El nivell d'aplicació

Joan Manuel Marquès Puig
Silvia Llorente Viejo

PID_00147706

Universitat Oberta de Catalunya
www.uoc.edu

Índex

Introducció

En aquest mòdul didàctic es descriu tota una sèrie d'aplicacions que utilitzen Internet com a mitjà de comunicació i també els protocols de comunicacions que tenen associats. Aquestes aplicacions es coneixen com a *aplicacions distribuïdes*, ja que estan formades per diferents parts i cadascuna es troba en màquines diferents.

Normalment, hi ha una part anomenada *servidor* que s'executa en un ordinador, a la qual es connecten els diferents clients (que es troben en altres ordinadors remots) per tal de demanar-ne els serveis (generalment, sol·liciten l'execució d'algun tipus d'operació).

En aquest mòdul, veurem primerament alguns conceptes bàsics referents a les aplicacions distribuïdes i després, una sèrie d'aplicacions distribuïdes a Internet, de les quals estudiarem les característiques següents:

- **El model**: descrivim els diferents elements que formen l'aplicació i les seves característiques.

- **El protocol**: exposem les idees principals del protocol de comunicacions de cada aplicació.

- **El model d'informació**: descrivim el format d'informació dels protocols que en tenen un de concret.

- **La funcionalitat**: expliquem la funcionalitat i les instruccions que la proporcionen.

Incloem exemples breus de l'intercanvi d'instruccions per a facilitar-ne la comprensió.

Objectius

En aquest mòdul didàctic trobareu les eines necessàries per a assolir els objectius següents:

1. Comprendre el model client-servidor i el model d'igual a igual, que serveixen com a base de la implementació d'aplicacions distribuïdes.

2. Conèixer les aplicacions distribuïdes més utilitzades a Internet i entendre els principis bàsics de funcionament dels protocols de comunicació que usen.

1. Arquitectures d'aplicacions distribuïdes

Hi ha moltes definicions d'*aplicacions distribuïdes*. Aquí us en posem una que creiem que encaixa molt bé amb l'enfocament d'aquest mòdul:

> Una aplicació distribuïda està formada per una col·lecció d'ordinadors autònoms enllaçats per una xarxa d'ordinadors i suportats per un programari que fa que la col·lecció actuï com un servei integrat.

Una arquitectura de programari determina com s'identifiquen i s'assignen els components del sistema, com interactuen els components per a formar el sistema, la quantitat i la granularitat de la comunicació que es necessita per a la interacció, i els protocols de la interfície usada per la comunicació.

L'arquitectura per utilitzar en cada cas depèn de molts factors. Ara esmentarem les característiques més significatives que cal considerar quan es dissenya una aplicació a escala Internet.

La primera característica és la **gran quantitat tant d'ordinadors com d'usuaris** que hi ha a Internet. Relacionat amb això, hi ha la seva dispersió geogràfica. La **dispersió geogràfica** afecta la manera en què els components del grup perceben les accions que passen a l'aplicació. Per exemple, els usuaris de l'aplicació poden percebre dues accions que passen en dos extrems oposats del món en diferent ordre, segons la proximitat de cada component al component generador de l'acció. Segons l'aplicació ens caldran o no mecanismes per a abordar aquestes situacions.

Una tercera característica és **l'autonomia** dels diferents ordinadors que formen l'aplicació. És molt habitual que diferents parts d'una aplicació estiguin sota dominis administrats per diferents administradors. Relacionada amb aquesta característica hi ha la **seguretat**. Cada organització té unes polítiques de seguretat diferents, com ara tallafocs. Cal que les aplicacions distribuïdes que dissenyem es puguin executar tenint en compte aquestes restriccions imposades pels diferents administradors.

Una quarta característica és la **qualitat de servei**. En una aplicació distribuïda a escala Internet aquest és un aspecte fonamental que estarà molt condicionat per la latència de la xarxa, els talls i altres fenòmens que la puguin afectar.

Finalment, hi ha els aspectes relacionats amb la **mobilitat**: dispositius que es connecten i desconnecten, accés des de diferents ubicacions, treball en mode desconnectat, qualitat d'accés menor (amplada de banda, fiabilitat...). Aquest darrer aspecte, però, millorarà molt a mesura que aquests serveis es generalitzin.

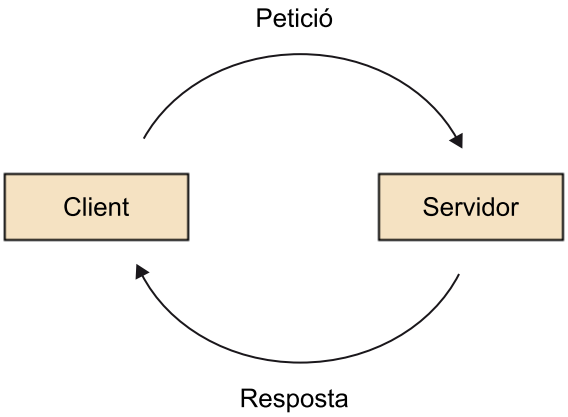
En aquest mòdul tractarem les dues arquitectures distribuïdes més utilitzades avui dia: client-servidor i d'igual a igual[1]. Hi ha altres arquitectures distribuïdes, com ara la graella, la publicació subscripció, o el codi mòbil, entre d'altres, que no tractarem en aquest material.

(1) *D'igual a igual* en anglès s'anomena ***peer-to-peer*** o ***p2p***.

1.1. Client-servidor (*client/server* en anglès)

En el model client-servidor hi ha dos tipus de components:

- **Clients**: fan peticions de servei. Normalment els clients inicien la comunicació amb el servidor.
- **Servidors**: proveeixen serveis. Normalment els servidors esperen rebre peticions. Un cop han rebut una petició, la resolen i retornen el resultat al client.

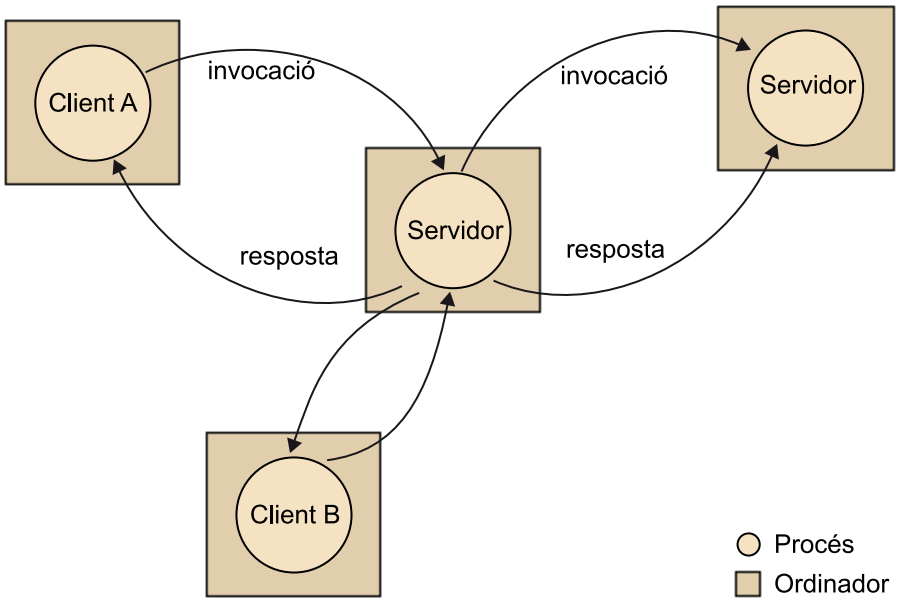


Aquesta idea es pot aplicar de manera molt variada a molts tipus diferents d'aplicacions. Un exemple que ho ajudarà a entendre és el Web. El navegador és el client i els ordinadors als quals ens connectem i dels quals obtenim les pàgines són els servidors.

Els servidors poden ser amb estat o sense estat. Un servidor sense estat no manté cap informació entre peticions. Un servidor amb estat pot recordar informació entre peticions. L'abast d'aquesta informació pot ser global o específica d'una sessió. Un servidor web amb pàgines web estàtiques és un exemple

de servidor sense estat, mentre que un servidor que generi les pàgines dinàmicament (com ara la Intrauoc del campus virtual de la UOC) és un exemple de servidor amb estat.

Un servidor també pot ser client d'altres servidors, tal com es veu en la figura següent. Per exemple, una aplicació de correu via Web actua com a servidor per al navegador i com a client del servidor de correu que gestiona els missatges de l'usuari en qüestió. Els servidors web i els altres serveis disponibles a Internet són clients del servei de resolució de noms (DNS). Un tercer exemple són els cercadors (*search engines*), que permeten als usuaris d'accedir a sumaris d'informació de pàgines web escampades per molts llocs web d'arreu d'Internet. Un cercador és alhora servidor i client: respon a peticions provinents dels navegadors clients i executa programes que, actuant com a clients (robots web; en anglès, *web crawlers*), accedeixen a servidors d'Internet cercant informació. De fet, un cercador inclourà diferents fils d'execució, alguns servint el client i els altres executant robots web.

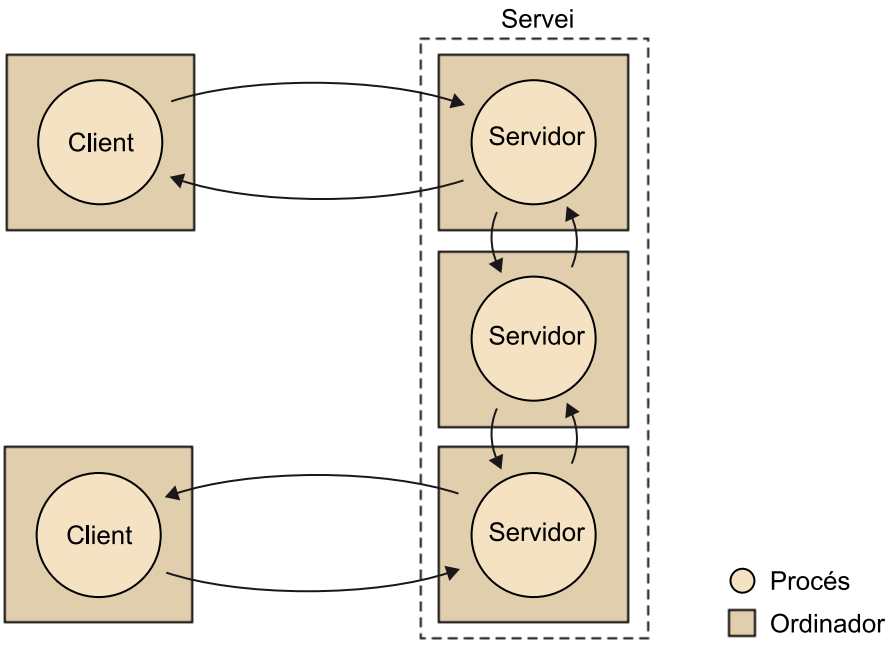


Els serveis també es poden implementar com a diferents processos servidors que s'executen en diferents ordinadors i que interactuen per a proporcionar un servei a processos clients següent. Els servidors es poden repartir els diferents objectes que componen el servei que proporcionen o poden mantenir rèpliques dels objectes en diversos ordinadors.

Exemple de partició de dades

El web proporciona un exemple habitual de partició de les dades, en què cada servidor gestiona el seu conjunt de recursos. Un usuari utilitza un navegador per a accedir a un recurs situat en qualsevol dels servidors.

La reproducció s'usa per a incrementar el rendiment i la disponibilitat i per a millorar la tolerància a fallades. Proporciona múltiples còpies consistents de les dades en processos que s'executen en diferents ordinadors. Per exemple, el web proporcionat a www.google.com (o www.uoc.edu) està mapat en diferents servidors que tenen dades reproduïdes.

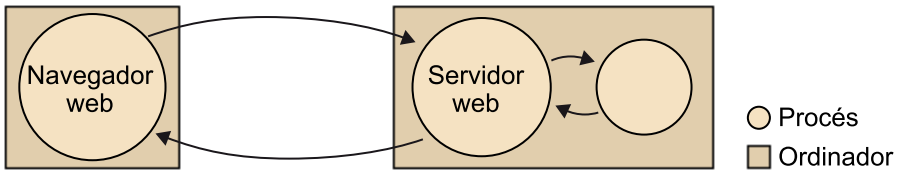


1.1.1. Aplicacions basades en el Web

Un cas particular d'aplicacions client-servidor són les aplicacions que s'executen aprofitant l'arquitectura del Web. Aquestes aplicacions es basen en el fet de tenir tota la capacitat de processament en un servidor web (o conjunt de servidors), als quals s'accedeix des d'un navegador web. Quan l'usuari fa clic sobre un enllaç de la pàgina web que té al seu navegador, es genera una petició al servidor que conté la informació. El servidor, en rebre la petició, retorna la pàgina demanada si la petició era a una pàgina, o retorna el resultat d'executar una aplicació si l'enllaç corresponia a un codi per executar (per exemple, CGI o Servlet). El navegador web proporciona a l'aplicació un entorn per a presentar la informació. La comunicació entre client i servidor es fa utilitzant el protocol HTTP. El resultat que retorna el servidor al client s'envia en format HTML, de manera que per al client web és totalment transparent si accedeix a una pàgina web o a una aplicació que genera un resultat formatat en HTML.

Les aplicacions basades en el Web tenen l'avantatge que són accessibles des de qualsevol ordinador que disposi d'un navegador (la pràctica totalitat dels ordinadors connectats avui dia a Internet) sense haver de tenir res més instal·lat a l'ordinador local. L'ús d'aquestes arquitectures també facilita el disseny de les aplicacions, ja que no cal implementar la comunicació entre el client i el servidor (s'utilitza el protocol HTTP), i la part de presentació de l'aplicació es facilita molt pel fet de formatar el document en HTML i aprofitar les funcionalitats que proporciona el navegador (com els intèrprets de Javascript i Java).

La facilitat i universalitat en l'accés a les aplicacions que proporciona aquesta arquitectura és la base dels serveis oferts a Internet. Alguns exemples són el campus virtual de la UOC, els servidors de correu web tipus Yahoo o Google i els bancs per Internet.



1.2. D'igual a igual (*peer-to-peer* en anglès)

D'una manera molt genèrica, podem dir que un sistema d'igual a igual es caracteritza per ser un sistema distribuït en què tots els nodes tenen les mateixes capacitats i responsabilitats, i en què tota la comunicació és simètrica.

A Internet, des dels seus orígens, hi ha hagut sistemes i aplicacions que s'han comportat seguint la filosofia d'igual a igual. Aquests sistemes s'han caracteritzat pel fet de ser totalment descentralitzats, de gran escala i amb capacitat per autoorganitzar-se. L'exemple paradigmàtic són els missatges Usenet.

Els missatges Usenet

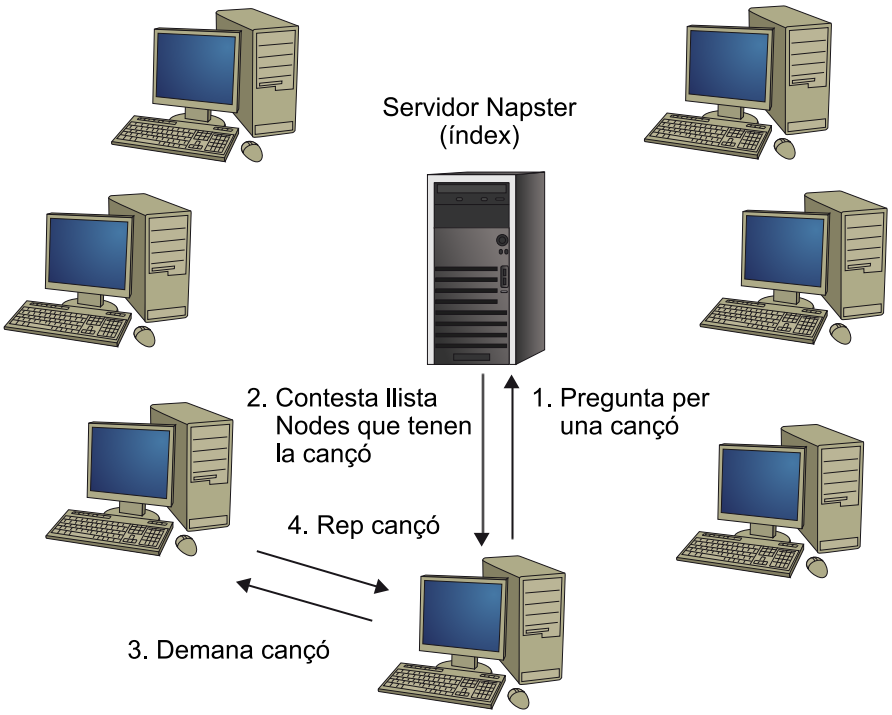
Els missatges Usenet (*Usenet news*, en anglès) és un sistema global i descentralitzat de grups de discussió a Internet. Els usuaris llegeixen i posen missatges que semblen correus electrònics (s'anomenen anomena *articles*) a un nombre determinat de grups de notícies, que estan organitzats formant jerarquies lògiques de temes (per exemple, informàtica.linux.distribucions o informàtica.llenguatgesProgramació.tutorials). Els missatges es distribueixen entre un gran nombre de servidors, que emmagatzemen i es reenvien missatges els uns als altres. Els usuaris es baixen els missatges i en posen de nous en un dels servidors. L'intercanvi de missatges entre els servidors fa que a la llarga els missatges arribin a tots els servidors.

Tot i això, el fenomen d'igual a igual comença com a tal al final dels anys noranta de la mà del Napster. En l'època de l'explosió d'Internet –a partir de 1994– el vessant de col·laboració que havia dominat Internet fins aquell moment va començar a perdre protagonisme enfront del vessant més comercial, que imposava l'arquitectura client-servidor, liderada pel Web com a arquitectura estrella.

Dins d'aquest context dominat per la centralització de l'arquitectura client-servidor, els usuaris del Napster van descobrir que l'efecte agregat de posar cada individu cançons al servei d'una comunitat era que els participants de la comunitat trobaven amb facilitat les cançons que els interessaven.

El funcionament del Napster era molt senzill. Usava un servidor (o índex) per a proporcionar un servei de directori. Quan un usuari arrencava un node del Napster, aquest es connectava al servidor i hi publicava la llista de cançons que el node local feia pública. D'aquesta manera, el servei de directori sabia, per cada igual, quins objectes tenia disponibles per compartir. Quan algú buscava

una cançó feia una petició de la cançó al servidor i aquest li contestava la llista de nodes que tenien un títol similar. L'usuari n'escollia un i es baixava la cançó directament d'aquest node.



Els grans canvis que aporta el Napster, tant des del punt de vista de l'arquitectura com del funcionament del sistema, respecte a les solucions centralitzades (client-servidor) que predominaven en aquell moment, són:

- Els fitxers que hi ha disponibles són els que els usuaris, de manera individual, decideixen aportar al sistema (autonomia dels usuaris).

- La disponibilitat d'un fitxer depèn de si els usuaris que el tenen estan connectats al sistema (connectivitat puntual o *ad hoc*, en anglès).

- Hi ha molts usuaris que proporcionen un mateix fitxer (tolerància a fallades).

- Els recursos necessaris per a emmagatzemar els fitxers els aporten els usuaris mateixos (cost del sistema).

- El sistema evoluciona i s'adapta a mesura que els usuaris es connecten o desconnecten (autoorganització).

- El sistema suporta molts usuaris (escalabilitat). De fet, hi va arribar a haver milions d'usuaris connectats al Napster.

- En haver-hi moltes reproduccions d'una cançó, la càrrega està repartida (millora de rendiment).

Encara que hi hagués un servidor o índex, es considera que el Napster era un sistema d'igual a igual, perquè els fitxers es trobaven en els ordinadors dels usuaris i la baixada es feia directament entre l'ordinador que buscava la cançó i el que l'oferia.

Aquesta manera d'organitzar grans quantitats d'informació a escala Internet per a facilitar-ne la compartició va resultar molt eficaç. La prova d'això la trobem tant en el fet que el sistema va arribar a tenir milions d'usuaris com en el fet que moltes empreses discogràfiques ho van veure com una amenaça. Precisament, el motiu que el Napster deixés de funcionar va ser que van denunciar els propietaris del servei de directori per infringir les lleis del *copyright*. El cas Napster va acabar amb una sentència judicial que va forçar el tancament del servidor índex.

Arran de l'èxit de la solució, molta gent es va animar a fer propostes més descentralitzades i que superessin les limitacions del Napster. Gnutella n'és un exemple: es basa en un algorisme d'inundació per a localitzar el fitxer que es busca i d'aquesta manera elimina el punt únic de fallada que representa tenir un servei centralitzat de directori. Un cop localitzat el fitxer, la baixada es fa directament entre qui vol el fitxer i qui el proporciona. Aquesta solució té l'inconvenient que no és determinista.

Determinisme

Per *determinisme* entenem que diferents execucions d'una mateixa operació donin el mateix resultat. Els sistemes de tipus Gnutella no són deterministes. L'algorisme que utilitzen per a localitzar fitxers dins el sistema no garanteix que si un fitxer està en algun dels iguals el trobi. Pot ser que, segons el camí que hagi seguit la consulta, ens digui que no l'ha trobat, quan sí que hi és.

1.2.1. Aplicacions d'igual a igual

Els sistemes i aplicacions d'igual a igual s'han fet populars de la mà de les aplicacions de compartició de fitxers, però hi ha altres tipus d'aplicacions. Skype és un altre sistema de tipus d'igual a igual que és molt popular. Skype proporciona telefonia a Internet. Utilitza un protocol de propietat, de la implementació del qual es coneixen poques coses. Funciona seguint una organització amb superiguals, tal com fa KaZaA. De fet, Skype va ser fundada pels fundadors de KaZaA. Un aspecte que cal destacar és que aconsegueix superar els problemes que tenen els iguals quan són darrere d'un tallafocs o els problemes derivats del NAT (*network address translation*).

Vegeu també

Les aplicacions d'igual a igual per a la compartició de fitxers com KaZaA estan explicades a l'apartat "Aplicacions d'igual a igual per la compartició de fitxers" d'aquest mòdul didàctic.

També hi ha altres sistemes d'igual a igual per a la comunicació síncrona (com ara la missatgeria instantània), jocs, sistemes de processament distribuït (com SETI@home) o programari per a la col·laboració (com ara Groove).

SETI@home

SETI@home és un projecte que té com a objectiu detectar vida intel·ligent fora de la Terra. Distribueix processament entre molts ordinadors personals que estan subscrits al projecte. Analitza dades de radiotelescopis aprofitant les grans quantitats de temps de processament que els PC desaprofiten perquè no fan res.

Bibliografia complementària

Trobareu més informació sobre el funcionament intern de Skype a:

S. A. Baset; H. Schulzrinne (2006, abril). "An analysis of the Skype peer-to-peer internet telephony protocol". *Proceedings of IEEE INFOCOM* 2006. Barcelona.

Groove

Groove és un sistema d'igual a igual per a facilitar la col·laboració i comunicació en grups petits. Proporciona eines per a la compartició de fitxers, la missatgeria instantània, el calendari, la gestió de projectes, etc.

1.3. Avantatges i desavantatges dels models client-servidor i d'igual a igual

Com s'ha vist, client-servidor i d'igual a igual són dues maneres de plantejar el disseny d'una aplicació. També s'ha esmentat que hi ha altres paradigmes. Aquest esforç de caracterització, però, no ens ha de confondre. A l'hora de la veritat, les aplicacions no implementen mai una arquitectura pura i moltes vegades són híbrids entre diferents models. Cada aplicació té les seves necessitats i cal utilitzar les característiques que ens ofereix cada paradigma per a construir una aplicació que satisfaci les necessitats dels seus usuaris.

La taula següent pretén resumir els avantatges i desavantatges tant del client-servidor com de l'igual a igual. No pretén ser exhaustiva. A més, parlar d'avantatges i desavantatges és molt personal o dependrà de cada situació. El que per a algú o en una situació és un avantatge, per a algú altre o en una altra situació és un desavantatge. I a l'inrevés passa el mateix.

	Avantatges	Desavantatges
Client-servidor	• Dades en els servidors, cosa que en facilita el control de la seguretat, control d'accés, consistència de la informació... • Hi ha moltes tecnologies madures per als sistemes client-servidor. Això fa que hi hagi solucions molt provades i que garanteixen la seguretat, facilitat d'ús i amigabilitat de la interfície. • Relativa facilitat per a actualitzar els elements d'un sistema client-servidor. • El desenvolupament d'aplicacions centralitzades és més senzill que les descentralitzades. • Hi ha uns responsables de garantir el servei, que són els proveïdors del servei (o de cadascuna de les parts d'aquests).	• La centralització del model: punt únic de fallada, dependència de l'autoritat administrativa que proveeix el servei, vulnerabilitat a atacs. • Escalabilitat condicionada a la capacitat de fer créixer el servidor o conjunt de servidors que proporcionen el servei. Relacionat amb això: és molt costós construir un sistema que suporti càrregues altes de continguts pesants. • Quan hi ha molts clients que fan peticions el servidor es pot congestionar. En entorns en què la demanda pot ser molt variable o incerta, cal sobredimensionar el servei per a atendre càrregues que potser només ocorren esporàdicament o assumir que en certes situacions hi pot haver congestió.
D'igual a igual	• Descentralització: els diferents membres (iguals) del sistema són els propietaris i tenen el control de les dades i els recursos del seu ordinador. • El cost de la propietat (o la major part d'aquesta si és un sistema d'igual a igual híbrid) està repartit entre els usuaris. • Escalable: atès que les operacions es fan directament entre els iguals, el sistema pot suportar més operacions que si hi hagués un node que centralitzés les operacions. • Autoorganització: no cal que un component del sistema supervisi l'evolució del sistema quan en canvia l'escala (augmenten els usuaris o la càrrega), hi ha fallades o els iguals es connecten o desconnecten.	• El comportament descentralitzat és complex d'implementar. • La responsabilitat del sistema està difosa entre els usuaris: si els usuaris no aporten recursos suficients, el sistema no pot funcionar. • Tecnologia molt nova. Encara que hi ha sistemes molt usats i que funcionen bé, les tècniques utilitzades encara han de madurar • El dinamisme del sistema fa que el sistema sigui utilitzable només si la seva disponibilitat satisfà les necessitats dels usuaris. En la major part dels casos això passa per la reproducció (tant d'informació com de capacitat de processament). • Seguretat: als problemes habituals dels sistemes distribuïts cal afegir-hi mecanismes perquè la informació que s'aporti sigui vàlida, que no hi hagi usuaris que obtinguin informació sense aportar-ne, etc. Aquests aspectes encara no estan ben resolts.

1.4. Requisits de les aplicacions

Les aplicacions dialoguen fent servir un protocol de transport. En terminologia de nivells, es diu que el transport ofereix uns serveis a l'aplicació, o dit al revés, l'aplicació requereix al nivell de transport unes capacitats. En general, aquestes capacitats giren entorn de tres eixos:

- Transferència fiable: algunes aplicacions, com el correu electrònic o la transferència d'arxius, necessiten que la informació que viatja arribi, i arribi bé. La fiabilitat de la transferència és, llavors, un requisit crucial. En canvi, hi ha aplicacions que es basen en la transmissió de veu o imatge en temps real que no necessiten aquesta fiabilitat, perquè si es perd un o diversos paquets, la reproducció del so o la imatge original encara és possible, per bé que amb defectes acceptables.

- Amplada de banda: una transmissió d'imatge en temps real necessita una amplada de banda concreta perquè es pugui fer de manera acceptable, però l'enviament d'un missatge de correu electrònic es pot fer amb qualsevol amplada de banda, per petita que sigui.

- Temporització: les aplicacions que permeten comunicacions en temps real, ja sigui amb so, imatge, o tots dos, necessiten que el ritme de recepció dels paquets sigui constant, perquè en la recepció es pugui reproduir el so o la imatge original de manera contínua, tal com es van captar en origen.

Per tant, cada aplicació té uns requisits diferents, i per això en el nivell de transport hi ha diferents opcions per a satisfer-los. Els protocols originals d'Internet (TCP i UDP) es concentraven en el primer requisit, perquè les primeres aplicacions que es van especificar no eren de temps real. A mesura que s'han desenvolupat aquest tipus d'aplicacions, s'han especificat nous protocols de transport que ofereixen els serveis adients.

2. DNS: Servei de noms a Internet

Els ordinadors connectats a Internet estan identificats de manera única per la seva adreça IP. Les adreces IP són difícils de recordar i per això s'utilitzen noms de l'estil de www.uoc.edu o smtp.uoc.edu per a identificar els diferents recursos o ordinadors d'Internet. El sistema de noms de domini[2] és un servei desplegat a Internet que permet fer la traducció entre noms i adreces IP.

[2]En anglès, *Domain Name System* (DNS).

Format de les adreces IP

Les adreces IP tenen l'aspecte següent: 213.73.40.99; cada punt separa un byte del 0 al 255 expressat en notació decimal.

El DNS és un servei que rep milions de peticions i ha de ser capaç de resoldre-les de manera molt ràpida. Per a fer-nos una idea del volum de peticions que ha de ser capaç de gestionar el Akamai usen el DNS[3], només cal que ens fixem en dues de les aplicacions més populars d'Internet: el Web i el correu electrònic. Cada vegada que escrivim una adreça web al navegador, aquest fa una consulta al DNS per a saber a quina adreça IP correspon. Cada vegada que enviem un missatge de correu electrònic, es fa una consulta al DNS per a saber a quin servidor de correu cal enviar el correu corresponent al domini de l'adreça de correu electrònic.

[3]El DNS està especificat en els RFC 1034 i PFC 1035 i actualitzat en altres RFC.

A continuació veurem els passos que se segueixen perquè l'ordinador de l'usuari pugui enviar una petició de pàgina web a l'URL www.uoc.edu/eimt/index.html:

1) L'ordinador de l'usuari és el que executa la part client de l'aplicació DNS.

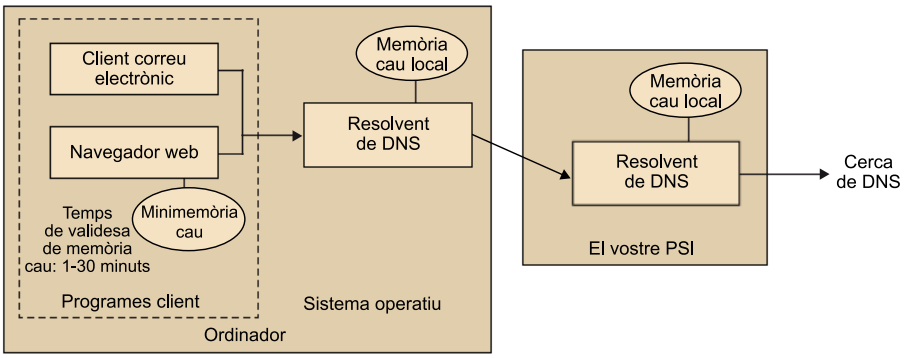
2) El navegador extreu el nom del lloc web –www.uoc.edu en aquest cas– de l'URL i el passa a la part client de l'aplicació DNS.

3) El DNS client envia la petició (que conté el nom del lloc web) a un servidor DNS.

4) En algun moment futur el client DNS rebrà la resposta, que conté l'adreça IP corresponent al lloc web.

5) Quan el navegador rep l'adreça IP del DNS, pot iniciar la connexió TCP amb el servidor HTTP ubicat al port 80 d'aquesta adreça IP.

La figura següent mostra com el correu electrònic o el navegador interactuen amb el DNS.



Us haureu adonat que la pregunta al DNS introdueix un retard addicional –a vegades substancial– a l'aplicació Internet que l'invoca. Afortunadament, com veurem més endavant, l'adreça IP està en forma de cau en algun DNS "proper", cosa que ajuda a reduir el trafic DNS a la xarxa i el retard mitjà del DNS.

Altres serveis importants que proporciona el DNS, a part dels ja vistos de traducció entre el nom de l'ordinador i la seva adreça IP, són:

- **Àlies**. A vegades interessa que un ordinador tingui més d'un nom. En aquest cas, té un nom que s'anomena **canònic**, que és el nom de l'ordinador, i després té altres noms que són els seus àlies. Aquests àlies normalment són més senzills de recordar. En aquest cas es pot invocar al DNS per a obtenir el nom canònic de l'ordinador i la seva adreça IP. En el cas de la UOC, www.uoc.edu és un àlies del nom canònic www-org.uoc.edu.

