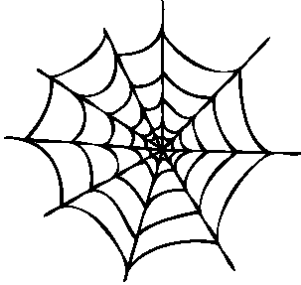
Universitat Autònoma de Barcelona



TECNOLOGIES DE DESENVOLUPAMENT PER A INTERNET I WEB

GRAU D'ENGINYERIA INFORMÀTICA - CURS 2019/2020

TEMA 2 – *EL PROTOCOL HTTP*

2. EL PROTOCOL HTTP

2.1 – Arquitectura

- *Arquitectura client/servidor*
- *Pila de protocols TCP/IP*
- *Característiques*
- *Format dels missatges*
- *Test i prova*

2.2 – Mecanismes

2.3 – Maneig de l'estat

2.1 Arquitectura. Arquitectura client/servidor

Cada pàgina web té un únic nom per identificar-la. El nom s'anomena *Uniform Resource Locator* (URL), i comença amb l'esquema de transmissió utilitzat per aquell URL. L'esquema de transmissió especifica quin format tindrà la URL. Quan l'esquema és http, la URL té la següent forma:

http: // [user:password] hostname [:port] / path [;parameters] [?query] [#bookmark]

- La URL pot ser absoluta o relativa, depenent de si el servidor és implícitament conegut.

- Si es demana el document nul (/), el servidor s'encarrega de respondre la pàgina adequada.

- *Atenció!* no confondre URL amb Uniform Resource Name (URN), que especifica un recurs però no la forma d'accedir-hi, o amb Uniform Resouce Identified (URI), que inclou la URL i la URN.

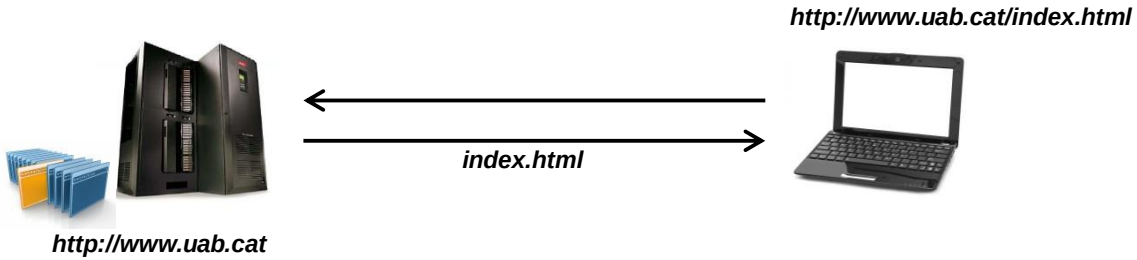
¡PREGUNTA D'EXAMEN!

Si el port 80 s'utilitza per les connexions HTTP, el servidor pot només respondre a un client a la vegada?

¡PREGUNTA D'EXAMEN!

Quina diferència hi ha entre els paràmetres i la query d'una URL?

