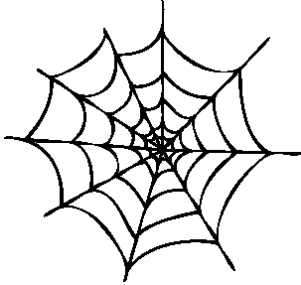
Universitat Autònoma de Barcelona



TECNOLOGIES DE DESENVOLUPAMENT PER A INTERNET I WEB

GRAU D'ENGINYERIA INFORMÀTICA - CURS 2019/2020

TEMA 3 – *PROTOCOLS DE SERVEIS*

0. INTRODUCCIÓ
1. LLENGUATGES DE PROGRAMACIÓ WEB
2. EL PROTOCOL HTTP
3. PROTOCOLS DE SERVEIS







3. PROTOCOLS DE SERVEIS

3.1 – TRANSPORT DE FITXERS I MISSATGES

- File Transport Protocol (FTP)
- Peer-To-Peer (P2P)
- Correu electrònic (SMTP, POP, IMAP)
- Notícies (Usenet)

3.2 – SERVEIS WEB

3.3 – REPRESENTATIONAL STATE TRANSFER (REST)




3.1 Transport de fitxers i missatges. Introducció

- La informació a Internet està organitzada en un conjunt d'unitats discretes: *els fitxers*. Quan els fitxers es creen específicament per la comunicació d'informació, es solen anomenar *missatges*.

- Existeixen quatre grans famílies d'aplicacions per la transmissió de fitxers i/o missatges a Internet:

  - *Transmissió explícita de fitxers:* File Transport Protocol (FTP), xarxes Peer-to-Peer (P2P).
  - *Correu electrònic:* Simple Mail Transfer Protocol (SMTP), Post Office Protocol (POP), Internet Message Access Protocol (IMAP).
  - *Notícies:* Usenet.
  - *Hypertext:* Hypertext Transfer Protocol (HTTP).

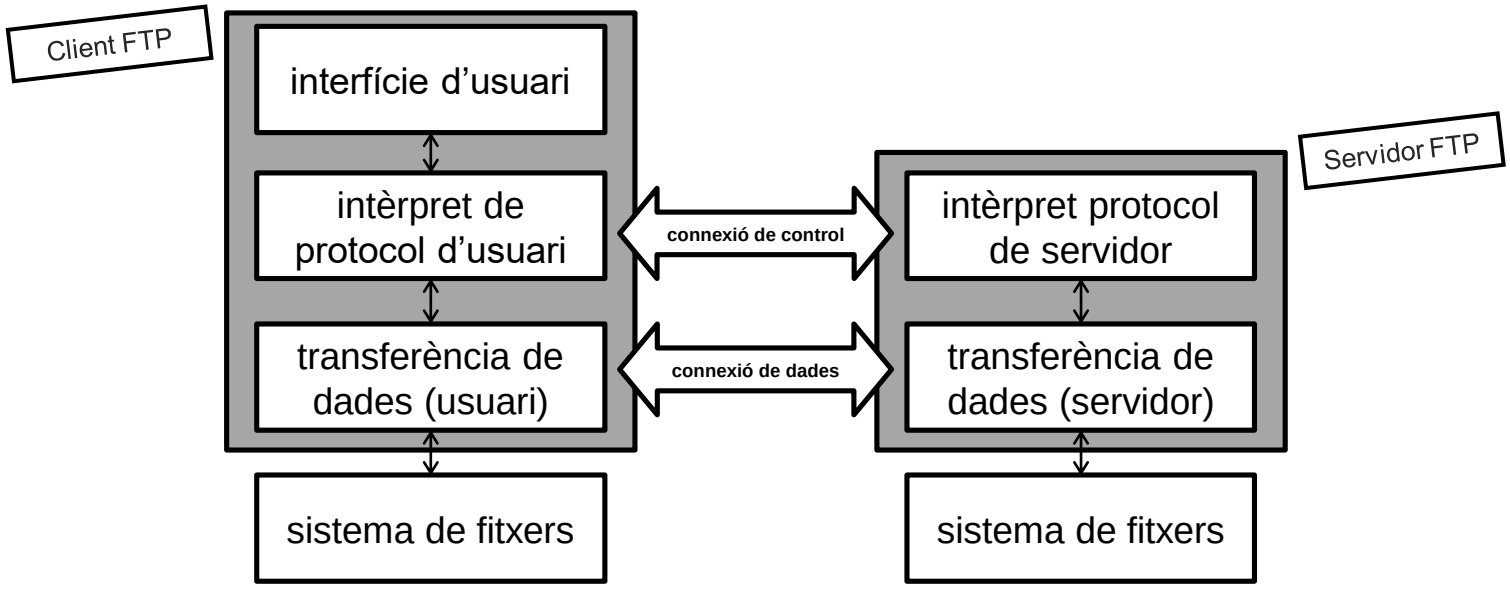
- Només els protocols de transmissió explícita de fitxers veuen el fitxer com una caixa negra que s'ha de moure d'un lloc a l'altre. Els altres protocols veuen el fitxer com un missatge que s'ha de comunicar a l'usuari i per tant el tracten de diferents maneres (e.g., en permeten crear de nous, visualitzar-los, etc.).

- Amb el temps, aquests protocols s'han ampliat per permetre més i més funcionalitats, esborrant les divisions entre ells.

3.1 Transport de fitxers i missatges. FTP

File Transport Protocol (FTP)

- A diferència de la majora de protocols, utilitza dos canals lògics (connexions TCP) per obtenir més flexibilitat:
  - *Control:* la connexió de control serveix per passar comandes interactivament entre l'usuari (client FTP) i el servidor.
  - *Dades:* la connexió serveix per transferir les dades d'un fitxer i se n'estableix una de nova per cada nova transmissió.
- Al iniciar la connexió s'utilitza un sistema d'autenticació per identificar l'usuari.



¡PREGUNTA D'EXAMEN!

Què és el que passa (tant a nivell d'usuari com de protocol de comunicació) quan escrivim

ftp://servidorFTP.org

en un navegador web?

¡PREGUNTA D'EXAMEN!

Quines avantatges té que el protocol FTP utilitzi dues connexions diferents per transmetre comandes i dades?

¡PREGUNTA D'EXAMEN!

És recomanable utilitzar el protocol ftp per accedir a recursos personals d'un servidor?

3.1 Transport de fitxers i missatges. P2P

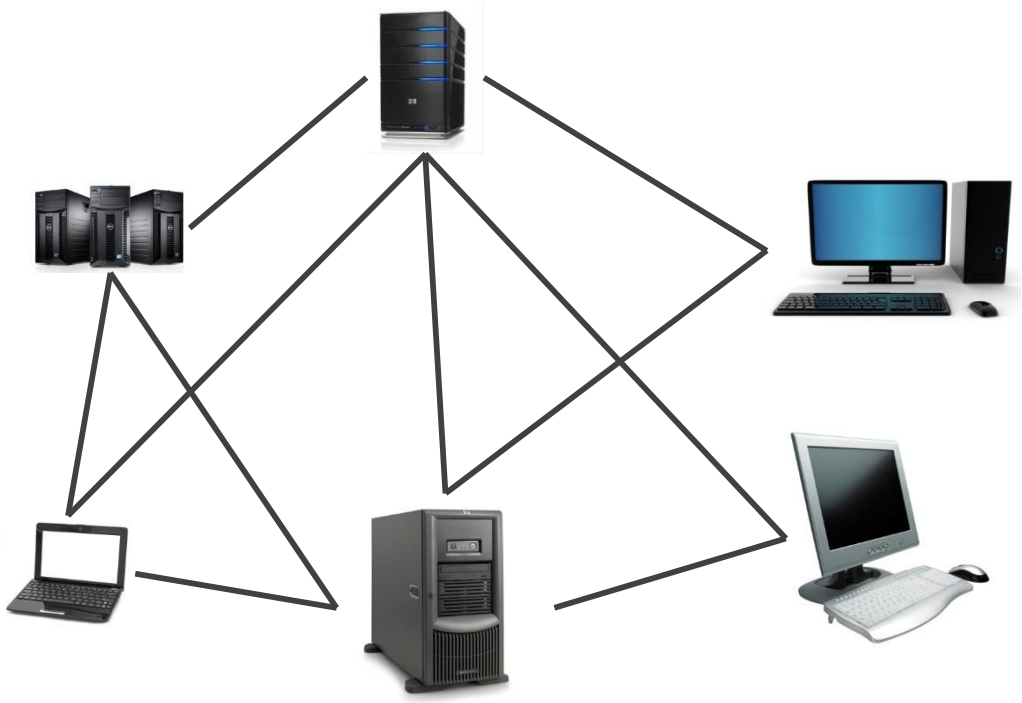
Xarxes Peer-To-Peer (P2P)

- A diferència de les tradicionals arquitectures client/servidor, en les arquitectures P2P els hosts, anomenats nodes, poden actuar de client i servidor a la vegada, permeten accés a recursos com fitxers, impressores, o perifèrics.

- És necessari que tots els nodes de la xarxa P2P utilitzin el mateix protocol per connectar i compartir recursos.