- **Àlies del servidor de correu**. És important que les adreces de correu electrònic siguin fàcils de recordar. El nom del servidor que gestiona el correu pot ser més complicat de recordar i hi pot haver més d'un servidor que respongui quan es demana per un domini determinat. De fet, el camp MX permet que una institució o empresa tingui el mateix nom (un àlies) per al domini de correu electrònic i per al servidor web. Per exemple, a la UOC tant el servidor web com el domini de correu electrònic es diuen uoc.edu.

- **Distribució de càrrega**. El DNS s'usa també per a fer distribució de càrrega entre servidors replicats, com ara servidors web. Llocs web molt accedits poden estar replicats en molts servidors, cadascun corrent en un ordinador diferent i amb una adreça IP diferent. En aquest cas, una manera de repartir la càrrega entre els servidors és associar un conjunt d'adreces IP a un nom canònic. El DNS conté aquest conjunt d'adreces. Quan un client fa una consulta al DNS per un nom al qual correspon un conjunt d'adreces, contesta totes les adreces IP, però en cada resposta en rota l'ordre. Com que normalment el servidor HTTP envia la petició a la primera adreça IP de la llista, s'aconsegueix una distribució de càrrega entre els servidors. Aquesta rotació també s'utilitza en el correu electrònic, i així molts servidors de correu poden tenir el mateix nom. Darrerament, companyies de distribució de continguts com ara Akamai usen el DNS de maneres més sofisticades per a proporcionar distribució de continguts web. Aquestes companyi-

Vegeu també

Per a saber-ne més sobre la distribució de contingut, vegeu el subapartat 3.8.

es utilitzen el DNS per a fer que els usuaris es descarreguin els continguts d'ubicacions properes.

2.1. DNS: base de dades jeràrquica i distribuïda

El DNS és una base de dades jeràrquica i distribuïda organitzada de manera autònoma que proporciona un rendiment en temps real a una audiència global amb contribuïdors globals.

Un disseny senzill per al DNS seria tenir un servidor que contingués totes les equivalències entre noms i adreces IP. Els clients enviarien les peticions de resolució al servidor DNS i aquest els contestaria directament. Aquesta solució centralitzada seria senzilla d'implementar però tindria nombrosos inconvenients, pel fet que actualment a Internet hi ha milions de servidors. Caldria que tingués una base de dades molt gran capaç de suportar un volum de consultes molt gran. A més d'haver de ser capaç de gestionar una freqüència alta d'actualitzacions de les entrades, com ara nous amfitrions o canvis en els amfitrions.

Per tal de poder atendre aquests requisits d'escala (nombre d'entrades i nombre peticions), el DNS usa molts servidors organitzats de manera jeràrquica i distribuïts arreu del món. Cap servidor DNS té totes les equivalències entre noms i adreces IP. Aquestes equivalències estan distribuïdes en els diferents servidors DNS.

Hi ha tres tipus de servidors DNS:

- **Servidors DNS arrel**. Hi ha 13 servidors DNS arrel (etiquetats de la A a la M). Cadascun d'aquests servidors arrel en realitat és un clúster de servidors reproduïts, per seguretat i fiabilitat.

- **Servidors DNS de nivell de domini superior**[4]. Són els responsables dels dominis de primer nivell, com ara .org, .com, .net, .edu o .cat, i també de tots els dominis de països (.us, .es, etc.).

- **Servidors DNS autoritzats**[5]. Cada organització amb ordinadors accessibles des d'Internet ha de proporcionar registres DNS accessibles públicament que permetin fer l'equivalència entre els noms dels seus ordinadors i les adreces IP. Un servidor DNS autoritzat hostatja a aquests registres. Aquests registres poden estar en servidors DNS autoritzats de l'organització mateixa o en servidors DNS autoritzats d'algun proveïdor de serveis. En aquest cas, l'organització ha de pagar per aquest servei. La majoria de grans em-

Adreça web recomanada

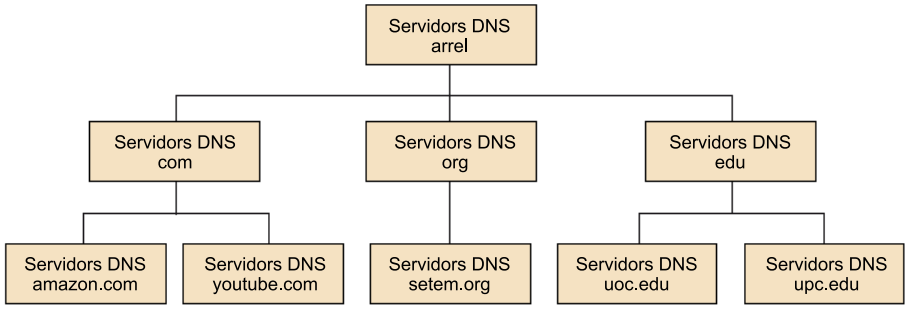
A http://www.root-servers.org/ podeu veure la ubicació dels servidors DNS arrel.

(4)En anglès, *top-level domain* (TLD).

(5)En anglès, *authoritative DNS servers.*

preses, universitats i centres de recerca mantenen els seus servidors DNS autoritzats primari i secundari (*backup*).

Porció de la jerarquia de servidors DNS



Hi ha un altre tipus de servidors DNS, anomenats **servidors DNS locals**, que encara que estrictament no pertanyen a la jerarquia de servidors, són claus per a l'arquitectura del DNS. Cada empresa proveïdora d'accés a Internet (PAI o ISP en anglès) –com ara una universitat o una empresa de telecomunicacions que proporciona accés a Internet a usuaris domèstics– té un servidor DNS local. Quan un node es connecta al seu PAI aquest li proporciona l'adreça IP d'un o més servidors DNS locals (normalment això es fa automàticament via DHCP). Quan un ordinador fa una consulta al DNS, la consulta s'envia al servidor DNS local, que actua com a proxy, i aquest l'envia a la jerarquia de servidors DNS perquè resolgui la petició.

Proveïdor d'accés a Internet

Proveïdor d'accés a Internet (PAI) o proveïdor de serveis Internet (en anglès, *Internet services provider*, ISP) és una empresa que ofereix als seus clients accés a Internet. Un PAI connecta els seus usuaris a Internet per mitjà de diferents tecnologies, com ara DSL (normalment ADSL), mòdem de xarxa de telefonia commutada, cable mòdem, Wi-Fi, etc.

Molts PAI també ofereixen altres serveis relacionats amb Internet com ara correu electrònic, allotjament de pàgines web, registre de dominis, etc.

DHCP

Dynamic Host Configuration Protocol (DHCP) és un protocol de xarxa que permet als nodes d'una xarxa IP obtenir els seus paràmetres de configuració automàticament. Es tracta d'un protocol de tipus client/servidor. El client fa difusió d'una petició d'informació de configuració. El servidor DHCP rep la petició i respon amb informació de configuració de la seva base de dades de configuració. Generalment un servidor posseeix una llista d'adreces IP dinàmiques i les va assignant als clients a mesura que estan lliures, i sap en tot moment qui ha estat en possessió d'aquella IP, quant temps l'ha tinguda o a qui se li ha assignat després.

En cas de no fer servir DHCP, cada ordinador de la xarxa s'ha de configurar manualment, tasca que és costosa i propensa a errors.

Exemple d'obtenció d'adreça IP

A continuació hi ha un exemple dels passos que se seguirien perquè l'ordinador `estudiant.pai.cat` obtingui l'adreça IP de l'ordinador `sd.eimt.uoc.edu`.

1) `estudiant.pai.cat` envia la petició al seu servidor DNS local.

2) El servidor DNS local reenvia la petició a un servidor DNS arrel.

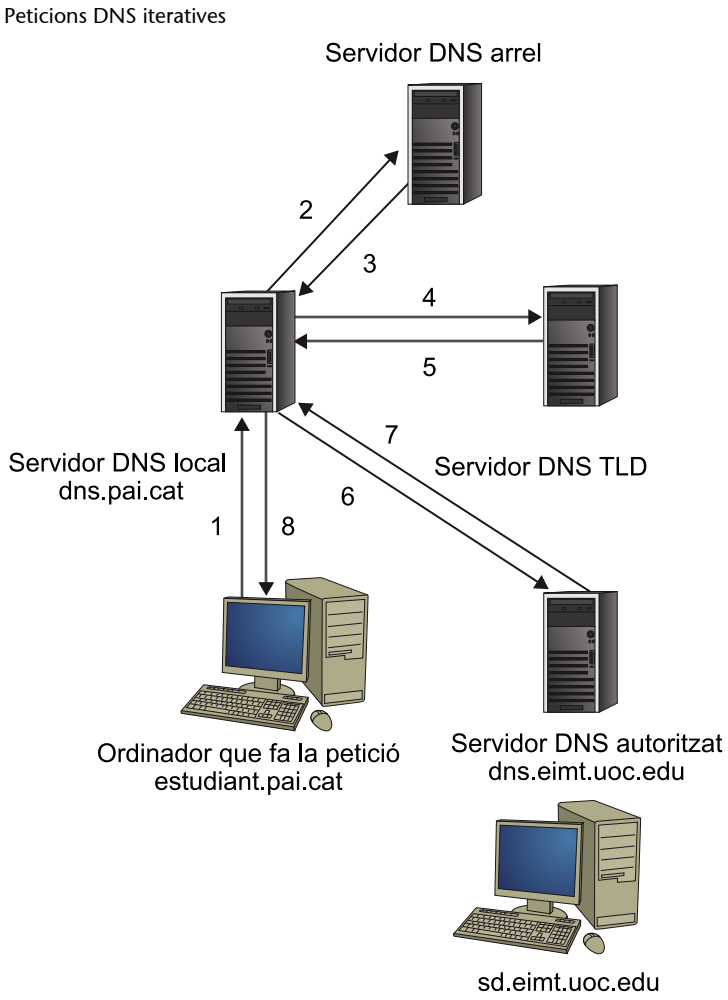
3) El servidor DNS arrel retorna una llista d'adreces IP de servidors DNS responsables del domini `edu`.

4) El servidor DNS local reenvia la petició a un d'aquests servidors DNS TLD.

5) El servidor DNS TLD contesta l'adreça IP del servidor autoritzat pel domini `uoc.edu`.

6) El servidor DNS local reenvia la petició al servidor DNS de la Universitat Oberta de Catalunya (`dns.uoc.edu`).

7) El servidor DNS de la Universitat Oberta de Catalunya respon l'adreça IP de l'ordinador `sd.eimt.uoc.edu`.



Sovint passa que el servidor TLD no coneix el nom del servidor autoritzat sinó només d'un servidor DNS intermedi, el qual sí que coneix el servidor DNS autoritzat de l'ordinador. Suposem que en l'exemple que acabem de veure la UOC tingui un servidor DNS per a tota la universitat –dns.uoc.edu-, i que el servidor DNS de cada departament sigui l'autoritzat per tots els ordinadors del departament. En aquest cas, quan el servidor DNS intermediari, dns.uoc.edu, rep una petició d'un ordinador amb el nom de l'ordinador acabat en `eimt.uoc.edu`, retorna a dns.pai.cat l'adreça IP de `dns.eimt.uoc.edu`, que és el servidor DNS autoritzat per tots els ordinadors acabats en `eimt.uoc.edu`. El servidor DNS local preguntaria, finalment, a `dns.eimt.uoc.edu`, l'adreça IP de `sd.eimt.uoc.edu`.

2.2. *Caching* de DNS

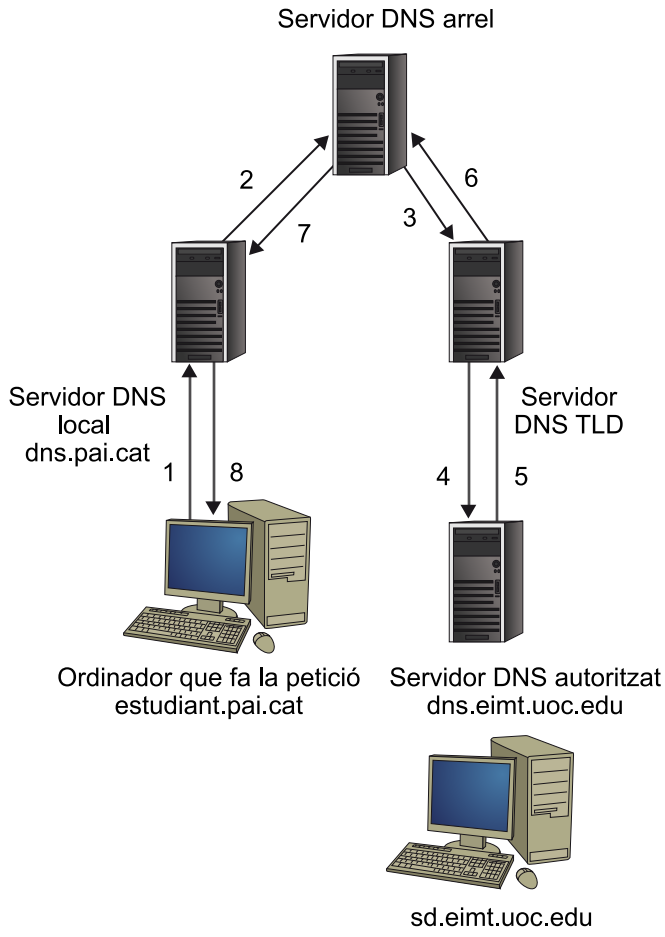
Com es pot veure, cada vegada que es demana la resolució d'un nom s'envien molts missatges, cosa que fa incrementar el temps per a obtenir la resposta i fa que hi hagi molts missatges DNS circulant per Internet. Per tal de reduir aquest nombre de missatges el DNS fa *caching*. En l'exemple de l'última figura del subpartat 2.1, el servidor DNS local es desaria una còpia de l'adreça IP responsable del domini `uoc.edu`. D'aquesta manera si al cap d'una estona rep una petició d'un ordinador del domini `uoc.edu` ja preguntaria directament

al servidor DNS autoritzat per `uoc.edu`. El servidor DNS local també es pot guardar una còpia de l'adreça IP dels servidors TLD i així estalviar-se preguntar als servidors DNS arrel. Atès que l'equivalència entre amfitrions i adreces IP no és permanent, els servidors DNS descarten la informació del cau al cap d'un cert temps (sovint dos dies).

2.3. Peticions recursives enfront de peticions iteratives

L'exemple de l'última figura del subpartat 2.1 utilitza tant **peticions recursives** com **peticions iteratives**. La petició d'`estudiant.pai.cat` a `dns.pai.cat` és recursiva, ja que la petició demana a `dns.pai.cat` que resolgui l'equivalència en lloc seu. Les altres tres consultes són iteratives, ja que les respostes es retornen directament a `dns.pai.cat`. En teoria les peticions al DNS poden ser recursives o iteratives. La figura següent mostra la cadena de peticions per al cas d'una resolució recursiva. En la pràctica, la majoria de peticions segueix l'esquema de la figura que hem esmentat abans.

Petició DNS recursiva



2.4. Registres DNS

Els servidors DNS emmagatzemen **registres de recursos**, que tenen els camps següents:

- Nom.
- Valor.
- Tipus.
- Temps de vida[6]. El TTL és el temps màxim perquè un recurs s'esborri de la memòria cau.

[6] En anglès, *time to live* (TTL).

Els tipus més usuals són:

- A: *Nom* és el nom de l'ordinador i *Valor* és la seva adreça IP.

- NS: *Nom* és un domini (per exemple, `uoc.edu`) i *Valor* és el nom d'un servidor autoritzat per aquest domini (per exemple, `dns.uoc.edu`).

- CNAME: *Valor* és el nom canònic d'un àlies (*Nom*). Permet obtenir el nom canònic d'un àlies. Per exemple, (`sd.uoc.edu, einfsun1.uoc.edu, CNAME`).

- MX: *Valor* és el nom canònic d'un servidor de correu que té *Name* com a àlies. Per exemple, (`uoc.edu, correu2.uoc.edu, MX`). MX permet que els servidors de correu tinguin noms més senzills. També permet que una organització pugui tenir el mateix àlies per al domini de correu i per al servidor web. Si es vol obtenir el nom canònic del servidor de correu es preguntarà pel camp MX i si es vol obtenir el de l'altre servidor es preguntarà pel camp CNAME.

2.5. Consideracions de seguretat

El DNS es va dissenyar sense considerar la seguretat. Un tipus de vulnerabilitat és la contaminació de la memòria cau del DNS, que fa creure a un servidor DNS que ha rebut informació autèntica quan en realitat no és així.

Contaminació de la memòria cau del DNS

La contaminació de la memòria cau del DNS (*DNS cache poisoning* o *DNS cache pollution* en anglès) és una situació creada maliciosament o intencionadament que proporciona dades a una memòria cau de DNS que no està originada pel servidor DNS autoritzat. Aquest fet pot ocórrer per un error en el programari, per una configuració dolenta dels servidors DNS o per accions malintencionades que s'aprofiten de l'arquitectura oberta del DNS.

Domain Name System Security Extensions (DNSSEC) modifica el DNS per a afegir respostes signades digitalment, cosa que proporciona als clients DNS (resoledors) autenticació de l'origen de les dades DNS, integritat de dades –però no disponibilitat ni confidencialitat– i denegació d'existència autenticada.

Extensió de DNSSEC

A partir del juliol de 2010, tots els servidors arrel hauri-en d'utilitzar DNSSEC (http://www.root-servers.org/).

3. El Web i l'HTTP

El Web és un sistema format per milions de pàgines escrites en hipertext mitjançant el llenguatge de marcatge HTML i connectades entre si mitjançant vincles, de manera que formin un sol cos de coneixement pel qual es pot navegar fàcilment. Amb un navegador web es poden visualitzar pàgines que poden contenir text, imatges, vídeos i altres mitjans multimèdia i navegar d'unes pàgines a altres usant els hiperenllaços.

HTML

Hyper text markup language (HTML) és un llenguatge de marcatge dissenyat per a estructurar textos i relacionar-los en forma d'hipertext. El consorci World Wide Web Consortium (W3C) controla l'evolució de l'HTML: http://www.w3.org.

La introducció del web al principi dels noranta va representar una revolució a Internet. Internet va deixar de ser una xarxa utilitzada majoritàriament per investigadors, acadèmics i alumnes d'universitat, per a passar a ser un mitjà de comunicació, interacció i publicació d'informació que ha entrat a formar part de la nostra quotidianitat.

El Web es basa en tres estàndards per a funcionar:

- L'*uniform resource locator* (URL), que s'encarrega de donar una adreça única per a localitzar cada objecte.
- L'Hyper Text Transfer Protocol (HTTP), que especifica la manera com s'enviarà i es rebrà la informació entre el navegador i el servidor.
- L'*hyper-text markup language* (HTML), un mètode per a especificar com s'ha de veure aquesta informació en el navegador.

En aquest mòdul ens centrarem en l'HTTP. Abans, però, veurem els aspectes bàsics dels URL. L'URL és una cadena de caràcters que informa al navegador del següent:

- l'ordinador on hi ha el recurs a què fa referència,
- el protocol que ha d'utilitzar per a obtenir aquest recurs,
- la manera com el servidor web trobarà quin és el recurs.

La sintaxi que s'utilitza en l'URL és:
`protocol://[usuari:contrasenya]@host[:port]/[camí]`,
en què `usuari:contrasenya`, el `port` i el `camí` són opcionals.

Els protocols més habituals són: http, ftp (File Transfer Protocol) i *file* (per a l'accés i localització d'arxius dins de sistemes de fitxers).

Vegeu també

El protocol ftp es tracta a l'apartat 4 d'aquest mòdul.

Host identifica la màquina on hi ha el recurs. Pot ser un nom (www.wikipedia.org) o una adreça IP (192.12.34.11). El port és el número de port TCP del servidor, on es connectarà el navegador.

Exemple

En l'adreça http://www.empresa.com/producte/descipcio.html, *http* és el nom del protocol, www.empresa.com és l'adreça del servidor de l'empresa i /producte/descripcio.html és el camí fins a arribar a l'objecte en el servidor. En aquest cas no hi ha cap port especificat, cosa que vol dir que es fa servir el port 80, que és el port per defecte dels servidors web.

3.1. HTTP: Hypertext Transfer Protocol

L'HTTP[7] és un protocol a nivell d'aplicació per a sistemes hipermèdia col·laboratius i distribuïts, que és la base del Web. L'HTTP segueix el paradigma client/servidor. Clients i servidors s'intercanvien missatges HTTP. L'HTTP defineix l'estructura d'aquests missatges i com el client i el servidor intercanvien els missatges. Es basa en un transport fiable, i el més habitual és TCP, encara que podria funcionar en altres transports fiables. El port per defecte per a establir les connexions és el port assignat oficialment al servei www, que és el port 80.

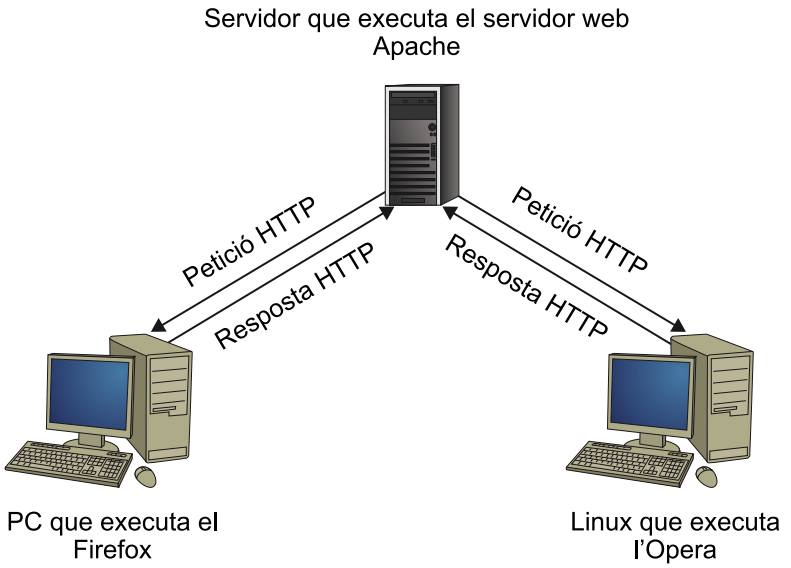
(7)L'HTTP està definit en l'RFC 1945 i l'RFC 2616.

Una **pàgina web** està formada per diferents objectes. Un **objecte** és un fitxer –que pot ser de diferents tipus: un fitxer HTML, una imatge, una miniaplicació Java o un vídeo– adreçable per un URL. La majoria de pàgines web consisteixen en un fitxer base formatat en HTML que referencia diferents objectes.

Exemple

Una pàgina que conté text en HTML i dues imatges en GIF conté tres objectes: la pàgina HTML base i les dues imatges.

La figura següent il·lustra el funcionament de la interacció entre un servidor web i uns clients web: un client envia un missatge HTTP al servidor i aquest el processa i el contesta amb un altre missatge HTTP. Atès que els navegadors web implementen la part client HTTP, sovint ens referirem a clients HTTP com a *navegadors*. Els navegadors també implementen altres parts, com ara el visualitzador que formata els documents HTML i els presenta als usuaris.



Els servidors HTTP no mantenen cap informació sobre els clients, i per aquest motiu es diu que l'HTTP és un **protocol sense estat**.

3.1.1. Connexions persistents i no persistents

L'HTTP té dos modes de connexió:

- **Connexions persistents**. En el cas de les connexions no persistents, s'obre una connexió TCP diferent per a enviar la petició/resposta per a cada objecte que conté la pàgina que es vol baixar (és a dir, la petició i resposta d'un mateix objecte viatgen per la mateixa connexió).

- **Connexions no persistents**. En el cas de les connexions persistents totes les peticions i respostes dels objectes d'una pàgina s'envien per la mateixa connexió TCP. Per defecte, l'HTTP usa connexions persistents, encara que tant els clients com els servidors HTTP es poden configurar per a usar connexions no persistents.

	Avantatges	Inconvenients
Connexions persistents	Després d'enviar una resposta el servidor deixa oberta la connexió TCP amb el client. Això permet que les futures peticions i respostes entre el client i el servidor es puguin enviar per aquesta connexió TCP (siguin d'objectes pertanyents a la mateixa pàgina o de diferents pàgines). A més, no cal que esperin que els toqui el torn de ser servides, ja que la connexió ja està oberta. El servidor tanca la connexió TCP amb el client quan fa una estona que no s'usa.	Triga més a baixar els objectes d'una pàgina, ja que no aprofita el paral·lelisme que es pot obtenir baixant cada objecte de la pàgina per separat.

	Avantatges	Inconvenients
Connexions no persistents	Permet tenir més d'una connexió en paral·lel, fet que redueix el temps de resposta.	Cal establir més connexions: • la baixada de cada objecte pateix un retard a causa del temps d'establiment de la connexió TCP i del temps de fer la petició de l'objecte. • Per a cada connexió cal mantenir informació: memòries intermèdies i altres variables del TCP, cosa que pot ser una molèstia per al servidor web, ja que pot estar atenent moltes peticions simultàniament.

3.2. Format dels missatges HTTP

A continuació, veurem els missatges HTTP de petició i resposta.

3.2.1. Missatge HTTP de petició

Tot seguit hi ha l'exemple d'un missatge de petició HTTP.

```
GET /elMeuDirectori/pagina.html HTTP/1.1
Host: www.servidor.edu
```

Com podeu veure, el missatge està escrit en ASCII. Cada línia acaba amb un *carry return* i un *line feed*. La darrera línia està acabada amb una línia en blanc –una línia que només conté un *carry return* i un *line feed*. Els missatges poden tenir més línies.

La primera línia s'anomena **línia de petició**. Les línies següents s'anomenen **línies de capçalera**. La línia de petició té tres camps:

- **El mètode**: pot tenir diferents valors: `GET, POST, HEAD, PUT, DELETE`, etc. El mètode més usat és el GET. El mètode GET s'usa quan el navegador demana un objecte.

- **L'URL**: identifica l'objecte. En l'exemple, es demana l'objecte `/elMeuDirectori/pagina.html`.

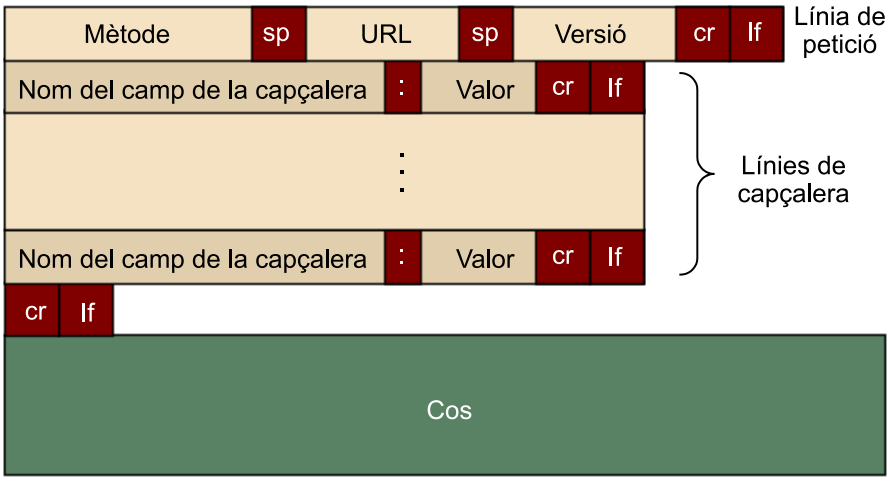
- **Versió d'HTTP**: en l'exemple, el navegador implementa la versió HTTP/1.1.

La línia de capçalera `Host: www.servidor.edu` indica en quin ordinador hi ha l'objecte. D'entrada aquesta capçalera podria semblar supèrflua, però és útil per als servidors de memòria cau.

Vegeu també

Al subapartat 3.6 es tracten els servidors de memòria cau.

De manera més general, a la figura següent trobareu el format dels missatges HTTP de petició.



És fàcil veure que el format dels missatges de petició segueix l'exemple que hem vist anteriorment, amb una diferència: el format general conté un "cos". El cos és buit en el cas del mètode GET, però es fa servir amb el mètode POST. Els clients HTTP acostumen a fer servir el mètode POST per a enviar dades d'un formulari –per exemple, quan un usuari proporciona paraules per a fer una cerca a un cercador. Amb un missatge POST, l'usuari està demanant una pàgina web a un servidor però el contingut de la pàgina web depèn del que l'usuari ha entrat en els camps del formulari. Si el valor del camp mètode és POST, el cos conté el que l'usuari ha entrat en els camps de formulari.

Val la pena esmentar que no sempre que s'utilitza un formulari les dades s'envien usant el mètode POST. Sovint s'usa el mètode GET i les dades s'inclouen en l'URL. Per exemple, si un formulari que utilitza el mètode GET té dos camps (model i quantitat) i els valors dels dos camps són AT i 23, llavors l'URL tindrà l'estructura següent:

```
www.negoci.com/compra?model=TA&quantitat=23
```

3.2.2. Missatge HTTP de resposta

A continuació es mostra un exemple de possible missatge HTTP de resposta.

```
HTTP/1.1 200 Ok
Date: Wed, 24 Feb 2010 19:05:40 GMT
Server: Apache/1.3.3 (Unix)
Last-Modified: Wed, 18 Feb 2009 21:11:55 GMT
Content-Length: 3245
Content-Type: text/html

(dades dades dades ...)
```

La resposta té tres parts:

- **Una línia d'estat**. Té tres parts:
  - la versió del protocol,
  - un codi d'estat i
  - el missatge corresponent al codi d'estat.

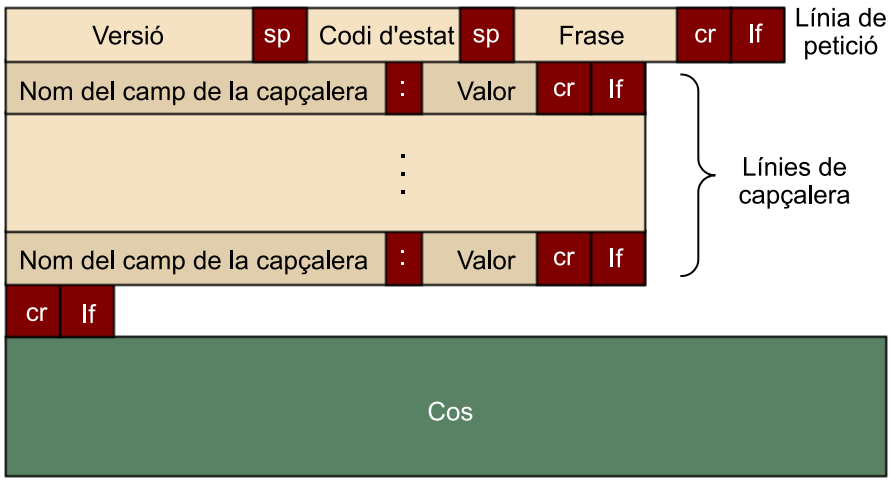
- **Unes capçaleres**. En l'exemple anterior:
  - `Date:` indica el dia i l'hora en què es va crear i enviar la resposta. No és el dia i l'hora en què es va crear o modificar per darrera vegada l'objecte, sinó el moment en què el servidor obté l'objecte del sistema de fitxers, l'inclou en el missatge de resposta i l'envia.
  - `Server:` indica que el missatge el va generar un servidor web Apache.
  - `Last-Modified:` indica el dia i la data en què es va crear o modificar per darrera vegada l'objecte.
  - `Content-Length:` indica el nombre de bytes de l'objecte que s'envia.
  - `Content-Type:` indica que l'objecte que conté el cos és del tipus text/HTML.

- **Un cos**: conté l'objecte demanat.

Vegeu també

Veureu en el subapartat 3.6 que la capçalera `Last-Modified` és una capçalera crítica per als servidors de memòria cau.

A la figura següent trobareu el format general dels missatges HTTP de resposta.



Alguns codis d'estat són:

- `200 OK`: la petició ha tingut èxit i la informació es retorna inclosa en la resposta.

- `301 Moved Permanently`: l'objecte demanat s'ha canviat d'ubicació. El nou URL està especificat a la capçalera `Location:` del missatge de resposta. El client obtindrà de manera automàtica el nou URL.

- `400 Bad Request`: codi d'error genèric que indica que el servidor no ha pogut entendre la petició.

- `404 Not Found`: el document demanat no és al servidor.

- `505 HTTP Version Not Supported`: el servidor no suporta la versió del protocol HTTP demanat.