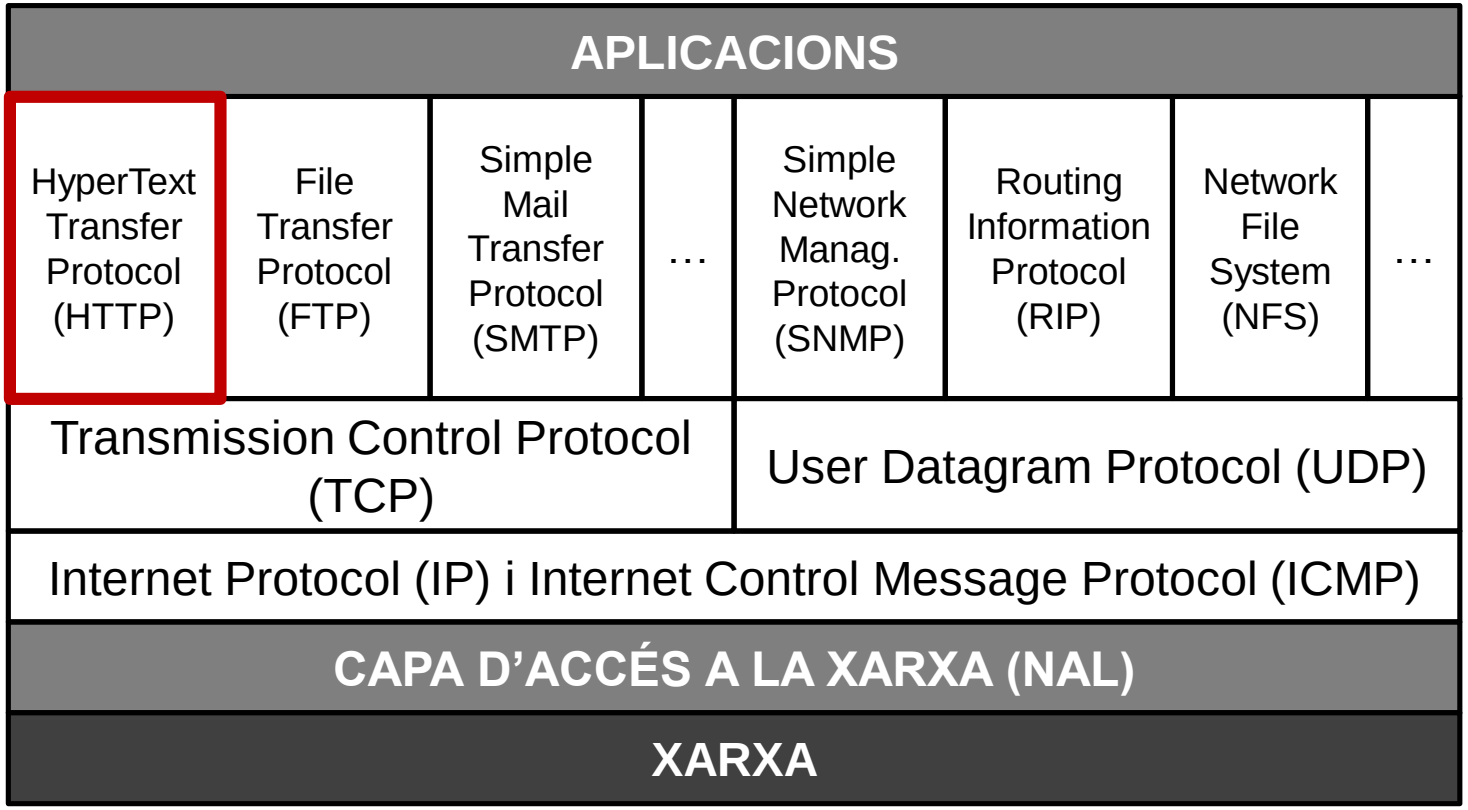
2.1 Arquitectura. Arquitectura client/servidor



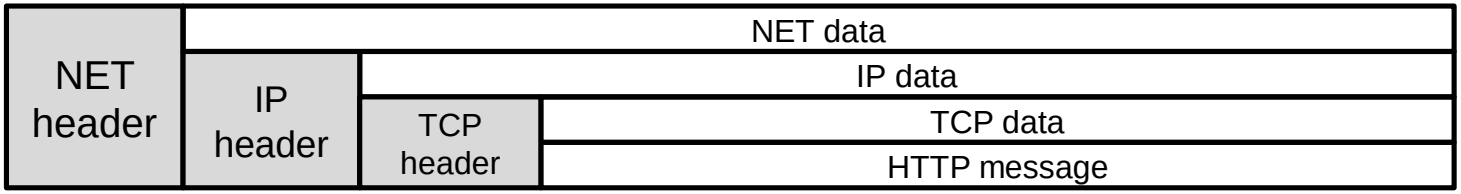
Funcionament (simplificat):

1) L'usuari especifica una URL en el navegador.
2) El navegador esdevé un client HTTP obrint una connexió utilitzant el protocol HTTP cap al servidor especificat en l'URL.
3) El servidor rep la petició del client i comprova que el document HTML existeixi en la seva biblioteca.
4) Si el document existeix, el retorna al client i s'acaba la connexió.
5) Si no existeix, retorna un codi d'error i s'acaba la connexió .

2.1 Arquitectura. Pila de protocols TCP/IP



missatge HTTP



2.1 Arquitectura. Característiques

- *Nivell d'aplicació:* funciona a nivell d'aplicació i assumeix una connexió orientada a la connexió fiable i estable.
- *Petició/resposta:* un cop la sessió de transport s'estableix, el navegador envia una petició al servidor, i aquest la respon.
- *Sense estat:* cada petició HTTP és auto-continguda. El servidor no manté una història de les pàgines servides.
- *Transmissió bidireccional:* en general, el navegador demana un document i el servidor el serveix, però HTTP també permet enviar informació des del client (e.g., els formularis).
- *Capacitat de negociació:* el servidor i client poden negociar certs aspectes com el conjunt de caràcters utilitzat.
- *Suport per "caching":* per millorar el temps de resposta, el navegador pot mantenir una cache dels documents descarregats.
- *Suport per intermediaris:* es permet que al llarg del camí hi hagi servidors "proxy" que emmagatzemen pàgines dels servidors i responen als navegadors.