"killer application"

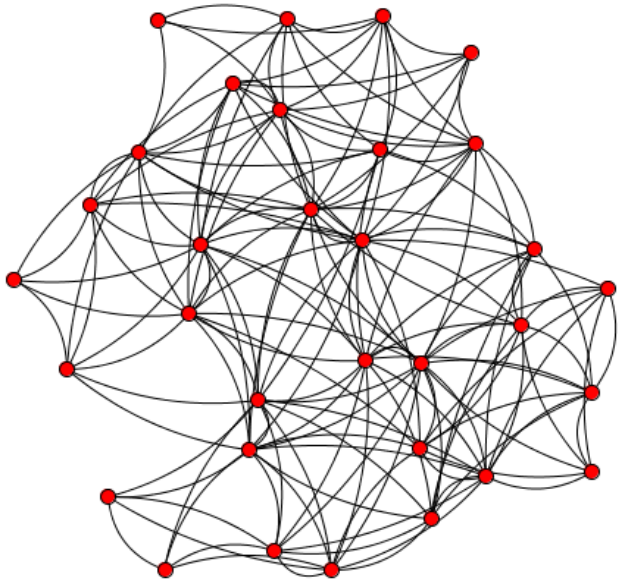
napster



3.1 Transport de fitxers i missatges. P2P

- Segons l'estructura de la "overlay network" (xarxa construïda a sobre una altra), les xarxes P2P es poden classificar en:

  - *Estructurades:* els nodes estan organitzats seguint un criteri, creant unes topologies clares. Tenen una escalabilitat molt alta i un bon rendiment. Qualsevol node de la xarxa pot accedir a qualsevol recurs. Els recursos s'indexen en taules "hash" distribuïdes (DHT). Les DHT són una forma descentralitzada de taules "hash" que permeten la recerca de qualsevol recurs de la xarxa des de qualsevol node.



Xarxes P2P estructurades:

- BitTorrent
- Kad
- Storm
- Sistemes de computació "grid"

3.1 Transport de fitxers i missatges. P2P

- *No estructurades:* no s'imposa cap criteri en l'estructura dels nodes. No sempre es pot accedir a tots els recursos disponibles a la xarxa. Les recerques s'escampen per la xarxa. En general, es necessita algun element centralitzador. Existeixen tres categories diferents:
  - Sistemes "peer-to-peer" purs, on tots els nodes són iguals.
  - Sistemes "peer-to-peer" centralitzats, on un o varis supernodes centrals s'utilitzen per indexació.
  - Sistemes "peer-to-peer" híbrids, on es permet l'ús de supernodes, però l'estructura no està fixada.

Xarxes P2P no estructurades:

- Napster
- Freenet
- Gnutella
- Kazza

¡PREGUNTA D'EXAMEN!

Per què és difícil de controlar/evitar la descàrrega de fitxers a través de protocols P2P?

¡PREGUNTA D'EXAMEN!

Què és el que passa, doncs, quan la policia atrapa i multa usuaris de xarxes P2P descarregant fitxers amb "copyright"?

¡PREGUNTA D'EXAMEN!

Què és un client i un servidor en una xarxa P2P?

¡PREGUNTA D'EXAMEN!

On s'emmagatzemen les DHT en les xarxes P2P?

¡PREGUNTA D'EXAMEN!

Es podria utilitzar el P2P per crear un "cloud"?

3.1 Transport de fitxers i missatges. Correu electrònic

Sistema general

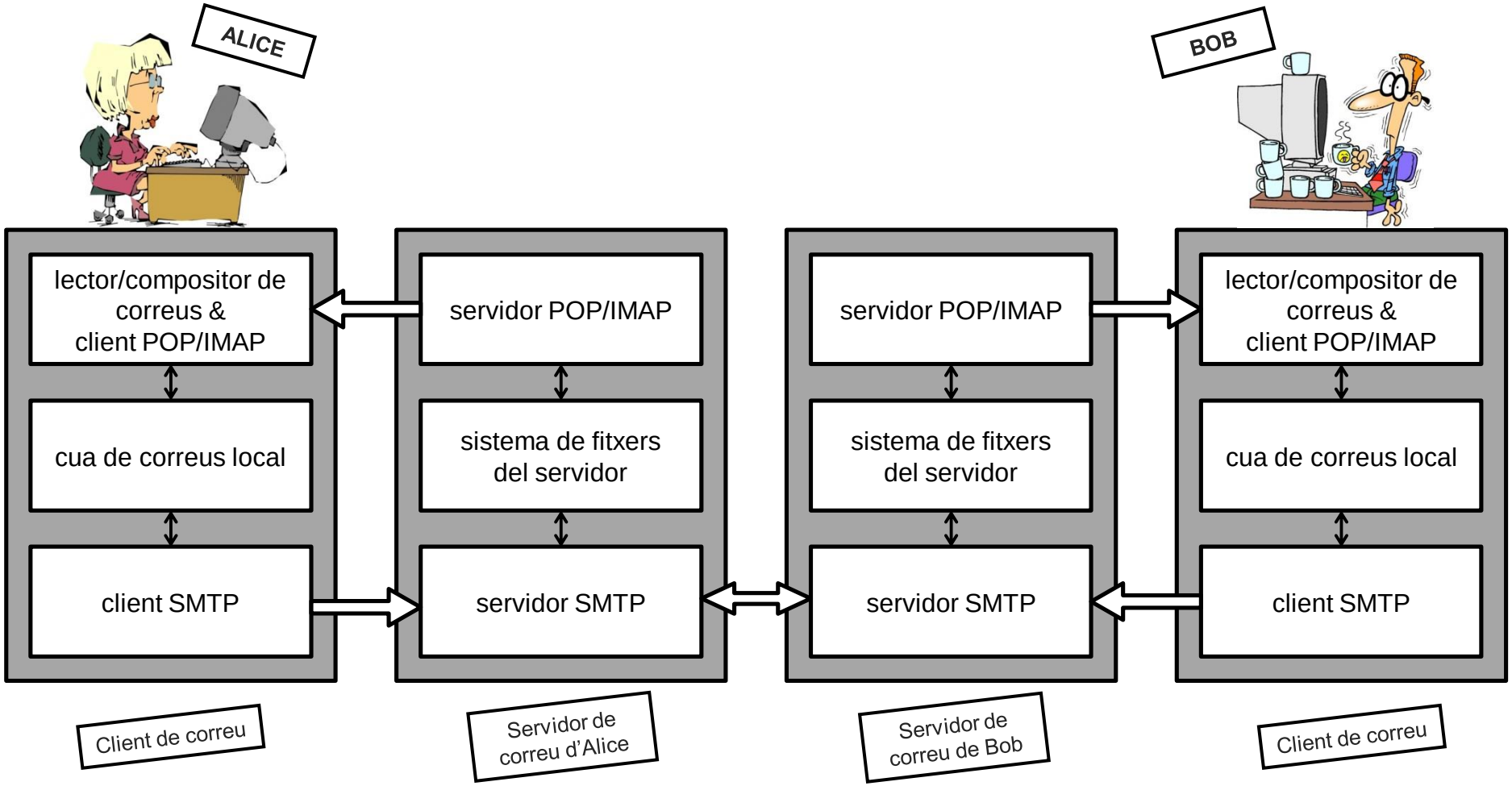
- El correu electrònic en la pila de protocols TCP/IP no està composat d'un sol protocol o tecnologia sinó que és un sistema que conté varis components i protocols que treballen conjuntament.

- Actualment, l'enviament d'un correu electrònic utilitza 5 passos:

1) *Composició:* l'usuari crea un correu que està compost del "header", que conté el tipus de missatge i a qui va dirigit, i el "body", que conté el missatge.
2) *Submissió:* a diferència de la resta de protocols, l'emissor i receptor del missatge no tenen per què està connectats necessàriament. L'usuari pot decidir quan s'entrega en el sistema de correu.
3) *Entrega:* la transmissió de correus es realitza utilitzant el protocol SMTP.
4) *Recepció i processament:* un cop el correu arriba al servidor, es deixa a la bústia de l'usuari corresponent on espera a ser retirat.
5) *Accés i retirada:* l'usuari receptor comprova amb el seu servidor de correu si disposa de més correus. Si és així, accedeix a la seva bústia utilitzant POP o IMAP i recupera els correus.

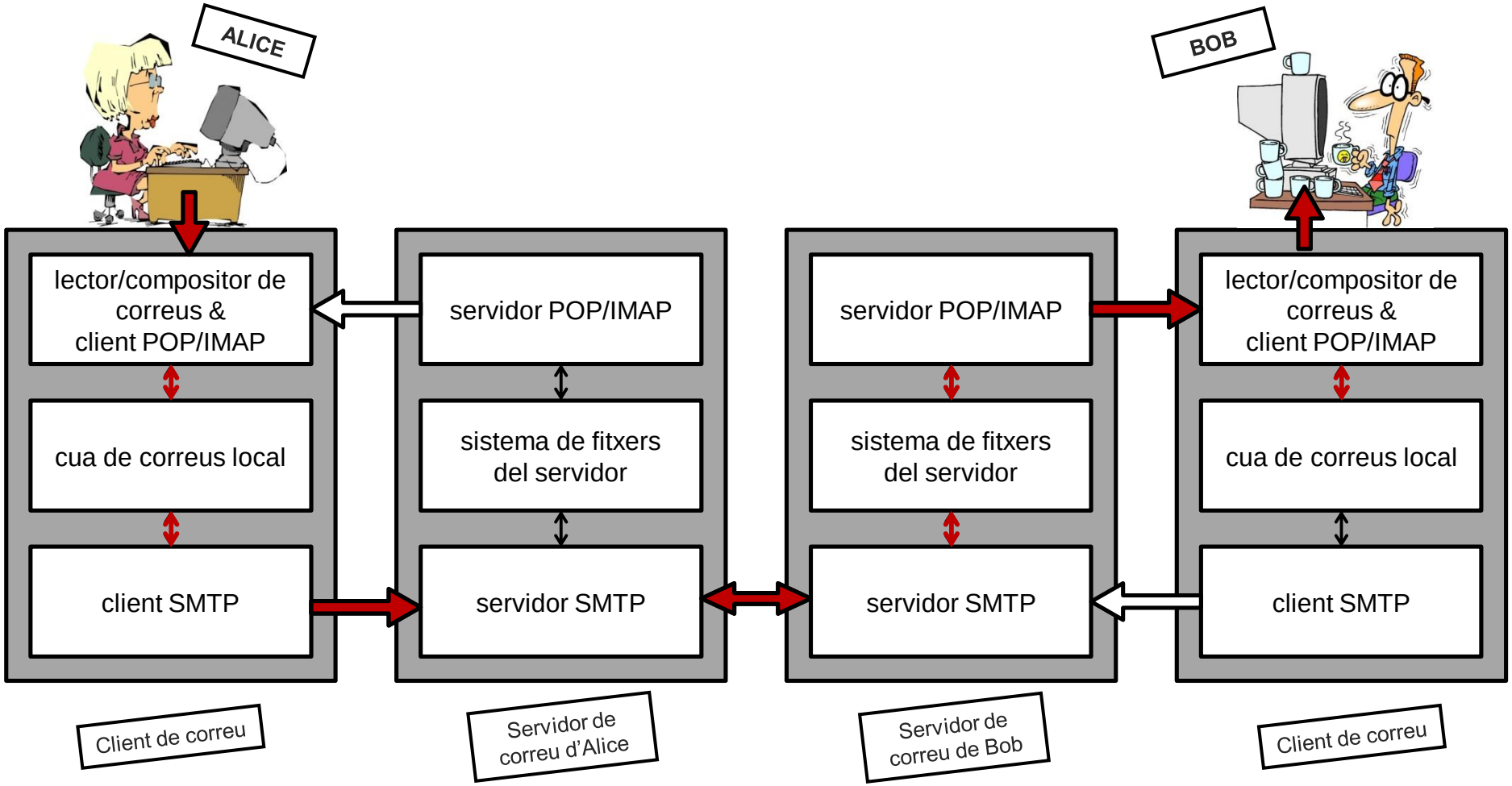
3.1 Transport de fitxers i missatges. Correu electrònic