Missatge HTTP de resposta real

És molt fàcil veure un missatge HTTP de resposta real. Només cal fer un Telnet a un servidor web i després teclejar en línia un missatge demanant un objecte que estigui hostatjat al servidor. Podeu provar amb l'exemple següent[8]:

```
telnet www.uoc.edu 80
GET /index.html HTTP/1.1
Host: www.uoc.edu
```

Amb el Telnet s'obre una connexió TCP en el port 80 del servidor www.uoc.edu. A continuació s'envia un missatge HTTP de petició. (Encara que no veieu el que esteu escrivint, s'enviarà un cop premeu el retorn. Si us equivoqueu no serveix de res esborrar, ja que s'enviaran tots els caràcters. Caldrà que torneu a començar.) Rebreu un missatge de resposta que contindrà el fitxer HTML de la pàgina que heu demanat.

(8)Recordeu que cal prémer dues vegades la tecla de retorn després de teclejar la darrera línia.

3.3. HTTPS (HTTP segur)

L'Hypertext Transfer Protocol Secure (HTTPS) és una combinació de l'HTTP amb el protocol SSL/TLS per a proporcionar xifratge i identificació segura del servidor. La idea principal darrere l'HTTPS és crear un canal segur sobre una xarxa insegura. Això assegura una protecció raonable davant intrusos que vulguin espiar la comunicació i atacs *man-in-the-middle*, sempre que el certificat del servidor estigui verificat i s'hi pugui confiar.

Atac *man-in-the-middle*

L'atac *man-in-the-middle* és una manera d'espiar en la qual l'atacant crea connexions independents amb les víctimes i fa passar els missatges entre aquestes connexions a través seu. Això fa que les víctimes pensin que estan parlant directament entre elles per mitjà d'una connexió privada quan, en realitat, la conversa està controlada per l'atacant.

Un atac *man-in-the-middle* té èxit només quan l'atacant es pot fer passar per cadascuna de les víctimes. La majoria de protocols de xifratge inclouen alguna forma d'autenticació per tal d'evitar aquest tipus d'atacs. Per exemple, l'SSL autentica el servidor per mitjà d'una autoritat de certificació en la qual confien totes dues parts.

HTTPS confia en unes autoritats de certificació que estan preinstal·lades en els navegadors web. A partir d'aquí, les connexions HTTPS són fiables si i només si es compleix el següent:

- L'usuari confia que l'autoritat certificadora genera certificats per a llocs web legítims.
- El lloc web proporciona un certificat vàlid (un certificat signat per una autoritat en la qual s'hi pot confiar).
- El certificat identifica de manera clara el lloc web.
- O bé els nodes d'Internet que intervenen són fiables o bé l'usuari confia que el protocol de xifratge usat (TLS o SSL) fa que no es pugui espiar la comunicació.

3.4. Galetes

L'HTTP és un protocol sense estat. Això simplifica el disseny del servidor i fa que aquest tingui un rendiment elevat, ja que és capaç de suportar milers de connexions TCP simultànies. A vegades, però, convé que el servidor web pugui identificar usuaris. Les galetes[9] s'usen per a aquest propòsit. Permeten als servidors web tenir informació dels usuaris. Les galetes es poden usar per a autenticar, emmagatzemar preferències del client, identificar una sessió, mantenir un seguiment de les compres en una botiga virtual o qualsevol altra cosa que es pugui fer emmagatzemant dades textuals. La majoria de llocs web comercials usen galetes.

[9] Les galetes (*cookies* en anglès) estan definides en l'RFC 2965.

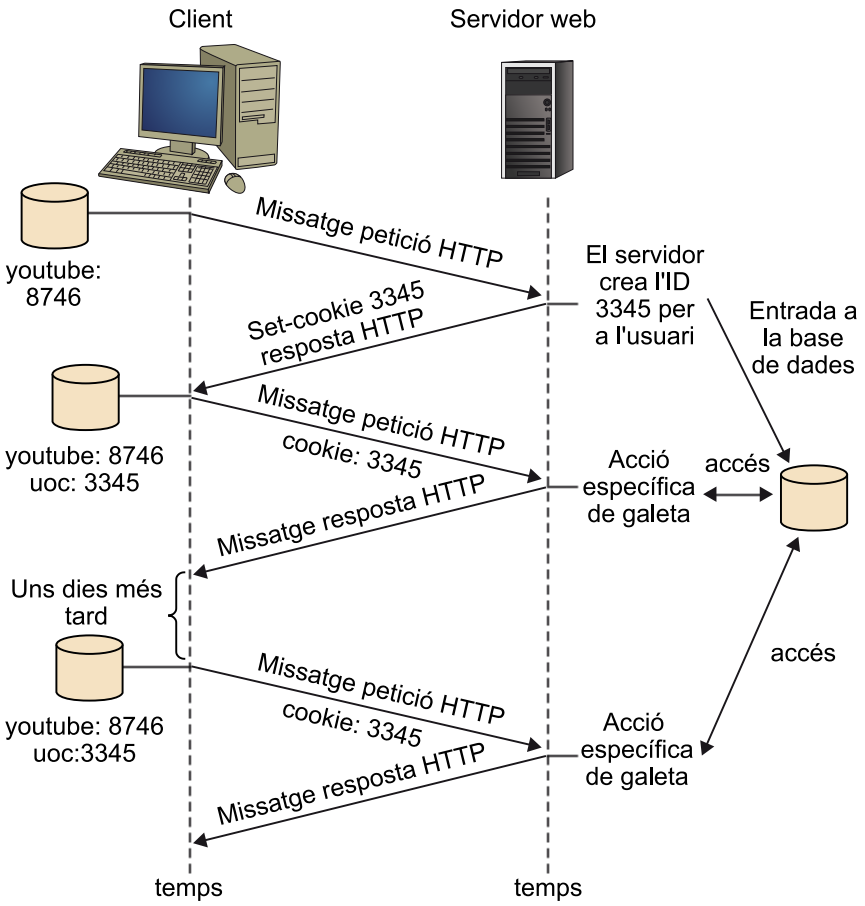
Una galeta és una petita porció de text emmagatzemat en l'ordinador de l'usuari pel navegador web, que consisteix en una o més parelles nom-valor que contenen porcions d'informació.

El servidor web envia la galeta al navegador web com una capçalera de la resposta. A partir d'aquest moment, el navegador web envia la galeta cada vegada que accedeix el servidor. El navegador web emmagatzema les galetes en un fitxer local gestionat pel navegador. El servidor web pot tenir una base de dades amb la informació relacionada amb les galetes.

Exemple d'ús d'una galeta

Suposem que hi ha un client que es vol connectar al servidor web de la UOC. Quan la petició arriba al servidor de la UOC, el servidor crea un identificador únic i crea una entrada a una base de dades que s'indexa per a aquest identificador. A continuació contesta al navegador del client incloent-hi la capçalera `Set-cookie:` en la resposta HTTP, que conté l'identificador. El navegador rep la resposta HTTP i emmagatzema la galeta rebuda en un fitxer, en què les emmagatzema.

En l'exemple de la figura següent, el fitxer de galetes ja en conté una d'una connexió anterior a Youtube. L'usuari continua navegant pel lloc web de la UOC. Cada cop que demana una pàgina web, el navegador consulta el fitxer de galetes i afegeix una línia de capçalera a la petició HTTP. Si l'usuari torna al lloc web de la UOC uns dies més tard, continua afegint la capçalera `Cookie:` en els missatges de petició.



Les galetes, en ser text, no són executables. Pel fet de no ser executables, no es poden replicar per si soles i, per tant, no són virus. A causa del mecanisme per a desar i llegir les galetes, es poden usar com a programari espia. Els productes antiespia poden avisar els usuaris sobre algunes galetes, perquè es poden usar per a fer un seguiment de les operacions de l'usuari o vulnerar la seva privadesa.

La majoria de navegadors moderns permeten als usuaris decidir si volen acceptar o no les galetes de les pàgines visitades, però no acceptar-les pot fer que algunes pàgines web no funcionin correctament.

3.5. Característiques de la demanda de pàgines web

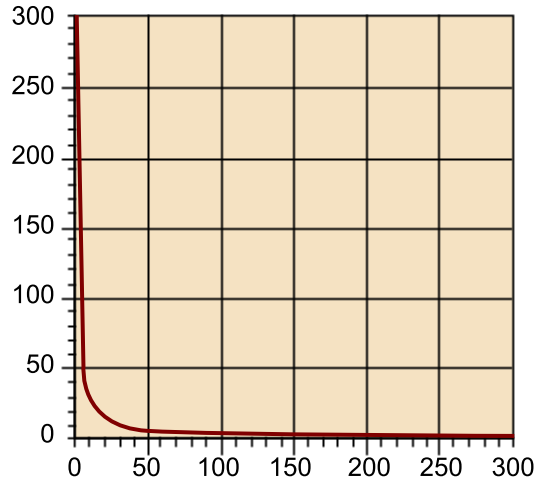
Un servidor web pot tenir milers de documents i, tanmateix, rebre la majoria de peticions per a un únic document.

Llei de Zipf

En molts casos, la popularitat relativa entre diversos llocs web o entre diverses pàgines d'un cert lloc web es regeix per la llei de Zipf[10], que, formulada de manera planera seria: "molts pocs casos tenen molta popularitat i molts casos en tenen molt poca".

[10]La llei de Zipf deu el seu nom a George Kingsley Zipf (1902-1950).

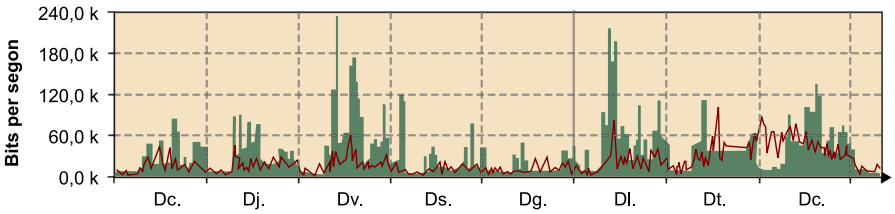
Distribució de popularitat que segueix la llei de Zipf



Casos ordenats per popularitat a l'eix *x*, i valor de popularitat a l'eix *y*

La popularitat d'un lloc web pot ser molt variable. Pot rebre molt poques visites durant molt de temps i, de sobte, rebre diverses vegades més peticions de les que pot servir: és un trànsit a ràfegues.

Evolució del trànsit entrant i sortint d'un lloc web típic durant una setmana



Podeu observar la gran variació horària i la reducció de trànsit durant el cap de setmana.

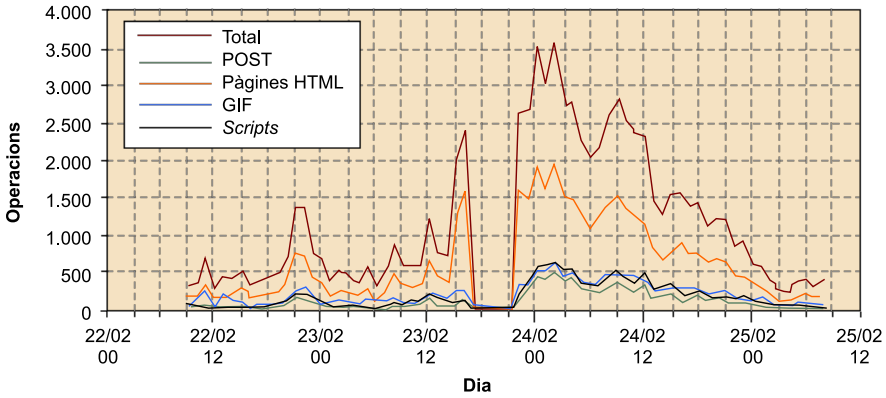
Un servidor pot rebre allaus sobtades de trànsit. Per exemple, per les estadístiques del servidor web que es veu a la figura següent sabem que, després de ser anunciat a la pàgina de notícies Slashdot, va patir un excés de visites tan alt que el servidor es va bloquejar.

Flash crowd

Un conte de ciència-ficció de Larry Niven (1973) va predir que una conseqüència d'un mecanisme de teletransport barat seria que grans multituds es materialitzarien instantàniament als llocs amb notícies interessants. Trenta anys després el terme s'usa a Internet per a descriure els pics de trànsit web quan un determinat lloc web esdevé sobtadament popular i és visitat de manera massiva.

També es coneix com a "efecte *slashdot*" o efecte "/.", que es dóna quan un lloc web resulta inaccessible per causa de les nombroses visites que rep quan apareix en un article del lloc web de notícies Slashdot (en castellà, Barrapunto).

Peticions web per hora, servides per http://counter.li.org durant tres dies

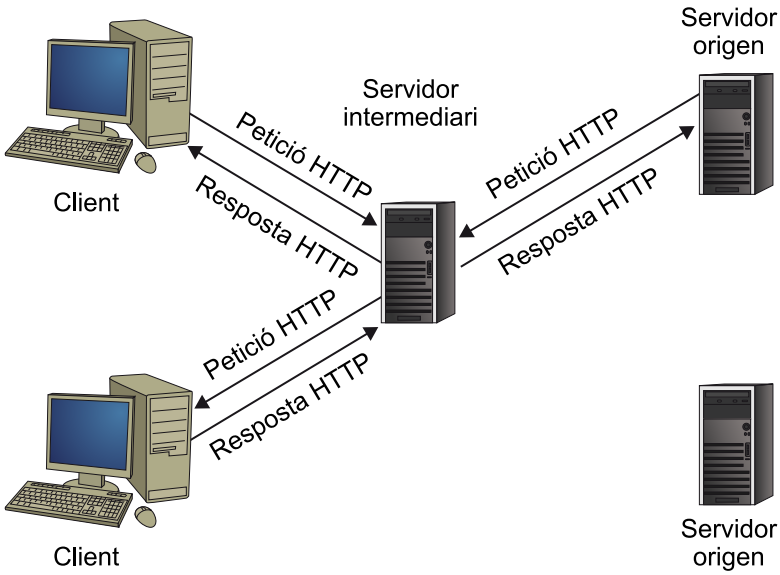


Es pot veure que, mentre el nombre habitual d'operacions (peticions web) estava per sota de 500, va pujar ràpidament a unes 2.500, fet que va provocar la fallada del sistema. Després de reconfigurar-lo, va estar suportant durant unes dotze hores al voltant de 3.000 peticions/hora per a baixar posteriorment a valors normals. La història completa és a l'adreça: http://counter.li.org/slashdot.

3.6. Servidors intermediaris

Un servidor intermediari[11] és una entitat de la xarxa que satisfà les peticions HTTP en lloc del servidor web al qual estava adreçada la petició. Els servidors intermediaris mantenen una còpia dels objectes accedits recentment en un disc del servidor. La figura següent mostra el funcionament general d'un servidor intermediari.

(11)En anglès, *web cache* o *proxy-cache*.



El navegador de l'usuari es pot configurar perquè totes les peticions HTTP de l'usuari vagin primer al servidor web intermediari. A partir d'aquest moment:

1) Cada nova petició de l'usuari establirà una connexió TCP amb el servidor intermediari i li enviarà la petició HTTP.

2) Si el servidor intermediari té una còpia de l'objecte el retorna dins d'un missatge HTTP de resposta al navegador client.

3) En cas que el servidor intermediari no tingui l'objecte demanat, aquest obre una nova connexió TCP amb el servidor origen de la petició per a demanar-li l'objecte.

4) En rebre la petició, el servidor origen envia l'objecte en una resposta HTTP al servidor intermediari.

5) Finalment, el servidor intermediari emmagatzema l'objecte rebut en el seu disc local i envia una còpia, en un missatge HTTP de resposta, al navegador client (en la connexió TCP establerta entre el client i el servidor intermediari).

El servidor intermediari actua com a servidor del client i com a client del servidor. Els proveïdors d'accés a Internet[12] són els que instal·len els servidor intermediari. Per exemple, una universitat pot instal·lar un servidor intermediari a la xarxa del campus i configurar tots els navegadors perquè apuntin al servidor intermediari.

(12)En anglès, *Internet service providers* (ISP)

Hi ha dues raons principals per a desplegar un servidor intermediari a Internet:

- es pot reduir significativament el temps de resposta de les peticions;
- pot reduir significativament el trafic de l'enllaç de la institució amb Internet.

Els servidors intermediaris s'usen també en altres protocols a més de l'HTTP. En general se situen en una discontinuïtat per a fer-hi una funció. Per exemple, per a fer una canvi de xarxa: una màquina connectada a la xarxa interna d'una organització, que usa adreces IPv4 privades (per exemple, 10.*.*.*), i també a Internet, pot fer de servidor intermediari per a la traducció d'adreces IP entre totes dues xarxes (NAT).

Network Address Translation (NAT)

En els paquets IP sortints: substituir l'adreça IP de les màquines internes (no vàlides a Internet) per la seva pròpia. En els paquets IP entrants: substituir la seva pròpia adreça IP per la d'una màquina interna i reenviar el paquet cap a la xarxa interna. Per poder saber a qui s'ha de lliurar, el servidor intermediari ha d'associar cadascun dels seus ports a les màquines internes, ja que una adreça de transport és una parella (adreça IP, port).

3.7. El GET condicional

Hem vist que el *caching* pot reduir el temps de resposta percebut per l'usuari, però la còpia que tingui el cau pot ser obsoleta. És a dir, l'objecte hostatjat al servidor web es pot haver modificat després que el client el copiés en la seva

memòria cau. Per a solucionar-ho el protocol HTTP inclou un mecanisme que permet a la memòria cau verificar que la seva còpia de l'objecte està actualitzada.

```
GET /elMeuDirectori/pagina.html HTTP/1.1
Host: www.servidor.edu
If-modified-since: Wed, 18 Feb 2009 21:11:55 GMT
```

La capçalera `If-modified-since:` conté el valor de la capçalera `Last-Modified:` de quan el servidor va enviar l'objecte. Aquesta petició GET, que s'anomena *GET condicional*, indica al servidor que enviï l'objecte només si l'objecte s'ha modificat des de la data especificada.

En cas que l'objecte no s'hagi modificat des de la data indicada, el servidor web contesta un missatge a la memòria cau com el següent:

```
HTTP/1.1 304 Not Modified
Date: Tue, 24 Mar 2009 18:23:40 GMT
(sense cos)
```

`304 Not Modified` a la línia d'estat del missatge de resposta indica a la memòria cau que pot passar la seva còpia de l'objecte al navegador que ha fet la sol·licitud. Cal destacar que el missatge de resposta del servidor no inclou l'objecte, ja que això només faria malgastar amplada de banda i la percepció de l'usuari sobre el temps de resposta.

3.8. Distribució de continguts

Els mecanismes de *proxy-cache* són útils perquè una comunitat d'usuaris que comparteixen una connexió a Internet puguin estalviar amplada de banda, reduint transferències repetitives d'objectes. Tanmateix, això només és una part del problema. Diversos agents participen en el rendiment d'una transferència web:

- **Proveïdor de continguts (servidor web)**: ha de planificar la seva capacitat per a poder donar servei en hores punta i aguantar possibles allaus de peticions, i així tenir una certa garantia que els seus clients tindran un bon servei amb certa independència de la demanda.

- **Client**: podria usar una memòria cau per a "economitzar" recursos de la xarxa. Quan un contingut es troba en més d'un lloc, hauria de triar el

millor servidor en cada moment: l'original o una rèplica "ràpida" (o portar l'objecte a trossos des de diversos llocs alhora).

- **Rèplica**: si un servidor desa una rèplica d'algun contingut probablement és perquè el vol oferir i reduir la càrrega del servidor original. Per tant, ha de ser "conegut" pels seus clients, però, a més, el contingut ha de ser consistent amb el contingut original.

- **Proveïdor de xarxa**: hauria de triar el millor camí per a una petició, via encaminament IP, via resolució DNS (resoldre noms a l'adreça IP més pròxima al client que consulti; no és el més habitual), via servidors intermediaris HTTP, i evitar zones congestionades (punts calents) o *flash crowd* (congestió en el servidor i en la xarxa pròxima deguda a una allau de demandes).

En aquest subapartat ens centrarem en els sistemes de distribució de documents per a ajudar a millorar el rendiment del Web. Més concretament ens centrarem en què pot fer el proveïdor dels continguts.

Les aplicacions que ofereixen continguts a Internet s'enfronten al repte de l'escala: un únic servidor davant milions de persones que eventualment poden demanar els seus serveis tots alhora: el proveïdor d'informació ha de posar tants recursos com audiència pugui tenir.

Una opció molt usada és que el proveïdor del contingut disposi de diferents rèpliques de la informació. Hi ha diversos "trucs" per a repartir peticions entre els diferents servidors que contenen les rèpliques:

- Rèpliques amb un programa que readreça la petició HTTP a la millor rèplica.

- Fer que el servei de noms DNS torni diverses adreces IP. Normalment s'envia la petició HTTP a la primera adreça de la llista d'adreces IP rebudes del DNS. Si aquesta llista està ordenada de manera diferent en cada petició al DNS (mètode *round robin*), cada vegada es farà la petició a una adreça IP diferent.

- Readreçament a escala de transport (commutador de nivell 4, L4 *switch*): un encaminador mira els paquets IP de connexions TCP cap a un servidor web (port 80) i els redirigeix a la màquina interna menys carregada

- Readreçament a escala d'aplicació (commutador de nivell 7, L7 *switch*): un encaminador que mira les connexions HTTP i pot decidir amb quina rèplica cal contactar en funció de l'URL sol·licitat. Molt complex.

- Enviar totes les peticions a un servidor intermediari invers que respongui amb contingut desat en la memòria o que passi la petició a un o diversos servidors interns.

Rèplica

Una rèplica (en anglès, *mirror*) és una còpia exacta d'un altre lloc Internet. Les rèpliques proporcionen múltiples fonts per una mateixa informació i són especialment útils per la fiabilitat que això comporta. També perquè l'usuari accedeixi a la ubicació més propera o per a repartir la càrrega entre diferents servidors.

3.8.1. Xarxes de distribució de continguts

Han sorgit empreses que s'han dedicat a instal·lar màquines en molts llocs del món i algorismes per a decidir quina màquina és la més adequada per a atendre les peticions, segons la ubicació del client i la càrrega de la xarxa. Aquestes empreses venen aquest "servidor web distribuït" a diversos clients, que paguen per a poder disposar d'un sistema de servei web de gran capacitat i que pot respondre demandes de continguts extraordinàriament altes. Aquests sistemes es denominen xarxes de distribució de continguts[13].

(13)En anglès, *content delivery networks* (CDN).

Akamai

Es poden trobar diverses empreses que ofereixen aquest servei de xarxes de distribució de continguts. Akamai és la més important. El febrer de 2010, la UOC utilitzava Akamai per a proporcionar el contingut del seu portal.

Una CDN és un programari intermediari *(middleware)* entre proveïdors de continguts i clients web. La CDN serveix continguts des de diversos servidors repartits per tot el planeta. La infraestructura de la CDN està compartida per diversos clients de la CDN i les peticions tenen en compte la situació de la xarxa per a fer que cada client web obtingui el contingut sol·licitat des del servidor més eficient de la CDN (usualment una combinació del més proper, el menys carregat, el que té un camí amb més amplitud de banda amb el client).

La CDN disposa d'un gran nombre de punts de servei en multitud de proveïdors d'Internet i pot actuar sobre el següent:

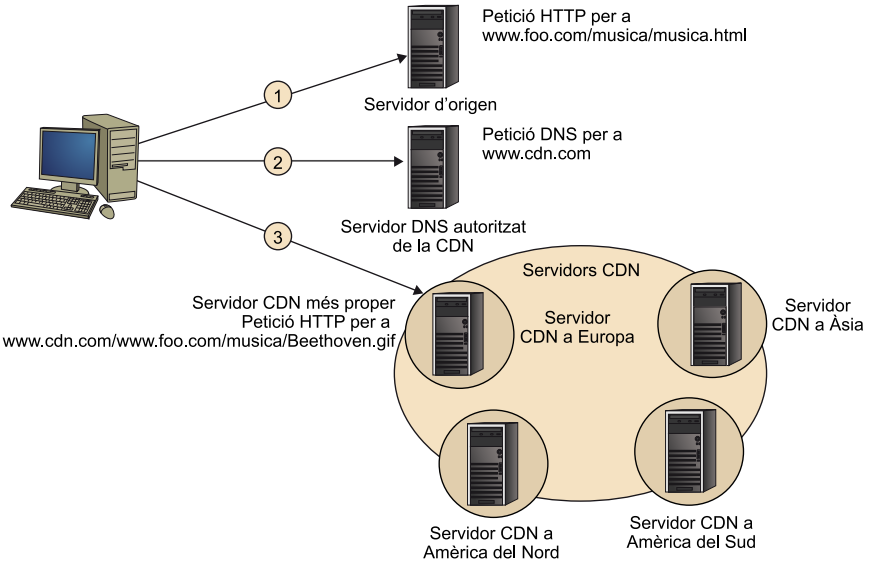
- El DNS: quan es resol un nom, el servidor DNS respon segons la ubicació de l'adreça IP del client.

- El servidor web: quan es demana una pàgina web, es reescriuen els URL interns segons la ubicació de l'adreça IP del client.

- Quan es demana un objecte a l'adreça IP d'un servidor de la CDN, el commutador (*switch*) d'accés a un conjunt de diversos servidors intermediaris, o rèpliques, pot redirigir la connexió al servidor menys carregat del grup (tot el conjunt respon sota una mateixa adreça IP).

Exemple de funcionament d'una CDN

La figura següent mostra l'exemple dels passos possibles que es poden donar quan es fa la petició d'una pàgina web amb imatges.



1) El client demana al servidor origen (www.foo.com) una pàgina sobre música. El servidor informa al navegador web que la pàgina demanada conté diversos objectes. L'URL d'aquests objectes es modifica perquè facin referència a la CDN. Per exemple, en lloc de contestar http://www.foo.com/musica/Beethoven.gif contesta http://www.cdn.com/www.foo.com/musica/Beethoven.gif.

2) El navegador demana al DNS que resolgui www.cdn.com. La petició arriba fins al DNS autoritzat per aquest domini, que és un DNS de l'empresa propietària de la CDN. La CDN respon amb l'adreça IP del node de la CDN més proper a la ubicació del navegador web (que és qui fa la petició pels continguts) al DNS de la CDN. Aquesta "proximitat" es calcula a partir d'informació que va recollint sobre les distàncies dels nodes de la CDN a l'ISP.

3) Finalment, el navegador web demana els objectes al node de la CDN que li han assignat.

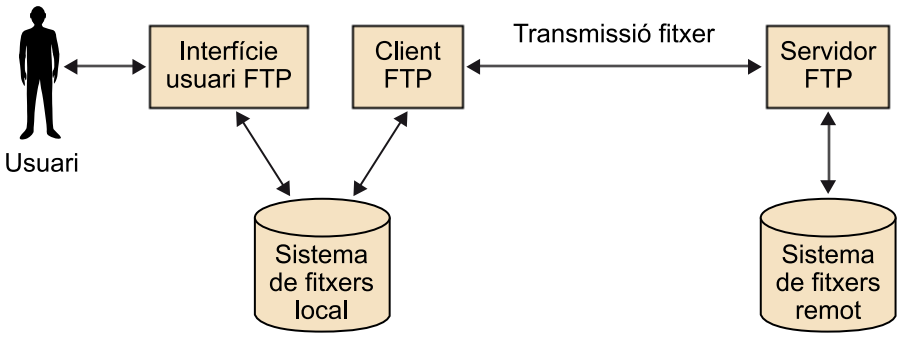
Les CDN no s'utilitzen només per a distribuir continguts web. També s'usen per a reproducció en temps real d'àudio i vídeo, tant emmagatzemat com en directe.

Finalment cal dir que, per a proporcionar el seu servei, les CDN usen el DNS per a unes funcions per a les quals no està dissenyat. Les CDN utilitzen el DNS per a redirigir les peticions de manera transparent però aquest mecanisme de redirecció sobrecarrega el DNS, ja que fa que les peticions adreçades a la CDN no es puguin posar a la memòria cau.

4. Transferència de fitxers

El protocol de transmissió de fitxers FTP[14] permet transferir fitxers d'un ordinador a un altre. Funciona seguint el model client-servidor sobre una connexió TCP/IP. A la figura següent es pot veure el funcionament a grans trets de l'FTP.

(14)Sigles en anglès de File Transfer Protocol



Per a poder fer la transferència, l'usuari s'ha de connectar a l'ordinador remot proporcionant un usuari i una contrasenya. La transferència es pot fer tant de l'ordinador de l'usuari a l'ordinador remot com a l'inrevés.

L'FTP utilitza dues connexions entre el client i el servidor, una per a les dades i l'altra per al control. La **connexió de control** usa el port TCP número 21 i s'utilitza per a enviar informació de control, com ara l'usuari i la contrasenya, instruccions per a canviar de directori o instruccions per a demanar i enviar fitxers. Aquesta connexió es manté oberta durant tota la sessió. La **connexió de dades** s'utilitza per a enviar els fitxers i es fa pel port TCP número 20. S'estableix una connexió de dades per cada fitxer. Un cop enviat la connexió es tanca. En cas que es vulgui enviar un altre fitxer s'obre una nova connexió de dades.

El servidor FTP manté informació d'estat durant una sessió. El servidor ha d'associar la connexió de control amb un compte d'usuari i ha de saber en tot moment en quin directori es troba l'usuari, ja que aquest pot anar canviant de directori en el servidor remot.

Les instruccions s'envien codificades en ASCII i finalitzades amb un *carriage return* i un *line feed*. Les respostes també acaben amb un *carriage return* i un *line feed*. Les instruccions consisteixen en quatre caràcters ASCII en majúscules, algunes amb arguments opcionals. Algunes de les instruccions més usuals són (aquestes instruccions són del protocol, no les que l'usuari posa a l'intèrpret d'instruccions de l'aplicació FTP client):

- `USER nomUsuari`: per a enviar el nom d'usuari al servidor.
- `PASS clau`: per a enviar la clau d'accés al servidor.
- `LIST`: per a demanar al servidor que enviï la llista de fitxers del directori remot actual. La llista de fitxers s'envia per una connexió de dades.
- `RETR fitxer`: per a obtenir un fitxer del directori actual de l'ordinador remot.
- `STOR fitxer`: per a enviar un fitxer al directori actual de l'ordinador remot.

Cada instrucció enviada a l'ordinador remot genera una resposta d'aquest ordinador. Les respostes són nombres de tres dígits seguits d'un missatge opcional. Algunes respostes usuals són:

- `331 Username Ok, password required`
- `125 Data connection already open; transfer starting`
- `425 Can't open data connection`
- `452 Error writing file`

Consulta recomanada

Sobre les comandes i les respostes, en trobareu més informació sobre les instruccions i respostes en l'RFC 959.

4.1. Seguretat en la transferència de fitxers

L'especificació original de l'FTP és inherentment insegura perquè no té cap mètode per a transferir dades de manera xifrada. Això vol dir que en la majoria de configuracions de xarxa, el nom d'usuari, la clau d'accés, les instruccions FTP i els fitxers transferits es poden capturar des de la mateixa xarxa utilitzant un detector. La solució habitual a aquest problema és usar SFTP[15] o FTPS[16].

(15)SFTP és la sigla de *SSH File Transfer Protocol* o *Secure File Transfer Protocol*.

(16)FTPS és la sigla d'*FTP sobre SSL*.

L'SFTP és un protocol que proporciona accés, transferència i gestió de fitxers sobre un canal fiable. Normalment s'utilitza amb SSH, i el SSH és el qui proporciona el canal fiable, encara que es podria utilitzar amb altres protocols de seguretat. Per tant, la seguretat no la proporciona directament el protocol SFTP, sinó SSH o el protocol que s'utilitzi per a aquest propòsit.

IETF

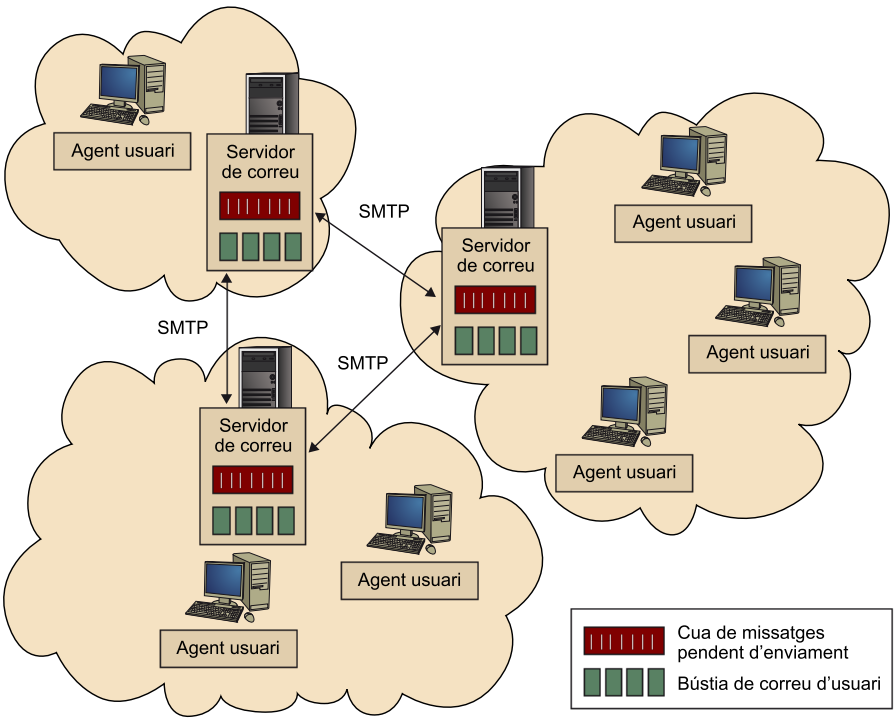
L'SFTP va ser dissenyat per l'Internet Engineering Task Force (IETF) com una ampliació del Secure Shell Protocol (SSH) versió 2.0.