¡PREGUNTA D'EXAMEN!

El protocol HTTP no té estat, però a la pila de protocols està per sobre TCP, que sí té estats. Què vol dir això?

¡PREGUNTA D'EXAMEN!

*Si els servidors HTTP **no** mantenen una història de les peticions rebudes, com es poden establir les sessions?*

2.1 Arquitectura. Format dels missatges

<START LINE>
<MESSAGE HEADERS>
<EMPTY LINE>
<MESSAGE BODY>
<MESSAGE TRAILERS>

"Start line" – indica si el missatge és una petició a un servidor o una resposta cap a un client, i de quin tipus de petició/resposta es tracta.

"Message headers" – hi ha múltiples "headers" organitzats en diferents grups. Pràcticament tots són opcionals excepte el "host header".

"Message body" – és opcional ja que només es necessita en alguns missatges. Conté el conjunt d'informació que s'ha d'intercanviar entre el client i el servidor, com missatges d'error a una petició o, més comunament, un fitxer o un recurs del servidor. En general, els recursos inclosos en el "body" s'anomenen entitats i es poden transmetre com text o bé com a informació binària.

"Message trailers" – ja que el "body" pot incloure vàries entitats, els "trailers" permeten identificar quan s'acaba una i comença la següent.

2.1 Arquitectura. Format dels missatges

<START LINE>
<MESSAGE HEADERS>
<EMPTY LINE>
<MESSAGE BODY>
<MESSAGE TRAILERS>

Format de les peticions HTTP

Les peticions les genera el client a partir d'una acció directa de l'usuari (e.g., accés a una URL), o bé a partir d'accions indirectes al navegar per una web (e.g., accés a una imatge de la web navegada).

"Start-line"

<METHOD> <REQUEST-URI> <HTTP-VERSION>

- GET: demana una URI al servidor. És el més comú. La resposta és condicional i pot dependre de "headers" com "*If-Modified-Since*".

- POST: permet al client enviar informació al servidor generada a partir d'una pàgina dinàmica o un formulari. La URI és el programa encarregat de processar la informació.

- HEAD: és idèntic al GET però es demana que només s'enviï les capçaleres i la longitud de les entitats de la URI. És útil per deixar decidir al client si vol tota la URI o no.

- OPTIONS, PUT, DELETE, TRACE

2.1 Arquitectura. Format dels missatges

<START LINE>
<MESSAGE HEADERS>
<EMPTY LINE>
<MESSAGE BODY>
<MESSAGE TRAILERS>

Format de les peticions HTTP

Les peticions les genera el client a partir d'una acció directa de l'usuari (e.g., accés a una URL), o bé a partir d'accions indirectes al navegar per una web (e.g., accés a una imatge de la web navegada).

"Start-line"

<METHOD> <REQUEST-URI> <HTTP-VERSION>

No és la mateixa especificada per l'usuari, sinó que es divideix. Per exemple, la URL *http://www.uab.cat:8080/chat/chatroom.php* genera

GET /chat/chatroom.php HTTP/1.1

Host: www.uab.cat:8080

2.1 Arquitectura. Format dels missatges

<START LINE>
<MESSAGE HEADERS>
<EMPTY LINE>
<MESSAGE BODY>
<MESSAGE TRAILERS>

Format de les peticions HTTP

Les peticions les genera el client a partir d'una acció directa de l'usuari (e.g., accés a una URL), o bé a partir d'accions indirectes al navegar per una web (e.g., accés a una imatge de la web navegada).

"Start-line"

<METHOD> <REQUEST-URI> <HTTP-VERSION>

És de la forma *HTTP/X.X* on X.X indica la versió de HTTP.

Existeixen les versions 0.9, 1.0, 1.1 i 2.0 essent la 1.1 la més comunament utilitzada, de moment.