Sistema general



3.1 Transport de fitxers i missatges. Correu electrònic

Sistema general



¡PREGUNTA D'EXAMEN!

Per què creus que el sistema de correus va diferenciar el protocol SMTP del POP/IMAP?

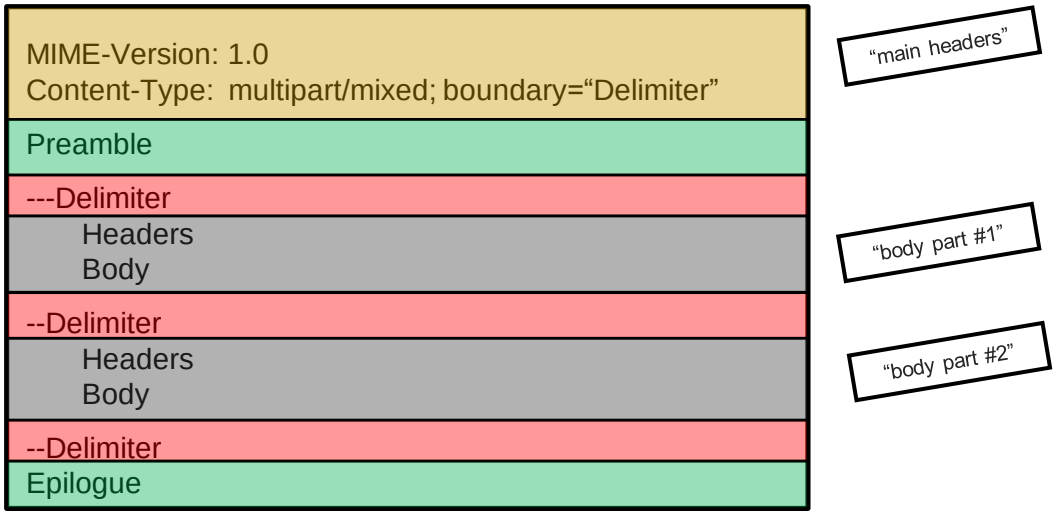
3.1 Transport de fitxers i missatges. Correu electrònic

Format dels missatges

- El format de les adreces de correu són del tipus <*usuari*> @ <domini>. Contenen un usuari "at" un servidor seguint les normes de sintaxi del Domain Name System (DNS).

- El registre MX de DNS especifica els servidors de SMTP que s'han d'utilitzar per un determinat nom de domini, ampliant la flexibilitat del sistema i evitant l'"spam".

- Per permetre qualsevol tipus de fitxer en l'enviament de correus, es creà l'estàndard Mutipurpose Internet Mail Extensions (MIME). MIME permet l'ús de missatges de text, no-text, fitxers binaris, i codificar múltiples fitxers en un sol missatge. El format utilitzat és del tipus:



3.1 Transport de fitxers i missatges. Correu electrònic

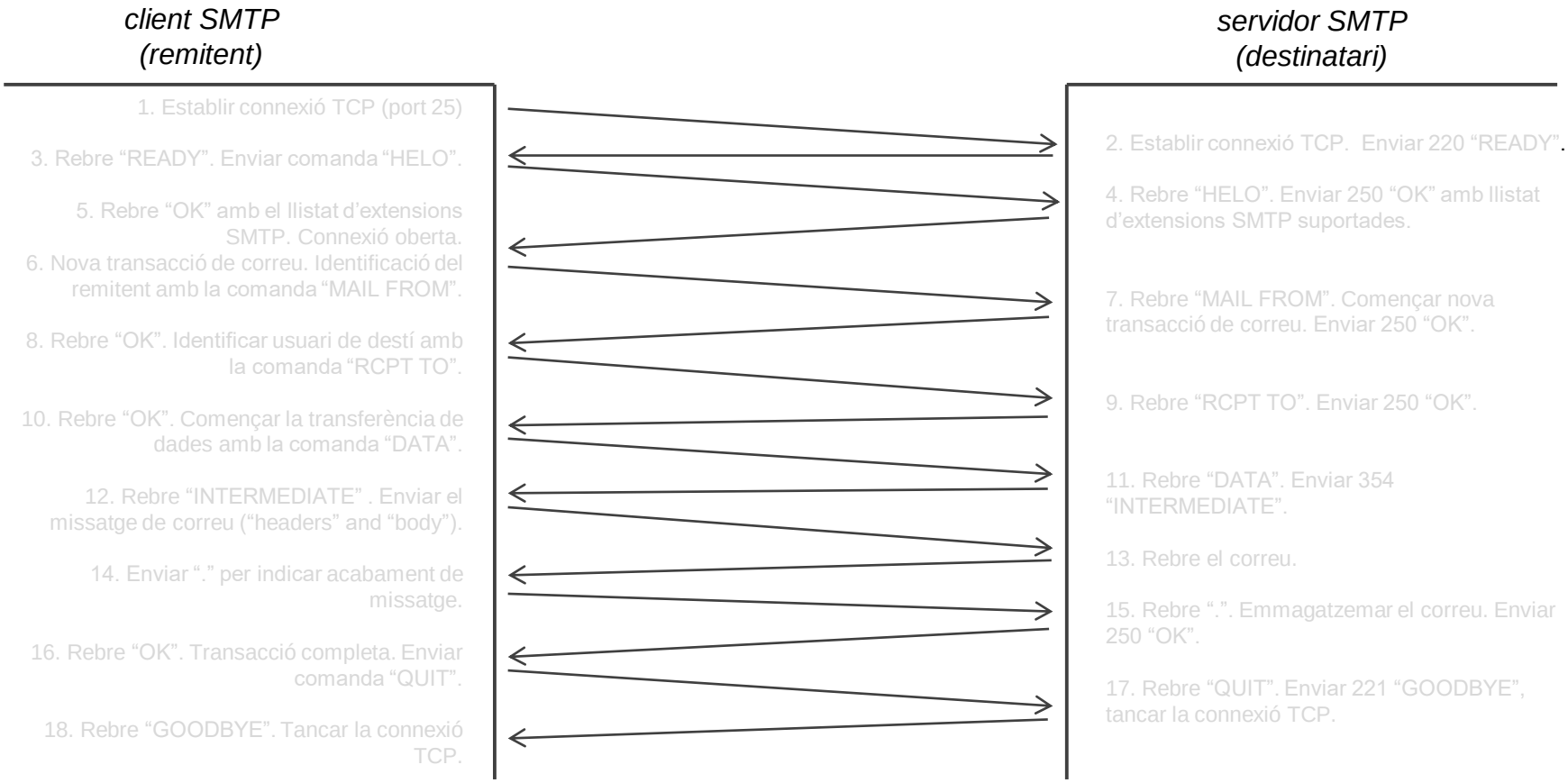
Format dels missatges

- Exemple de correu electrònic utilitzant un missatge MIME (es pot veure amb qualsevol client de correu utilitzant l'opció de *mostrar original*):

```
From: Alice <alice@someplace.org>
To: Bob <bob@somewhereelse.com>
Date: Tue, 11 Sept 2012 18:00:12
Subject: Photo from our recently date
MIME-Version: 1.0
Content-Type: multipart/mixe; boundary="delimiter123"
This is a multipart message in MIME format

--delimiter123
Content-Type: text/plain
Dear Bob, here is the photo that we took in our first date.
Bla bla bla...
Alice.

--delimiter123
Content-Type: image/jpeg; name="firstDate.jpg"
Content-Transfer-Encoding: base64
0M8R4KGxGuEAAAAAAAAAAAAAAAAAAAAPgADAP7/CQAGAAAAAAAAAAAAAA
CAAAAhgAAAAAAAAAEAAAiAAAAAEAAAD+////AAAAAIQ...

--delimiter123
(Epilogue)
```

3.1 Transport de fitxers i missatges. Correu electrònic

Simple Mail Transport Protocol (SMTP)



3.1 Transport de fitxers i missatges. Correu electrònic

Exemple de SMTP

```
$ telnet smtp.uab.es 25
220 damascus.uab.es -- Server ESMTP (Sun Java System
Messaging Server 6.1 HotFix 0.10 (built Jan  6 2005))
HELO smtp.uab.es
250 damascus.uab.es OK, [81.60.165.181].
MAIL FROM: bob@uab.es
250 2.5.0 Address Ok.
RCPT TO: alice@uab.es
250 2.1.0 alice@uab.es Recipient ok
DATA
Subject:-With love from Bob-
Hi again Alice, I've been thinking of you since our last
date...
.
250 2.0.0 Message accepted for delivery
QUIT
221 2.0.0 damascus.uab.es closing connection
```

3.1 Transport de fitxers i missatges. Correu electrònic

Simple Mail Transport Protocol (SMTP)

- Característiques especials:

- *"Mail relaying":* "relaying" significa que el servidor SMTP accepta i redirigeix correus dirigits a qualsevol altre servidor. Actualment en desús degut a l'ús del registre MX del DNS, que permet entregues directes, i a l'abús d'"spam".
- *"Mail forwarding":* sota determinades circumstàncies es permet que el correu per un determinat usuari s'enviï a una altra direcció de correu.
- *"Mail gatewaying":* servidors especials que permeten la conversió d'un sistema de correu basat en la pila de protocols TCP/IP a una altre, i viceversa.
- *"Address debugging":* verificació que un usuari existeixi en un servidor sense l'enviament de cap correu.
- *"Mail list expansion":* determinar els usuaris individuals associats a una llista de distribució.