5. Correu electrònic a Internet

El correu electrònic existeix des dels inicis de la Internet. En els anys 60 els sistemes *mainframe* ja tenien formes de comunicació un a un de tipus missatgeria i al llarg dels anys aquests sistemes han anat evolucionant fins arribar al nivell de sofisticació i potència actuals. Actualment és una de les aplicacions més importants i usades d'Internet.

A la figura següent es presenta una visió del sistema de correu a Internet. Quan un usuari vol enviar un missatge, l'agent usuari de missatgeria que està utilitzant envia el missatge a un servidor de correu. Aquest –el servidor de correu origen– pregunta al DNS l'adreça corresponent al camp MX del domini de destinació del missatge i envia el missatge al port 25 d'aquest servidor de correu via SMTP. La transmissió del missatge es fa utilitzant el protocol TCP perquè proporciona una transmissió fiable de les dades. Un cop el servidor de destinació ha acceptat el missatge l'emmagatzema a la bústia de l'usuari destinatari perquè més endavant l'usuari destinatari –amb prèvia autenticació– el llegeixi via un agent usuari de missatgeria. Cal destacar el fet que el protocol SMTP normalment no utilitza cap servidor intermediari entre el servidor origen ni el de destinació. La comunicació es fa directament encara que cadascú estigui a una punta del món.

Visió d'alt nivell del sistema de correu electrònic a Internet



En cas que el servidor origen no pugui enviar el missatge al servidor de destinació per algun error, el servidor origen posa el missatge en cua i prova d'enviar-lo més tard. Els reintents es fan normalment cada 30 minuts. Si després d'uns quants dies no es pot lliurar el missatge el servidor origen elimina el missatge i notifica al remitent que no l'ha pogut lliurar.

5.1. SMTP

El protocol **Simple Mail Transfer Protocol** (SMTP) està definit en l'RFC 2821 i és l'estàndard actual de transmissió de correu electrònic a Internet.

RFC de l'SMTP

El primer RFC que descriu l'SMTP data del 1982 (RFC 821), però abans d'aquesta data ja hi havia versions del que ara és l'SMTP.

El protocol SMTP té molt bones qualitats, com es pot deduir per l'èxit que ha tingut, però té alguns inconvenients heretats del passat. Per exemple, tant el cos dels missatges com les capçaleres estan codificats en caràcters ASCII de 7 bits. Això fa que qualsevol dada que no estigui codificada en aquest format s'hagi de codificar a ASCII de 7 bits abans d'enviar, i cal tornar-lo a descodificar un cop rebut.

Exemple d'enviament d'un missatge

A continuació posem l'exemple de l'enviament d'un missatge de `uoc.edu` a `upc.cat` utilitzant SMTP. L'exemple mostra les instruccions que s'intercanvien un cop la connexió fiable (la connexió TCP) ha estat establerta. Per a clarificar la comprensió de l'exemple, les instruccions que envia el client (originador) s'han prefixat amb `C:` i les del servidor (receptor) amb `S:`. També hem numerat les línies per a poder referir-nos-hi més fàcilment.

```
1 S: 220 upc.cat
2 C: HELO uoc.edu
3 S: 250 Hello uoc.edu please to meet you
4 C: MAIL FROM: <marques@uoc.edu>
5 S: 250 marques@uoc.edu ... Sender OK
6 C: RCPT TO: <puig@upc.cat>
7 S: 250 puig@upc.cat ... Recipient OK
8 C: DATA
9 S: 354 Enter mail, end with "." on a line by itself
10 C: Aquest és un missatge de correu d'exemple.
11 C: Oi que és senzill?
12 C: .
13 S: 250 Message accepted for delivery
14 C: QUIT
15 S: 221 upc.cat closing connection
```

En aquest exemple el client envia el missatge `Aquest és un missatge de correu d'exemple. Oi que és senzill?` des del servidor de correu `uoc.edu` a `upc.cat`.

Com a part del diàleg, el client envia les instruccions: `HELO`, `MAIL FROM`, `RCPT TO`, `DATA` i `QUIT`. El client envia el missatge (línies 10 a 11) després de la instrucció DATA i l'acaba amb una línia que només conté un punt (línia 12: `CRLF.CFLR`), que indica el final del missatge enviat al servidor. El servidor contesta cada instrucció amb un codi de resposta i un text opcional explicatiu (en anglès).

El protocol SMTP usa connexions persistents. En cas que el servidor origen tingui més d'un missatge per enviar al mateix servidor de destinació els missatges s'envien aprofitant la mateixa connexió TCP. El client comença cada missatge amb un nou `MAIL FROM: uoc.edu`, acaba cada missatge amb un punt en una línia que només conté un punt, i envia la instrucció `QUIT` un cop ha acabat d'enviar tots els missatges.

Diàleg amb el servidor SMTP

Si teniu accés a un servidor SMTP que no requereixi autenticació, vosaltres mateixos podeu provar de dialogar directament amb el servidor SMTP. Per a fer-ho només heu de fer: `telnet nomServidor 25`, sent `nomServidor` el del servidor SMTP, al qual us connecteu. Fent això esteu establint una connexió TCP amb el servidor SMTP. Podeu repetir les instruccions de l'exemple prefixades amb `C:` i veureu que s'envia un missatge de correu electrònic. (Proveu de posar com a recipient una adreça de correu electrònica vostra i rebreu el missatge.)

5.2. Format dels missatges

Tal com passa en les cartes postals, que tenen un sobre i una carta, els missatges de correu electrònic tenen dues parts:

- **Capçalera**: recull la informació general del missatge. Seria equivalent al sobre de la carta postal i està format per diversos camps de capçalera.

- **Cos del missatge**: conté el missatge en si. Correspon al contingut de la carta postal. Aquesta part és opcional.

Format de les capçaleres

L'RFC 822 especifica el format de les capçaleres i la seva semàntica. Cada camp de la capçalera consta d'un **nom del camp** seguit del caràcter `":"`, opcionalment acompanyat per un **cos del camp**, i acaba amb un `CRLF`.

Exemple de capçalera d'un missatge

L'exemple del subpartat 5.1 no conté cap capçalera. A continuació repetim el mateix missatge incloent-hi la capçalera del missatge. Així, primer trobem la **capçalera** (línies 10 a 13), seguida del **cos** del missatge (línies 14 i 15). Tal com es pot veure a la línia 13 el separador entre la capçalera i el cos del missatge és una línia en blanc (o sigui, CRLF). Aquests camps de la capçalera són diferents de les instruccions de l'SMTP (`HELO`, `MAIL FROM`, `RCPT TO`, `DATA` i `QUIT`), encara que puguin semblar similars.

```
1 S: 220 upc.cat
2 C: HELO uoc.edu
3 S: 250 Hello uoc.edu please to meet you
4 C: MAIL FROM: <marques@uoc.edu>
5 S: 250 marques@uoc.edu ... Sender OK
6 C: RCPT TO: <puig@upc.cat>
7 S: 250 puig@upc.cat ... Recipient OK
8 C: DATA
9 S: 354 Enter mail, end with "." on a line by itself
10 C: From: marques@uoc.edu
11 C: To: puig@upc.cat
12 C: Subject: Missatge d'exemple
13 C:
14 C: Aquest és un missatge de correu d'exemple.
15 C: Oi que és senzill?
16 C: .
17 S: 250 Message accepted for delivery
18 C: QUIT
19 S: 221 upc.cat closing connection
```

5.2.1. Format de missatges MIME

Tal com hem dit anteriorment, una limitació de la norma RFC 822 és que defineix un format de missatge i un contingut amb una única part de text en ASCII de 7 bits. Es va veure que aquest format era molt pobre i que calia algun

[17] *MIME* és la sigla de *Multipurpose Internet Mail Extensions*.

mètode per a superar-ne les limitacions. El format MIME[17] redefineix el format del missatge per tal de permetre, sense perdre la compatibilitat amb el format definit per l'RFC 822, donar suport al següent:

RFC del format MIME

El format de missatges MIME està especificat en els RFC 2045, RFC 2046, RFC 2047, RFC 4288, RFC 4289 i RFC 2049.

- Text codificat amb un joc de caràcters diferent de l'ASCII de 7 bits.
- Adjuncions que no siguin text.
- Cossos del missatge amb múltiples parts.
- Capçaleres codificades amb un joc de caràcters diferent de l'ASCII de 7 bits.

Per a fer-ho MIME defineix un conjunt de nous camps de capçalera: `MIME-version`, `Content-ID`, `Content-Type`, `Content-Disposition` i `Content-Transfer-Encoding`.

En aquest mòdul ens centrarem en `Content-Type:` i `Content-Transfer-Encoding:`.

Content-Type

`Content-Type:` indica a l'agent d'usuari de missatgeria receptor del missatge quin tipus de contingut conté el cos del missatge, cosa que permet que aquest prengui l'acció apropiada per a cada contingut de missatge. El tipus de contingut s'especifica amb un tipus i un subtipus.

Exemple

```
Content-Type: image/JPEG
```

Indica que el cos del missatge conté una imatge formatada en JPEG. L'agent usuari de missatgeria que rebi el missatge pot fer així les accions necessàries, com ara enviar el cos del missatge a un descompressor JPEG.

Format: `Content-Type: type/subtype; parameters`

Alguns tipus	Alguns subtipus
text	plain, html
image	jpeg, gif
audio	basic, mp4, mpeg
video	mpeg, quicktime
application	msword, octet-stream, zip

La Internet Assigned Numbers Authority (IANA) manté la llista de tipus i subtipus. La podeu trobar a http://www.iana.org/assignments/media-types.

Content-Transfer-Encoding

Recordeu que hem dit que els missatges SMTP han d'estar codificats en ASCII de 7 bits. La capçalera `Content-Transfer-Encoding:` indica que el cos del missatge ha estat codificat en ASCII i indica el tipus de codificació usat.

Exemple

```
Content-Transfer-Encoding: base64
```

Indica que s'ha utilitza la codificació base64 per a codificar el cos del missatge. La codificació base64 codifica una seqüència arbitrària de bytes en una forma que satisfà les regles de l'ASCII de 7 bit, i per tant, no confondrà l'SMTP.

Una altra tècnica popular per a codificar que també satisfà l'ASCII de 7 bits és la *quoted-printable*, que s'acostuma a utilitzar per a convertir text codificat en ASCII de 8 bits (que pot contenir caràcters no anglesos) a ASCII de 7 bits.

Exemple de *quoted-printable*

El següent és un exemple de *quoted-printable*:

```
Subject: =?iso-8859-1?Q?Qu=E8?= tal =?iso-8859-1?Q?est=E0s=3F?=
```

que correspon a "Subject: Què tal estàs?" ęn què:

- `=` delimita la seqüència codificada amb *quoted-printable*. En aquest cas, hi ha delimitades dues seqüències: `Què` i `estàs?`
- `?` separa les parts
- codificat en ISO-8859-1
- `Q`: Codificació *quoted-printable*
- `E8` és el codi corresponent a è en la codificació iso-8859-1

Quan torneu a rebre un missatge (sovint reenviat) amb una seqüència com aquesta ja sabreu d'on surt :-)

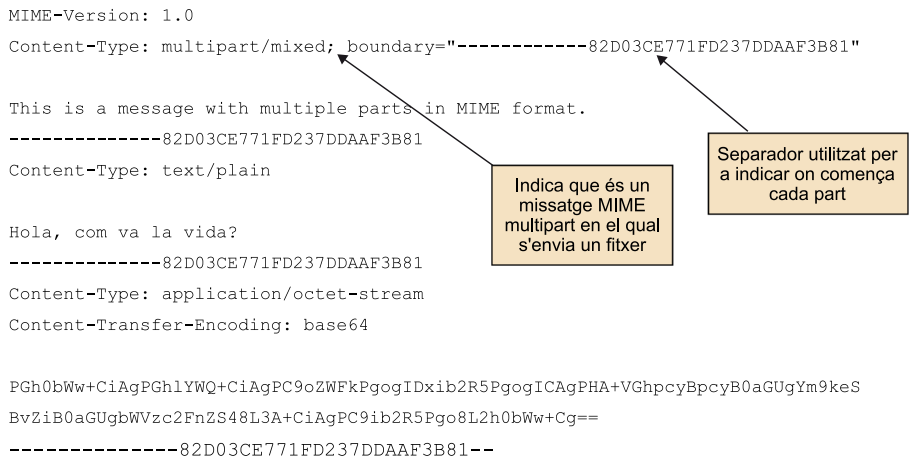
A continuació posarem un exemple que recollirà els diferents aspectes vistos sobre el format MIME:

```
From: marques@uoc.edu
To: puig@upc.cat
Subject: Foto del nen
MIME-version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

(dades codificades en base64
.............................................
dades codificades en base64)
```

La capçalera `MIME-version` indica la versió de MIME que s'està utilitzant.

La capçalera `Content-Type:` també s'utilitza perquè un missatge pugui contenir text i fitxers adjunts. En aquest cas a la capçalera `Content-Type:` s'hi afegeix un indicador d'inici i final de cada part (`boundary`).



S/MIME (Secure/Multipurpose Internet Mail Extensions) és un estàndard per xifratge de clau pública i signatura de correu electrònic encapsulat amb MIME. S/MIME proporciona:

- autenticació, integritat i no-repudi (usant signatura digital);
- privacitat i seguretat de les dades (usant xifratge).

5.3. Protocols per a accedir a les bústies de correu

Els servidors SMTP desen els missatges rebuts a les bústies de correu dels destinataris dels missatges. Aquests missatges es poden veure connectant-se al servidor SMTP i executant un programa per a llegir el correu. La majoria d'usuaris, però, llegeixen el correu electrònic des d'un ordinador personal utilitzant un client de correu que els ofereix un conjunt ampli de funcionalitats, incloent-hi la possibilitat de veure missatges multimèdia i adjuncions.

5.3.1. POP3

POP3[18] és un protocol de nivell d'aplicació perquè un agent d'usuari de correu electrònic (el client) pugui obtenir el correu d'un servidor de correu remot. El funcionament és molt senzill. El client obre una connexió TCP amb el servidor al port 110. Un cop establerta aquesta connexió, el POP3 passa per tres fases:

[18]El protocol POP3 (Post Office Protocol versió 3) està especificat en l'RFC 1939.

1) Autorització: l'usuari envia un usuari i una clau d'accés (en clar) per autenticar-se.

2) Transacció: l'agent d'usuari baixa missatges. En aquesta fase l'agent d'usuari pot marcar els missatges perquè s'esborrin, eliminar marques d'esborrar, i obtenir informació sobre la bústia de l'usuari.

3) Actualització: ocorre després que el client ha executat la instrucció `QUIT`, que finalitza la sessió POP3. És en aquest moment quan el servidor esborra els missatges que s'havien marcat per a ser esborrats.

Les instruccions són paraules clau, que poden estar seguides d'arguments, i que acaben amb un salt de línia (CRLF). Tant les paraules clau com els arguments són caràcters ASCII imprimibles.

Les respostes consisteixen en un indicador d'estat (`+OK`) o (`-ERR`) que pot estar seguit per una informació addicional:

- `+OK`: el servidor indica al client que la instrucció rebuda era correcta
- `-ERR`: el servidor indica al client que hi havia alguna cosa errònia en la instrucció rebuda.

Exemple d'una sessió d'autenticació

El següent és un exemple d'una sessió d'autenticació (un cop establerta la connexió al port TCP 110 del servidor de correu).

```
+OK POP3 server ready
USER marques
+OK
PASS uoc
+OK user successfully
logged on
```

L'usuari pot configurar el seu agent d'usuari –utilitzant POP3– perquè apliqui la política "baixa els missatges i esborra'ls" o la política "baixa els missatges i deixa'n còpia al servidor".

Connexió a un servidor de correu

Si teniu accés a un servidor de correu que no requereixi utilitzar una connexió segura podeu provar de connectar-vos-hi amb `telnet servidorCorreu 110`.

Exemple de baixar i esborrar els missatges

L'agent d'usuari obté una llista dels missatges (`LIST`), i els va baixant (`RETR`) i esborrant (`DELE`) un per un. Per claredat identifiquem les línies emeses per l'agent d'usuari amb **C:** i pel servidor de correu amb **S:**

```
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
C: RETR 1
S: +OK 120 octets
S: <el servidor de POP3 envia el missatge 1>
S: .
C: DELE 1
S: +OK message 1 deleted
C: RETR 2
S: +OK 200 octets
S: <el servidor de POP3 envia el missatge 2>
S: .
C: DELE 2
S: +OK message 2 deleted
C: QUIT
S: +OK POP3 server signing off
```

Un problema del mode "baixa els missatges i esborra'ls" és que el correu es baixa a un ordinador. Si l'usuari posteriorment vol accedir al correu des d'un altre ordinador no ho podrà fer, ja que el servidor ja no contindrà els missatges. Això és problemàtic per a usuaris que acostumen a llegir el correu des de diferents ordinadors; per exemple, l'ordinador de casa, un ordinador portàtil, la feina, etc.

El servidor POP3 no desa estat entre sessions.

5.3.2. IMAP

El protocol IMAP[19] és un protocol d'accés a correu, com el POP3, però té moltes més funcionalitats. També és més complex d'implementar, tant en la part client com en la part del servidor.

(19)L'IMAP està especificat en l'RFC 3501.

Els servidors IMAP[20] associen cada missatge a una carpeta. Quan un missatge arriba al servidor, s'associa a la carpeta INBOX. El receptor pot, posteriorment, moure aquest missatge a una altra carpeta creada per l'usuari mateix. També el pot llegir, esborrar, etc. El protocol IMAP proporciona instruccions per a permetre als usuaris crear carpetes i per a moure missatges d'una carpeta a una altra. L'IMAP també proporciona instruccions que permeten als usuaris fer cerques de missatges que satisfacin certs criteris en les carpetes remotes. De tot això ja es veu que els servidors IMAP han de mantenir informació entre sessions IMAP: els noms de les carpetes, quins missatges estan associats a cada carpeta, etc.

(20)*IMAP* és la sigla d'*Internet Mail Acess Protocol.*

Una altra característica interessant de l'IMAP és que té instruccions que permeten a l'agent d'usuari obtenir parts d'un missatge: pot obtenir només les capçaleres del missatge, o només una part d'un missatge multipart (per exemple, el text del missatge sense el fitxer adjunt). Això és especialment útil quan es fa servir una connexió amb poca amplada de banda.

5.3.3. Web

Una manera molt popular d'accedir al correu és per mitjà d'un navegador web. En aquest cas, l'agent d'usuari és el navegador web i l'usuari es comunica amb la seva bústia via HTTP. D'aquesta manera quan es vol llegir un missatge el navegador l'obté del servidor de correu usant el protocol HTTP i quan es vol enviar un missatge, el navegador l'envia al servidor de correu usant el protocol HTTP.

6. Aplicacions d'igual a igual per a la compartició de fitxers

L'arquitectura d'igual a igual és una manera de construir aplicacions que persegueix la utilització dels recursos informàtics i l'amplada de banda que aporten els usuaris de l'aplicació encara que aquests estiguin connectats de manera intermitent.

Bibliografia recomanada

En l'article de Lua i altres (2005). "A survey and comparison of peer-to-peer overlay network schemes". *IEEE Communications Surveys & Tutorials* (vol. 7, núm. 2), trobareu un resum de com funcionen les xarxes superposades d'igual a igual més populars, i també una comparativa entre aquestes. També trobareu més detall dels sistemes explicats en aquest apartat.

Les aplicacions d'igual a igual s'han popularitzat de la mà de les aplicacions de compartició de fitxers. La primera va ser el Napster, però l'han anat seguint altres aplicacions. En aquest material en presentarem diverses, que ens han d'ajudar a entendre com funcionen aquests tipus d'aplicacions. Cadascuna d'aquestes aplicacions utilitza la seva organització interna i protocols propis.

Els nodes que formen el sistema o aplicació d'igual a igual s'organitzen formant una xarxa superposada (*overlay network*, en anglès), que funciona sobre la xarxa física que connecta els nodes. Hi ha diferents maneres d'organitzar aquesta xarxa superposada, les quals es poden dividir en dues categories: estructurades i no estructurades.

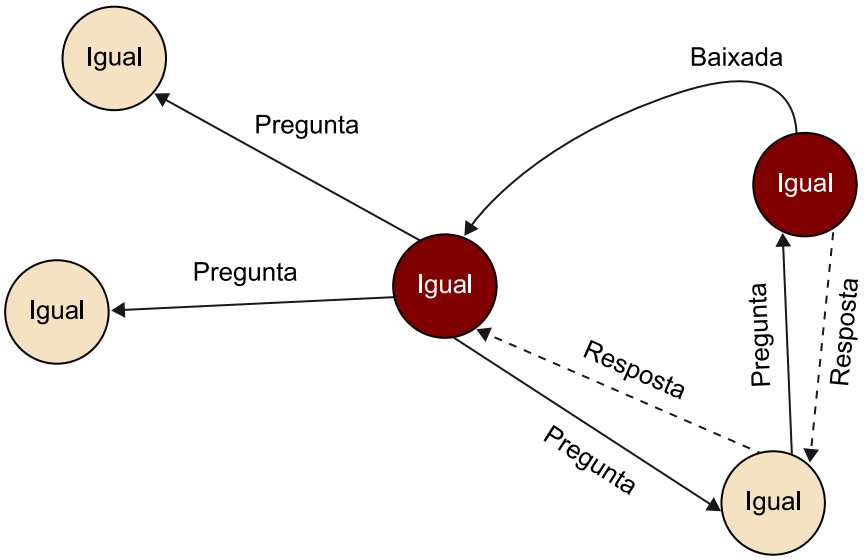
6.1. Xarxes superposades no estructurades

Un sistema d'igual a igual que utilitzi una xarxa superposada de tipus no estructurat és un sistema que està compost d'iguals que es connecten a la Xarxa sense conèixer-ne la topologia. Aquests sistemes usen mecanismes d'inundació per a enviar consultes a través de la xarxa superposada. Quan un igual rep la pregunta, envia a l'igual que l'ha originada una llista de tot el contingut que encaixa amb la pregunta. Mentre que les tècniques basades en la inundació van bé per a localitzar objectes altament reproduïts i són resilients davant de les connexions i desconnexions dels nodes, no tenen un comportament gaire bo quan es fan cerques d'objectes poc reproduïts. D'aquesta manera, les xarxes superposades no estructurades tenen fonamentalment un problema: quan han de gestionar un ritme elevat de consultes o quan hi ha creixements sobtats de la mida del sistema se sobrecarreguen i tenen problemes d'escalabilitat.

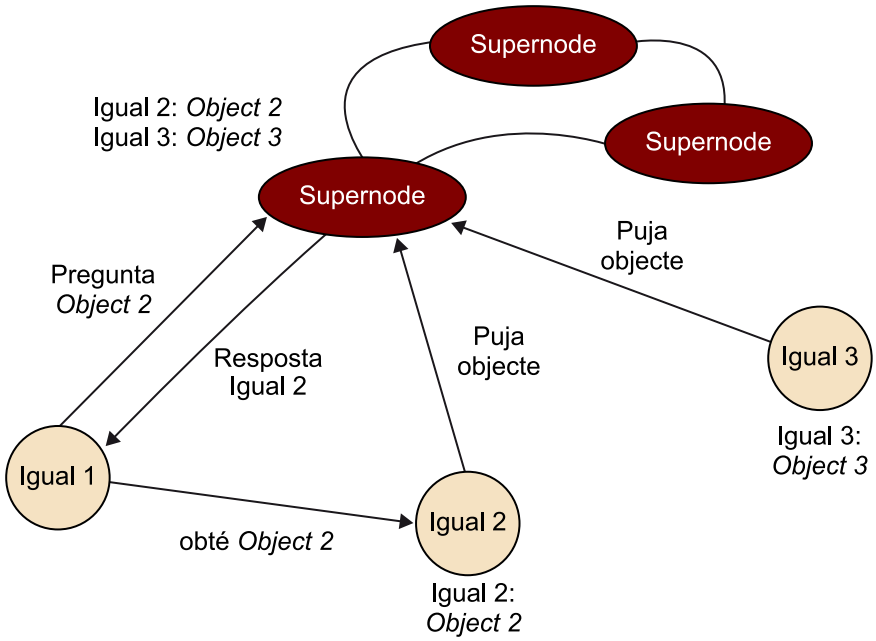
Tot i que el sistema d'encaminament basat en claus que fan servir els sistemes d'igual a igual estructurats –veurem aquest tipus de sistemes en el proper subapartat– pugui localitzar de manera eficient objectes i, a més, sigui escalable, pateixen sobrecàrregues significativament més grans que els sistemes no estructurats per a continguts populars. Per això, els sistemes descentralitzats no estructurats s'usen més.

Alguns exemples de sistemes no estructurats són Gnutella, FastTrack/KaZaA, BitTorrent i Overnet/eDonkey2000.

Gnutella: protocol totalment descentralitzat per a fer cerques sobre una topologia d'iguals totalment plana. Ha estat (i encara és) molt usat. Per a localitzar un objecte, un igual pregunta als seus veïns. Aquests inunden els seus veïns i així fins a un cert radi. Aquest mecanisme és extremament resilient davant d'entrades i sortides de nodes, però els mecanismes actuals de cerca no són escalables i generen càrregues no esperades en el sistema. La figura següent mostra un exemple de cerca.



KaZaA: és un sistema de fitxers descentralitzat en el qual els nodes s'agrupen al voltant de superiguals (*super-peers* en anglès) per a fer les cerques més eficients, tal com es mostra en la figura següent. La comunicació entre els iguals a KaZaA es fa utilitzant el protocol Fast Track, que és un protocol de propietat. Els superiguals són iguals del sistema que s'han escollit perquè mantinguin metainformació que farà les cerques més eficients. En el moment d'una cerca, l'igual pregunta al superigual al qual està connectat. Aquest, de manera similar al que fa Gnutella, fa una difusió als altres superiguals.



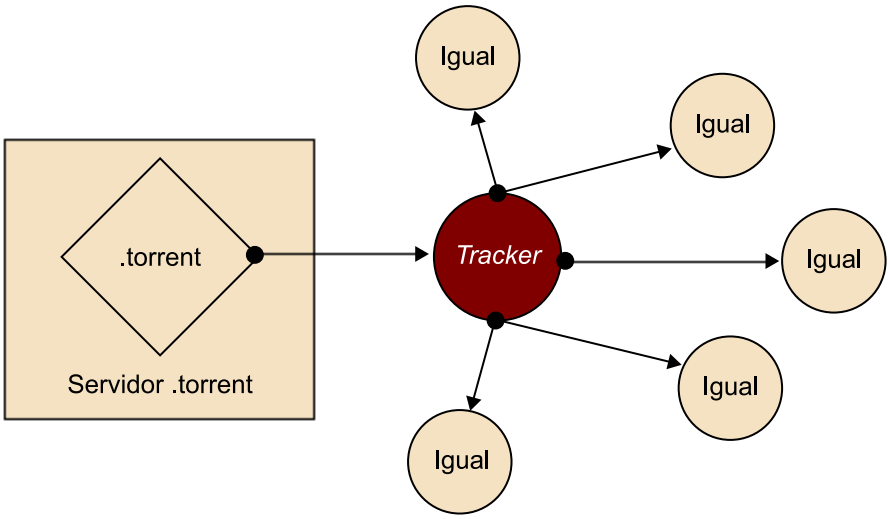
Els iguals es connecten a un superigual. Les consultes s'encaminen cap als superiguals. Les baixades es fan entre iguals.

BitTorrent: és un sistema d'igual a igual per a distribuir grans volums de dades sense que l'originador de la informació hagi de suportar tot el cost dels recursos necessaris per a servir el contingut. Aquest tipus de solucions són útils per a distribuir continguts que són molt populars. BitTorrent fa servir servidors per a gestionar les baixades. Aquests servidors emmagatzemen un fitxer que conté informació sobre el fitxer: llargada, nom, informació de resum (*hashing information*, en anglès) i l'URL del *tracker*. El *tracker* (vegeu la figura següent) coneix tots els iguals que tenen el fitxer (tant totalment com parcialment) i fa que els iguals es connectin els uns amb els altres per a baixar o pujar els fitxers. Quan un node vol baixar un fitxer envia un missatge al *tracker*, que li contesta amb una llista aleatòria de nodes que estan baixant el mateix fitxer. BitTorrent parteix els fitxers en trossos (de 256 kB) per a poder saber què té cadascun dels iguals. Cada igual que està baixant el fitxer anuncia als seus iguals els trossos que té. El protocol proporciona mecanismes per a penalitzar els usuaris que obtenen informació sense proporcionar-ne. D'aquesta manera, a l'hora de pujar informació, un igual escollirà un altre igual del qual hagi rebut dades.

Bibliografia complementària

Per a més informació sobre el funcionament de BitTorrent podeu consultar l'article següent:

B. Cohen (2003, juny). "Incentives Build Robustness in BitTorrent". *Proc. First Workshop the Economics of Peer-to-Peer Systems*. Berkeley. Califòrnia: University of Berkeley.



eDonkey: és un sistema d'igual a igual híbrid organitzat en dos nivells per a l'emmagatzematge d'informació. Està format per clients i servidors. Els servidors actuen com a concentradors per als clients i permeten als usuaris localitzar els fitxers que hi ha a la Xarxa. Aquesta arquitectura proporciona baixada concurrent d'un fitxer des de diverses ubicacions, ús de funcions resum (*hash*, en anglès) per a la detecció de fitxers corruptes, compartició parcial de fitxers mentre es baixen, i mètodes expressius per a fer cerques de fitxers. Perquè un node es pugui connectar al sistema cal que conegui un igual que actuï com a servidor. En el procés de connexió, el client proporciona al servidor la informació sobre els fitxers que comparteix. Quan un client busca un fitxer, els servidors proporcionen les ubicacions dels fitxers. D'aquesta manera els clients poden baixar els fitxers directament de les ubicacions indicades.

eMule pot funcionar tant sobre la xarxa eDonkey com sobre Kad. Kad és una implementació de Kademlia, una xarxa superposada estructurada (en el proper apartat s'explica com funcionen aquests tipus de sistemes).

6.2. Xarxes superposades estructurades

La topologia de la xarxa superposada sobre la qual es construeixen aquests sistemes està fortament controlada i el contingut no va a qualsevol lloc, sinó a un lloc determinat que fa que les consultes siguin més eficients. Aquests sistemes fan servir taules de resum distribuïdes (*distributed hash tables* o DHT en anglès) com a substrats, en els quals la ubicació dels objectes (o valors) es fa de manera determinista. Els sistemes que es basen en DHT tenen la propietat que assignen els identificadors de nodes d'una manera consistent als iguals dins d'un espai amb molts identificadors. Als objectes de dades els assigna un identificador, que s'anomena *clau*, escollit dins del mateix espai de noms. El protocol de la xarxa superposada mapa les claus a un únic node entre els connectats. Aquestes xarxes superposades suporten l'emmagatzematge i la recuperació de parells {clau,valor} en la xarxa superposada.

Aquest tipus de sistemes usen un sistema d'encaminament estructurat, tot mantenint tant la descentralització de Gnutella com l'eficiència i garantia de localitzar els resultats del Napster.

Per això, cada igual manté una taula d'encaminament petita. Els missatges s'encaminen d'una manera progressiva cap als iguals a través de camins de superposició. Cada node envia el missatge al node de la seva taula d'encaminament que té un identificador més proper a la clau en l'espai d'identificadors. Els diferents sistemes basats en DHT tenen diferents esquemes d'organització per als objectes de dades i el seu espai de claus i estratègies d'encaminament. En teoria, els sistemes basats en DHT garanteixen que, de mitjana, es pot localitzar qualsevol objecte en $O(\log N)$ salts a la xarxa superposada, en què N és el nombre d'iguals en el sistema. El camí entre dos nodes en la xarxa física pot ser molt diferent del camí en la xarxa superposada DHT. Això pot provocar que la latència en les cerques en un sistema d'igual a igual basat en una xarxa DHT sigui força gran i pugui afectar negativament el rendiment de l'aplicació que funcioni per sobre.