2.1 Arquitectura. Format dels missatges

<START LINE>
<MESSAGE HEADERS>
<EMPTY LINE>
<MESSAGE BODY>
<MESSAGE TRAILERS>

Format de les peticions HTTP

"Message-headers"

- *"General Headers":* es refereixen al mateix missatge i no al seu contingut. Serveixen per controlar el seu processament o proveir d'informació extra.
- *"Request Headers":* proveeixen més detalls de la natura de la petició i permeten al client controlar millor com es realitza la petició. Permeten fer peticions condicionals.
- *"Entity Headers":* descriuen les entitats de la petició, si n'hi ha.

2.1 Arquitectura. Format dels missatges

<START LINE>
<MESSAGE HEADERS>
<EMPTY LINE>
<MESSAGE BODY>
<MESSAGE TRAILERS>

Format de les peticions HTTP

START-LINE
GENERAL-HEADER
REQUEST-HEADER
EMPTY-LINE

GET /index.html HTTP/1.1
Date: Thu, 6 Sept 2012 10:57:34 GMT
Connection: close
Host: www.uab.es
Accept: text/html, text/plain
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows 7)

2.1 Arquitectura. Format dels missatges

<START LINE>
<MESSAGE HEADERS>
<EMPTY LINE>
<MESSAGE BODY>
<MESSAGE TRAILERS>

Format de les respostes HTTP

Cada petició a un servidor web genera un missatge de resposta. En general, la resposta conté una o vàries entitats corresponent a la URI demanada.

"Start-line"

<HTTP-VERSION> <STATUS-CODE> <REASON-PHRASE>

Versió HTTP utilitzada en forma *HTTP/X.X*

2.1 Arquitectura. Format dels missatges

<START LINE>
<MESSAGE HEADERS>
<EMPTY LINE>
<MESSAGE BODY>
<MESSAGE TRAILERS>

Format de les respostes HTTP

Cada petició a un servidor web genera un missatge de resposta. En general, la resposta conté una o vàries entitats corresponent a la URI demanada.

"Start-line"

<HTTP-VERSION> <STATUS-CODE> <REASON-PHRASE>

S'informa al client del resultat de la petició. L'"status-code" és un número de 3 dígits que indica el resultat formal a l'aplicació que processa el client. La "reason-phrase" és un text addicional que pot ser mostrat a l'usuari perquè sàpiga com el servidor ha respost.

2.1 Arquitectura. Format dels missatges

<START LINE>
<MESSAGE HEADERS>
<EMPTY LINE>
<MESSAGE BODY>
<MESSAGE TRAILERS>

Format de les respostes HTTP

Cada petició a un servidor web genera un missatge de resposta. En general, la resposta conté una o vàries entitats corresponent a la URI demanada.

"Start-line"

<HTTP-VERSION> <STATUS-CODE> <REASON-PHRASE>

Els "status-code" estan organitzats com:

1xx – missatges informatius

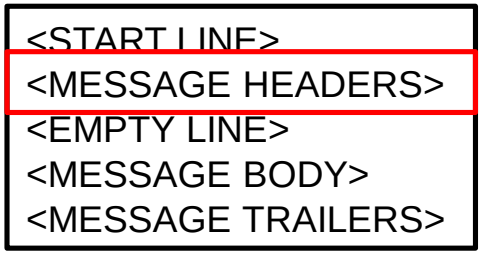
2xx – èxit

3xx – redireccions

4xx – error de client (peticions invàlides, mala sintaxi, etc.)

5xx – error del servidor (petició vàlida però el servidor ha estat incapaç de processar-la)

2.1 Arquitectura. Format dels missatges



Format de les respostes HTTP

"Message-headers"

- *"General Headers":* el mateix que per les peticions.
- *"Response Headers":* proveeixen més detalls sobre la resposta, especialment si hi ha hagut algun error.
- *"Entity Headers":* descriuen les entitats de la petició, si n'hi ha.

2.1 Arquitectura. Format dels missatges

<START LINE>
<MESSAGE HEADERS>
<EMPTY LINE>
<MESSAGE BODY>
<MESSAGE TRAILERS>

Format de les respostes HTTP

START-LINE: *HTTP/1.1 200 OK*

GENERAL-HEADER:
Date: Thu, 6 Sept 2012 10:57:36 GMT
Connection: close

RESPONSE-HEADER:
Server: Apache/1.3.27
Accept-Ranges: bytes

ENTITY-HEADER:
Content-Type: text/html
Content-Length: 78
Last-Modified: Tue, 28 Aug 2012 21:31:12 GMT