¡PREGUNTA D'EXAMEN!

Per què el mail "relaying" crea dificultats per controlar el correu "spam"?

¡PREGUNTA D'EXAMEN!

Què és un "email scam"?

3.1 Transport de fitxers i missatges. Correu electrònic

Simple Mail Transport Protocol (SMTP)

- Els aspectes de seguretat són un *malson*. Tot el sistema està basat en la confiança entre usuaris i entre servidors. Amb l'explosió d'Internet es van haver d'adoptar algunes mesures:

  - Tenir una llista de servidors autoritzats i comprovar l'adreça al iniciar la sessió.
  - Restricció d'algunes comandes o serveis com el "relaying", o l'expansió.
  - Autenticació a partir d'una extensió d'SMTP.
  - Validar el sobre abans d'acceptar el correu, com per exemple validant l'adreça origen.
  - Limitant la mida del missatges.
  - Creant "logs" de tots els intents de connexió per analitzar-los a posteriori.

¡PREGUNTA D'EXAMEN!

Per què el sistema de mail TCP/IP pot tenir problemes de seguretat si no es prenen les mesures adequades?

3.1 Transport de fitxers i missatges. Correu electrònic

POP i IMAP

- Existeixen tres formes per accedir i recuperar el correu electrònic:

  - *Model d'accés "online":* tots els hosts estan sempre connectats i executant un servidor SMTP, per tant els usuaris tenen accés directe a la seva bústia.
  - *Model d'accés "offline":* l'usuari estableix una connexió al servidor SMTP on té la seva bústia, es descarrega els correus i els elimina del servidor. Els correus es llegeixen de forma local. S'utilitza el protocol POP.
  - *Model d'accés desconnectat:* l'usuari es connecta al seu servidor SMTP, sincronitza la bústia del servidor amb la seva local, es desconnecta per llegir/escriure/... i es torna a connectar per sincronitzar la seva bústia local amb la del servidor. S'utilitza el protocol IMAP.

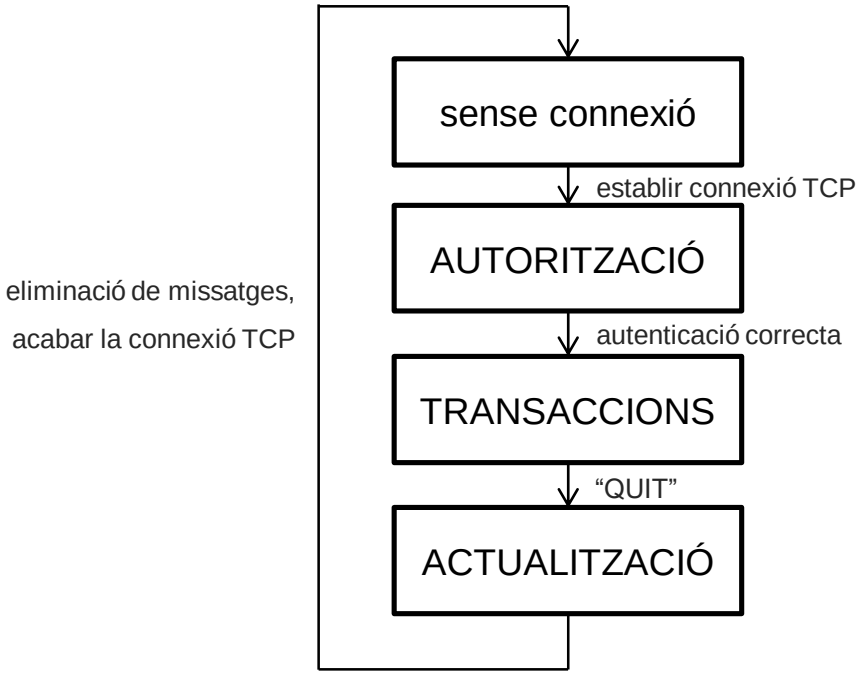
- POP3 (3 és la versió actual del protocol) és el protocol més senzill i el que més popularitat ha tingut degut a la seva fàcil implementació. Les peticions obtenen dues respostes bàsiques: +OK, -ERR. El protocol té 3 estats:

  - *Autorització:* el servidor envia una salutació dient que està a punt, i el client envia la seva identificació.

3.1 Transport de fitxers i missatges. Correu electrònic

POP i IMAP

- *Transacció:* el client és autoritzat per realitzar operacions a la seva bústia, com recuperar correus o eliminar-ne.
- *Actualització:* les operacions realitzades pel client es porten a terme quan el client envia la comanda "QUIT".




```
+OK POP3 server ready
USER alice@uab.es
+OK
PASS *******
+OK alice@uab.es has 1 message
STAT
+OK 2 574
LIST
1 414
.
RETR 1
+OK
(mail from Bob sent, 414 bytes)
.
DELE 1
+OK message 1 deleted
QUIT
```

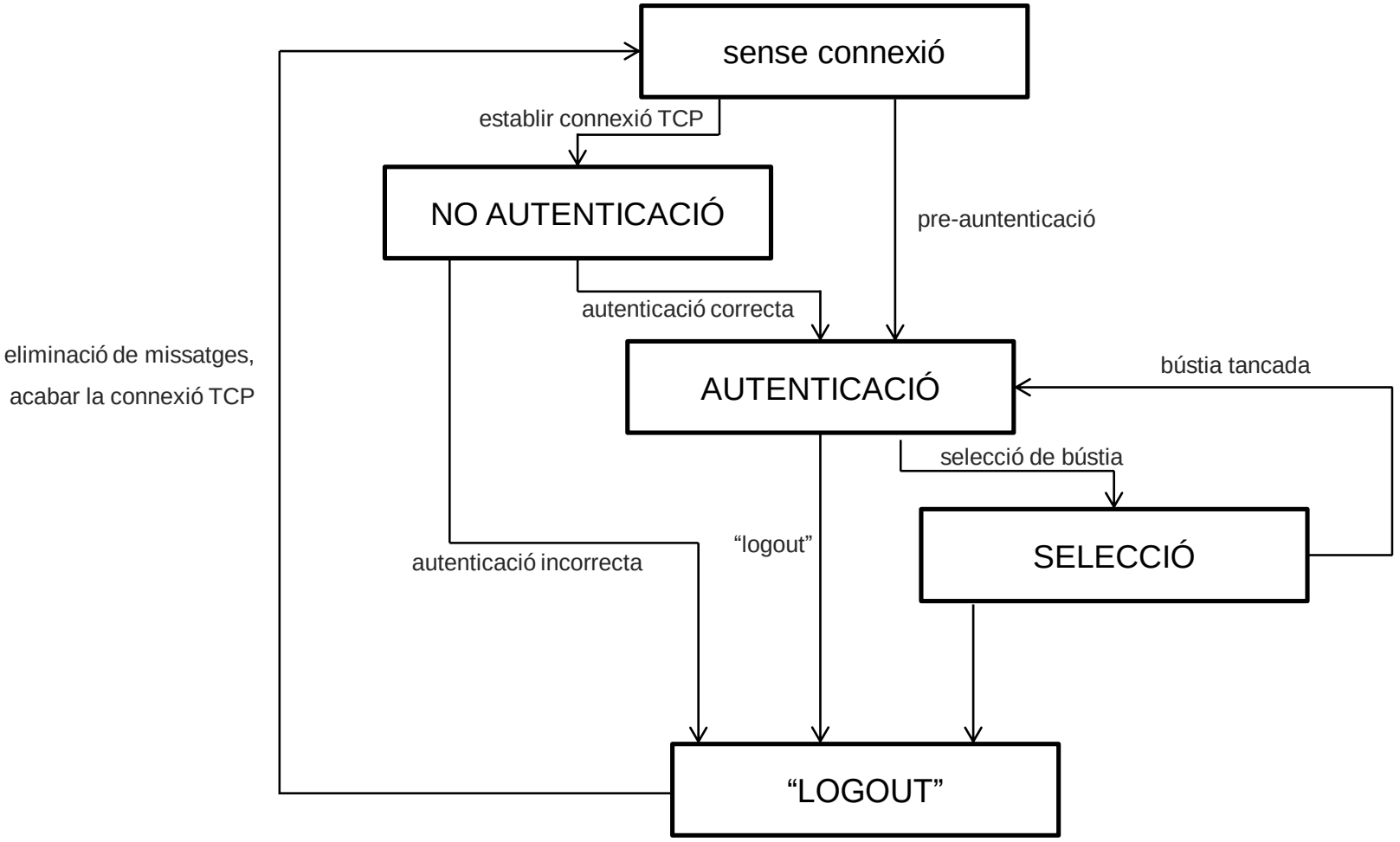
3.1 Transport de fitxers i missatges. Correu electrònic

POP i IMAP

- IMAP4 (4 és la versió actual del protocol) incorpora mecanismes més avançats per la manipulació de correu com:
  - Sincronització de la bústia entre el servidor i de forma local.
  - Ús de "flags" per saber quins missatges s'han llegit i quins no.
  - Maneig de múltiples bústies i transferència de correus entre elles (comunament vist com carpetes).
  - Recuperació parcial dels correus abans de descarregar-los completament.
  - Recuperació de porcions dels correus (quan s'utilitza "MIME multipart").
  - Manipulació de missatges de Usenet.
- Els estats del protocol són:
  - *No autenticació:* després d'establir la connexió s'entra en aquest estat.
  - *Autenticació:* el client a completat l'autenticació i pot fer operacions.
  - *Selecció:* després de selecciona una bústia el client pot manipular missatges individuals.
  - *"Logout":* es tanca la sessió.

