed document that also **conforms to a set of rules specified in a restrictions document**

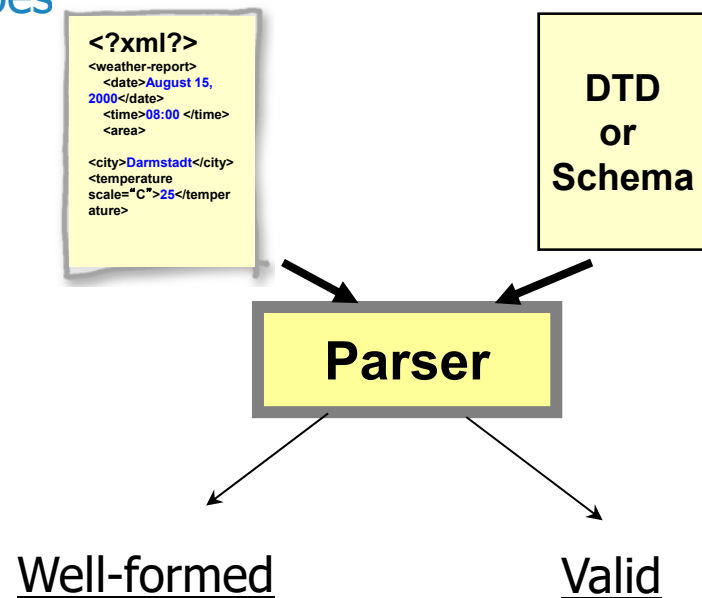


XML: validity

- Different formats for restrictions/rules documents/files:

1. DTD (Document Type Definition):
 - 1st generation: based on SGML,
 - syntax not XML, few data types

2. **XML Schema:**
 - XML format,
 - more data types,
 - more restrictions



Schema Languages

- Define the valid structure (grammar) of a set of XML documents (a XML application)
- Initially DTD (Document Type Definition):
 - Simple but limited
- Now XML Schema
 - Higher expressiveness, but very complex

Example with a DTD

```
<?xml version="1.0"?>
```

```
<!DOCTYPE film "film.dtd">
```

```
<films>
```

```
  <film>
```

```
    <title id="1">Tootsie</title>
```

```
    <genre>&COM;</genre>
```

```
    <year>1982</year>
```

```
  </film>
```

```
</films>
```

```
<!DOCTYPE film [
```

```
  <!ENTITY COM "Comedy">
```

```
  <!ENTITY SF "Science Fiction">
```

```
  <!ELEMENT films (film+)>
```

```
  <!ELEMENT film (title,genre,year)>
```

```
  <!ELEMENT title (#PCDATA)>
```

```
  <!ATTLIST title
    id ID>
```

```
  <!ELEMENT genre (#PCDATA)>
```

```
  <!ELEMENT year (#PCDATA)>
```

Example with Schema

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.films.org"
  xmlns=http://www.films.org>
  <xsd:element name="films">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="film" type="filmType" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="filmType">
    <xsd:sequence>
      <xsd:element name="title" type="xsd:string"/>
      <xsd:element name="genre" type="xsd:string"/>
      <xsd:element name="year" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Example of instance

<?xml version="1.0"?>

Referencia al schema

<films xmlns="http://www.films.org"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation= "http://www.films.org films.xsd">

<film>

<title>Tootsie</Title>

<genre>Comedy</genre>

<year>1982</year>

</film>

</films>

No hemos definido atributo id y
no hay entidad para definir el
género

XML Schema & namespaces

`<films xmlns="http://www.films.org"` (1)
`xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` (2)
`xsi:schemaLocation="http://www.films.org films.xsd">` (3)

- 1) Names without prefix belong to the *default* namespace
“http://www.films.org”
- 2) Names with prefix “xsi” (only the schemaLocation) belong to
the namespace “http://www.w3.org/2001/XMLSchema-
instance”
- 3) The schema corresponding to the namespace
“http://www.films.org” is in file “films.xsd”

XML Schema: Types

Simple Types (Datatypes) – Primitive

string	<i>any Unicode string</i>
boolean	true, false, 1, 0
decimal	3.1415
float	6.02214199E23
double	42E970
dateTime	2004-09-26T16:29:00-05:00
time	16:29:00-05:00
date	2004-09-26
hexBinary	48656c6c6f0a
base64Binary	SGVsbG8K
anyURI	http://www.brics.dk/ixwt/
QName	rcp:recipe, recipe
...	