EMPTY-LINE

MESSAGE-BODY:
<!DOCTYPE HTML><HTML><HEAD>...</HEAD><BODY><CENTER> Hello world!
</CENTER></BODY></HTML>

2.1 Arquitectura. Test i prova

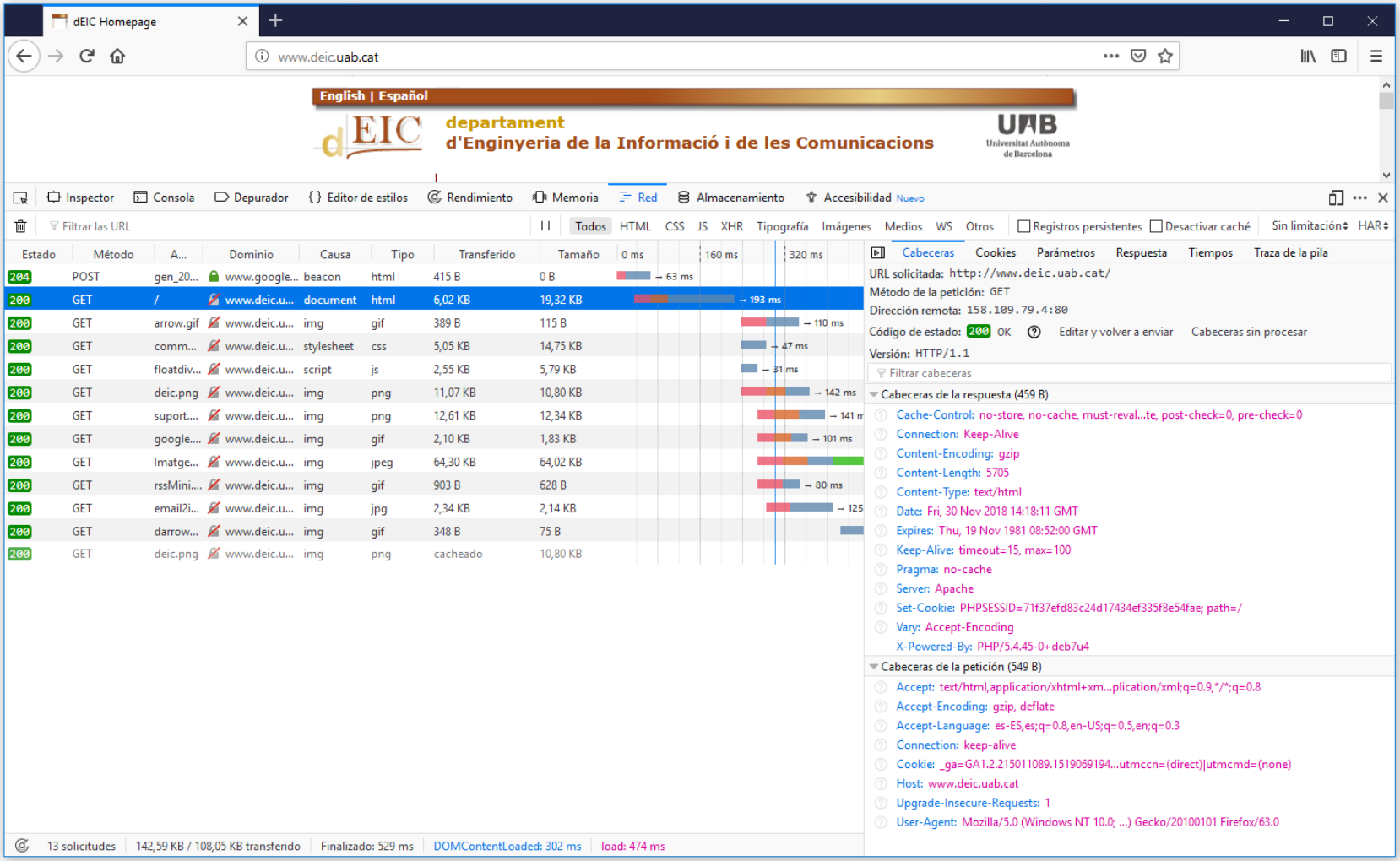
```
$ telnet www.uab.es 80
GET / HTTP/1.1
Host: www.uab.es

HTTP/1.1 200 OK
Date: Tue, 04 Sep 2012 10:31:23 GMT
Server: Apache/2.2.16 (Unix) DAV/2 mod_ssl/2.2.16
OpenSSL/0.9.7e
Last-Modified: Tue, 04 Sep 2012 09:33:18 GMT
ETag: "248ab-c594-4c8dcf2c1f676"
Accept-Ranges: bytes
Content-Length: 50580
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3c.org/TR/html4/strict.dtd">
<html lang='cat'>
<head>
          <title>Universitat Aut&ograve;noma de
Barcelona</title>
...
```

Si feu servir el *putty* o similars recordeu de posar l'opció de negotiation mode a passive (opcions de "connection – telnet").