Exemples de xarxes superposades estructurades

Can, Chord, Tapestry, Pastry, Kademlia, DKS o Viceroy són exemples de xarxes superposades estructurades.

Podeu trobar més informació d'aquests sistemes en la bibliografia següent:

CAN
S. Ratnasamy i altres (2001). "A Scalable Content Addressable Network". *Proc. ACM SIGCOMM* (pàg. 161-172).

Chord
I. Stoica; R. Morris i altres (2003). "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications". *IEEE/ACM Trans. Net.* (vol. 11, núm. 1, pàg. 17-32).

Tapestry
B. Y. Zhao i altres (2004, gener). "Tapestry: A Resilient Global-Scale Overlay for Service Deployment". *IEEE JSAC* (vol. 22, núm. 1, pàg. 41-53).

Pastry
A. Rowstron; P. Druschel (2001). "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems". *Proc. Middleware*.

Kademlia
P. Maymounkov; D. Mazieres (2002, febrer). "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric". *Proc. IPTPS* (pàg. 53-65). Cambridge, EUA.

DKS
DKS(N,k,f): A Familiy of Low Communication, Scalable and Fault-Tolerant Infrastructures for P2P Applications.

Viceroy
D. Malkhi; M. Naor; D. Ratajczak (2002, juliol). "Viceroy: A Scalable and Dynamic Emulation of the Butterfly". *Proc. ACM PODC 2002* (pàg. 183-92). Monterey, EUA.

eMule té una versió que funciona sobre la xarxa Kad, que és una implementació de Kademlia. Kad millora la localització de fitxers a la Xarxa i la fa més resistent a atacs als servidors centrals.

7. Missatgeria instantània

La missatgeria instantània és una forma de comunicació textual en temps real entre dues o més persones per mitjà d'Internet o algun altre tipus de xarxa. La missatgeria instantània es diferencia del correu electrònic en el fet que l'usuari percep sincronisme en la comunicació, encara que molts dels sistemes de missatgeria instantània també permeten enviar missatges a persones que en aquell moment no estan connectades. En aquest cas, el destinatari rebrà el missatge quan es torni a connectar al sistema.

La missatgeria instantània permet una comunicació efectiva i eficient i aconsegueix una recepció immediata de la confirmació o la resposta. Normalment les respostes es poden desar i consultar posteriorment.

Sovint amb la missatgeria instantània també trobem combinades altres funcionalitats que la fan encara més completa i popular, com càmeres de vídeo o poder parlar directament per mitjà d'Internet. En aquest apartat, però, ens centrarem en la missatgeria instantània textual.

7.1. XMPP

XMPP[21] és un protocol lliure de missatgeria instantània d'especificacions obertes basat en XML que ha estat estandarditzat per l'IETF.

(21) *XMPP* és la sigla d'*Extensible Messaging Presence Protocol*. L'XMPP abans es coneixia com a Jabber.

Com es pot veure a la figura següent, XMPP[22] utilitza una arquitectura client-servidor descentralitzada: de manera similar com fa l'SMTP, no hi ha un servidor central que coordini la missatgeria. Els clients no es comuniquen directament els uns amb els altres, ho fan a través dels servidors.

(22) El port estàndard per a l'XMPP és el 5222.

Arquitectures centralitzades

Altres sistemes de missatgeria instantània com Windows Live Messenger o AOL instant Messenger utilitzen arquitectures centralitzades.

De manera similar al correu electrònic cada usuari té una adreça única formada per dos camps: un nom d'usuari (únic per a cada servidor) seguit de l'adreça DNS del servidor que hostatja l'usuari i separats pel símbol @, com ara nom@domini.com.

Els passos perquè un missatge enviat pel client 7 (client7@server3) de la figura anterior arribi al client 3 (client3@server1) són els següents:

1) El client 7 envia el missatge al seu servidor (en aquest cas el servidor 3).

a) El servidor 3 demana al DNS l'adreça IP corresponent a servidor 1.

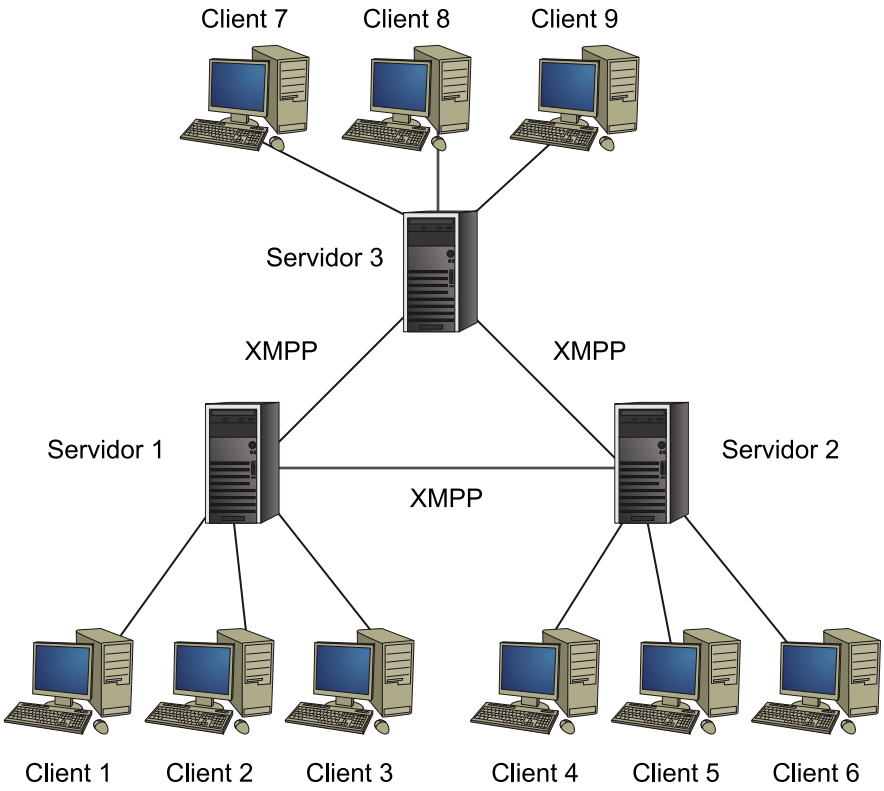
b) Si el servidor 3 bloqueja la comunicació cap al servidor 1, el missatge es descarta.

2) El servidor 3 obre una connexió amb el servidor 1.

a) Si el servidor 1 bloqueja els missatges procedents del servidor 3, el missatge es descarta.

b) Si el client 3 no està connectat en aquests moments, el servidor 1 emmagatzema el missatge per a lliurar-lo més endavant.

3) El servidor 1 lliura el missatge al client 3.



Els punts forts del protocol XMPP són els següents:

- **Descentralització**: qualsevol pot tenir un servidor XMPP. No hi ha un servidor centralitzat que coordini la missatgeria.

- **Estàndards lliures**: l'Internet Engineering Task Force té l'XMPP aprovat com un estàndard de missatgeria instantània i presencial (RFC 3920 i RFC 3921).

- **Seguretat**: els servidors XMPP es poden aïllar de la xarxa XMPP pública (per exemple, a la intranet d'una companyia), i el nucli d'XMPP inclou seguretat (via SASL i TLS).

- **Flexibilitat**: per sobre d'XMPP es poden afegir funcionalitats a mida.

- En ser un protocol obert, cada usuari pot fer servir un client XMPP diferent, sempre que compleixi les especificacions d'XMPP.

Els punts dèbils del protocol XMPP són aquests:

- ***Overhead* de la informació de presència**: entorn del 70% del tràfic entre servidors XMPP és informació de presència, el 60% de la qual és redundant. La informació de presència permet saber quins amics o contactes hi ha connectats i, per tant, a qui podem enviar missatges instantanis.

- **La inclusió de dades binàries és ineficient**: XMPP es codifica com un únic document XML. Per a poder incloure dades binàries s'han de codificar en base64. Quan cal enviar quantitats considerables de dades binàries (per exemple, enviament de fitxers) el millor és fer l'enviament de manera externa a XMPP i utilitzar missatges XMPP per a coordinar-se.

SASL

Simple Authentication and Security Layer (SASL) és un entorn de treball per a l'autenticació i la seguretat de les dades en protocols d'Internet. Separa els mecanismes d'autenticació dels protocols d'aplicació i fa possible –en teoria– que qualsevol mecanisme d'autenticació suportat per SASL es pugui utilitzar en qualsevol protocol d'aplicació. Està especificat en l'RFC 4422.

Jingle

Jingle és una ampliació d'XMPP per a iniciar i gestionar sessions multimèdia entre dues entitats XMPP de manera que sigui interoperable amb estàndards existents d'Internet.

8. Telnet i Secure Shell: accés a ordinadors remots

El protocol Telnet[23] s'usa a Internet o en xarxes d'àrea local per a proporcionar una comunicació bidireccional interactiva en l'accés a ordinadors remots. Està basat en el protocol de transport TCP. En una comunicació Telnet, normalment se segueix el model client/servidor, és a dir, el sistema usuari estableix una connexió amb el sistema proveïdor, que està esperant peticions de connexió en un port determinat. Es pot utilitzar qualsevol número de port per a les connexions i, de fet, hi ha moltes aplicacions que utilitzen el protocol Telnet per a la comunicació, cadascuna amb el seu número propi.

(23)L'especificació de Telnet es va publicar l'any 1983 en l'estàndard RFC 854.

L'aplicació bàsica, però, és establir una sessió de treball interactiva amb el sistema servidor. En aquest cas el número de port utilitzat és el 23. Aquesta sessió de treball a l'ordinador remot es fa via un terminal virtual. Per a resoldre el problema del control del terminal en l'aplicació de sessions interactives, en el protocol Telnet s'utilitza el concepte de terminal virtual de xarxa o NVT. Un NVT és un terminal virtual amb una funcionalitat molt bàsica, definida en l'especificació mateixa del protocol.

Terminal virtual de xarxa

Un terminal virtual de xarxa, en anglès *network virtual terminal* (NVT), és un dispositiu imaginari per al qual es defineixen unes funcions de control canòniques, de manera que es pot establir una correspondència entre aquestes funcions i les de cada tipus de terminal real.

Quan s'estableix la connexió entre el client i el servidor, inicialment se suposa que la comunicació es produeix entre dos NVT. Això vol dir que tant el sistema client com el sistema servidor han de mapar les seves característiques a les d'un NVT i suposar que a l'altre extrem de la connexió hi ha un altre NVT. En el mode d'operació normal, cada terminal accepta dades de l'usuari i les envia mitjançant la connexió establerta a l'altre terminal, i accepta també les dades que arriben per la connexió i les presenta a l'usuari. La comunicació es fa en bytes de 8 bits.

8.1. Seguretat del protocol Telnet

Telnet no és un protocol segur per les raons següents:

- No xifra cap dada enviada a través de la connexió (ni les claus d'accés), i això permet espiar la comunicació i poder usar la clau d'accés –i qualsevol altra informació enviada– més endavant per usos malintencionats.

- La majoria d'implementacions de Telnet no proporcionen autenticació que asseguri que la comunicació es du a terme entre els dos ordinadors desitjats i no està interceptada pel camí.

8.2. Secure Shell

La solució a la falta de seguretat del protocol Telnet ha estat el protocol Secure Shell (SSH).

Port TCP 22

El port estàndard per a contactar un servidor SSH és el port TCP 22.

L'SSH proporciona la funcionalitat de Telnet amb els afegits següents:

a) Permet el xifratge per a evitar que es puguin interceptar les dades que s'envien.

b) Permet l'autenticació amb clau pública per a assegurar que l'ordinador remot és qui diu ser. L'SSH té la debilitat que l'usuari ha de confiar en la primera sessió amb un ordinador quan encara no ha adquirit la clau pública del servidor.

9. Aplicacions multimèdia en xarxa

Les aplicacions multimèdia en xarxa tenen uns requisits molt diferents de les aplicacions tradicionals a la xarxa Internet (correu electrònic, transferència de fitxers, etc.).

Si tenim en compte dos dels requisits més rellevants que tenen les aplicacions de xarxa, com són la tolerància a pèrdues de dades i les consideracions temporals, ens adonem que aquests són especialment importants per a les aplicacions multimèdia en xarxa.

Vegeu també

Per a una descripció completa dels requisits de les aplicacions en xarxa, vegeu en aquest mateix mòdul el subapartat 1.4: "Requisits de les aplicacions".

Respecte a la pèrdua de dades, s'ha de dir que les aplicacions multimèdia hi són molt tolerants, ja que les pèrdues ocasionals de dades l'únic que fan és que hi hagi un salt en la imatge o en el so, però si es continua rebent informació, es pot continuar normalment la comunicació. Per contra, són aplicacions molt sensibles al retard, la qual cosa fa que les consideracions temporals s'hagin de tenir molt en compte. Així doncs, al llarg dels subapartats següents veurem que els paquets que arriben amb un retard de més d'uns centenars de mil·lisegons no serveixen per a res en una aplicació multimèdia i es poden descartar (un so que s'ha emès fa dos segons respecte al que s'està escoltant o una imatge que ja ha passat, no cal mostrar-los a l'usuari).

Aquestes característiques d'alta tolerància a pèrdues i molta sensibilitat al retard fan que les aplicacions multimèdia siguin molt diferents de les aplicacions tradicionals, en què és molt més important que arribin totes les dades (si falta alguna part, la informació rebuda serà inservible), que el temps que aquestes dades triguin a arribar al destinatari (que pot fer que l'experiència d'usuari no sigui gaire bona, però si al final rep totes les dades, haurà acomplert els requisits). En els propers subapartats veurem alguns exemples d'aplicacions multimèdia que ens podem trobar actualment a Internet per al suport d'aquestes aplicacions.

9.1. Exemples d'aplicacions multimèdia

Hi ha molts tipus d'aplicacions multimèdia actualment a la xarxa Internet. En aquest subapartat descriurem breument tres dels grans tipus d'aplicacions multimèdia que ens podem trobar: reproducció d'àudio/vídeo emmagatzemats, reproducció en directe d'àudio/vídeo i àudio/vídeo en temps real interactiu. Cap d'aquestes aplicacions cobreixen el cas que ens baixem un contingut i després el veiem. Aquest cas té a veure amb la transferència de fitxers que es pot fer amb protocols com HTTP (Hypertext Transfer Protocol) i FTP (File Transfer Protocol).

9.1.1. Reproducció en temps real d'àudio i vídeo emmagatzemats

En aquest tipus d'aplicacions, els clients demanen continguts d'àudio[24] i vídeo comprimits emmagatzemats en servidors. Aquests continguts poder ser, per exemple, programes de televisió, vídeos casolans, cançons, simfonies o programes de ràdio. Aquesta classe d'aplicacions tenen tres característiques distintives:

(24)En anglès *streaming*.

- **Continguts emmagatzemats**. El contingut multimèdia està ja gravat i s'emmagatzema al servidor. Com a resultat, l'usuari pot parar la reproducció, rebobinar o indexar el contingut. El temps de resposta d'un contingut d'aquesta mena pot ser d'un a deu segons per a ser acceptable.

- **Reproducció en temps real**. Quan un usuari rep un contingut amb la tècnica de la reproducció en temps real, aquesta comença poc després d'haver fet la petició del contingut. D'aquesta manera, l'usuari veu una part del contingut, mentre la resta es va rebent a mesura que es reprodueix. En aquest cas, el temps de resposta és més baix. Amb aquesta tècnica evitem el retard que podem tenir si ens hem de baixar tot el contingut.

- **Reproducció contínua**. Una vegada que la reproducció del contingut multimèdia comença ha de transcórrer tal com es va gravar originalment. Això fa que hi hagi uns forts requisits de retard en el lliurament de les dades en aquest tipus d'aplicació. L'aplicació d'usuari ha de rebre a temps les dades del servidor per a poder fer la reproducció de manera correcta. Encara que aquesta aplicació té uns requisits de retard importants, no són tan forts com per a la reproducció en directe o les aplicacions de temps real, que veurem a continuació.

9.2. Reproducció en directe d'àudio i vídeo

Aquest tipus d'aplicacions funcionen com l'emissió tradicional de ràdio i televisió (un emissor transmet a molts receptors), només que es fa per Internet. Amb aquesta mena de reproducció es poden rebre emissions de ràdio i televisió en directe des de qualsevol punt del món.

Com que amb la reproducció en temps real d'àudio i vídeo la informació no està emmagatzemada, l'usuari no pot tirar endavant en la reproducció del contingut rebut. El que sí que permeten algunes aplicacions és parar la reproducció i fins i tot anar cap enrere (encara que això últim només és possible si s'ha emmagatzemat el contingut en el disc de l'usuari mentre s'anava reproduint).

Encara que per a enviar aquest tipus de flux de dades es podrien fer servir connexions multidestinació o *multicast* (un emissor transmet a diversos receptors en una única connexió), la veritat és que es fan múltiples connexions unidestinació o *unicast* (una connexió per cada receptor).

Es requereix una reproducció continuada, però els requisits d'interactivitat no són tan crítics com en les aplicacions interactives en temps real, i es poden acceptar retards en l'inici de la reproducció de fins a 10 segons.

9.2.1. Àudio i vídeo en temps real interactiu

Aquest tipus d'aplicacions permeten que els usuaris interactuïn en temps real. Actualment, hi ha bastants aplicacions d'aquest tipus que permeten que els usuaris facin àudio o videoconferències per mitjà d'Internet. Windows Messenger, Google Talk o Skype són alguns exemples d'aplicacions que permeten establir connexions d'àudio o vídeo. Algunes d'aquestes aplicacions també permeten fer trucades telefòniques a telèfons fixos o mòbils (fent un petit pagament), com és el cas de Skype.

Adreces web recomanades

Aplicacions d'àudio i vídeo interactiu:

Windows Messenger: http://www.microsoft.com/windows/messenger/features.asp.

Google Talk: http://www.google.com/talk/ intl/es/

Skype: http://www.skype.com/intl/es/

El retard permès en aquestes aplicacions no ha de ser més gran que uns centenars de mil·lisegons. Així doncs, retards de més de 400 mil·lisegons farien que una comunicació de veu fos pràcticament impossible.

9.3. Multimèdia a Internet

IP és un protocol de tipus tant-com-puc (*best-effort* en anglès), és a dir, que prova de lliurar els paquets tan aviat com pot, però sense donar cap garantia. Els protocols de transport TCP/UDP funcionen sobre IP i, per tant, tampoc no poden donar garanties de quan arribarà un determinat paquet a la seva destinació. Per això, l'enviament de dades multimèdia per Internet és una tasca difícil que ha tingut fins ara un èxit limitat.

En les aplicacions de reproducció en temps real d'àudio i vídeo emmagatzemat, retards entre 5 i 10 segons poden ser acceptables. Així doncs, si no es produeix un pic de trànsit o els paquets no han de passar per enllaços congestionats, podrem gaudir de la reproducció de manera satisfactòria.

D'altra banda, l'ús de les aplicacions de reproducció interactiva en temps real d'àudio i vídeo fins ara no ha estat tan satisfactori a causa de les restriccions en el retard dels paquets i en l'anomenat *paquet jitter*. El *paquet jitter* és la variabilitat que poden tenir els retards de paquets enviats entre un origen i una destinació dintre del mateix flux de dades. Com que l'àudio i el vídeo s'ha de reproduir amb la mateixa temporització amb la qual es van gravar, el *jitter* s'ha d'eliminar abans de fer la reproducció de les dades. El reproductor ha d'emmagatzemar els paquets durant un curt període de temps per a poder eliminar el *jitter* abans de reproduir el flux de dades.

Per a eliminar el *jitter*, habitualment es combinen tres mecanismes:

1) Afegir un número de seqüència a cada paquet de dades enviat. L'emissor incrementa aquest número en cada paquet enviat.

2) Afegir una marca de temps a cada paquet. L'emissor posa a cada paquet el moment en què es va generar.

3) Endarrerir la reproducció del paquet en el receptor. El retard en la reproducció ha de ser prou gran per a garantir que els paquets s'han rebut abans del moment de la reproducció. Aquest retard pot ser fix o pot ser adaptatiu al llarg de la sessió. Els paquets que no arriben abans del moment de la reproducció es consideren perduts i es descarten.

Amb aquests tres mecanismes, es poden utilitzar dues tècniques de reproducció dependents del retard: **retard fix** o **retard adaptatiu** en el moment de la reproducció de les dades. El retard fix consisteix a reproduir el paquet de dades exactament x mil·lisegons després de la creació del paquet de dades. Així doncs, si la marca de temps del paquet era t, la reproducció s'ha de fer en l'instant $t + x$, sempre que el paquet hagi arribat abans d'aquest moment. El valor de x depèn de l'aplicació, però per a telefonia per Internet es poden suportar fins a 400 mil·lisegons. Si és menys, hi ha el risc que no arribin. El retard adaptatiu consisteix a adaptar-se a les condicions de pèrdua de paquets i retards que tinguem durant la comunicació. La manera de fer això és estimar el retard de la xarxa i la variació del retard i anar ajustant el retard en la reproducció d'acord amb aquestes dades.

La reproducció en temps real interactiva funciona bé si es disposa d'una bona amplada de banda, en què el retard i el *jitter* són petits.

Les aplicacions multimèdia funcionarien millor a la xarxa Internet si hi hagués la possibilitat de reservar una part de l'amplada de banda per a l'enviament d'aquest tipus d'informació. Tanmateix, això no sembla que hagi de passar, almenys de moment. Per això, és necessari fer servir algun altre tipus de tècniques, com per exemple, utilitzar UDP, en comptes de TCP, per a l'enviament de les dades, introduir retards en l'enviament de les dades o, si es tracta de reproducció de continguts emmagatzemats, enviar dades per endavant (fent servir tècniques d'emmagatzematge a la memòria intermèdia) quan la connexió ho permet. Fins i tot es pot enviar informació redundant per a mitigar els efectes de la pèrdua de paquets.

Com que per aquest tipus de dades no és gaire adequat reenviar els paquets, el que es fa és utilitzar altres tècniques de recuperació d'errors, com són *forward error correction* (FEC) i l'*interleaving*.

El FEC consisteix a enviar dades redundants al flux de dades original. D'aquesta manera, es poden reconstruir versions aproximades dels paquets originals perduts. Una manera de fer FEC és enviar un paquet redundant cada n paquets. Això fa que si es perd un paquet d'aquests n, es pugui recuperar. Si se'n perden més, aleshores això ja no és possible. De tota manera, si s'envien molts paquets redundants, aleshores hi ha més retard. L'altra tècnica per a fer FEC és enviar les dades redundants en un flux de dades amb més baixa qualitat. Si es perd un paquet del flux de dades original, aleshores s'agafa el paquet del flux de dades de baixa qualitat i es reprodueix en el moment que li toca. D'aquesta manera, encara que es barregin paquets d'alta i baixa qualitat, no es perd cap paquet i l'experiència de l'usuari és prou bona.

L'*interleaving* consisteix en el fet que l'emissor envia els paquets en un ordre diferent del de la reproducció, per a minimitzar les pèrdues en un grup de paquets. D'aquesta manera es redueixen les pèrdues, però no és una tècnica adequada per a àudio i vídeo interactiu, pel retard en la recepció de les dades, però sí que funcionaria bé per a àudio i vídeo emmagatzemats. L'avantatge d'aquesta tècnica és que no es necessita més amplada de banda, ja que el flux de dades enviat és el mateix.

9.3.1. Compressió d'àudio i vídeo

Per a enviar dades multimèdia (àudio i vídeo) per Internet és necessari digitalitzar i comprimir aquestes dades. La raó per la qual cal digitalitzar les dades és molt senzilla: les xarxes de computadors transmeten bits; així doncs, tota la informació que es transmet ha d'estar representada amb bits. La compressió és important perquè l'àudio i el vídeo sense comprimir ocupen molt espai, amb el consum d'amplada de banda que això implica. Eliminar la redundància en els senyals d'àudio i vídeo redueix en diverses ordres de magnitud l'amplada de banda que es necessita per a transmetre la informació.

L'amplada de banda necessària i la compressió

Una imatge de 1.024 píxels (en què cada píxel es presenta amb 24 bits, 8 per als colors vermell, blau i verd), necessita 3 MB sense compressió.

Si s'aplica una taxa de compressió 1:10, tindrem que aquesta mateixa imatge necessita 300 KB per a ser representada.

En cas que haguéssim d'enviar les dues imatges per un canal de 64 kbps, en el primer cas trigaria 7 minuts a enviar-se, mentre que en el segon el temps de transmissió es reduiria en un factor de 10.

Compressió d'àudio

La compressió de tipus **PCM (*pulse code modulation*)** es basa en la recollida de mostres d'àudio a una freqüència determinada. El valor de cada mostra és un nombre real arbitrari. El valor de cada mostra s'arrodoneix a un determinat valor finit (tenim un nombre de valors finits per a les mostres). Cada un d'aquests valors es representa amb un nombre finit de bits, que depèn del nombre de valors que poden prendre les mostres. Per exemple, si es tenen 256 valors possibles de mostres, utilitzaríem un byte per a representar-les.

Per al cas de PCM, es recullen 8.000 mostres per segon i cada mostra es representa amb 8 bits. Això ens dóna un senyal digital amb una taxa de 64.000 bits per segon. El senyal digital es pot descodificar i convertir de nou en un senyal analògic, però aquest senyal serà diferent del senyal analògic original com a conseqüència del mostreig. Si es recullen més mostres i es prenen més valors possibles per a aquestes mostres, el senyal analògic descodificat serà molt més semblant al senyal original.

Exemples en l'ús de PCM

Alguns exemples en l'ús de PCM són la codificació de veu amb 8.000 mostres/segon i 8 bits/mostra, la qual cosa dóna una taxa de 64 kbps. El CD d'àudio també fa servir PCM, amb 44.100 mostres/segon i 16 bits/mostra. Això dóna una taxa de 705,6 kbps per a canals mono i 1,411 Mbps per a estèreo.

Encara que la velocitat de les connexions a Internet ha millorat (recordem que un mòdem proporcionava una velocitat màxima de 56 kbps), la compressió d'àudio i vídeo és encara molt important. Així doncs, podem trobar alguns exemples de compressió de veu amb taxes de bits més baixes, com ara GSM (13 kbps) o G.729 (8 kbps).

Per a la compressió de so amb qualitat quasi de CD, la tècnica més popular és l'estàndard de compressió MPEG 1 Layer 3, més conegut com a MP3. Les taxes de compressió d'MP3 solen ser 96 kbps, 128 kbps i 160 kbps, amb poca degradació del so.

Compressió de vídeo

El vídeo és una successió d'imatges, transmeses a una taxa constant, com ara 24 o 30 imatges per segon. Una imatge sense comprimir és una successió de píxels, en què cada píxel es representa amb un nombre de bits que indiquen color i lluminositat. Hi ha dos tipus de redundància en els vídeos, que es poden aprofitar per a comprimir: redundància espacial i redundància temporal. La redundància espacial consisteix a tenir repeticions dins de la mateixa imatge i la temporal és redundància entre imatges consecutives.

Per a vídeo, els estàndards de compressió MPEG són també els més populars. Aquests inclouen MPEG 1 per a la compressió amb qualitat de CD de vídeo (1,5 Mbps), MPEG 2 per a qualitat de DVD (3-6 Mbps) i MPEG 4 per a compressió orientada a objectes. Les tècniques de compressió MPEG es basen en el fet d'explotar la redundància temporal entre imatges a més de la compres-

sió que té en compte la redundància espacial que ja proporciona JPEG. Altres estàndards de compressió de vídeo són H.261 o Quicktime (aquest últim és un format propietat d'Apple).

9.3.2. Formats d'àudio i vídeo

Hi ha molts formats diferents d'àudio i vídeo; presentarem aquí els més coneguts i utilitzats. Els formats de vídeo més coneguts per a treballar a la xarxa Internet són Quicktime Movie (Apple), Real Media (Real Networks), Windows Media (Microsoft), Audio / Video Interleaved (Microsoft), MPEG (MPG) i Flash Video (Adobe). A continuació fem un resum de les seves característiques principals:

Bibliografia recomanada

Més informació sobre els formats d'àudio i vídeo:

J. Niederst (2006). *Web Design in a Nutshell* (3a. ed.). Sebastopol: O'Reilly.

- **Quicktime Movie**: originalment estava pensat com a format de vídeo, però ara es pot fer servir per a contenir qualsevol tipus de mitjà (imatges, àudio, vídeo, Flash, etc.). És un format propietat d'Apple. És possible fer reproducció d'aquest format fent servir algun servidor de reproducció en temps real dedicat. Es pot simular també la reproducció en temps real sobre HTTP si es fa servir FastStart, que fa que el vídeo es comenci a reproduir al mateix temps que s'està baixant.

- **Real Media**: és un dels formats més utilitzats per a fer reproducció en temps real. És un format propietat de la companyia Real Networks i cal tenir un sistema Real Media per a poder fer-ne la reproducció. També permet fer *pseudostreaming* via HTTP.

- **Windows Media**: es tracta del format de vídeo creat per Microsoft. Hi ha dos formats, Windows Media Video (.wmv) i Advanced Streaming Format (.asf). Es necessita un servidor de tipus Windows Media Server per a crear aquest tipus de fitxers. Permet fer reproducció i transmissions de vídeo en directe.

- **Audio / Video Interleaved**: és un format de vídeo creat també per Microsoft per al seu Video For Windows (VFW), l'arquitectura multimèdia del Windows 95. Ha estat substituït pel format Windows Media. Amb aquest format no es pot fer reproducció en temps real, s'ha de baixar el contingut i després reproduir-lo.

- **MPEG**: format estàndard definit pel Moving Picture Experts Group (MPEG). Suporta tres tipus d'informació: àudio, vídeo i reproducció (que vol dir que suporta àudio i vídeo sincronitzats). Aquest format ofereix una alta compressió amb poques pèrdues.

- **Flash Video**: és un format de vídeo creat per Flash (.flv) que permet fer reproducció en temps real. Funciona amb l'aplicació Flash Player, la qual cosa fa que no es necessiti un reproductor dedicat per a poder veure els

vídeos. A més, permet les mateixes característiques d'interactivitat que ara ofereixen les animacions fetes amb Flash (fitxers .swf).

Els formats d'àudio més coneguts per a treballar a la xarxa Internet són WAV/AIFF, MP3, Quicktime Audio (Apple), MIDI, Real Media Audio (Real Networks), Windows Media Audio (Microsoft) i Flash (Adobe). A continuació fem un resum de les seves característiques principals:

- **WAV/AIFF**: aquests dos formats tenen característiques molt similars. El Waveform Audio Format (.wav) va ser originalment dissenyat per als sistemes operatius Windows, mentre que Audio Interchange File Format (.aiff) va ser dissenyat per a sistemes Apple. Ara ja no es fan servir tant, ja que hi ha altres formats que ofereixen més compressió amb una qualitat semblant, com ara l'MP3. No permeten fer reproducció en temps real, però són els formats que es fan servir de base per a altres formats (com ara RealAudio). Si es comprimeixen, perden molta qualitat de so.

- **MP3**: es tracta del format més popular a la xarxa Internet. Ofereix una molt bona qualitat de so, alhora que permet fer una alta compressió de dades. Segueix l'estàndard MPEG-1 nivell 3, i es basa a comprimir la informació partint de la percepció auditiva de les persones. Es pot fer reproducció en temps real d'aquest format i també es pot baixar amb HTTP o FTP.

- **Quicktime Audio**: encara que el format Quicktime està pensant per vídeo, també permet incloure àudio. Aquest format permet fer reproducció en temps real i *pseudostreaming* amb HTTP, a més de poder-se baixar.

- **MIDI**: *MIDI* vol dir *musical instrument digital interface* i es tracta d'un tipus diferent de format d'àudio dels vistos fins ara. Es va dissenyar inicialment com a estàndard perquè els instruments musicals electrònics es poguessin comunicar entre ells. No conté informació d'àudio, sinó ordres de com s'haurien de tocar les diferents notes, amb instruccions sobre longitud i volum. Els fitxers MIDI ocupen molt poc i són ideals per a connexions amb una amplada de banda baixa. No es pot fer reproducció en temps real d'aquest format.

- **Real Media Audio**: va ser un dels primers formats que permetia fer reproducció en temps real d'àudio a través de la xarxa. Necessita un servidor dedicat, el Real Server, que permet negociar l'amplada de banda i la transmissió RTSP. Es pot fer *pseudostreaming*, d'aquest format amb un servidor HTTP, si hi ha un trànsit limitat.