XML Schema: Types

Derivation of Simple Types – Restriction

Constraining facets:

- length
- minLength
- maxLength
- pattern
- enumeration
- whitespace
- maxInclusive
- maxExclusive
- minInclusive
- minExclusive
- totalDigits
- fractionDigits


XML Schema: Types

Examples

```
<simpleType name="score_from_0_to_100">  
  <restriction base="integer">  
    <minInclusive value="0"/>  
    <maxInclusive value="100"/>  
  </restriction>  
</simpleType>
```

```
<simpleType name="percentage">  
  <restriction base="string">  
    <pattern value="([0-9]|[1-9][0-9]|100)%"/>  
  </restriction>  
</simpleType>
```

regular expression



XML Schema: Types

Built-In Derived Simple Types

- `normalizedString`
- `token`
- `language`
- `Name`
- `NCName`
- `ID`
- `IDREF`
- `integer`
- `nonNegativeInteger`
- `unsignedLong`
- `long`
- `int`
- `short`
- `byte`
- ...

XML Schema: Types

Complex Types with Complex Contents

- Content models as regular expressions:
 - Element reference `<element ref="name"/>`
 - Concatenation `<sequence> ... </sequence>`
 - Union `<choice> ... </choice>`
 - All `<all> ... </all>`
 - Element wildcard: `<any namespace="..."
 processContents="..." />`
 - Attribute reference: `<attribute ref="..." />`
 - Attribute wildcard: `<anyAttribute namespace="..."
 processContents="..." />`
- Cardinalities: minOccurs, maxOccurs, use
- Mixed content: mixed="true"

XML Schema: Types

Example

```
<element name="order" type="n:order_type"/>

<complexType name="order_type" mixed="true">
  <choice>
    <element ref="n:address"/>
    <sequence>
      <element ref="n:email"
        minOccurs="0" maxOccurs="unbounded"/>
      <element ref="n:phone"/>
    </sequence>
  </choice>
  <attribute ref="n:id" use="required"/>
</complexType>
```

XML Schema: Types

Complex Types with Simple Content

```
<complexType name="category">
  <simpleContent>
    <extension base="integer">
      <attribute ref="r:class"/>
    </extension>
  </simpleContent>
</complexType>

<complexType name="extended_category">
  <simpleContent>
    <extension base="n:category">
      <attribute ref="r:kind"/>
    </extension>
  </simpleContent>
</complexType>

<complexType name="restricted_category">
  <simpleContent>
    <restriction base="n:category">
      <totalDigits value="3"/>
      <attribute ref="r:class" use="required"/>
    </restriction>
  </simpleContent>
</complexType>
```

XML Schema: Types

Global vs. Local Descriptions

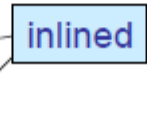
Global (toplevel) style:

```
<element name="card"
  type="b:card_type"/>
<element name="name"
  type="string"/>

<complexType name="card_type">
  <sequence>
    <element ref="b:name"/>
    ...
  </sequence>
</complexType>
```