2.1 Arquitectura. Test i prova



Les eines del desenvolupador dels navegadors també ens permeten veure les capçaleres HTTP (pestanya de xarxa).

¡PREGUNTA D'EXAMEN!

El formularis en HTML es poden sotmetre amb el mètode "GET" o "POST". Quina diferència hi ha entre ells?

¡PREGUNTA D'EXAMEN!

Quina diferència fonamental existeix entre els protocols que estan a les diferents capes de la pila TCP/IP? Per què creus que és deguda?

¡PREGUNTA D'EXAMEN!

Què és la propaganda computacional?

2. EL PROTOCOL HTTP

2.1 – Arquitectura

2.2 – Mecanismes

- *Negociació*
- *"Caching"*
- *Seguretat i privacitat*
- *Persistència*

2.3 – Maneig de l'estat

2.2 Mecanismes. Negociació

- Quan es demana un recurs únic, l'únic tipus de negociació la realitza el client a partir de condicions "*If-*" especificades en els "headers" de la petició.

- Alguns recursos poden tenir vàries representacions. Per exemple, quan un document està en múltiples llenguatges, en diferents conjunts de caràcters, o imatges de diferent qualitat. Existeixen dos tipus de negociació:

  - *Negociació guiada pel servidor:* el client inclou preferències en els "headers" de la petició i el servidor processa aquests preferències per servir la versió del recurs que més s'hi ajusta.
  - *Negociació guiada per l'agent:* en aquest tipus de negociació el client inclou les seves preferències en la petició però el servidor respon amb un missatge *300 "Multiple Choices"*, que conté les diferents opcions disponibles. El client fa una segona petició on escull l'opció que prefereix.

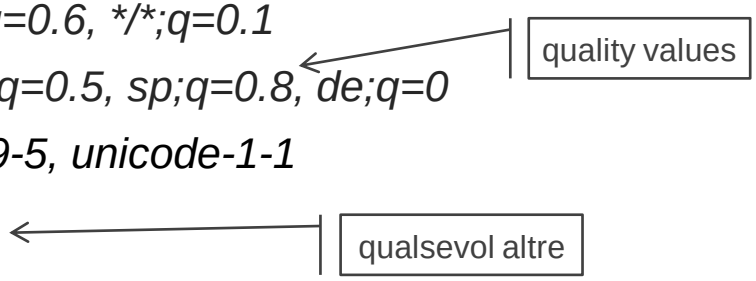
- La negociació guiada pel servidor és la més utilitzada. En general, el client especifica les seves preferències a partir dels següents headers:

Accept: text/html, text/;q=0.6, */*;q=0.1*

Accept-Language: en, fr;q=0.5, sp;q=0.8, de;q=0

Accept-Charset: iso-8859-5, unicode-1-1

*Accept-Encoding: gzip, **



¡PREGUNTA D'EXAMEN!

Per què creus que la negociació HTTP guiada pel servidor és la més utilitzada?

¡PREGUNTA D'EXAMEN!

Què canvia quan a les preferències del navegador li posem que volem pàgines en un idioma o altre?

2.2 Mecanismes. "Caching"

- A partir de HTTP 1.1 s'incorporen mecanismes de "caching" per fer el protocol més eficient (opcions en els "headers"). El "caching" és necessari degut a l'estructura de les web (e.g., les imatges estructurals d'un pàgina de notícies).

- Redueix el consum d'ample de banda i millora el temps de resposta.

- Tipus de "caching":

  - *En el client:* privada i no compartida. Instantània.
  - *En un servidor "proxy":* en un servidor intermediari entre el client i el servidor. Compartida per varis usuaris i molt ràpida per xarxes locals. Escalable (al llarg de la connexió poden existir varis "proxies"). Seguretat (es poden afegir regles de filtratge).

  - *En el servidor:* útil quan el servidor necessita molts recursos per crear una pàgina web dinàmica.

- Aspectes clau a tenir en compte: envelliment, validació, "headers", actualitzacions del recurs (mètodes PUT), aspectes de privacitat.