- **Windows Media Audio**: és el format d'àudio per a reproducció en temps real creat per Microsoft. Hi ha dues opcions: Windows Media Audio (.wma), que no permet reproducció en temps real, i Active Streaming File (.asf) per a reproducció en temps real. El còdec per a aquest format és de

propietat i es necessita un servidor dedicat per a fer reproducció en temps real.

- **Flash**: es tracta d'un format que permet afegir sorolls de fons a les pàgines HTML i, a més, dóna característiques d'interactivitat i animació. En aquest cas, no hi ha retards en la reproducció i, un cop baixat el fitxer Flash, la interactivitat i el so estan permanentment disponibles.

10. Reproducció en temps real d'àudio i vídeo emmagatzemats

Les aplicacions de reproducció en temps real d'àudio i vídeo s'han popularitzat en els últims anys per diversos motius. D'una banda, els usuaris cada cop tenen com més va més capacitat d'emmagatzematge i també xarxes amb més amplada de banda, que permeten rebre la informació multimèdia demanada en un espai de temps curt.

Tanmateix, per a rebre les dades de reproducció en temps real cal tenir una aplicació dedicada (el que es coneix com a *media player* o reproductor), que sigui capaç d'oferir les característiques següents:

- **Descompressió**: les dades d'àudio i vídeo estan gairebé sempre comprimides per a estalviar espai de disc i amplada de banda. El reproductor haurà de descomprimir l'àudio i el vídeo a mesura que es reben.

- **Eliminació del *jitter***: el *paquet jitter* és la variabilitat del retard que hi ha entre origen i destinació dintre d'un mateix flux de dades. Com que l'àudio i el vídeo s'han de reproduir amb la mateixa temporització amb la qual es van gravar, el *jitter* s'ha d'eliminar abans de fer la reproducció. El receptor emmagatzemarà els paquets per un curt període de temps per a poder eliminar el *jitter* abans de reproduir el flux de dades.

- **Correcció d'errors**: a causa dels errors imprevisibles a la xarxa, una part dels paquets es podrien perdre. Si es perden massa paquets, aleshores la qualitat del flux de dades rebut és inacceptable. Algunes tècniques per a recuperar les dades perdudes són:
  - reconstruir els paquets perduts mitjançant l'enviament de paquets redundants,
  - fer que el client demani explícitament els paquets perduts o
  - interpolar les dades perdudes a partir de les dades rebudes.

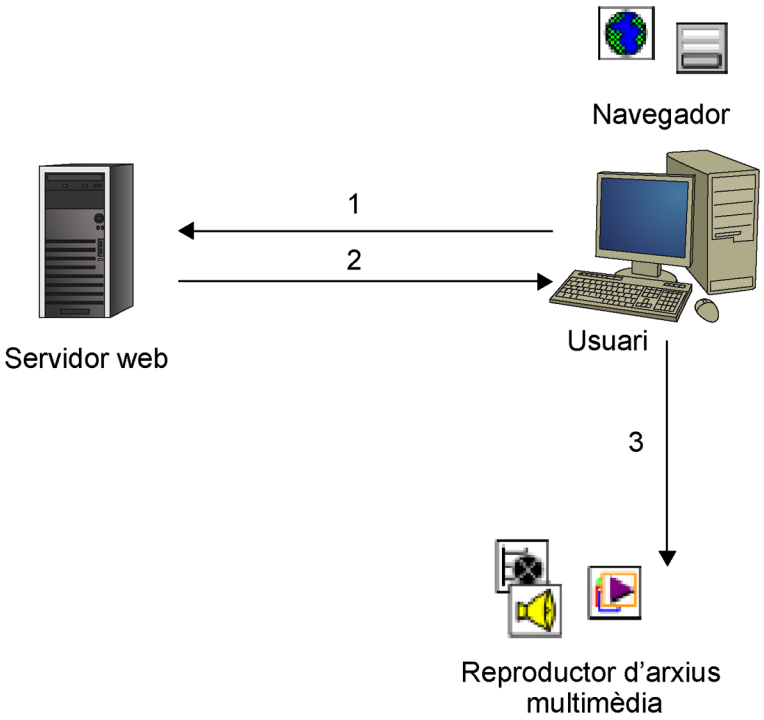
En els subapartats següents veurem dues maneres de fer reproducció en temps real d'àudio i vídeo emmagatzemats. La primera consisteix a utilitzar un servidor web tradicional, en què l'àudio i el vídeo estan emmagatzemats en el sistema de fitxers. L'altra manera de rebre un corrent de dades és l'ús d'un protocol de reproducció en temps real específic, com és el Real Time Streaming Protocol (RTSP).

10.1. Accés via servidor web

Des del punt de vista d'un servidor web, els fitxers d'àudio i vídeo són objectes accessibles des del servidor com qualsevol altre tipus de fitxer, com ara una pàgina HTML o una imatge.

Així doncs, l'usuari faria servir el seu navegador per a accedir al fitxer d'àudio o vídeo. Aquest accés establiria una connexió TCP amb el servidor i, un cop establerta aquesta connexió, demanaria el recurs (fitxer d'àudio o vídeo) amb un missatge de petició HTTP amb l'ordre corresponent HTTP (podria ser una ordre GET). El servidor genera un missatge de resposta, en què hi haurà encapsulat el fitxer demanat. La recepció d'aquest fitxer farà que s'obri el programa reproductor i, quan aquest tingui prou dades, començarà a reproduir el fitxer rebut.

La figura següent mostra de manera molt simple com es produiria aquesta comunicació:



1) L'usuari demana amb el seu navegador un arxiu multimèdia (per exemple, pitjant un enllaç web). Aquesta petició genera un missatge de petició HTTP.

2) El servidor busca el recurs en el seu sistema de fitxers i l'envia al navegador del client. Aquesta resposta viatja en un missatge de resposta HTTP.

3) El navegador desa les dades en el disc i aquestes dades són reproduïdes per un reproductor d'arxius multimèdia, que haurà de descomprimir les dades, eliminar els retards i solucionar els errors de dades que s'hagin pogut produir.

En aquest cas, ens trobem el problema que fins que el navegador no rep l'objecte multimèdia sencer, no el pot passar al reproductor. Si l'arxiu és molt gran, el retard serà inacceptable. Per això, el que es fa normalment és que el servidor web es comuniqui directament amb el reproductor, i li envia les dades directament. D'aquesta manera, el reproductor rep les dades i, quan en té prou per a reproduir el contingut, ho comença a fer, a la vegada que continua rebent dades.

La manera de fer que el servidor web es connecti directament amb el reproductor és substituint el recurs multimèdia per un fitxer de metadades, que conté la informació d'on es troba el recurs multimèdia del qual s'ha de fer reproducció en temps real i fins i tot del tipus de codificació. D'aquesta manera, el navegador rep un fitxer de metadades, que passa al reproductor, i aquest s'encarrega de connectar amb l'adreça continguda en el fitxer de metadades. Així, l'inici de la reproducció pot començar molt abans d'haver rebut tot el fitxer multimèdia.

10.2. Real Time Streaming Protocol (RTSP)

El protocol RTSP[25] estableix i controla tant un com diversos fluxos de dades sincronitzats de dades multimèdia, com poden ser l'àudio i el vídeo. No fa l'enviament de les dades, encara que l'enviament d'informació de control al mig de la transmissió de dades és possible. El que fa l'RTSP és de control remot a través de la xarxa per als servidors de dades multimèdia.

[25]L'RFC que defineix RTSP és la 2326.

En RTSP no hi ha connexions, el que tenim són sessions mantingudes pel servidor. Cada sessió té el seu identificador. Una sessió RTSP no està lligada a una connexió de nivell de transport. Això vol dir que durant una sessió RTSP es poden obrir i tancar tantes sessions de transport com calgui. També es pot utilitzar UDP com a protocol de transport, un protocol sense connexió.

Els fluxos de dades controlats per RTSP poden fer servir RTP, però el funcionament d'RTSP no depèn del mecanisme de control utilitzat per a enviar les dades multimèdia. El protocol RTSP és molt semblant al protocol HTTP/1.1 (la versió 1.1 del protocol HTTP), la qual cosa fa que es puguin afegir mecanismes d'extensió per a HTTP que també es poden aplicar a RTSP. Tanmateix, RTSP és diferent d'HTTP en un bon nombre d'aspectes importants:

Vegeu també

Sobre l'RTP, vegeu més endavant el subapartat 11.1 d'aquest mòdul didàctic.

- RTSP introdueix mètodes nous i té un identificador de protocol propi (rtsp://).

- Un servidor RTSP ha de mantenir l'estat en la majoria de casos, just al contrari del que fa HTTP.

- Tant el client com el servidor RTSP poden fer peticions.

- Les dades les transporta un protocol diferent d'RTSP.

- L'URI de petició d'RTSP sempre conté l'URI absoluta. En HTTP l'URI de petició només porta la ruta absoluta al fitxer i el nom del servidor va en una capçalera a part.

El protocol suporta les operacions següents:

a) Recuperació de dades del servidor de continguts multimèdia: el client pot demanar una descripció d'una presentació via HTTP o qualsevol altre mètode. Si la presentació s'està emetent en multidestinació (o *multicast*), la seva descripció conté la seva adreça multidestinació i els ports que es faran servir. Si la presentació s'envia en unidestinació (o *unicast*) només a aquest client, el client dóna la destinació per motius de seguretat.

b) Invitació d'un servidor de dades a una conferència: un servidor de dades pot ser convidat a unir-se a una conferència existent, tant per a reproduir dades a la presentació com per a gravar tot o una part de les dades transmeses en la presentació. Aquest mode és molt útil per a aplicacions d'ensenyament distribuït. Molts participants poden prendre-hi part, pitjant els botons de control remot.

c) Afegir contingut a una presentació existent: és molt útil especialment per al cas de presentacions en directe, ja que el servidor pot dir al client que hi ha noves dades disponibles.

Un fitxer de descripció d'una presentació seria un exemple del que en la distribució de continguts a través d'un servidor web hem anomenat un *fitxer de metadades*. Aquest fitxer conté una descripció dels fluxos de dades que formen part de la presentació, incloent-hi el llenguatge de codificació i altres paràmetres que ajuden l'usuari a escollir la millor combinació de dades. Cada contingut (per exemple, l'àudio i el vídeo es podrien transmetre en fluxos de dades separats) té la seva URL pròpia, que apunta a un determinat contingut dins un determinat servidor. El fitxer també descriu els mètodes de transport que se suporten. A més de les dades del contingut, la descripció de l'adreça de destinació de la xarxa i el port s'han de determinar. Hi ha diversos modes d'operació:

1) Unidestinació: les dades es transmeten a l'origen de la petició RTSP, al número de port seleccionat pel client.

2) Multidestinació (el servidor escull adreça): el servidor escull l'adreça multidestinació i el port. Aquest cas és el típic per a transmissions de dades multimèdia en directe.

3) Multidestinació (el client escull adreça): si el servidor participa en una conferència multidestinació existent, l'adreça multidestinació, el port i la clau de xifratge són donades per la descripció de la conferència.

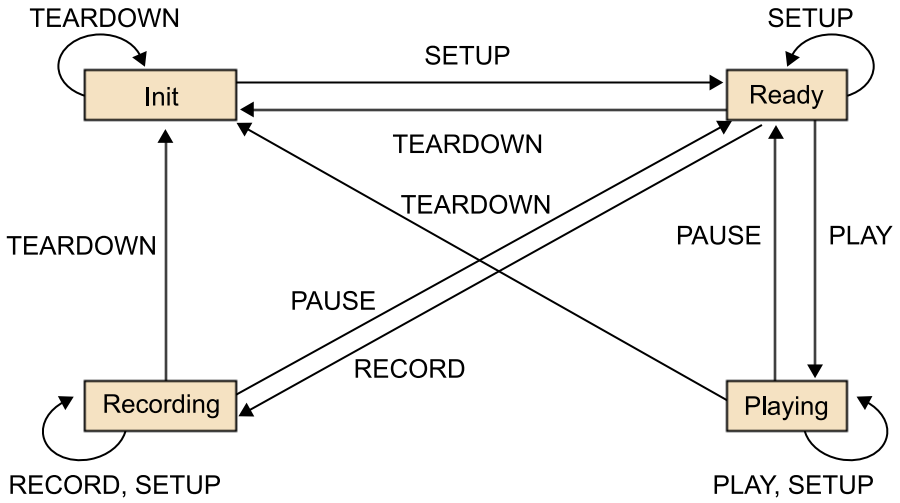
10.2.1. Els estats d'RTSP

RTSP controla fluxos de dades que es poden enviar per un protocol separat, independent del protocol de control. També pot passar que RTSP funcioni sobre connexions TCP, mentre que les dades s'envien per UDP. Així doncs, la transmissió de dades continua ocorrent encara que no s'enviïn dades de control RTSP. Una altra possibilitat és que un flux de dades estigui controlat per peticions RTSP i que aquestes peticions viatgin en diferents connexions TCP. Per totes aquestes raons, és necessari mantenir l'estat de la sessió RTSP, per tal de correlacionar les peticions que es refereixen al mateix flux de dades.

10.2.2. Diagrama d'estats del client

El client pot estar en quatre estats possibles: *Init*, *Ready*, *Playing* i *Recording*. L'estat *Init* indica que el client ha enviat una ordre SETUP i espera resposta. L'estat *Ready* indica que, o s'ha rebut una resposta afirmativa al SETUP o s'ha rebut una confirmació a l'enviament de l'ordre PAUSE estant en *Playing* o *Recording*. Els estats *Playing* i *Recording* indiquen que s'ha rebut una confirmació afirmativa a les ordres PLAY i RECORD, respectivament.

La figura següent mostra els estats pels quals ha passat el client, com a resposta a les ordres que envia. Les transicions es produeixen quan el client rep una confirmació positiva a l'ordre enviada (l'ordre s'indica a la fletxa corresponent).

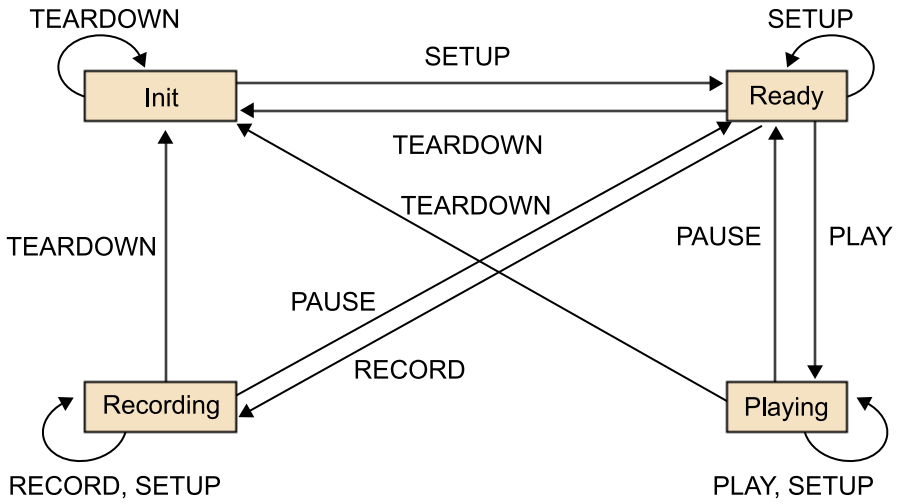


10.2.3. Diagrama d'estats del servidor

El servidor pot estar en quatre estats possibles: *Init*, *Ready*, *Playing* i *Recording*. L'estat *Init* indica que el servidor està a l'espera de rebre una ordre SETUP correcta. És l'estat inicial. L'estat *Ready* indica que l'últim SETUP rebut va ser correcte i es va enviar la confirmació corresponent o que, en els estats *Playing*

i *Recording*, l'ordre PAUSE es va rebre i es va confirmar. L'estat *Playing* indica que s'ha rebut l'ordre PLAY i es va confirmar i que s'estan enviant les dades al client. L'estat *Recording* indica que el servidor està gravant les dades.

La figura següent mostra els estats pels quals ha passat el servidor, com a resposta a les ordres que rep (i de les quals envia confirmació al client). Les transicions es produeixen quan el servidor rep una ordre i ha enviat una confirmació positiva (l'ordre s'indica a la fletxa corresponent).



10.2.4. Mètodes RTSP

Els mètodes RTSP següents són els més rellevants a l'hora de canviar d'estat i de fer la reserva de recursos per al flux de dades al cantó servidor:

- SETUP: el servidor reserva recursos per a un flux de dades i inicia una sessió RTSP.

- PLAY i RECORD: s'inicia la transmissió de dades d'un flux de dades inicialitzat amb SETUP.

- PAUSE: para temporalment la transmissió de dades sense alliberar els recursos del servidor.

- TEARDOWN: allibera els recursos associats al flux de dades. La sessió RTSP s'esborra del servidor.

Els mètodes RTSP que modifiquen l'estat de la sessió fan servir el camp "Sessió de la capçalera RTSP", que veurem en el subapartat següent. Per a consultar la llista sencera de mètodes, vegeu l'RFC d'RTSP.

10.2.5. El protocol RTSP

En aquest subapartat veurem el protocol RTSP, incloent-hi el format del missatge de petició i resposta i els camps de capçalera. Moltes de les característiques del protocol s'hereten d'HTTP, però en farem aquí una descripció completa. La URL del protocol RTSP té la forma següent:

Esquemes RTSP

En RTSP les URL poden començar amb *rtsp* o *rtspu*. L'ús d'RTSP indica que funciona sobre un nivell de transport fiable (TCP), mentre que RTSPU indica que el protocol funciona sobre un nivell de transport no fiable (UDP).

```
rtsp://host:port/cami/al/recurs
```

Exemple:

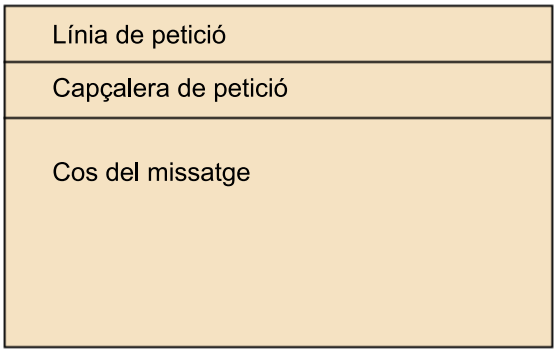
```
rtsp://servidor.multimedia.com:554/programa_tv.html
```

Les URL haurien d'evitar fer servir adreces IP en comptes del nom de l'amfitrió i es poden referir a un únic flux de dades o a una sèrie de fluxos de dades agregats.

RTSP és un protocol basat en missatges de text. Això en facilita la implementació i ampliació (només cal definir un nou mètode o una nova capçalera). A continuació veurem el format dels missatges de petició i resposta d'RTSP.

10.2.6. El missatge de petició RTSP

La figura següent mostra el format del missatge de petició. Aquest missatge consta de tres parts ben diferenciades: la línia de petició, la capçalera de petició i el cos del missatge.



La línia de petició té la forma següent:

```
OrdreRTSP [Espai] URI-petició [Espai] Versió-RTSP [Salt línia]
```

L'ordre RTSP indica quina ordre del protocol estem fent servir. L'URI-petició identifica el recurs al qual volem accedir, i la versió RTSP indica la versió del protocol que estem utilitzant. [Espai] és un espai en blanc i [Salt línia] és representat pels caràcters de salt de línia (ASCII 10) i retorn de carro (ASCII 13). La resta d'espais en blanc s'ha posat per claredat.

Vegeu també

Els valors de les ordres RTSP, i també el format de la versió RTSP, està explicat amb detall en l'annex 1.

```
Nom-camp: [Espai] valor-camp [Salt línia]
```

La capçalera de petició té tres parts: la capçalera general, la capçalera pròpiament de petició i la capçalera d'entitat. El format de tots els camps de les capçaleres és el següent:

- La capçalera general representa informació general que és independent de si el missatge és de petició o de resposta.
- La capçalera de petició conté camps que només tenen sentit per a un missatge de petició.
- La capçalera d'entitat conté informació sobre el cos del missatge (que també es coneix com a entitat).

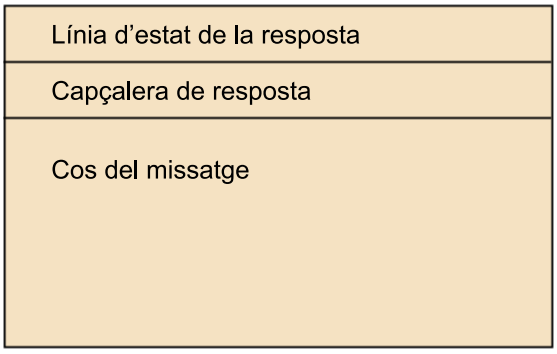
Vegeu també

Els valors que poden prendre les capçaleres d'entitat estan explicats amb detall en l'annex 1.

El cos del missatge porta la informació, si n'hi ha, associada a la petició.

10.2.7. El missatge de resposta RTSP

La figura següent mostra el format del missatge de resposta. Aquest missatge consta de tres parts ben diferenciades: la línia d'estat de la resposta, la capçalera i el cos del missatge.



La línia d'estat de la resposta té la forma següent:

```
Versió-RTSP [Espai] Codi-Estat [Espai] Descripció-Estat [Salt línia]
```

La versió RTSP defineix la versió del protocol que s'està fent servir, el codi d'estat defineix l'èxit o error en la petició i la descripció de l'estat, que és una descripció textual del codi d'estat. [Espai] és un espai en blanc i [Salt línia] és representat pels caràcters de salt de línia (ASCII 10) i retorn de carro (ASCII 13). La resta d'espais en blanc s'ha posat per claredat.

Vegeu també

Els valors dels codis i les descripcions d'estat estan explicats amb detall en l'annex 1.

La capçalera de resposta té tres parts: la capçalera general, la capçalera pròpiament de resposta i la capçalera d'entitat. El format de tots els camps de les capçaleres és el següent:

```
Nom-camp: [Espai] valor-camp [Salt línia]
```

La capçalera general representa informació general que és independent de si el missatge és de petició o de resposta. La capçalera de resposta conté camps que només tenen sentit per a un missatge de resposta. La capçalera d'entitat conté informació sobre el cos del missatge (que també es coneix com a *entitat)*.

Vegeu també

Els valors que poden prendre aquestes tres capçaleres estan explicats amb detall en l'annex 1.

El cos del missatge porta la informació, si n'hi ha, associada a la resposta.

10.2.8. Exemple d'ús del protocol RTSP

En aquest subapartat veurem un exemple d'ús del protocol RTSP, amb les peticions i respostes que ens trobaríem en una comunicació real.

En aquest exemple, el client estableix una connexió TCP al port 554 del servidor i envia l'ordre OPTIONS, indicant la versió del protocol RTSP que farà servir. El servidor confirma aquesta ordre amb un missatge 200 OK (que indica que ha anat bé). Aquest codi de resposta és igual que l'equivalent a HTTP.

Petició del client:

```
OPTIONS rtsp://servidor.multimedia.com:554 RTSP/1.0
Cseq: 1
```

Resposta del servidor:

```
RTSP/1.0 200 OK Cseq: 1
```

En un segon pas, el client envia una ordre de tipus DESCRIBE, que indica al servidor el contingut concret que volem. El servidor torna a respondre amb un 200 OK, incloent-hi una descripció completa del contingut, en algun format estandarditzat –per exemple, Session Description Protocol (SDP) o Multimedia and Hypermedia Experts Group (MHEG).

SDP i MHEG

El Session Description Protocol (SDP) està definit en l'RFC 4566.

Multimedia and Hypermedia Experts Group (MHEG): http://www.mheg.org.

Petició del client:

```
DESCRIBE rtsp://servidor.multimedia.com:554/videos/
video.html RTSP/1.0
Cseq: 2
```

Petició del client:

```
SETUP rtsp:// servidor.multimedia.com:554/videos/
video.html RTSP/1.0
Cseq: 3
Transport: rtp/udp;unicast;client_port=5067-5068
```

Resposta del servidor:

```
RTSP/1.0 200 OK Cseq: 3
Session: 12345678
Transport: rtp/udp;client_port=5067-5068;server_port=6023-6024
```

Com es pot veure, a més de les línies de petició i resposta, els missatges porten capçaleres que indiquen el número de seqüència del missatge (*Cseq*) o les característiques de l'entitat enviada (*Content-type* o *Content-length*).

En el pas següent de la negociació, el client envia una ordre SETUP en què diu al servidor els mecanismes de transport que vol fer servir i l'ordre de preferència. El client indica que vol utilitzar UDP com a protocol de transport i els ports 5067 i 5068 per a la transferència de dades. El servidor confirma les dades de transport i ports del client i afegeix l'identificador de sessió i els seus ports de transferència.

Després d'establir la connexió, el client està llest per a començar a rebre el corrent de dades i envia l'ordre PLAY. Aquesta ordre només conté l'URL del contingut i l'identificador de la sessió enviat prèviament pel servidor. El servidor confirma l'ordre i es comença a enviar el corrent de dades.

Si el client decideix parar el corrent de dades, ho pot fer amb l'ordre TEARDOWN. El servidor envia la confirmació que ha rebut l'ordre i es para l'enviament RTSP.

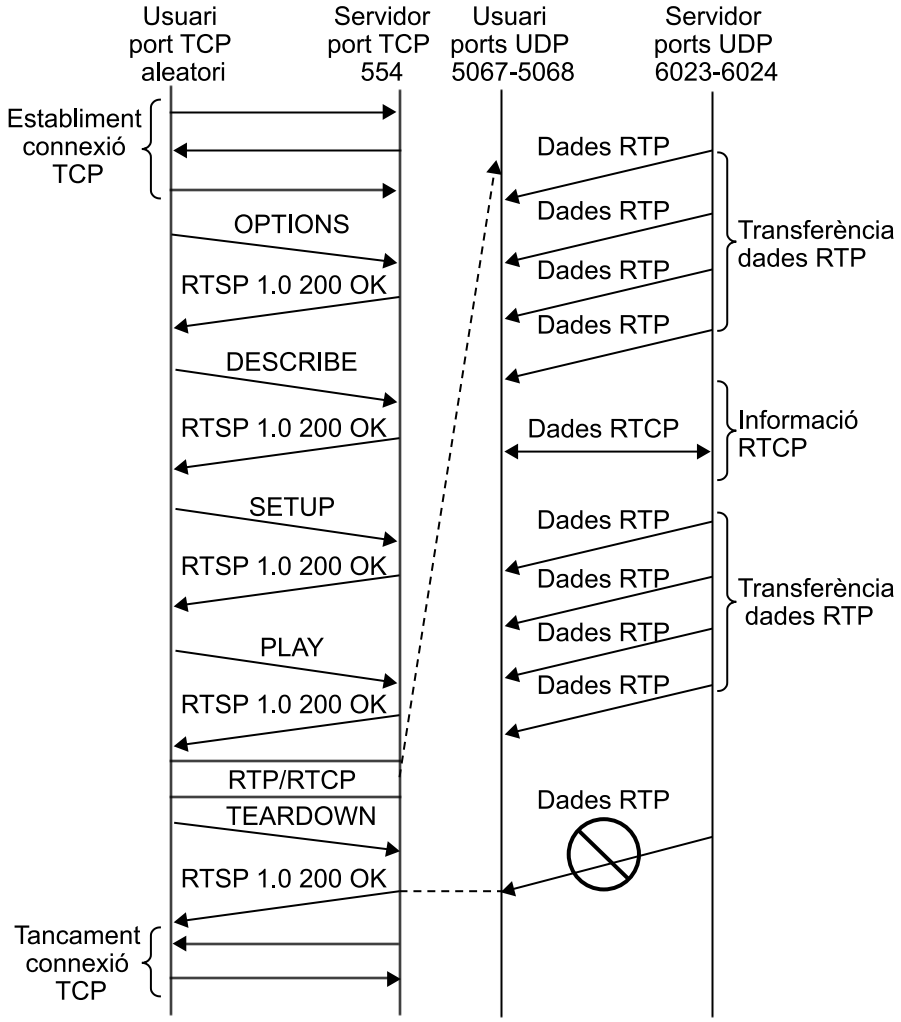
Petició del client:

```
PLAY rtsp:// servidor.multimedia.com:554/videos/
video.mpg RTSP/1.0
Cseq: 4
Session: 12345678
```

Resposta del servidor:

```
RTSP/1.0 200 OK Cseq: 4
```

La figura següent resumeix els intercanvis anteriors:


Usuari
port TCP
aleatori
Servidor
port TCP
554
Usuari
ports UDP
5067-5068
Servidor
ports UDP
6023-6024
Establiment
connexió
TCP
OPTIONS
RTSP 1.0 200 OK
DESCRIBE
RTSP 1.0 200 OK
SETUP
RTSP 1.0 200 OK
PLAY
RTSP 1.0 200 OK
RTP/RTCP
TEARDOWN
RTSP 1.0 200 OK
Tancament
connexió
TCP
Dades RTP
Dades RTP
Dades RTP
Dades RTP
Transferència
dades RTP
Dades RTCP
Informació
RTCP
Dades RTP
Dades RTP
Dades RTP
Dades RTP
Transferència
dades RTP
Dades RTP

11. Protocols per a aplicacions interactives en temps real

11.1. Real Time Transport Protocol (RTP)

El Real Time Transport Protocol (RTP)[26] és un protocol definit per Internet Engineering Task Force (IETF). Aquest protocol es pot fer servir per a enviar qualsevol tipus de format: PCM, GSM o MP3 per a àudio i MPEG o H.263 per a vídeo. Aquest protocol és complementari a altres protocols de temps real, com ara SIP o H.323.

[26]L'RFC que defineix RTP i RTCP és la 3550.

Vegeu també

Els protocols SIP o H.323 es veuran més endavant dins d'aquest mateix apartat.

11.1.1. Descripció d'RTP

L'RTP funciona sobre el protocol de transport UDP. L'emissor encapsula un tros de dades (*chunk*, en anglès) dintre del paquet RTP, que també s'encapsula dintre d'un datagrama UDP, que viatja en un paquet IP. El receptor extreu les dades RTP del datagrama UDP i passa les dades al reproductor perquè aquest descodifiqui el contingut i el reprodueixi.

Vegeu també

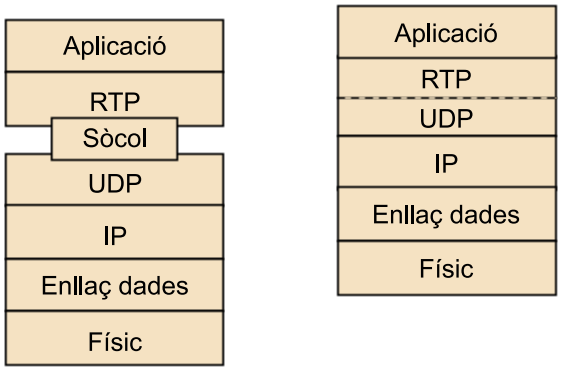
En aquest subapartat 11.1 parlarem d'RTP i en l'11.2 parlarem del protocol de control d'RTP, l'RTCP.

Si una aplicació inclou l'RTP serà més fàcil que pugui interoperar amb altres aplicacions multimèdia. Per exemple, si dos fabricants ofereixen un programari de telefonia sobre IP i incorporen RTP en el seu producte, la interconnexió entre els seus productes serà més factible.

L'RTP no ofereix cap mecanisme que permeti assegurar que les dades arriben a la seva destinació a temps o amb la qualitat de servei adequada. Tampoc no garanteix que els paquets arribin en ordre, ja que el protocol RTP només es reconeix en els extrems, els encaminadors prenen els paquets IP que contenen RTP com si fos qualsevol altre paquet IP i no els diferencien de la resta.

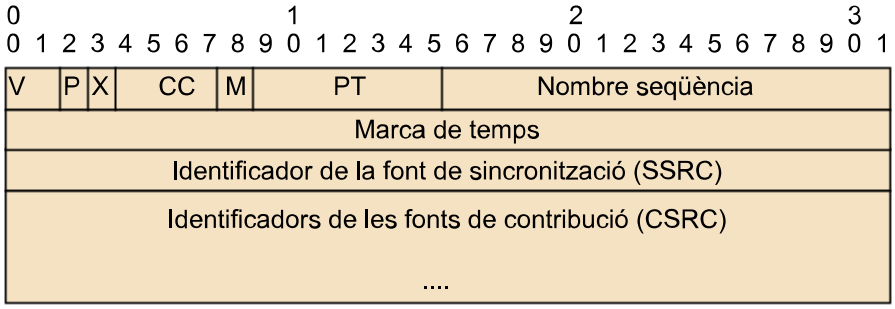
L'RTP permet associar a qualsevol dispositiu (una càmera, un micròfon, etc.) el seu flux de dades propi de paquets. Així doncs, si dos usuaris volen fer una videoconferència, es poden obrir dos fluxos de dades d'àudio i vídeo, un en cada sentit per a cada tipus de dades. La realitat és el que els fluxos de dades d'àudio i vídeo estan entrellaçats, s'envien com un únic flux de dades i s'estableix una única connexió per a enviar els dos fluxos de dades (àudio i vídeo).