3.1 Transport de fitxers i missatges. Correu electrònic

POP i IMAP



¡PREGUNTA D'EXAMEN!

Digues les principals avantatges i inconvenients que tenen els tres sistemes d'accés i recuperació del correu electrònic.

¡PREGUNTA D'EXAMEN!

El protocol per enviar i rebre correus electrònics és l'SMTP. Què és doncs un servei de correu via web com el Gmail?

¡PREGUNTA D'EXAMEN!

Per a què s'utilitzen els protocols SMTP, IMAP, i POP3. Quins protocols utilitzen els mètodes d'accés i recuperació de mails "online," "offline", i desconnectat?

3.1 Transport de fitxers i missatges. Notícies

Usenet

- "User network" (Usenet) és un sistema per enviar missatges a un gran grup de participants. Típicament, s'utilitza per distribuir notícies a una comunitat.

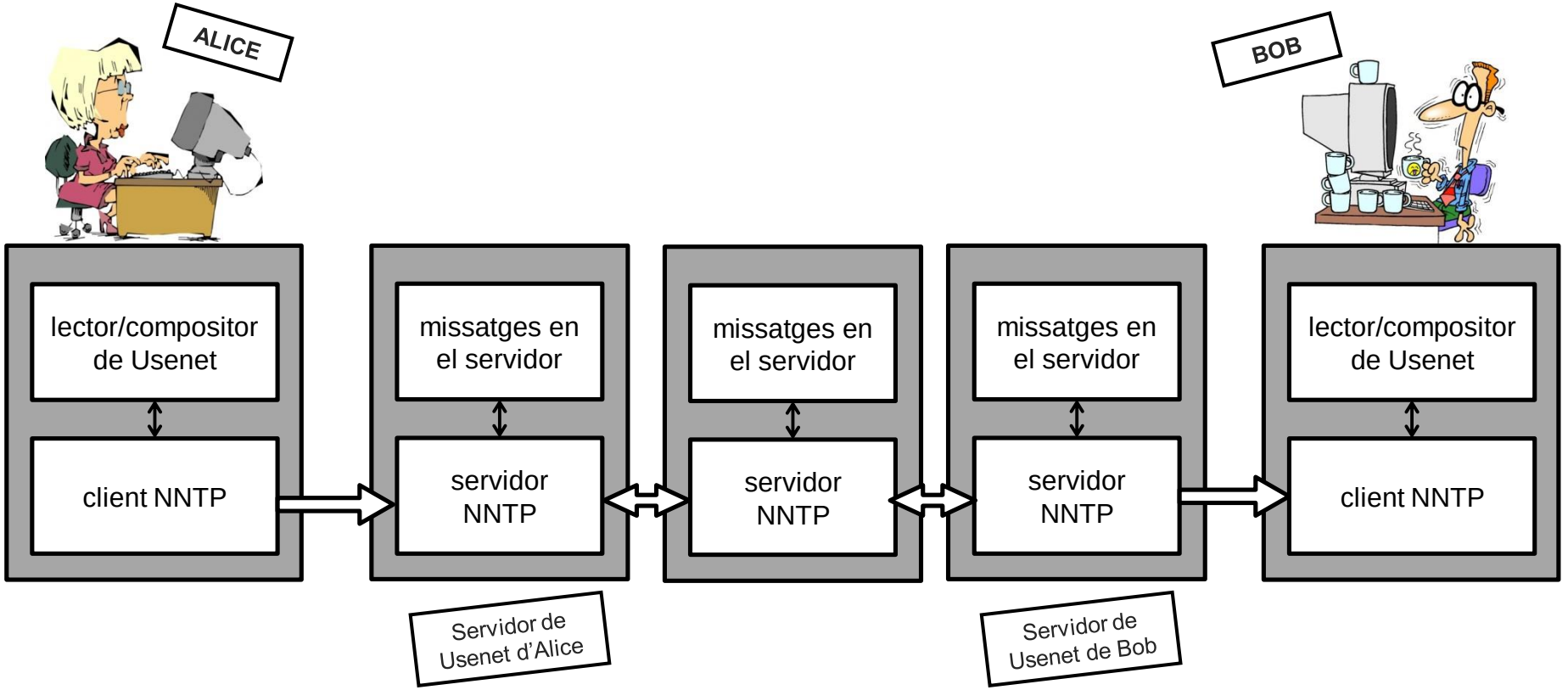
- Els grups de notícies estan organitzats de forma jeràrquica i la seva creació i estructura es controla fortament. Les "big eight" jerarquies són: comp.*, humanitites.*, misc.*, news.*, rec.*, sci.*, soc.*, talk.*.

- El procés de comunicació és el següent:

1. *Composició:* l'usuari crea el missatge (semblant a un correu electrònic).
2. *Enviament:* s'envia el missatge a servidor Usenet de l'usuari.
3. *Propagació:* els missatges es propaguen entre tots els servidors Usenet d'Internet. Aquesta és la part més complexa.
4. *Accés i recuperació:* altres usuaris accedeixen als seus servidors Usenet i recuperen el missatge escrit.

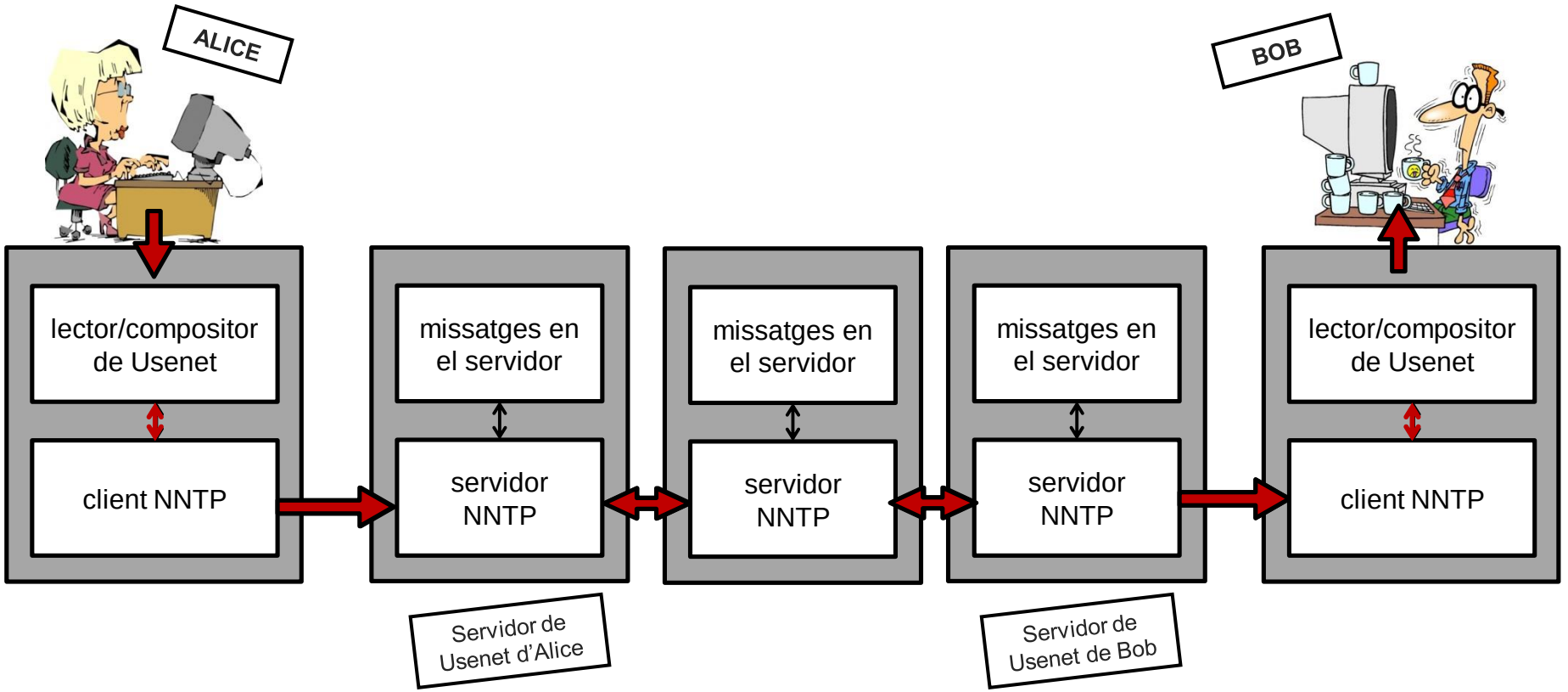
3.1 Transport de fitxers i missatges. Notícies

Usenet



3.1 Transport de fitxers i missatges. Notícies

Usenet



¡PREGUNTA D'EXAMEN!

Creus que Usenet es podria re-emplaçar de forma eficient per algun tipus de mecanisme que utilitzés el correu?