¡PREGUNTA D'EXAMEN!

Quan una xarxa local té un proxy HTTP, per què és necessari configurar-lo en els navegadors o en el sistema operatiu?

¡PREGUNTA D'EXAMEN!

Quines són les avantatges d'utilitzar servidors "proxy" HTTP? I els inconvenients?

¡PREGUNTA D'EXAMEN!

Els clients es poden saltar el "proxy" HTTP en alguna ocasió?

¡PREGUNTA D'EXAMEN!

*La utilització d'un "proxy" HTTP ens assegura que el contingut que rebem està **sempre** actualitzat?*

¡PREGUNTA D'EXAMEN!

De quines formes es pot comprovar que un recurs ha estat actualitzat en un servidor web respecte al que es té en una "cache" i, si ho ha estat, recuperar-lo?

2.2 Mecanismes. Seguretat i privacitat

- La seguretat i privacitat és important a HTTP ja que el contingut que es transporta conté una gran quantitat de dades personals.

- Un atac molt discutit és el de "*man in the middle"*, on un atacant s'interposa entre el client i el servidor per interceptar i/o canviar dades. Ja que els "proxy" servers estan inherentment "in the middle", també existeixen "headers" per autenticar-los.

- Els aspectes més sensibles són:

  - *Maneig d'informació sensible:* el contingut dels missatges no és xifrat.
  - *Privacitat en la informació continguda en les URLs:* pàgines mal construïdes poden portar informació privada en la URL (que es guarda en els "logs" del servidor).
  - *Informació privada en els "Accept headers":* a partir dels "headers" es pot recollir informació del client.
  - *Informació obtinguda a partir dels "Referer headers":* aquests "headers" aporten informació de la web de la qual es prové.

2.2 Mecanismes. Seguretat i privacitat

- Existeixen dos mètodes HTTP d'autenticació:
  - *Autenticació bàsica:* mètode convencional de usuari/clau. Quan el client demana un pàgina que demana autenticació, el servidor respon amb un *WWW-Authenticate* "header". No és segura ja que s'envia en clar.
  - *"Digest Authentication":* és el mateix mecanisme que la bàsica però incorpora xifratge.

- Existeixen dos mètodes per protegir la privacitat dels missatges HTTP:
  - Xifrar el recurs al servidor i proveir les claus als usuaris identificats.
  - Utilitzar un "add-on" protocol dissenyat específicament per assegurar la privacitat dels missatges HTTP. El més utilitzat és el *Transport Layer Security (TLS),* antigament Secure Sockets Layer (SSL), al que accedim a partir de pàgines *https://*

TLS és un protocol construït sobre TCP/IP que utilitza criptografia de clau pública (en general). HTTPS utilitza el mateix protocol HTTP però sobre TLS, de forma que tot l'intercanvi d'informació entre client i servidor està xifrat. La seguretat depèn del mètode de xifratge.

¡PREGUNTA D'EXAMEN!

Quina és la tecnologia més habitual per enfortir la seguretat en la navegació web?

¡PREGUNTA D'EXAMEN!

Per què s'utilitza el protocol TLS? A quin nivell de la pila de protocols TCP/IP s'ubica? Quin esquema criptogràfic utilitza?

¡PREGUNTA D'EXAMEN!

Què fa el "proxy" quan s'utilitzen connexions https?

2.2 Mecanismes. Persistència

- Les versions 0.9 i 1.0 de HTTP només suportaven connexions de tipus transitori, on el client fa una petició, el servidor respon, i s'acaba la connexió. A mesura que la web incorporà documents d'hipermèdia, aquest esquema resulta poc eficient degut a que una web pot necessitar molts elements emmagatzemats al mateix servidor.

- Les connexions persistents aparegueren en HTTP/1.1. Permeten que el client iniciï una connexió persistent amb la qual després de cada petició/resposta *no* s'acaba la connexió sinó que es deixa el canal TCP obert per poder realitzar més peticions. Això té vàries avantatges: menor latència, càrrega i memòria en el servidor reduïdes, i ús de l'ample de banda més eficient. La connexió acaba quan el client especifica el "header" "*Connection: close*".

- Les connexions persistents permeten l'ús de "pipelining", que consisteix en enviar vàries peticions simultàniament sense esperar la resposta a cadascuna d'elles.

¡PREGUNTA D'EXAMEN!

Quina capçalera canvia en el missatge HTTP quan utilitzem connexions persistents?

¡PREGUNTA D'EXAMEN!