L'RTP forma part del nivell d'aplicació (estaria per sobre d'UDP, que és el nivell de transport), però també es pot veure com un subnivell dins el nivell de transport. A la figura següent es pot veure aquesta distinció de manera gràfica:



11.1.2. La capçalera d'RTP

El format de la capçalera RTP es pot veure en la figura següent.



Els primers 12 octets estan presents en tots els paquets RTP, mentre que la llista d'identificadors CSRC (*contributing source*) només estan presents quan hi ha un mesclador.

Mesclador RTP

Un mesclador RTP s'encarrega de canviar el format del flux de dades per a adequar-lo a l'amplada de banda del receptor.

Els camps tenen el significat següent:

- Versió (V): 2 bits. Aquest camp identifica la versió RTP. La versió actual és la 2.

- Farciment (P): 1 bit. Si aquest bit està activat, el paquet conté un o més octets de farciment addicional a les dades. L'últim octet de farciment indica quants octets s'han de descartar, incloent-hi ell mateix. Aquesta tècnica pot ser necessària per a alguns algoritmes de xifratge que necessiten una mida de bloc fixa.

- Extensió (X): 1 bit. Si està activat, la capçalera fixa només pot portar una extensió.

- Nombre de CSRC (CC): 4 bits. Aquest camp conté el nombre d'identificadors CSRC que hi ha a la capçalera fixa.

- Marcador (M): 1 bit. La interpretació d'aquest camp està definida per un perfil, que indica per què s'ha d'utilitzar en un determinat context.

- Tipus de contingut (PT): 7 bits. Aquest camp identifica el format de les dades que s'estan enviant amb RTP.

- Nombre de seqüència: 16 bits. Aquest nombre s'incrementa en un cada vegada que s'envia un paquet de dades RTP i es pot utilitzar per a detectar pèrdues de dades i restaurar la seqüència del paquet. El valor inicial ha de ser aleatori.

- Marca de temps: 32 bits. Indica el moment de temps del primer octet enviat en aquest paquet de dades RTP.

- SSRC: 32 bits. Aquest camp identifica la font de sincronització. Ha de ser aleatori perquè dues fonts de sincronització d'una sessió RTP no tinguin el mateix identificador.

- Llista CSRC: de 0 a 15 elements, 32 bits per element. Identifica les fonts que contribueixen a les dades contingudes en aquest paquet RTP.

11.2. Real Time Control Protocol (RTCP)

El protocol de control d'RTP (RTCP) es basa en la transmissió periòdica de paquets de control a tots els participants en una sessió, utilitzant el mateix mecanisme de distribució que els paquets de dades enviats amb RTP. El protocol que hi hagi per sota ha d'oferir la possibilitat de multiplexar paquets de dades i de control, per exemple, per mitjà de números de port UDP diferents. RTCP fa quatre funcions bàsiques:

1) Dóna informació sobre la qualitat de les dades distribuïdes. Això forma part del paper d'RTP com a protocol de transport i està molt relacionat amb les funcionalitats de control de congestió ofertes per altres protocols de transport.

2) Manté un identificador persistent a escala de transport d'una font RTP, que s'anomena *nom canònic* o *CNAME*, ja que, en el transcurs de la comunicació, l'identificador SSRC podria canviar si es detecta un conflicte o es reinicia un programa. Amb el CNAME, si l'identificador SSRC canvia, es poden recuperar els participants de la sessió.

3) Les dues funcionalitats anteriors necessiten que tots els participants enviïn paquets RTCP, encara que s'ha de controlar la taxa d'enviament per si hi ha un nombre elevat de participants. Si cada participant envia els paquets de control

a tots els altres, cada un pot observar independentment el nombre de participants. Aquest nombre serveix per a calcular la taxa a la qual s'envien els paquets.

4) L'última funció és opcional, i consisteix a comunicar un mínim d'informació de control de la sessió, com ara, que es mostri la identificació d'un participant a la interfície d'usuari.

11.2.1. Els paquets RTCP

Els paquets RTCP poden transportar diferents tipus d'informació de control:

- SR: informe de l'emissor (*sender report*, en anglès), per a transmetre i rebre estadístiques dels participants que són emissors actius.

- RR: informe del receptor (*receiver report*, en anglès), per a rebre estadístiques dels participants que no són emissors actius. També es pot combinar amb l'SR per als emissors actius que han d'informar sobre més de 31 fonts de dades (el màxim que es pot indicar en un paquet de tipus SR).

- SDES: elements de descripció de la font, incloent-hi el CNAME.

- BYE: indica que s'ha deixat de participar.

- APP: funcionalitats específiques de l'aplicació.

Cada paquet RTCP comença amb una part fixa similar a la que tenen els paquets de dades RTP, seguida d'una sèrie d'elements estructurats que poden tenir una estructura variable d'acord amb el tipus de paquet, però que han de tenir una longitud múltiple de 32 bits. S'inclouen camps per a indicar l'alineació i un camp de longitud per a fer que els paquets RTCP siguin apilables. Els paquets RTCP es poden enviar de manera consecutiva, sense haver-hi de posar cap separador, i formar un paquet RTCP compost, que es pot enviar en un únic paquet del nivell inferior (per exemple, UDP). Cada paquet RTCP dintre del paquet compost s'ha de processar de manera independent de la resta de paquets. Tanmateix, per a poder portar a terme les funcions del protocol, s'imposen les restriccions següents:

a) La recepció d'estadístiques (en SR o RR) s'ha d'enviar tan freqüentment com ho permetin les restriccions d'amplada de banda per a maximitzar la resolució de les estadístiques. És per això que cada paquet RTCP compost ha de portar obligatòriament un paquet de tipus informe.

b) Els nous receptors han de rebre el CNAME tan aviat com sigui possible per a identificar la font i poder associar el contingut multimèdia.

c) El nombre de tipus de paquets que poden aparèixer inicialment en el paquet compost s'ha de limitar per tal d'augmentar el nombre de bits constants en la primera paraula del paquet i la probabilitat de validar amb èxit els paquets RTCP contra paquets de dades RTP que tinguin l'adreça malament o que no estiguin relacionats.

Tots els paquets RTCP s'han d'enviar obligatòriament en paquets compostos d'almenys dos paquets individuals, amb el format següent:

- Prefix de xifratge: si el paquet RCTP s'ha de xifrar, aleshores s'ha de posar davant un prefix aleatori de 32 bits, que s'ha de recalcular per a cada paquet compost. Si cal farciment, s'ha de posar a l'últim paquet del paquet compost.

- SR o RR: el primer paquet del paquet compost ha de ser de tipus informe, per a facilitar la validació de la capçalera. Això s'ha de fer sempre, encara que no s'hagin enviat o rebut dades. En aquest cas s'ha d'enviar un paquet RR buit.

- RR addicionals: si el nombre de fonts de les quals s'ha d'informar és més gran que 31, aleshores s'han d'enviar tants paquets RR addicionals fins a completar les dades del primer paquet SR o RR enviat.

- SDES: un paquet de tipus SDES que contingui un CNAME s'ha d'incloure sempre en el paquet compost.

- BYE o APP: la resta de paquets poden aparèixer en qualsevol ordre i fins i tot repetits, excepte els de tipus BYE, que han d'anar al final per a un SSRC/ CSRC donat.

Un participant RTP ha d'enviar un únic paquet compost per cada interval d'enviament d'informes per a poder calcular correctament l'amplada de banda d'RTCP per participant.

Vegeu també

El format específic de cada paquet RTCP es defineix en l'annex 2 d'aquest mòdul didàctic.

11.2.2. Ús de l'amplada de banda en el protocol RTCP

Per al funcionament d'RTCP, es pot veure que, en una connexió amb un emissor i molts receptors (que es poden comunicar amb multidestinació), les dades enviades amb RTCP poden excedir en gran mesura les dades enviades amb RTP per l'emissor. És per això que RTCP modifica la taxa d'enviament dels paquets RTCP a mesura que augmenta el nombre de participants en la sessió.

RTCP intenta mantenir el seu trànsit en el cinc per cent de l'amplada de banda de la sessió. Vegem això amb un exemple.

Amplada de banda RTSP

Si un emissor transmet a una velocitat de 2 Mbps, l'RTCP intenta limitar el seu trànsit en el 5 per cent d'aquests 2 Mbps, que serien 100 kbps.

El protocol dóna el 75% d'aquesta taxa, 75 kbps, als receptors, i la resta, els 25 kbps restants, a l'emissor. Els 75 Kbps dels receptors es reparteixen entre ells de manera igualitària. Així, si hi ha *R* receptors, cada receptor pot enviar trànsit RTCP a una taxa de 75 kbps/*R* i l'emissor envia a una taxa de 25 kbps. Els participants (emissors o receptors) determinen el període de transmissió d'un paquet RTCP calculant dinàmicament la mida mitjana de paquet RTCP (en tota la sessió) i dividint-la entre la mida mitjana de paquet RTCP que pot enviar a la seva taxa assignada.

En resum, el període en què un emissor pot enviar un paquet RTCP és:

$$T = \frac{\text{nombre emissors}}{0{,}25 * 0{,}05 * \text{amplada banda sessió}} \bullet (\text{mida mitjana paquet RTCP})$$

En resum, el període en què un receptor pot enviar un paquet RTCP és:

$$T = \frac{\text{nombre receptors}}{0{,}75 * 0{,}05 * \text{amplada banda sessió}} \bullet (\text{mida mitjana paquet RTCP})$$

11.3. Session Initiation Protocol (SIP)

Hi ha moltes aplicacions a Internet que necessiten la creació i gestió de sessions, entenent *sessió* com un intercanvi de dades entre diversos participants. La implementació de les sessions es fa difícil pel comportament dels que hi participen: els usuaris canvien de localització, poden tenir diverses maneres d'identificar-se i intercanvien diferents tipus de dades, de vegades de manera simultània. Hi ha molts protocols que permeten treballar amb sessions multimèdia en temps real que permeten transmetre text, àudio o vídeo.

El Session Initiation Protocol (SIP)[27] treballa en cooperació amb aquests protocols, i permet que els diferents agents d'usuari (aplicacions que fan servir els usuaris per a comunicar-se) es puguin trobar i posar d'acord amb el tipus de sessió que volen compartir. Per a localitzar els participants d'una sessió i per a altres funcions, el SIP permet la creació d'una infraestructura d'amfitrions (anomenats *servidors intermediaris* o *proxy servers*), als quals les aplicacions d'usuari poden enviar peticions de registre, invitació a les sessions i altres peticions. El SIP és una eina que permet gestionar sessions independentment dels protocols de transport que hi hagi per sota i sense cap dependència del tipus de sessió establerta.

[27]L'RFC en què es defineix el SIP és la 3261.

Aplicacions que fan servir el SIP

Windows Live Messenger i Yahoo Messenger són dues aplicacions que fan servir SIP.

11.3.1. Funcionalitat del SIP

El SIP és un protocol de control de nivell d'aplicació que pot establir, modificar i finalitzar sessions multimèdia (conferències). Com a exemple de sessió tenim les trucades telefòniques per mitjà d'Internet. El SIP també permet convidar participants a sessions existents, com ara les conferències multidestinació.

També es poden afegir i treure continguts multimèdia d'una sessió existent. Amb SIP, els usuaris poden tenir un únic identificador extern, independentment de la seva localització de xarxa.

SIP considera cinc aspectes diferents per a l'establiment i terminació de comunicacions multimèdia:

- Localització de l'usuari: fa referència al sistema final que es farà servir per a la comunicació.

- Disponibilitat de l'usuari: indica si l'usuari vol participar en les comunicacions.

- Capacitat dels usuaris: indica el medi i els paràmetres del medi que s'utilitzaran.

- Inici de la sessió: trucada, establiment de la sessió i paràmetres als dos extrems de la comunicació.

- Gestió de la sessió: inclou la transferència i finalització de sessions, modificant els paràmetres de la sessió i cridant els serveis.

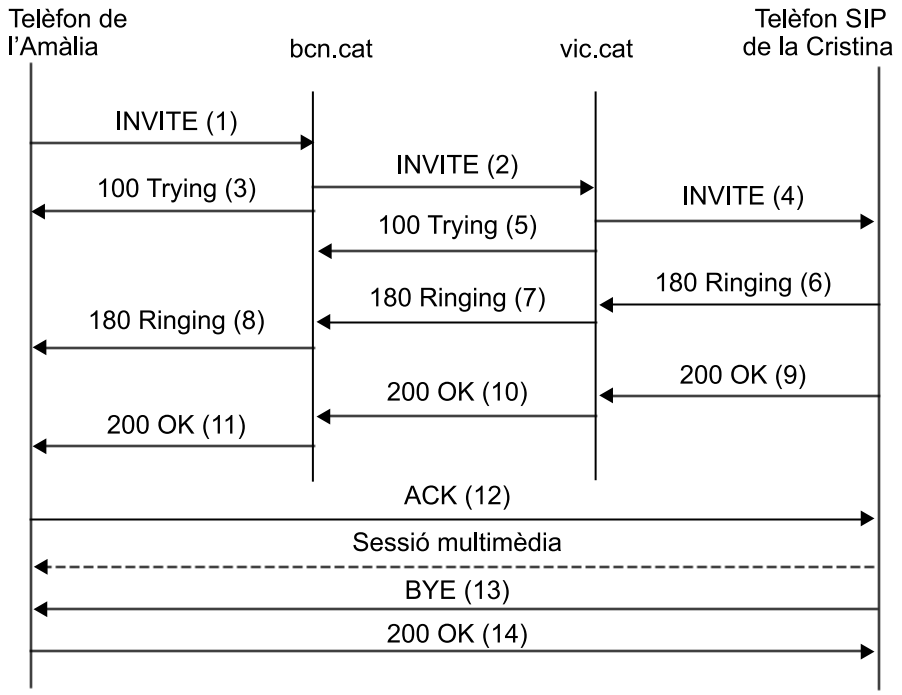
SIP no és un sistema complet de comunicacions, sinó que es tracta d'un component que es pot complementar amb altres protocols d'Internet per a implementar una aplicació multimèdia sencera. Per exemple, es poden fer servir els protocols RTP i RTCP per al transport i control de l'enviament de dades o el protocol RTSP per al control de la reproducció en temps real de dades multimèdia.

SIP no ofereix serveis, sinó que ofereix primitives que es poden utilitzar per a oferir diferents tipus de servei. Per exemple, SIP té una primitiva que permet trobar un usuari i enviar-li dades opaques a la seva localització. Si aquestes dades consisteixen en una descripció de sessió definida, per exemple amb SDP, els extrems de la comunicació es poden posar d'acord amb els paràmetres de la sessió.

SIP tampoc no ofereix serveis de control de les conferències, ni defineix com s'ha de gestionar una conferència. Tanmateix, SIP es pot fer servir per a iniciar una conferència que fa servir un altre protocol per al control de conferències.

La naturalesa dels serveis que es poden oferir amb SIP fa que la seguretat sigui molt important. És per això que SIP ofereix tota una sèrie de mecanismes de seguretat, que inclouen la prevenció de la denegació de servei, l'autenticació d'usuaris i *proxies*, la protecció de la integritat i els serveis de xifratge i privacitat.

La figura següent mostra un exemple de funcionament de SIP:



En aquest exemple, l'Amàlia i la Cristina volen establir una sessió SIP. Cada intercanvi té al costat un nombre entre parèntesis (*n*), per a fer de referència en l'explicació. També es mostren els dos servidors intermediaris que actuen en nom de l'Amàlia i la Cristina per a facilitar l'establiment de la sessió.

L'Amàlia truca la Cristina fent servir la seva identitat SIP, una mena d'URI, anomenada *URI SIP*. Una URI SIP és semblant a una adreça de correu electrònic; la definirem més endavant, però conté un amfitrió i un nom d'usuari. En aquest cas, aquesta URI és sip:amalia@bcn.cat, en què bcn.cat és el domini del proveïdor de servei de l'Amàlia. La Cristina té l'URI SIP sip:cris@vic.cat. L'Amàlia pot haver escrit l'adreça de la Cristina o la pot haver escollit d'un enllaç o una agenda. També hi ha les URI SIP segures, que s'anomenen SIPS, com per exemple sips:cris@vic.cat. En aquest cas, es faria servir el Transport Layer Security (TLS)[28], per a proveir d'una connexió segura i xifrada per a transportar els missatges SIP.

[28]L'RFC del TLS és la 4346.

SIP és basa en un mecanisme de petició/resposta molt similar al model de transaccions HTTP. Cada transacció consisteix en una petició que invoca un mètode del servidor i, com a mínim, una resposta. En aquest exemple, la transacció comença quan l'Amàlia envia una ordre INVITE adreçada a l'URI SIP de la Cristina.

A continuació es mostra la informació que portaria associada l'ordre INVITE:

```
INVITE sip:cris@vic.cat SIP/2.0
```

```
Via: SIP/2.0/UDP pc01.bcn.cat;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Cristina <sip:cris@vic.cat>
From: Amalia <sip:amalia@bcn.cat>;tag=1928301774
Call-ID: a84b4c76e66710@pc01.bcn.cat
CSeq: 314159 INVITE
Contact: <sip:amalia@pc01.bcn.cat> Content-Type: application/sdp Content-Length: 142

... Resta de dades SDP ...
```

El contingut del missatge és el següent:

- La primera línia conté el mètode, amb l'URI SIP origen i la versió de SIP.

- Les línies següents són capçaleres, que contenen informació sobre l'origen i la destinació de la comunicació (*From*, *To*), l'adreça on l'Amàlia vol rebre les respostes, indicada en *Via*, i alguns identificadors únics. *Cseq* és un número de seqüència que s'incrementa amb cada nova petició. *Contact* conté l'URI SIP que representa la ruta directa cap a l'Amàlia, formada per un nom d'usuari i un nom de domini. El nombre *Max-Forwards* indica el nombre màxim de salts que pot fer aquesta petició i, finalment, *Content-Type* i *Content-Length* contenen informació sobre el cos del missatge.

Vegeu també

Es pot trobar una llista completa de les capçaleres SIP en l'annex 3 d'aquest mòdul didàctic.

Els paràmetres específics de la sessió no es defineixen amb SIP, sinó amb un altre protocol, com ara SDP, que no veurem aquí.

Com que el telèfon de l'Amàlia no sap on pot trobar la Cristina o el servidor SIP de bcn.cat, envia l'ordre INVITE al seu servidor de domini, bcn.cat (1). El servidor bcn.cat és un tipus de servidor SIP conegut com a *servidor intermediari*. La seva funció és redireccionar les peticions SIP en nom de qui les envia. En aquest exemple, el servidor intermediari rep la petició INVITE (2) i respon amb una resposta 100 (*Trying*) (3). Això indica que la petició s'ha rebut bé i que s'està intentant enviar l'INVITE a la seva destinació. La resposta SIP té la forma d'un codi de 3 xifres i una frase descriptiva (com en l'HTTP).

El servidor intermediari bcn.cat troba l'adreça SIP del domini vic.cat. D'aquesta manera, aconsegueix l'adreça del servidor intermediari de vic.cat i li reenvia l'ordre INVITE (2). Aquest contesta amb un 100 (*Trying*) (5), per a indicar que està treballant a servir la petició. El servidor intermediari consulta una base de dades per a trobar l'adreça de la Cristina i li envia la petició INVITE (4) al seu telèfon SIP. Quan el telèfon rep la petició, sona per a informar la Cristina que està rebent una trucada. El telèfon SIP indica que s'està fent la trucada enviant un missatge de tipus 180 (*Ringing*) (6), (7), (8), que arriba fins al telèfon de l'Amàlia. Quan la resposta arriba al telèfon de l'Amàlia, aquest fa alguna indicació que està sonant el telèfon a l'altre extrem. En aquest cas, la Cristina respon la trucada, la qual cosa fa que s'enviï una resposta de tipus 200 OK (9),

(10), (11). L'Amàlia envia un ACK (12). En aquest moment, cal negociar els paràmetres de la sessió amb el protocol SDP. A continuació es mostra com seria el missatge de resposta enviat pel telèfon SIP de la Cristina:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP proxy03.vic.cat
;branch=z9hG4bKnashds8;received=192.0.2.3
Via: SIP/2.0/UDP proxy01.bcn.cat
;branch=z9hG4bK77ef4c2312983.1;received=192.0.2.2
Via: SIP/2.0/UDP pc01.bcn.cat
;branch=z9hG4bK776asdhds ;received=192.0.2.1
To: Cristina <sip:cris@vic.cat>;tag=a6c85cf
From: Amalia <sip:amalia@bcn.cat>;tag=1928301774
Call-ID: a84b4c76e66710@pc01.bcn.cat
CSeq: 314159 INVITE
Contact: <sip:cristina@192.0.2.4> Content-Type: application/sdp Content-Length: 131

... Resta de dades SDP ...
```

El contingut del missatge és el següent:

- La primera línia conté la resposta, amb la versió de SIP, el codi de resposta i la descripció.

- Les línies següents són capçaleres. *Via*, *From*, *Call-ID* i *Cseq* es copien de la petició INVITE (ara hi ha diverses capçaleres de tipus *Via* per a indicar la comunicació per mitjà dels servidors intermediaris). *Contact* conté l'URI SIP que representa la ruta directa cap a la Cristina, formada per un nom d'usuari i un nom de domini. *Content-Type* i *Content-Length* contenen informació sobre el cos del missatge.

La sessió multimèdia entre la Cristina i l'Amàlia ha començat i poden enviar les dades multimèdia en el format acordat. En general, aquests paquets seguiran una ruta diferent de la dels missatges SIP. Durant la comunicació, es poden canviar les característiques de la sessió tornant a enviar una ordre INVITE.

Per a acabar la sessió, la Cristina envia una ordre BYE (penja) (13). Aquest BYE s'envia directament al telèfon de l'Amàlia, sense passar pels servidors intermediaris. L'Amàlia respon amb un 200 OK (14). En aquest cas no hi ha ACK.

11.3.2. El protocol SIP

En aquest subapartat veurem el protocol SIP, incloent-hi el format dels missatges de petició i resposta i els camps de capçalera. Moltes de les característiques del protocol s'hereten d'HTTP, però no es tracta d'una extensió. Per exemple, SIP pot fer servir UDP per a enviar les seves peticions i respostes. Les adreces (URI SIP) dels usuaris que fan servir aquest protocol tenen la forma:

URI SIP

En SIP, les URI (adreces) poden començar amb SIP o *sips*. L'ús de SIPS indica que s'ha d'establir una connexió segura (amb TLS) per a fer la comunicació.

```
sip:user:password@host:port;uri-parameters?headers
```

Exemple:

```
sip:amalia@bcn.cat
```

Les URI per a SIPS segueixen el mateix format, canviant *sip* per *sips* a l'inici de l'URI.

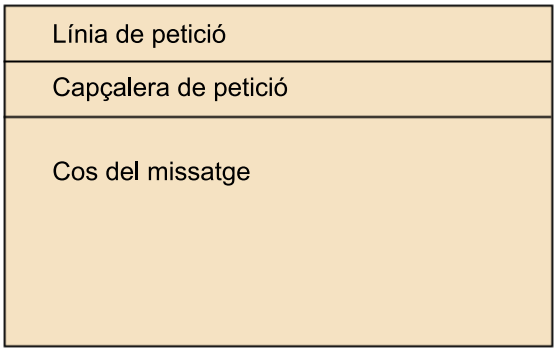
La resta de camps de l'URI:

- *user*: l'identificador d'un recurs a la màquina (*host*) que s'està adreçant. En aquest context, *host* es refereix a un domini.
- *password*: la contrasenya associada a l'usuari. Encara que es pot posar, no es recomana, perquè significa enviar-la "en clar" i és un risc de seguretat.
- *host*: l'ordinador que ofereix el recurs SIP. Aquest camp pot contenir un nom de domini o una adreça IP numèrica. Es recomana utilitzar el nom de domini.
- *port*: el número de port on s'ha d'enviar la petició.
- *uri parameters*: paràmetres que afecten la petició que porta l'URI. Els paràmetres es representen com a *nom-param=valor-param*. Estan separats del *host:port* amb punt i coma i entre ells també se separen amb un punt i coma.
- *headers*: camps de capçalera que es poden incloure en la petició representada per l'URI.

SIP és un protocol basat en missatges de text. Això facilita la implementació i l'ampliació. A continuació veurem el format dels missatges de petició i resposta SIP.

11.3.3. El missatge de petició SIP

La figura següent mostra el format del missatge de petició. Aquest missatge consta de tres parts ben diferenciades: la línia de petició, la capçalera de petició i el cos del missatge.



La línia de petició té la forma següent:

```
OrdreSIP [Espai] URI-petició [Espai] Versió-SIP [Salt línia]
```

L'ordre SIP indica quina ordre del protocol estem fent servir. L'*URI-petició* identifica el recurs al qual volem accedir, i la versió SIP indica la versió del protocol que estem utilitzant. [Espai] és un espai en blanc i [Salt línia] està representat pels caràcters de salt de línia (ASCII 10) i retorn de carro (ASCII 13). La resta d'espais en blanc s'ha posat per claredat. Les ordres SIP són: REGISTER, INVITE, ACK, CANCEL, BYE i OPTIONS. La funcionalitat de cada mètode és la següent:

- REGISTER: permet enregistrar un recurs, perquè després es puguin fer comunicacions SIP.
- INVITE: permet iniciar una sessió per part d'un usuari.
- ACK: indica que s'ha acceptat la petició enviada amb INVITE i que la sessió s'ha establert.
- CANCEL: permet cancel·lar una petició ja enviada. Es genera un missatge d'error i s'ha de deixar de processar la petició. Només s'hauria de fer servir per a cancel·lar un INVITE.
- BYE: permet finalitzar la sessió.
- OPTIONS: permet demanar les característiques d'un programa client o d'un *proxy server*.

La capçalera de petició té tres parts: la capçalera general, la capçalera pròpiament de petició i la capçalera d'entitat. El format de tots els camps de les capçaleres és el següent:

```
Nom-camp: [Espai] valor-camp [Salt línia]
```

La capçalera general representa informació general que és independent de si el missatge és de petició o de resposta. La capçalera de petició conté camps que només tenen sentit per a un missatge de petició. La capçalera d'entitat conté informació sobre el cos del missatge (que també es coneix com a *entitat*).

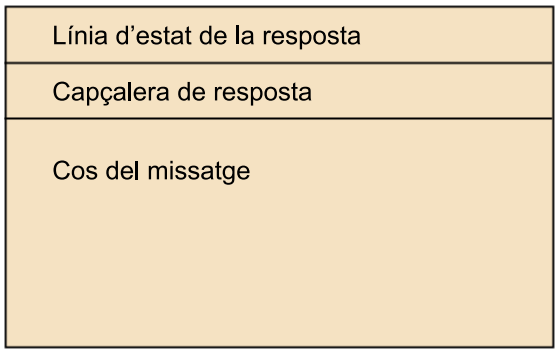
Vegeu també

Els valors que poden prendre aquestes tres capçaleres estan explicats amb detall en l'annex 3.

El cos del missatge porta la informació, si n'hi ha, associada a la petició.

11.3.4. El missatge de resposta SIP

La figura següent mostra el format del missatge de resposta. Aquest missatge consta de tres parts ben diferenciades: la línia d'estat de la resposta, la capçalera i el cos del missatge.



La línia d'estat de la resposta té la forma següent:

```
Versió-SIP [Espai] Codi-Estat [Espai] Descripció-Estat [Salt línia]
```

La versió SIP defineix la versió del protocol que s'està fent servir, el codi d'estat defineix l'èxit o error en la petició i la descripció de l'estat, que és una descripció textual del codi d'estat. [Espai] és un espai en blanc i [Salt línia] és representat pels caràcters de salt de línia (ASCII 10) i retorn de carro (ASCII 13). La resta d'espais en blanc s'ha posat per claredat.

Vegeu també

Els valors dels codis i descripcions d'estat estan explicats en detall en l'annex 3.

La capçalera de resposta té tres parts: la capçalera general, la capçalera pròpiament de resposta i la capçalera d'entitat. El format de tots els camps de les capçaleres és:

```
Nom-camp: [Espai] valor-camp [Salt línia]
```

La capçalera general representa informació general que és independent de si el missatge és de petició o de resposta. La capçalera de resposta conté camps que només tenen sentit per a un missatge de resposta. La capçalera d'entitat conté informació sobre el cos del missatge (que també es coneix com a *entitat*).

Vegeu també

Els valors que poden prendre aquestes tres capçaleres estan explicats en detall en l'annex 3.

El cos del missatge porta la informació, si n'hi ha, associada a la resposta.

11.4. H.323

Aquest protocol està considerat com una alternativa al SIP i és molt popular a l'hora de transmetre àudio i vídeo en temps real. Està dedicat a la transmissió de veu sobre IP (*voice over IP* –VoIP–, en anglès). Amb H.323 és possible comunicar un equip connectat a Internet amb un telèfon connectat a la xarxa telefònica.

H.323 inclou diverses especificacions:

- Negociació de codificacions d'àudio i vídeo entre els extrems de la comunicació.
- Com s'envien els trossos de dades d'àudio i vídeo. H.323 obliga a fer servir RTP.
- Com es comuniquen els equips amb els seus *gatekeepers*.
- Com es connecten els equips connectats a Internet amb els telèfons connectats a la xarxa telefònica.

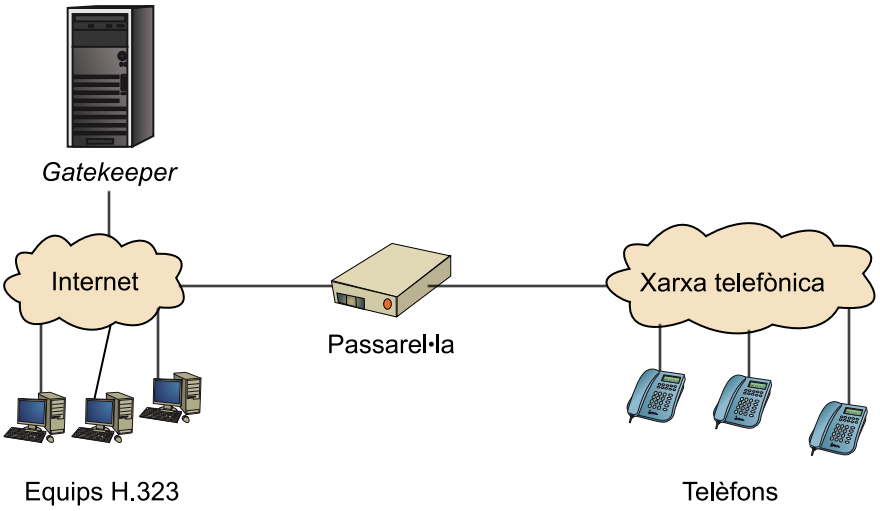
Els requisits mínims d'H.323 són suportar l'àudio, i les característiques de vídeo són opcionals. D'aquesta manera, els fabricants poden oferir uns equips senzills amb les característiques d'àudio i deixar per a equips més sofisticats la part de vídeo. Per a transmetre les dades d'àudio, s'ha d'utilitzar un estàndard que utilitza PCM (G.711) per a generar veu digitalitzada tant a 56 kbps com a 64 kbps. Encara que el vídeo és opcional, en cas que el terminal en tingui, ha de suportar com a mínim l'estàndard QCIF H.261.

Formats de dades utilitzats en H.323

G.711 - Àudio http://www.itu.int/rec/T-REC-G.711/en.

QCIF H.261 - Vídeo http://www.itu.int/rec/T-REC-H.261/en.

La figura següent mostra un exemple d'arquitectura d'un sistema H.323:



La passarel·la és un element opcional que només es necessita si volem connectar equips que estan en una xarxa diferent, com ara la xarxa telefònica.

El *gatekeeper* és un element opcional en la comunicació entre terminals H.323. Tot i això, són l'element més important d'una xarxa H.323, ja que fan de punt central de totes les trucades que hi ha dintre d'una zona i proporcionen serveis als terminals registrats i control de les trucades. Es podria dir que el porter fa de commutador virtual.

Per a acabar aquest subapartat, es descriuen les diferències principals entre SIP i H.323:

- H.323 és un conjunt integrat de protocols per a fer conferències multimèdia que inclou: senyalització, registre, control d'admissió, transport i còdecs. D'altra banda, SIP només s'encarrega d'iniciar i gestionar la sessió, sense fer cap imposició en el transport o els formats d'àudio o vídeo suportats.

- H.323 va ser definit per ITU (telefonia), mentre que SIP va ser definit per IETF (Internet). Això fa que el punt de vista dels dos sigui molt diferent.

- H.323 és un estàndard complex, format per moltes parts, mentre que SIP és molt més simple i, per tant, més fàcil d'implementar.

11.5. Skype

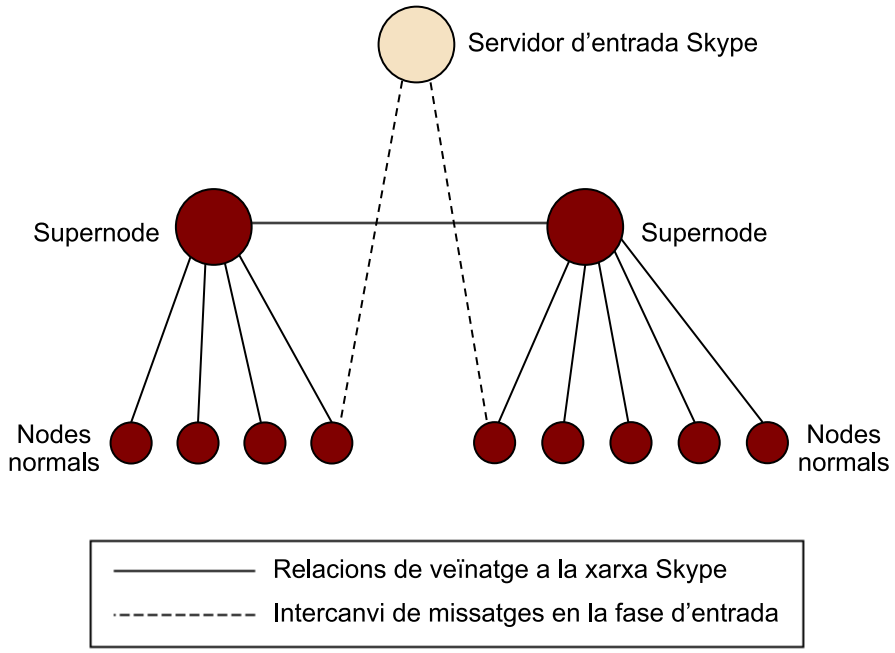
Skype és un sistema de telefonia d'igual a igual (*peer-too-peer*, P2P) que funciona sobre la xarxa Internet. És la competència d'estàndards de transmissió de veu sobre IP (VoIP) com ara SIP o H.323. Va ser desenvolupada per l'equip que va fer el KaZaA l'any 2003.

KaZaA

KaZaA és una aplicació P2P de descàrrega de continguts que va tenir molt èxit fa uns anys. Utilitza el mateix protocol P2P que Skype, FastTrack.

Es tracta d'una aplicació molt utilitzada d'ençà que es va crear, tant en els seus serveis gratuïts com de pagament. Ofereix moltes funcionalitats, incloent-hi àudio i videoconferències gratuïtes, l'ús de tecnologies P2P (descentralitzades) per a no tenir problemes amb tallafocs o amb la traducció d'adreces (en anglès, *network address translation*, NAT) i les múltiples mesures per a no poder reconèixer el protocol o el programari que l'implementen.

La identificació de l'usuari que truca està emmascarada quan es fa una trucada. La diferència principal entre l'Skype i altres serveis és que fa servir tecnologia P2P, mentre que la resta fan servir arquitectures client-servidor. La implementació de P2P que fa servir és FastTrack, implementada a l'aplicació KaZaA. L'arquitectura que implementa FastTrack és una xarxa superposada (*overlay network*, en anglès) amb dos tipus de nodes: els nodes normals i els supernodes. Un node normal és un ordinador en què s'ha instal·lat l'aplicació Skype, i permet fer trucades i enviar missatges de text. Els supernodes són punts límit de la xarxa Skype. Qualsevol node que tingui una IP pública i prou recursos (memòria, amplada de banda, etc.) es pot convertir en un supernode. Un node normal ha d'entrar a la xarxa Skype amb el seu usuari i paraula de pas i d'aquesta manera estarà connectat a un supernode. Encara tenim un altre element a la xarxa, el servidor d'entrada, que és el que emmagatzema els usuaris i paraules de pas a tota la xarxa Skype (els noms d'usuari són únics). A part d'aquest servidor, la resta d'operacions a la xarxa, com cerca d'usuaris, enviament de missatges i trucades, etc., es fa de manera totalment descentralitzada.



El directori de Skype està completament descentralitzat i distribuït entre els nodes de la xarxa, fet que significa que és fàcilment escalable quan es té un gran nombre d'usuaris sense necessitat d'una estructura complexa. Les trucades s'envien per mitjà d'altres usuaris de Skype en la Xarxa per a facilitar la comunicació quan hi ha tallafocs o NAT. Això fa que els usuaris que es con-

necten directament a la xarxa (sense NAT) tinguin més trànsit per a encaminar les trucades d'altres usuaris. Com que es tracta d'una xarxa superposada, cada client Skype ha de mantenir una llista de nodes accessibles. Aquesta llista és la llista d'amfitrions i conté l'adreça IP i el número de port dels supernodes. En sistemes Windows es guarda en el registre.

És possible desenvolupar programes que facin servir l'API de Skype per a accedir a la seva xarxa. Tanmateix, el codi és tancat i el protocol de propietat, fet que en dificulta la interoperabilitat amb altres sistemes.

12. Annexos

12.1. Annex 1. El protocol RTSP

12.1.1. Ordres

La taula següent descriu les ordres RTSP:

Ordre	Descripció
OPTIONS	Permet al client demanar informació sobre les opcions de comunicació amb el servidor
DESCRIBE	Recupera la informació del contingut multimèdia identificat per l'URL que s'envia amb l'ordre
ANNOUNCE	Si l'envia el client, s'està indicant la descripció d'una presentació. Si l'envia el servidor, s'actualitza la informació de la descripció de la sessió en temps real
SETUP	Especifica els paràmetres del mecanisme de transport. Es pot enviar a mitjan sessió, per canviar aquests paràmetres
PLAY	S'indica al servidor que s'ha d'iniciar l'enviament de dades
PAUSE	S'indica al servidor que es vol parar temporalment la recepció de dades de la sessió
TEARDOWN	Es tanca la recepció de dades
GET_PARAMETER	Demana el valor d'un paràmetre de la descripció de la sessió
SET_PARAMETER	Envia el valor d'un paràmetre de la descripció de la sessió
REDIRECT	S'indica al client que s'ha de connectar a un altre servidor
RECORD	S'inicia la gravació de dades d'acord amb la descripció de la presentació

12.1.2. Codis d'estat

Els codis d'estat són de 5 tipus: 1XX, informatius, 2XX, petició reeixida, 3XX, redireccionament, 4XX, error en la petició del client i 5XX, error en el servidor.

Només s'han posat els codis específics d'RTSP, la resta estan definits en l'RFC d'HTTP.

Tipus de codi	Valor	Descripció
Informatiu	100	El client pot continuar amb la petició

Tipus de codi	Valor	Descripció
Èxit	250	Poc espai d'emmagatzematge. Pot ocórrer amb l'ordre RECORD
Redirecció	3XX	Els mateixos que HTTP
Error part client	405	El mètode demanat no es pot executar per al recurs
Error part client	451	El servidor no suporta algun dels paràmetres indicats en la petició
Error part client	452	La conferència no s'ha trobat
Error part client	453	No hi ha prou amplada de banda per a rebre el contingut
Error part client	454	La sessió no s'ha trobat
Error part client	455	El mètode no és vàlid en l'estat actual del protocol
Error part client	456	Un paràmetre indicat a la capçalera no es pot processar
Error part client	457	El rang demanat està fora dels límits
Error part client	458	Un paràmetre que es volia modificar amb SET_PARAMETER és només de lectura
Error part client	459	El paràmetre no es pot aplicar al recurs. Només es pot aplicar a un corrent de dades
Error part client	460	El paràmetre no es pot aplicar al recurs. Només es pot aplicar a una presentació
Error part client	461	El tipus de transport no està suportat
Error part client	462	El client no és accessible
Error part servidor	551	Una opció enviada a la capçalera no és suportada

12.1.3. Capçaleres

A la taula següent es descriuen les capçaleres del protocol RTSP que són específiques d'aquest protocol. La resta estan definides en l'RFC d'HTTP:

Capçalera	Descripció
Accept	Tipus de continguts acceptats en la presentació
Allow	Mètodes RTSP suportats pel servidor
Bandwidth	Amplada de banda estimada del client
Blocksize	Grandària de dades preferida pel client. No inclou les capçaleres de protocols de nivells inferiors (IP, UDP, RTSP)
Cache-Control	Control del mecanisme de cau dels continguts
Conference	Estableix una connexió entre aquesta conferència i un corrent de dades existent
Content-Length	Longitud del contingut enviat en el cos del missatge

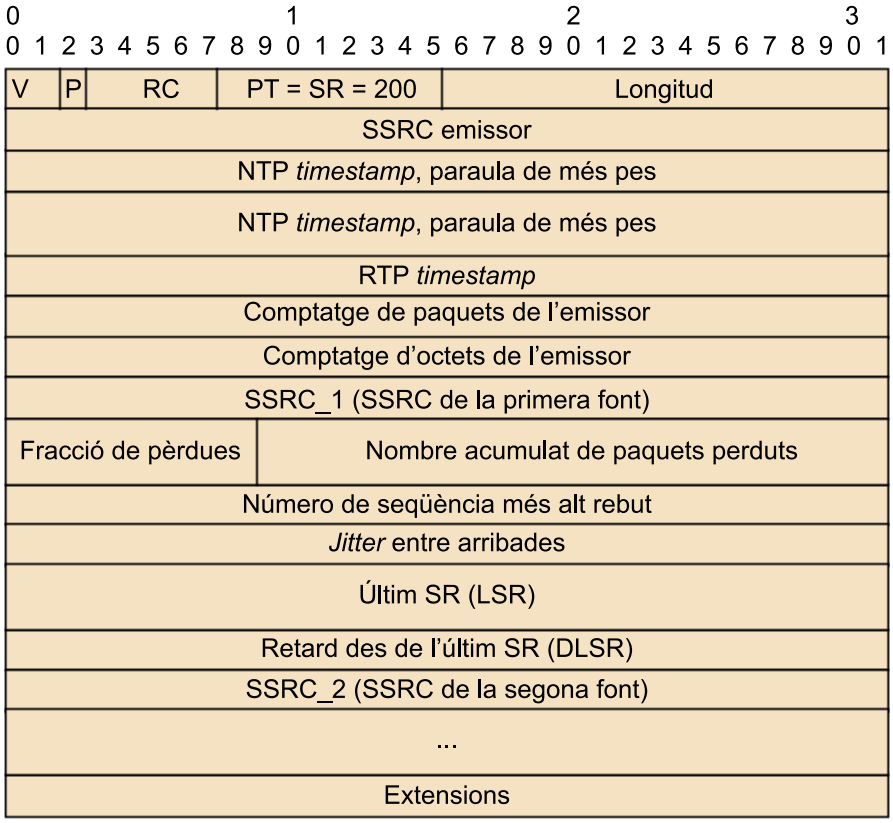
Capçalera	Descripció
CSeq	Número de seqüència en la sessió RTSP de parells petició-resposta
Expires	Moment en què el contingut estarà obsolet
If-Modified-Since	S'utilitza amb SETUP i DESCRIBE. Si la descripció no s'ha modificat des del moment indicat en aquesta capçalera, no s'envien dades
Last-Modified	Ultima vegada que es va modificar la descripció del recurs
Proxy-Require	Paràmetres que ha de suportar un servidor intermediari
Require	Serveix per a demanar al servidor quines opcions suporta
RTP-Info	Indica paràmetres específics d'RTP en una ordre PLAY
Scale	Permet indicar si el contingut es vol veure a velocitat normal, velocitat més ràpida o més lenta
Session	Identificador de la sessió
Speed	Demana que la transferència de dades es faci a una determinada velocitat
Transport	Protocol de transport utilitzat
Unsupported	Característiques no suportades pel servidor

12.2. Annex 2. El format dels paquets RTCP

En aquest annex es descriuen les dades que contenen cada un dels paquets del protocol RTCP.

12.2.1. Informe d'emissor (SR)

La figura següent mostra els camps del paquet de tipus SR:



Aquest tipus de paquet té tres seccions i una quarta, opcional, que són les extensions, que són definides pels perfils.

La primera secció és la capçalera i té vuit octets. Els camps són els següents:

- Versió (V): 2 bits. Identifica la versió d'RTP i és la mateixa que per als paquets de dades. La versió actual és la 2.

- Farciment (P): 1 bit. Si el bit de farciment està activat, aquest paquet RTCP té octets de farciment que no formen part de la informació de control però que estan inclosos en la longitud del paquet. L'últim octet de farciment indica quants octets s'han d'ignorar, incloent-hi l'octet mateix. En un paquet compost, només s'ha d'afegir farciment als paquets individuals.

- Comptador de recepció d'informes (RC): 5 bits. El nombre de blocs de recepció d'informes que hi ha en aquest paquet. 0 també és vàlid.

- Tipus de paquet (PT): 8 bits. Conté una constant amb valor 200 per a identificar que es tracta d'un paquet de tipus SR.

- Longitud: 16 bits. Longitud del paquet en paraules de 32 bits menys 1, incloent-hi capçalera i farciment.

- SSRC: 32 bits. Identificador de la font de sincronització del creador d'aquest paquet SR.

La segona secció, la secció de l'emissor, té 20 octets de longitud i està present en qualsevol paquet de tipus SR. Fa un resum de les dades transmeses per aquest emissor. Els camps són els següents:

- NTP *timestamp*: 64 bits. Indica el temps en què aquest informe es va enviar. Es pot fer servir per a calcular el temps de propagació de la informació si es combina amb marques de temps de paquets de tipus RR.

- RTP *timestamp*: 32 bits. Té el mateix valor que el camp anterior, però amb les mateixes unitats que les marques de temps dels paquets RTP.

- Nombre de paquets enviats per l'emissor: 32 bits. El nombre de paquets RTP de dades transmesos per l'emissor des que es va iniciar la transmissió fins a l'enviament d'aquest paquet.

- Nombre d'octets enviats per l'emissor: 32 bits. Nombre total d'octets de dades (sense incloure capçaleres ni farciment) transmesos en paquets de dades RTP per l'emissor des que es va iniciar la transmissió fins a l'enviament d'aquest paquet.

La tercera secció conté 0 o més blocs d'informes de recepció, depenent del nombre de fonts rebudes per aquest emissor des de l'últim informe. Cada bloc conté les estadístiques següents:

- SSRC_n (identificador de la font): 32 bits. L'identificador de la font SSRC a la qual pertany la informació enviada en aquest bloc.

- Fracció de pèrdues: 8 bits. La fracció de paquets RTP perduts per la font SSRC_n des que es va enviar l'últim paquet de tipus SR o RR.

- Nombre acumulat de paquets perduts: 24 bits. El nombre total de paquets perduts des de l'inici de la recepció.

- Nombre de seqüència més alt rebut: 32 bits. Els 16 bits més alts contenen el nombre de seqüència més alt rebut i la resta són extensions al nombre de seqüència.

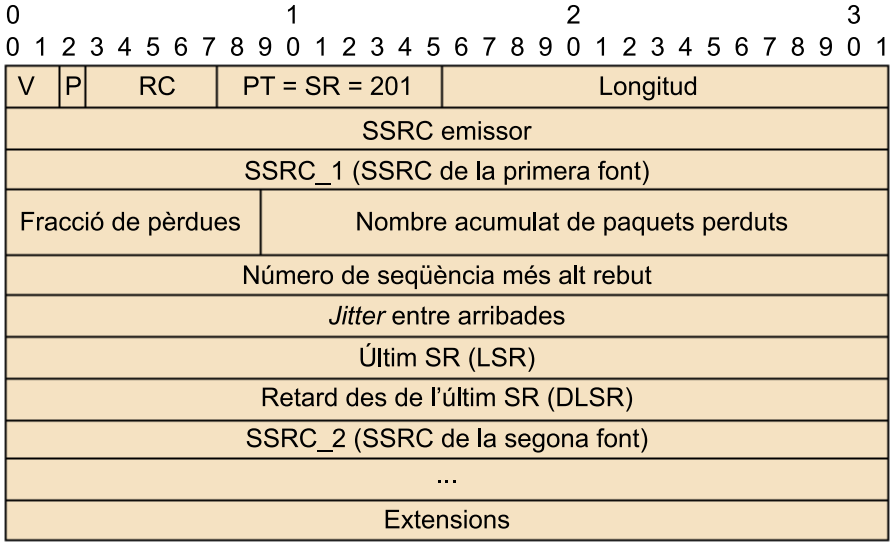
- *Jitter* entre arribades: 32 bits. Una estimació estadística de la variància del temps entre arribades dels paquets de dades RTP, mesurat en unitats de marca de temps i expressat com un enter sense signe.

- Últim SR (LSR): 32 bits. La meitat dels 64 bits rebuts en l'NTP *timestamp* de l'últim paquet de tipus SR rebut des de la font SSRC_n.

- Retard des de l'últim SR (DLSR): 32 bits. El retard entre l'últim SR rebut des de SSRC_n i l'enviament d'aquest paquet d'informació.

12.2.2. Informe de receptor (RR)

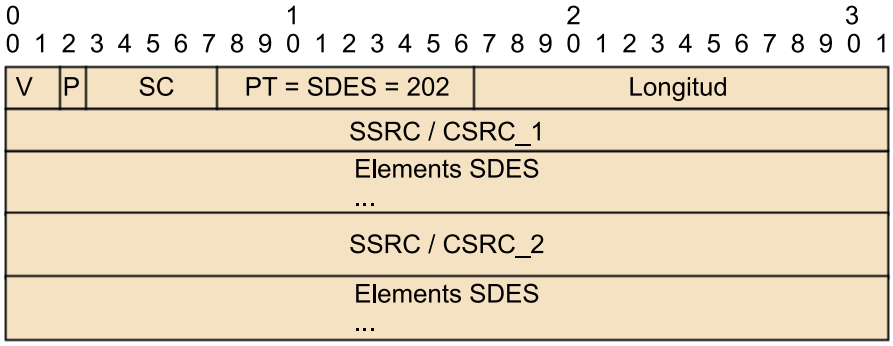
La figura següent mostra els camps del paquet de tipus RR:



El format d'aquest tipus de paquet és igual que el del paquet SR, tret que el tipus de paquet té la constant 201 i la informació de l'emissor (*timestamps* NTP i RTP i nombre de paquets i octets enviats) no hi és.

12.2.3. Descripció d'elements de la font (SDES)

La figura següent mostra els camps del paquet de tipus SDES:



Aquest tipus de paquet té una estructura en tres nivells composta per una capçalera i zero o més porcions, cada una formada per elements que descriuen la font identificada per la porció.

La primera secció és la capçalera i té vuit octets. Els camps són els següents:

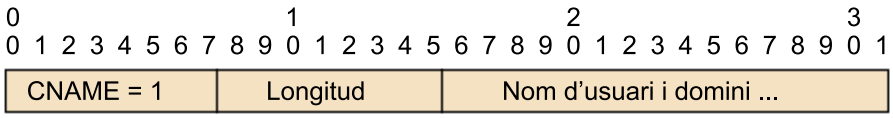
- Versió (V), farciment (P) i longitud tenen el mateix significat que els definits pel paquet SR.

- Nombre de fonts emissores (SC): 5 bits. El nombre de porcions SSRC/CRSC contingudes en un paquet SDES.

- Tipus de paquet (PT): 8 bits. Conté la constant 202.

- Cada porció conté un identificador d'SSRC/CSRC, seguit d'una llista d'elements SDES. Cada element consisteix en un camp de 8 bits que indica el tipus, un camp de 8 bits que és un comptador de nombre d'octets que indica la longitud del camp (sense la capçalera) i el text de l'element, que no pot passar de 255 octets.

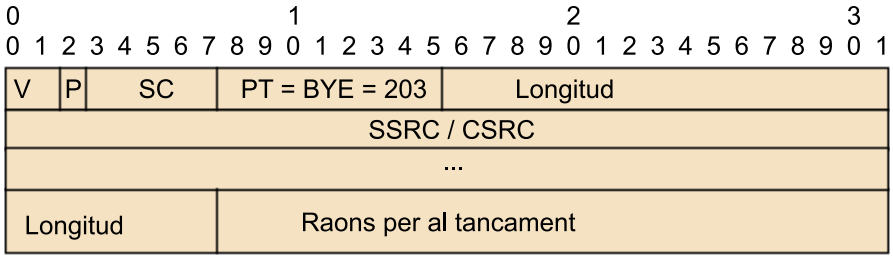
Com a exemple d'element SDES, veurem com es representa el camp CNAME.



El camp CNAME ha de ser únic i fa d'enllaç entre un identificador SSRC (que pot canviar) i l'identificador de la font, que ha de ser constant.

12.2.4. Paquet de tancament (BYE)

La figura següent mostra els camps del paquet de tipus BYE:



El paquet BYE indica que una o més fons ja no estan actives. Els camps són els següents:

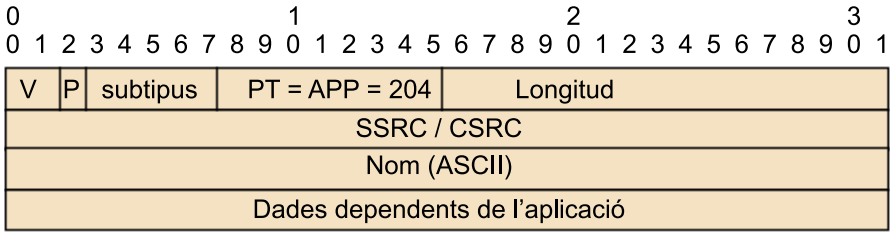
- Versió (V), farciment (P) i longitud tenen el mateix significat que els definits pel paquet SR.

- Nombre de fonts emissores (SC): 5 bits. El nombre de porcions SSRC/CRSC contingudes en un paquet SDES.

- Tipus de paquet (PT): 8 bits. Conté la constant 203.

12.2.5. Paquet definit per l'aplicació (APP)

La figura següent mostra els camps del paquet de tipus APP:



El paquet APP està pensat perquè les aplicacions afegeixin funcionalitat. Els camps són:

- Versió (V), farciment (P) i longitud tenen el mateix significat que els definits pel paquet SR.
- Subtipus: 5 bits. Es pot fer servir per a definir un subtipus d'aplicacions.
- Tipus de paquet (PT): 8 bits. Conté la constant 204.
- Nom: 8 octets. Nom escollit per la persona que ha definit els paquets de tipus APP, per a diferenciar-los d'altres paquets de tipus APP.
- Dades dependents de l'aplicació: longitud variable. Aquest camp pot aparèixer o no. És directament interpretat per l'aplicació i no per RTP. Ha de ser múltiple de 32 bits.

12.3. Annex 3. El protocol SIP

12.3.1. Codis d'estat

Els codis d'estat són de 5 tipus: 1XX, respostes provisionals, 2XX, petició reeixida, 3XX, redireccionament, 4XX, error en la petició del client, 5XX, error en el servidor i 6XX, errors globals (específics de SIP).

Molts dels codis d'estat estan heretats del protocol HTTP, però també hi ha codis específics de SIP. Els codis HTTP que no tinguin sentit en SIP no s'han de fer servir.

Tipus de codi	Valor	Descripció
Respostes provisionals	100	Indica que l'ordinador següent (salt, *hop* en anglès) ha rebut la petició
Respostes provisionals	180	Indica que la petició INVITE s'ha rebut a l'altra banda i s'avisa l'usuari
Respostes provisionals	181	La trucada s'està redirigint
Respostes provisionals	182	La trucada s'ha posat a la cua, perquè el receptor de la trucada no està disponible
Respostes provisionals	183	Informació sobre el progrés de la trucada
Èxit	200	La petició ha anat bé
Redirecció	300	Indica que hi ha múltiples opcions en la localització de l'adreça de l'usuari a qui es truca
Redirecció	301	L'usuari ha canviat d'adreça i se li ha de trucar a la nova adreça indicada a la capçalera *Contact*
Redirecció	302	L'usuari ha canviat d'adreça i se li ha de trucar a la nova adreça indicada en la capçalera *Contact*
Redirecció	305	S'ha de fer servir un servidor intermediari
Redirecció	380	La trucada no ha estat possible, però hi ha serveis alternatius disponibles
Redirecció	400	Format de petició erroni
Redirecció	401	L'usuari necessita autenticar-se
Redirecció	402	Reservat per a ús futur
Error part client	403	El servidor no vol servir la petició del client, encara que tenia un format correcte
Error part client	404	L'usuari a qui s'ha trucat no es troba en el servidor
Error part client	405	El mètode demanat no es pot executar per a l'adreça indicada
Error part client	406	El recurs no és acceptable segons els valors enviats en la capçalera *Accept*
Error part client	407	El client s'ha d'autoritzar davant un servidor intermediari
Error part client	408	El servidor no ha pogut produir una resposta a temps
Error part client	410	El recurs ja no està disponible en el servidor
Error part client	413	La grandària de l'entitat enviada pel client és massa gran perquè el servidor la pugui tractar
Error part client	414	La grandària de l'URI és massa gran perquè el servidor la pugui tractar
Error part client	415	El tipus de dades del contingut no està suportat pel servidor
Error part client	416	La petició no es pot servir, perquè l'esquema indicat en l'URI del recurs no es reconeix

Tipus de codi	Valor	Descripció
Error part client	420	El servidor no reconeix les extensions indicades en la capçalera *Proxy-Require*
Error part client	421	Els programes necessiten una extensió específica per a poder servir la petició
Error part client	423	El servidor refusa la petició, perquè expira massa aviat
Error part client	480	S'ha pogut arribar fins al servidor, però l'usuari a qui s'ha trucat no està disponible temporalment
Error part client	481	La transacció no existeix
Error part client	482	S'ha detectat un bucle en la trucada
Error part client	483	S'ha passat el màxim d'amfitrions indicats en la capçalera *MaxHops*
Error part client	484	L'adreça de l'usuari a qui s'ha trucat és incompleta
Error part client	485	L'adreça és ambigua
Error part client	486	L'usuari no accepta la trucada
Error part client	487	La petició s'ha acabat amb un CANCEL o un BYE
Error part client	488	La petició no és acceptable, per algun paràmetre indicat en la sessió
Error part client	491	Hi ha una altra petició pendent
Error part client	493	El cos del missatge porta un tipus MIME que el servidor no pot tractar
Error part servidor	500	Error intern del servidor, que fa que no pugui servir la petició
Error part servidor	501	El servidor no té implementada la funcionalitat necessària per a servir la petició
Error part servidor	502	El servidor rep un error quan fa de servidor intermediari o passarel·la
Error part servidor	503	El servei està temporalment no disponible
Error part servidor	504	El servidor no rep una resposta d'un servidor extern a temps
Error part servidor	505	El servidor no suporta la versió del protocol indicada pel client
Error part servidor	513	El missatge és massa llarg perquè el servidor el pugui processar
Error general	600	L'usuari trucat no està disponible
Error general	603	L'usuari trucat no vol acceptar la trucada
Error general	604	L'usuari trucat no existeix
Error general	606	La petició no és acceptable, per algun paràmetre indicat en la sessió

12.3.2. Capçaleres

En la taula següent es descriuen les capçaleres del protocol SIP que són específiques d'aquest protocol. La resta estan definides en l'RFC d'HTTP:

Capçalera	Descripció
Alert-Info	To de trucada alternatiu
Allow	Mètodes suportats
Authentication-Info	Informació per autenticació mútua
Authorization	Credencials d'autorització del client
Call-ID	Identificació d'una trucada o d'una sèrie dels registres d'un client
Call-Info	Informació addicional sobre els participants en la trucada
Contact	Conté una URI que pot tenir diferents significats segons el context
Content-Disposition	Indicació de l'organització del cos del missatge, en termes de les parts que pugui tenir i els formats MIME
Content-Encoding	Codificació preferida per al contingut
Content-Length	Longitud del contingut enviat en el cos del missatge
Cseq	Número de seqüència associat a una petició
Date	Data i hora en què es va crear el missatge
Error-Info	Informació addicional sobre l'error expressat en el codi de resposta
Expires	Moment en què el contingut estarà obsolet
From	Iniciador de la petició
In-Reply-To	Referència als identificadors de la trucada associats a la trucada actual
Max-Forwards	Límit al nombre de servidors intermediaris o passarel·les que poden reenviar la petició
Min-Expires	Interval de refresc mínim
Organization	Nom de l'organització a la qual pertany qui genera la petició
Priority	Prioritat de la trucada des del punt de vista del client
Proxy-Authorization	Identificació del client enfront d'un servidor intermediari que demana autenticació
Proxy-Require	Requisits que ha de tenir el servidor intermediari
Record-Route	El servidor intermediari omple aquesta capçalera per a indicar que les futures peticions han de passar per ell
Reply-To	URI de retorn que no ha de ser la mateixa que la de la capçalera *From*
Require	Característiques esperades del programa d'usuari

Capçalera	Descripció
Retry-After	Indicació de quan el servei tornarà a estar disponible
Route	Descripció dels servidors intermediaris pels quals hauria de passar la petició
Server	Descripció del programari que fa servir el servidor
Subject	Tema de la trucada
Supported	Extensions suportades pel programa d'usuari
Timestamp	Indicació de quan va fer la trucada el programa client
To	Receptor de la trucada
Unsupported	Descripció de les característiques no suportades
Via	Indicació del camí que ha seguit la petició i que s'ha d'utilitzar per a les respostes
WWW-Authenticate	Valor de resposta d'autenticació

Resum

En aquest mòdul s'ha començat descrivint les arquitectures d'aplicacions més usuals a Internet: client-servidor i d'igual a igual, i també els requisits de les aplicacions orientades a l'enviament de dades o de missatges.

A continuació s'han descrit les aplicacions distribuïdes més usuals a Internet i diferents estàndards que hi estan relacionats: Web i HTTP, DNS, SMTP, FTP, XMPP, Telnet i SSH, Gnutella, KaZaA, BitTorrent i eDonkey.

Seguidament s'han introduït els requisits de les aplicacions multimèdia en xarxa.

Finalment, s'han vist un seguit de tècniques i protocols per a treballar amb continguts multimèdia a Internet, tant per a reproducció en temps real d'àudio i vídeo, com RTSP, com per a aplicacions interactives en temps real, com RTP, RTCP, SIP, H3.23 i Skype.

Bibliografia

Kurose, J. F.; Ross, K. W. *Computer Networking* (5a. ed.). Boston: Addison Wesley. ISBN 0-321-49770-8

Aquest llibre proporciona una visió completa dels diferents aspectes relacionats amb les xarxes d'ordinadors. En aquest mòdul interessa el capítol 2 ("Application Layer"), en què es poden trobar els conceptes bàsics del Web, el correu electrònic, la transferència de fitxers i el servei de directori d'Internet. També hi ha un apartat d'igual a igual que pot complementar la visió dels sistemes d'igual a igual donada en aquest mòdul.

Keagy, S. (2000). *Integrating Voice and Data Networks*. Indianapolis: Cisco Press.

En aquest llibre es descriu el protocol SIP.

Niederst, J. (2006). *Web Design in a Nutshell* (3a. ed.). Sebastopol: O'Reilly.

En aquest llibre hi ha dues seccions dedicades als formats d'àudio i vídeo que es transmeten per Internet.

Hersent, O.; Gurle, D.; Petit, J. P. (1999). *IP Telephony: Packet-Based Multimedia Communications Systems.* Indianapolis: Addison Wesley Professional.

En aquest llibre es descriuen els protocols SIP i H.323 per a la implementació de telefonia sobre IP.

Articles

Lua i altres (2005). "A survey and comparison of peer-to-peer overlay network schemes". *IEEE Communications Surveys & Tutorials* (vol. 7, núm. 2).

En aquest article trobareu un resum de com funcionen les xarxes superposades d'igual a igual més populars, i també una comparativa. També trobareu més detalls dels sistemes explicats en aquest mòdul.

Baset, S.; Schulzrinne, H. (2004). *An Analysis of the Skype Peer-To-Peer Internet Telephony Protocol*.

Aquest article descriu l'arquitectura i el protocol de Skype basant-se en la captura de les dades enviades per la xarxa que han fet els autors.