3. PROTOCOLS DE SERVEIS

3.1 – TRANSPORT DE FITXERS I MISSATGES

3.2 – SERVEIS WEB

- Extensible Markup Language (XML)
- Universal Description, Discovery & Integration (UDDI)
- Web Services Description Language (WSDL)
- Simple Object Access Protocol (SOAP)

3.3 – REPRESENTATIONAL STATE TRANSFER (REST)

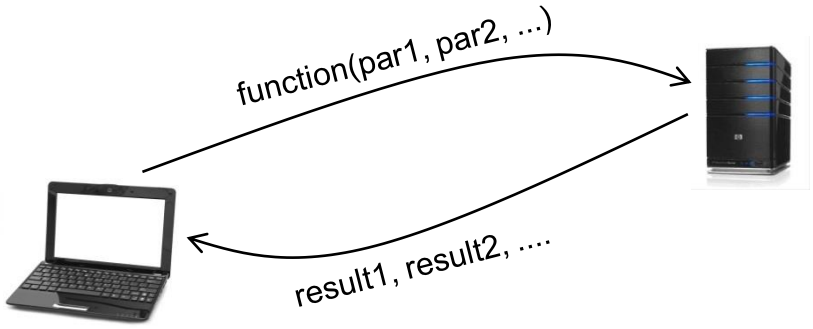



3.2 Serveis web. Introducció

- Es defineix *servei web* com un programari dissenyat per suportar la interacció entre dos dispositius d'una xarxa utilitzant els protocols de la web.

- Conceptualment, es pot veure com una crida a una funció remota però utilitzant el protocol HTTP perquè transporti els missatges.

- Les principals avantatges són: 1) l'aprofitament de tota la infrastructura desplegada per la web, 2) l'absència de tallafocs que ens privin d'accedir als hosts, i 3) compatibilitat entre diferents aplicacions i plataformes.



3.2 Serveis web. XML

Extensible Markup Language (XML)

```
<?xml version="1.0"?>
<note>
   <to>Alice</to>
   <from>Bob</from>
   <heading>Reminder</heading>
   <body>Don't forget our date!</body>
</note>
```

3.2 Serveis web. XML

- El llenguatge XML proporciona una sèrie de normes i formats per codificar informació de forma que sigui llegible tant per humans com per ordinadors.

- El va desenvolupar la W3C amb l'objectiu de proporcionar un llenguatge que transportés dades, i no per mostrar-les. XML no "fa" res!

- Les seves principals avantatges són:
  - Separa les dades (contingut) de la presentació.
  - Simplifica la compartició de continguts.
  - Simplifica el transport de dades.
  - Simplifica canvis en les plataformes.
  - Ajuda en la creació de nous llenguatges i protocols a Internet.
  - Proporciona molta flexibilitat.

- En l'actualitat hi ha molts llenguatges basats en XML, com RSS, Atom, SOAP, o XHTML, també protocols com XMPP, i moltes aplicacions ofimàtiques com Micro$oft Office o OpenOffice també l'utilitzen.

- L'XML *"parser"* és el processador que analitza els "markups" i passa la informació estructurada a l'aplicació.

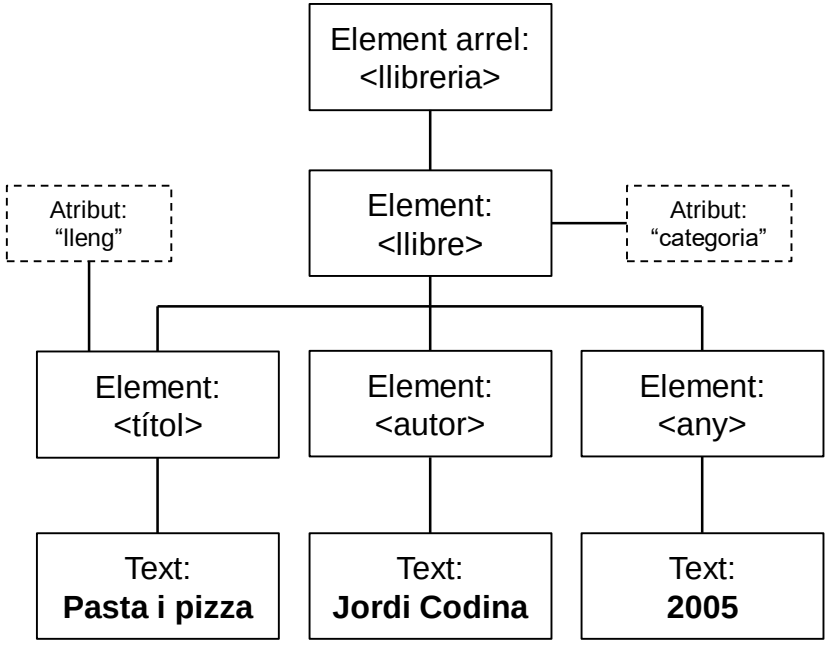
3.2 Serveis web. XML

- Un document XML està format per els següents elements:
  - *Declaració XML:* tots els documents comencen amb una declaració del l'estil
    <?xml version="1.0" encoding="UTF-8" ?>
  - *"Markups" i contingut:* els caràcters del document o bé pertanyen al contingut o als "markups". Els "markups" generalment són del tipus
    <XXX**>** o **&**XXX**;**
  - *"Tags":* és un "markup" del tipus **<**XXX**>**. O bé són de començament **<**XXX**>**, final **</** XXX**>**, o d'elements buits **<**XXX **/>**.
  - *Elements:* és el contingut dins un "tag". Pot contenir altres "tags".
  - *Atributs:* consisteix en una parella de nom/valor existent dins un "tag" de començament o d'element buit.
  - *Declaració DTD:* després de la declaració de XML, és comú trobar-se una Document Type Definition (DTD), que especifica el tipus de "tags" vàlids per aquell document.

3.2 Serveis web. XML

- S'utilitza una sintaxi molt senzilla i s'estructura de forma jeràrquica.

```
<llibreria>
   <llibre categoria="cuina">
      <títol lleng="cat">Pasta i pizza</títol>
      <autor>Jordi Codina</autor>
      <any>2005</any>
   </llibre>
   <llibre categoria="web">
      <títol lleng="cat">Aprendre XML</títol>
      <autor>Joan Garcia</autor>
      <any>2009</any>
   </llibre>
</llibreria>
```



3.2 Serveis web. XML

- Alguns aspectes a tenir en compte són:
  - Tots els elements tenen un "tag" de tancament.
  - Els "tags" han d'estar ben ordenats.
  - Ha d'existir l'element arrel.
  - Els atributs han d'estar entre cometes.
  - És "recomanable" no utilitzar atributs i convertir-los en elements.

???

```
<p> Això és un paràgraf.
<br>
<b><i> Això és negreta i itàlica </b></i>
<img src=imatge.png>
```

XML

```
<arrel>
   <p> Això és un paràgraf.</p>
   <br />
   <b><i> Això és negreta i itàlica </i></b>
   <img src="imatge.png" />
</arrel>
```

¡PREGUNTA D'EXAMEN!

Per què creus que XML és actualment utilitzat de forma tan comuna i en aplicacions tan diferents?

3.2 Serveis web. UDDI

Universal Description, Discovery and Integration (UDDI)

- És un registre general perquè negocis d'arreu es puguin llistar a Internet. També proporciona mecanismes per registrar i localitzar serveis web.

- Un registre UDDI consisteix en tres components:

  - *"White pages":* proporciona informació del negoci que proporciona el servei (nom, descripció, etc.).
  - *"Yellow pages":* proporciona una classificació del servei o negoci basada en taxonomies.
  - *"Green pages":* s'utilitzen per descriure com accedir a un determinat servei web proporcionat per un negoci.

- Un exemple on UDDI és d'utilitat és per l'intercanvi de documents entre companyies com comandes o factures.

- En l'actualitat no sembla que s'utilitzi massa excepte internament en algunes empreses grans.

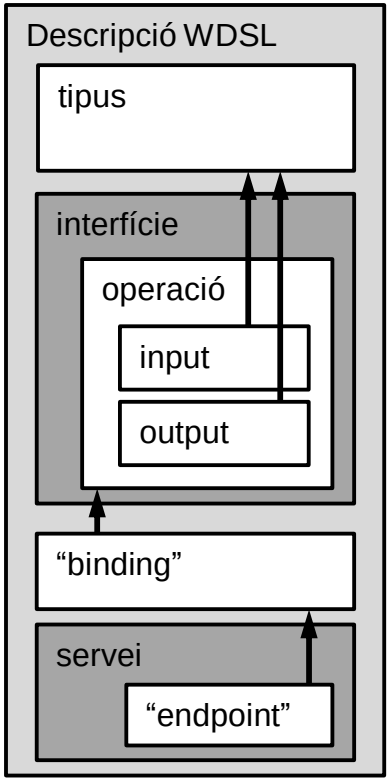
3.2 Serveis web. WSDL

Web Services Description Language (WSDL)

- WSDL és un llenguatge basat en XML que descriu les funcionalitats d'un servei web.

- Una descripció WSDL (o fitxer WSDL) conté les funcions que el servei web accepta. Està estructurada de la següent forma:

- *Servei:* conjunt de funcions que defineixen el servei.
- *"Endpoint":* adreça o punt de connexió al servei web (URL).
- *"Binding":* defineix la interfície, l'estil, i les operacions.
- *Interfície:* defineix totes les operacions suportades i els missatges utilitzats per realitzar-les.
- *Operació:* defineix les accions (de tipus SOAP). És com una crida a funció.
- *Missatge:* conté la informació necessària per realitzar l'acció.
- *Tipus:* descriu el tipus de data utilitzat per cridar la operació, i el tipus que retorna.



¡PREGUNTA D'EXAMEN!

Com estan relacionats WSDL i UDDI?

3.2 Serveis web. SOAP

Simple Object Access Protocol (SOAP)

- En l'actualitat les aplicacions distribuïdes es comuniquen a partir de Remote Procedure Calls (RPC) entre objectes com DCOM i CORBA.

- HTTP i la web no es varen dissenyar per aquest tipus de comunicació. RPC poden representar un problema de compatibilitat i seguretat, i els tallafocs i "proxies" els poden bloquejar.

- SOAP es va crear com una altra forma de comunicació entre aplicacions utilitzant HTTP. Per què? Perquè HTTP és suportat per tots els navegadors i servidors i proporciona una forma de comunicació entre diferents llenguatges de programació, arquitectures, i tecnologies.