Es pot utilitzar "pipelining" en connexions HTTP transitòries? Per què?

¡PREGUNTA D'EXAMEN!

Es pot utilitzar algun altre mecanisme semblant a "pipelining" per accelerar la transmissió de dades a la web?

2. EL PROTOCOL HTTP

2.1 – Arquitectura

2.2 – Mecanismes

2.3 – Maneig de l'estat

- ***Utilització de "cookies"***

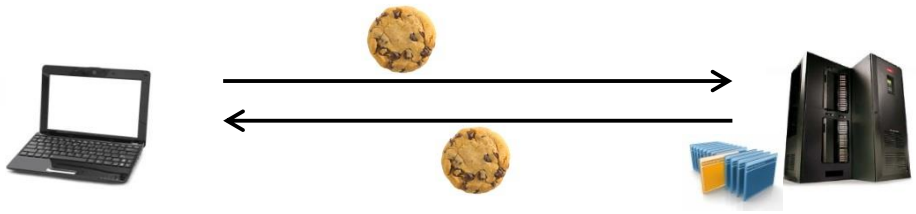
2.3 Maneig de l'estat. Utilització de "cookies"

- Si bé el protocol HTTP té moltes habilitats, no deixa de ser un protocol de petició/resposta. Com a tal, no manté l'estat d'una connexió. Cada nova petició d'un client es tracta independentment de les anteriors.

Com podem gestionar, doncs, la cistella d'una botiga virtual?

- El mecanisme més comú per mantenir un estat és utilitzar "*cookies.*" Són un element opcional del protocol HTTP. El seu funcionament és el següent:

1) Quan el servidor processa una petició que necessita l'estat, envia una quantitat petita d'informació al client anomenada "cookie" que conté informació d'estat (e.g., el nom d'usuari, els articles comprats, etc.).
2) El client manté la "cookie", i per cada nova petició que fa al servidor li envia.
3) El servidor rep la "cookie", la modifica quan és necessari, i la retorna al client cada vegada que li envia una resposta.



2.3 Maneig de l'estat. Utilització de "cookies"

- L'ús de "cookies" presenta varis aspectes delicats:
  - *Transmissió d'informació sensible:* s'ha de tenir en compte que la informació que transporten les "cookies" pot ser interceptada durant la transmissió o a l'ordinador del client.
  - *Ús no desitjat de "cookies":* les "cookies" es poden utilitzar per recollir informació de l'usuari, com les webs que visita, els seus accessos, etc.
  - *"Third-party cookies":* qualsevol servidor pot generar una "cookie", fins i tot quan es demana tan sols una imatge des d'una altra pàgina web.
- No hi ha una solució per aquests aspectes. En general, el maneig de "cookies" recau sobre el navegador i l'usuari. La configuració de "cookies" del navegador permetrà donar més o menys llibertat a l'ús d'aquests elements.

¡PREGUNTA D'EXAMEN!

Què es pot fer per minimitzar la transmissió d'informació al utilitzar "cookies"?

¡PREGUNTA D'EXAMEN!

Una sessió és un conjunt d'informació compartida entre un client i un servidor HTTP. Com creus que s'implementen a la pràctica?

¡PREGUNTA D'EXAMEN!

Quines diferències hi ha entre una "cookie" i una sessió?

¡PREGUNTA D'EXAMEN!

Creus que es poden llegir "cookies" d'altres dominis? I des de subdominis? Com ho podem fer si ens identifiquem en un domini i volem continuar la sessió en un altre?

¡PREGUNTA D'EXAMEN!

Si no es poden llegir "cookies" d'altres dominis, com pot ser que de vegades tinguem publicitat personalitzada a les pàgines que visitem?

¡PREGUNTA D'EXAMEN!

Digues les avantatges i els inconvenients de la utilització de "cookies".

¡PREGUNTA D'EXAMEN!

A la pràctica, un client pot modificar i/o crear cookies?

¡PREGUNTA D'EXAMEN!

S'utilitzen "cookies" quan activem el navegador en mode incògnit?

¡PREGUNTA D'EXAMEN!

Encara que posem el navegador en mode incògnit, creus que hi ha alguna forma a través de la qual companyies d'anuncis poden esbrinar la nostra identitat digital?

¡PREGUNTA D'EXAMEN!

Existeixen serveis que permeten muntar un servidor web en una línia ADSL, per exemple, tot i tenir una IP dinàmica. Quin és el mecanisme que utilitzen?