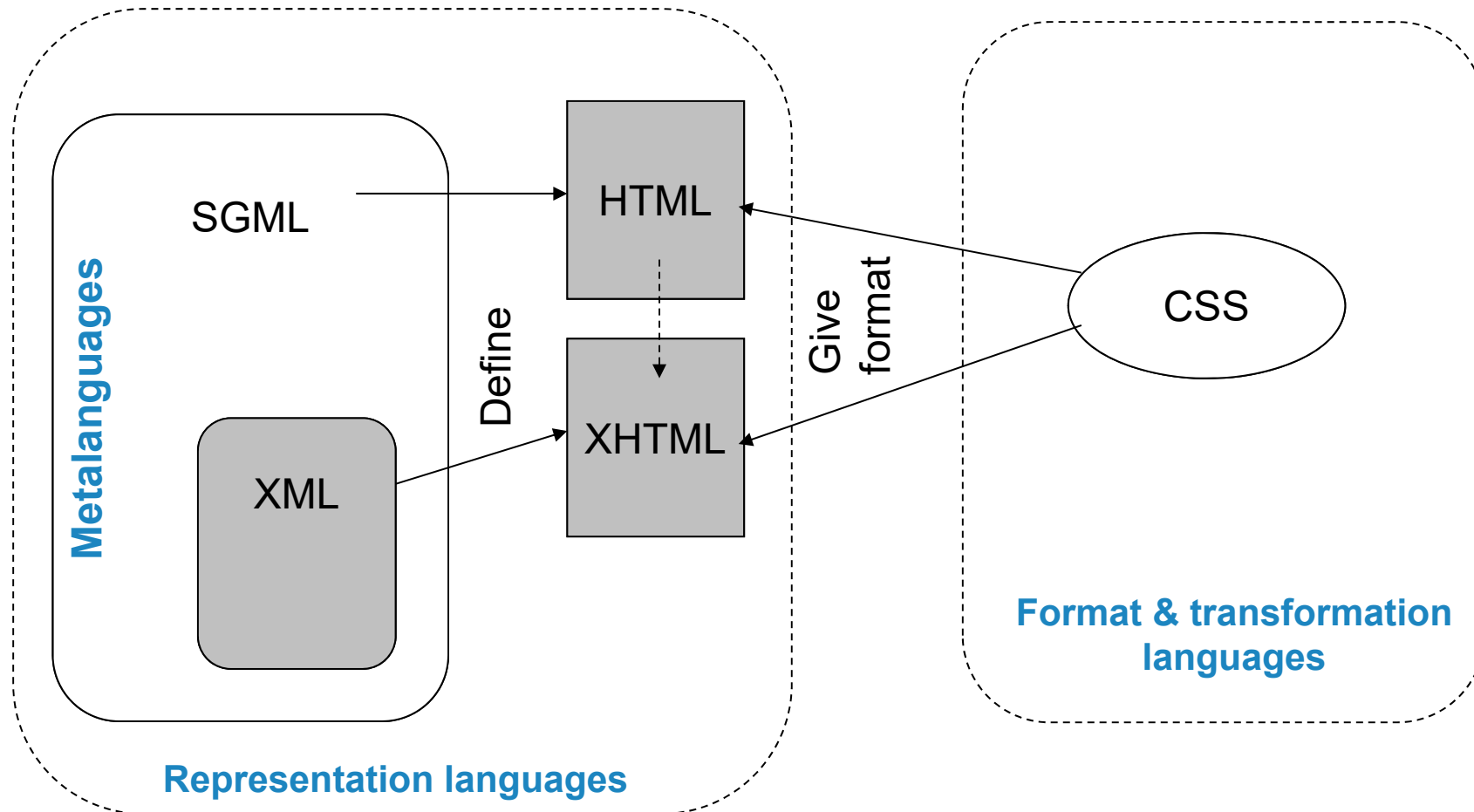
Local (inlined) style:

```
<element name="card">
  <complexType>
    <sequence>
      <element name="name"
        type="string"/>
      ...
    </sequence>
  </complexType>
</element>
```



The diagram shows a blue box labeled "inlined" with two arrows pointing to the `<complexType>` and `<sequence>` tags in the local style example.

Languages relationships



Aplicacions Distribuïdes

Silvia Llorente
silviall@ac.upc.edu

Distributed Multimedia Applications Group
Departament d'Arquitectura de Computadors