- Un missatge SOAP utilitza XML i es composa dels següents elements:
    - *"Envelope":* identifica el document XML com a un missatge SOAP.
    - *"Header":* informació específica de l'aplicació.
    - *"Body":* conté informació de la crida i/o dades de resposta.
    - *"Fault":* conté errors i informació d'estat.

3.2 Serveis web. SOAP

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  ....
</soap:Envelope>
```

L'"envelope" és l'element arrel del document. L'atribut xmlns:soap sempre ha de tenir el mateix contingut, mentre que soap:encodingStyle s'utilitza per definir el tipus de dades del document.

3.2 Serveis web. SOAP

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
        <m:Trans xmlns:m="http://www.stockMarket.com/transaction/"
          soap:mustUnderstand="1">234</m:Trans>
    </soap:Header>

    ....
</soap:Envelope>
```

La "header" és el primer element del document. Cada element del "header" pot tenir un dels següents atributs:

- *mustUnderstand:* indica si el seu processament és opcional o obligatori

- *actor:* indica a qui va dirigit el missatge, ja que pot passar per varis servidors.

- *encodingStyle:* indica el tipus de dades utilitzat en el document.

3.2 Serveis web. SOAP

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
        <m:Trans xmlns:m="http://www.stockMarket.com/transaction/"
          soap:mustUnderstand="1">234</m:Trans>
    </soap:Header>

    <soap:Body>
        <m:GetPrice xmlns:m="http://www.stockMarket.com/price">
            <m:Item>Apple</m:Item>
        </m:GetPrice>
    </soap:Body>

</soap:Envelope>
```

El "body" conté el missatge. Els seus elements són específics de l'aplicació. Poden contenir la petició, o la resposta.

3.2 Serveis web. SOAP

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
        <m:Trans xmlns:m="http://www.stockMarket.com/transaction/"
          soap:mustUnderstand="1">234</m:Trans>
    </soap:Header>

    <soap:Body>
        <m:GetPriceResponse xmlns:m="http://www.stockMarket.com/price">
            <m:Price>1.90</m:Price>
        </m:GetPriceResponse>
    </soap:Body>

</soap:Envelope>
```

El "body" conté el missatge. Els seus elements són específics de l'aplicació. Poden contenir la petició, o la resposta.

3.2 Serveis web. SOAP

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
        <m:Trans xmlns:m="http://www.stockMarket.com/transaction/"
          soap:mustUnderstand="1">234</m:Trans>
    </soap:Header>

    <soap:Body>
        <m:GetPriceResponse xmlns:m="http://www.stockMarket.com/price">
            <m:Price>1.90</m:Price>
        </m:GetPriceResponse>

        <soap:Fault>
            <faultcode>VersionMismatch</faultcode>
        </soap:Fault>

    </soap:Body>

</soap:Envelope>
```

L'element "Fault" és opcional i serveix per indicar missatges. Pot contenir els quatre elements:

<faultcode>: codi identificador de l'error.

<faultstring>: explicació de l'error.

<faultactor>: qui ha causat l'error.

<detail>: error específic de l'aplicació.

3.2 Serveis web. SOAP

```
POST /transaction HTTP/1.1
Host: www.stockMarket.com
Content-type: application/soap+xml
Content-Length: 210
```

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
        <m:Trans xmlns:m="http://www.stockMarket.com/transaction/"
          soap:mustUnderstand="1">234</m:Trans>
    </soap:Header>

    <soap:Body>
        <m:GetPrice xmlns:m="http://www.stockMarket.com/price">
            <m:Item>Apple</m:Item>
        </m:GetPrice>

        <soap:Fault>
            <faultcode>VersionMismatch</faultcode>
        </soap:Fault>

    </soap:Body>

</soap:Envelope>
```

Sobre HTTP, el missatge SOAP es transmet com una petició de tipus POST.

3.2 Serveis web. SOAP

```
HTTP/1.1 200 OK
Content-type: application/soap+xml
Content-Length: 210
```

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
        <m:Trans xmlns:m="http://www.stockMarket.com/transaction/"
          soap:mustUnderstand="1">234</m:Trans>
    </soap:Header>

    <soap:Body>
        <m:GetPriceResponse xmlns:m="http://www.stockMarket.com/price">
            <m:Price>1.90</m:Price>
        </m:GetPriceResponse>

        <soap:Fault>
            <faultcode>VersionMismatch</faultcode>
        </soap:Fault>

    </soap:Body>

</soap:Envelope>
```

Un cop el servidor HTTP rep la petició, la passa a l'aplicació corresponent i respon.

¡PREGUNTA D'EXAMEN!

Quines avantatges té SOAP respecte a RMI o RPC?




3. PROTOCOLS DE SERVEIS

3.1 – TRANSPORT DE FITXERS I MISSATGES

3.2 – SERVEIS WEB

3.3 – REPRESENTATIONAL STATE TRANSFER (REST)

- **Utilització de mètodes HTTP**
- **Sense estat**
- **Exposició de l'estructura de directori**
- **Transferència de contingut amb XML o JSON**




3.3 REST. Introducció

- REST defineix una sèrie de principis arquitectònics amb els quals definir serveis web. És un model de disseny per aplicacions web que utilitza adequadament els protocols de la web.

- Qualsevol implementació d'un servei web basat en REST segueix quatre principis de disseny bàsics:

1. S'utilitzen únicament mètodes HTTP
2. No es manté l'estat
3. S'exposa l'estructura de directori
4. Es transfereix el contingut amb XML, JavaScript Object Notation (JSON), o amb els dos

- La principal diferència entre REST i SOAP és que SOAP és completament independent de HTTP, és a dir, no utilitza els mètodes definits a HTTP sinó que implementa els seus propis. Aquest aspecte té avantatges i inconvenients: SOAP es pot utilitzar sobre qualsevol altre protocol, és més flexible, però l'arquitectura de les aplicacions no està estructurada per la web i per tant no aprofita els mecanismes existents.

¡PREGUNTA D'EXAMEN!

Quina és la principal diferència entre SOAP i REST?

3.3 REST. Utilització de mètodes HTTP

- Els diferents mètodes HTTP s'utilitzen de forma explícita, és a dir, el seu sentit és el definit en el protocol (RFC 2616). Per exemple, HTTP GET és un mètode de producció de dades que utilitza un client per estirar un recurs.

- Amb REST, els mètodes HTTP s'utilitzen com:

  - *GET:* per estirar un recurs del servidor.
  - *POST:* per crear un recurs en el servidor.
  - *PUT:* canviar l'estat d'un recurs o actualitzar-lo.
  - *DELETE:* per eliminar un recurs.

- Per què? Perquè un error de disseny en moltes aplicacions web és l'ús de mètodes HTTP de forma incorrecte, e.g.,

GET /adduser?name=Alice HTTP/1.1

Aquesta petició utilitza el mètode GET per fer una crida transaccional, i.e., afegir dades en una base de dades. Això és inconsistent amb la definició del protocol HTTP i per tant pot tenir efectes col·laterals.

3.3 REST. Utilització de mètodes HTTP

- Els diferents mètodes HTTP s'utilitzen de forma explícita, és a dir, el seu sentit és el definit en el protocol (RFC 2616). Per exemple, HTTP GET és un mètode de producció de dades que utilitza un client per estirar un recurs.

- Amb REST, els mètodes HTTP s'utilitzen com:
  - *GET:* per estirar un recurs del servidor.
  - *POST:* per crear un recurs en el servidor.
  - *PUT:* canviar l'estat d'un recurs o actualitzar-lo.
  - *DELETE:* per eliminar un recurs.

- La forma correcte ("RESTfully") d'utilitzar aquesta petició seria:

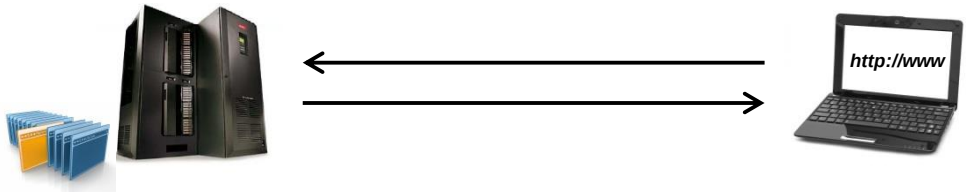
```
POST /users HTTP/1.1
Host: example.com
Content-Type: application/xml

<?xml version="1.0"?>
<user>
   <name>Alice</name>
</user>
```

¡PREGUNTA D'EXAMEN!

Quins efectes col·laterals pot tenir l'ús inconsistent de mètodes HTTP GET?

3.3 REST. Sense estat

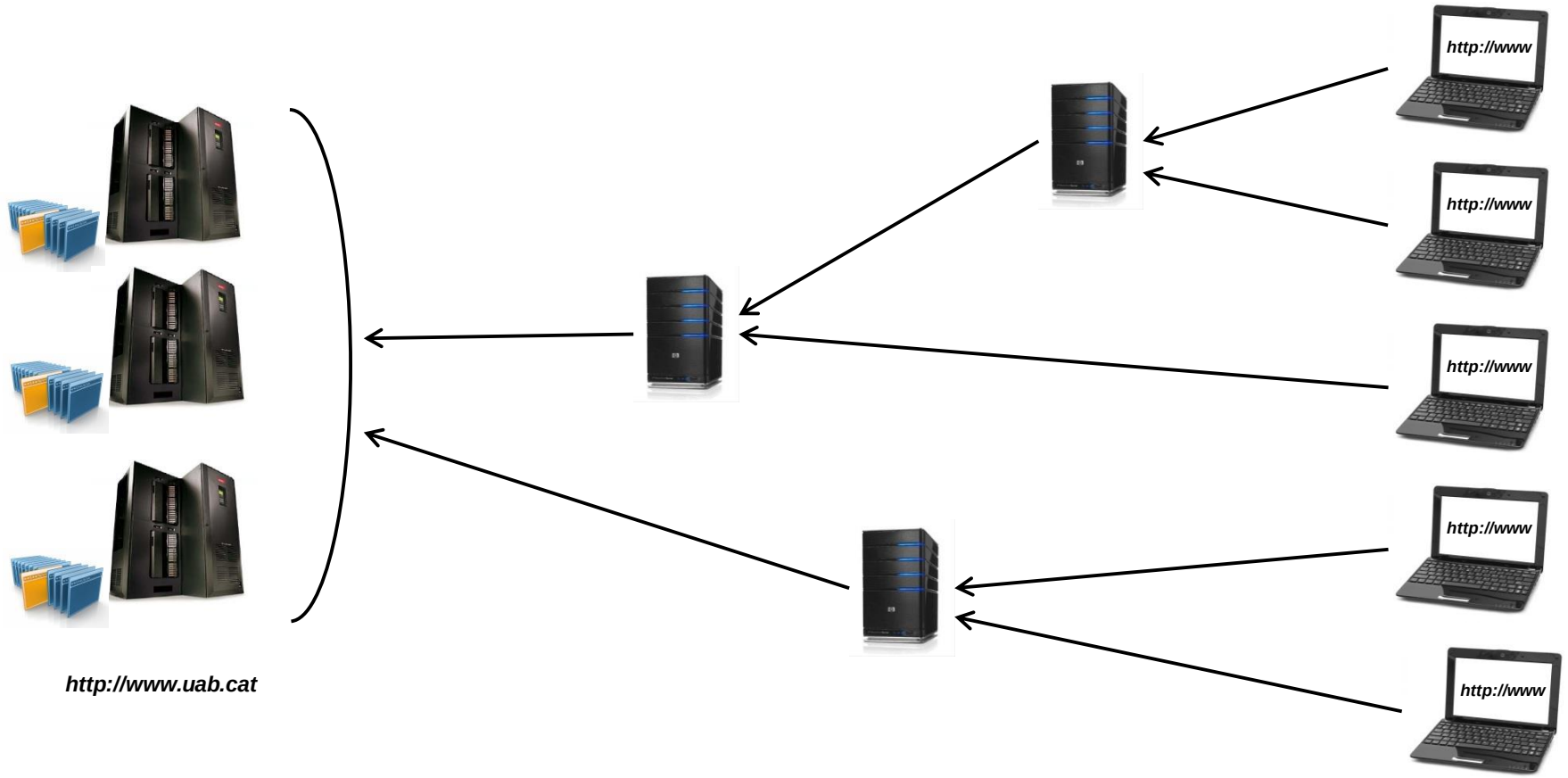
- Els serveis web es necessiten escalar per poder satisfer les demandes de rendiment. Això acaba esdevenint amb "clusters" de servidors amb capacitat de balanceig de càrrega, sistemes jeràrquics de "proxies", etc.



http://www.uab.cat

3.3 REST. Sense estat

- Els serveis web es necessiten escalar per poder satisfer les demandes de rendiment. Això acaba esdevenint amb "clusters" de servidors amb capacitat de balanceig de càrrega, sistemes jeràrquics de "proxies", etc.

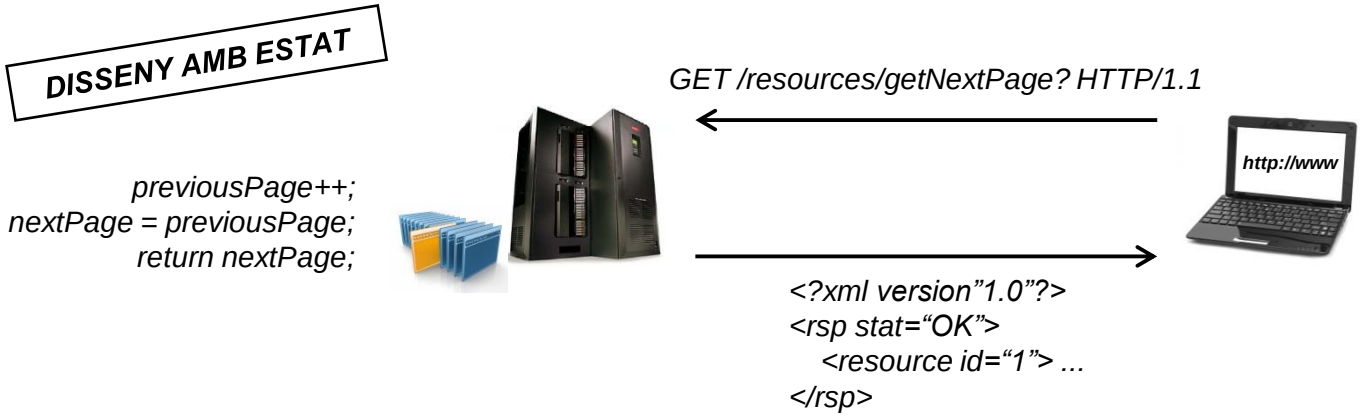


3.3 REST. Sense estat

- Els serveis web es necessiten escalar per poder satisfer les demandes de rendiment. Això acaba esdevenint amb "clusters" de servidors amb capacitat de balanceig de càrrega, sistemes jeràrquics de "proxies", etc.

- Perquè la utilització de servidors entremig del client i servidor web i/o "proxies" és necessari que els clients enviïn peticions complertes i independents, és a dir, que la petició inclogui totes les dades perquè es puguin respondre completament sense necessitar altres dades. Això permet que els servidors enmig puguin balancejar, enrutar, o re-enviar la petició allà on més convingui.

- Les peticions NO haurien de necessitar recuperar cap estat del servidor. Això simplifica el disseny i la implementació.

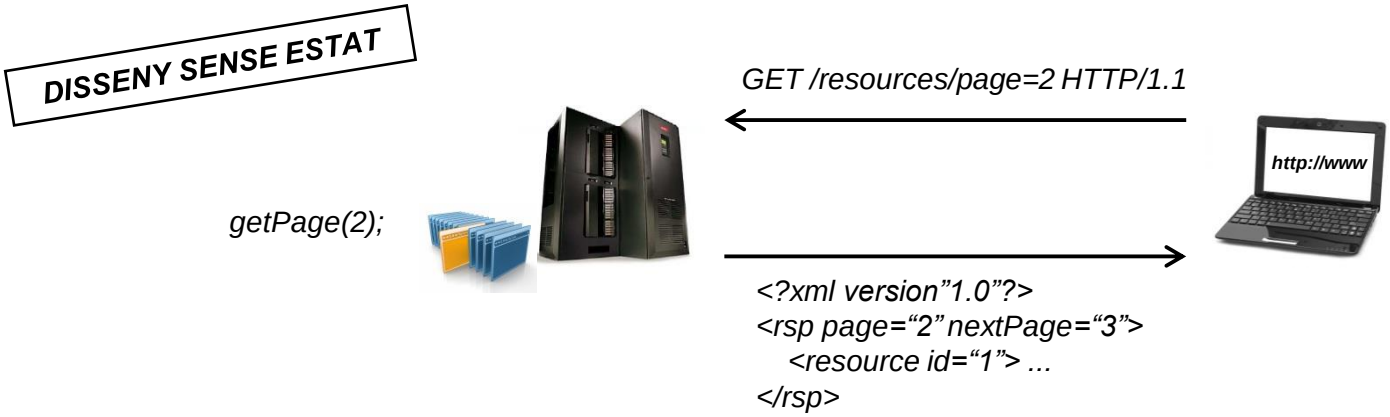


3.3 REST. Sense estat

- Els serveis web es necessiten escalar per poder satisfer les demandes de rendiment. Això acaba esdevenint amb "clusters" de servidors amb capacitat de balanceig de càrrega, sistemes jeràrquics de "proxies", etc.

- Perquè la utilització de servidors entremig del client i servidor web i/o "proxies" és necessari que els clients enviïn peticions complertes i independents, és a dir, que la petició inclogui totes les dades perquè es puguin respondre completament sense necessitar altres dades. Això permet que els servidors enmig puguin balancejar, enrutar, o re-enviar la petició allà on més convingui.

- Les peticions NO haurien de necessitar recuperar cap estat del servidor. Això simplifica el disseny i la implementació.



3.3 REST. Exposició de l'estructura de directori

- L'estructura de la URI ha de ser senzilla i intuïtiva, com si fos un interfície auto-continguda que no necessités cap explicació.

- Una forma d'aconseguir aquest objectiu és utilitzar URIs amb una estructura semblant a la de directoris. D'aquesta forma, la URI té forma d'arbre, amb branques i fulles.

http://www.forumworld.org/discussions/topics/rest
✖ *http://www.forumworld.org/getDiscussion?topic=rest*

- Algunes recomanacions d'ús són:
  - Amagar la tecnologia d'"scripting" del servidor (e.g., .jsp, .pgp, .asp) per poder portar l'aplicació a altres tecnologies sense canviar les URIs.
  - Utilitzar "lowercase".
  - Substituir espais per "_" o "~".
  - Evitar l'ús de "queries" en la URI tant com sigui possible.
  - Enlloc d'utilitzar el codi "404 Not Found", retornar un recurs per defecte.

3.3 REST. Transferència de contingut amb XML o JSON

- Un recurs és una representació de la informació en el moment en què el client fa la petició.

- Com a tal, el recurs s'ha de poder mantenir de forma senzilla i s'ha de transmetre utilitzant els estàndards de transmissió de contingut.

- El contingut ha d'estar estructurat utilitzant les relacions que hi ha entre ell, de forma que es pugui utilitzar l'estructura jeràrquica dels documents XML.

- És recomanable construir les webs de forma que els seus continguts estiguin disponibles amb diferents formats MIME com application/json, application/xml, application/xhtml+xml. D'aquesta forma les capacitats de negociació de HTTP es poden utilitzar i es pot respondre els clients de forma més adequada.

¡PREGUNTA D'EXAMEN!

Es poden utilitzar "cookies" en una aplicació RESTful?

¡PREGUNTA D'EXAMEN!

Quins protocols d'aplicació s'han vist a classe? Situa'ls a la pila de protocols TCP/IP i digues si s'utilitzen a Internet, la Web o a ambdós.