Arquitectures distribuïdes a escala Internet

Joan Manuel Marquès i Puig

P07/11068/00080

UOC
www.uoc.edu

Índex

Introducció

Internet és un recurs compartit format per milions d'ordinadors repartits arreu del món que estan connectats per mitjà de xarxes. El nombre d'aplicacions i usuaris connectats a aquest megasistema informàtic no para de créixer. La introducció del Web el 1994 va comportar un canvi radical a Internet. Aquesta va començar un procés de generalització frenètic que encara dura avui. Però és durant l'any 2000 que molta gent s'adona que Internet és alguna cosa més que correu electrònic, web i serveis comercials per web. Amb l'explosió del fenomen Napster una nova visió d'Internet es fa realitat: la Xarxa és un entorn idoni per a l'intercanvi i la compartició. Els usuaris connectats deixen de ser simples elements passius que es connecten i extreuen continguts d'Internet i es converteixen en subministradors de continguts i de recursos. I el que és més important, aquesta compartició es fa a base de formar comunitats i col·laborar directament utilitzant els mateixos ordinadors dels usuaris. El que el Napster va proporcionar al públic d'Internet no va ser només una manera fàcil d'obtenir qualsevol cançó en format MP3 sense haver de pagar. Sobretot va significar un canvi de mentalitat respecte què és, com s'organitza i per a què pot servir Internet.

Napster

El Napster va ser la primera aplicació popular d'intercanvi de fitxers que seguia el model d'igual a igual. Permetia l'intercanvi de fitxers en format MP3 entre usuaris finals d'Internet. Seguia un model d'igual a igual no pur: un índex centralitzat i les cançons repartides en els ordinadors dels usuaris que les aportaven. Un cop localitzada la ubicació de la cançó preguntant a l'índex, la baixada de cançons es feia directament entre els ordinadors dels usuaris.

Centrant aquesta reflexió entorn dels objectius d'aquest mòdul, és important adonar-nos que tant el Napster com els seus successors –Gnutella, KaZaA, e-mule, BitTorrent, etc.– aposten per la descentralització –tant en l'aportació de continguts com en l'administració del sistema–, l'escala –són sistemes formats per milions d'usuaris repartits arreu d'Internet–, l'autonomia –els recursos perquè el sistema funcioni (o com a mínim la major part d'aquests) els aporten els mateixos usuaris del sistema–, i l'autoorganització –el sistema s'adapta de manera automàtica als canvis que ocorren en el sistema (connexions, desconnexions, fallades, mobilitat, etc.). El model d'arquitectura que segueixen l'anomenem d'igual a igual o entre iguals (*peer-to-peer*, en anglès) i es diferencia molt clarament de l'arquitectura web, la gran dominadora d'Internet durant la dècada dels noranta, que és una arquitectura centralitzada (en l'arquitectura web els nodes servidors aporten continguts i recursos; i els nodes clients hi accedeixen). Abans de començar a concretar cadascuna de les arquitectures que tractarem en aquest mòdul, farem un breu repàs de la història d'Internet, i això ens ajudarà a entendre les diferents arquitectures de sistemes distribuïts que han dominat a cada moment.

Internet es va crear a finals dels seixanta amb l'objectiu de compartir recursos de processament que estaven escampats arreu dels Estats Units i que fins aquell moment estaven aïllats els uns dels altres. El repte que tenien al davant era aconseguir d'integrar les xarxes existents en aquell moment, i també les possibles tecnologies futures, en una única arquitectura de xarxa que permetés

a tots els nodes de tenir un paper al mateix nivell (tots els nodes iguals entre si). Qualsevol ordinador podia enviar paquets a qualsevol altre. L'ús principal que es donava a la xarxa era que diferents recercaires col·laboressin. Una mostra d'aquest caire col·laboratiu és que fins al final dels vuitanta no comencen a aparèixer els primers tallafocs. Tot i que des de bon començament hi havia aplicacions com l'FTP i el Telnet, que per naturalesa són client-servidor, com que tots els nodes implementaven tant la part client com la part servidora, l'ús que se'n feia era totalment simètric. Qualsevol node podia fer Telnet o FTP a qualsevol altre node. Altres aplicacions d'aquesta primera època són els missatges Usenet (*Usenet news*), que implentaven un control totalment descentralitzat dels continguts dels grups; i el correu SMTP, en què cada servidor de correu es connecta directament amb qualsevol altre per a fer arribar el correu adreçat a aquest servidor.

Usenet

Usenet és un sistema que, sense utilitzar un control centralitzat, copia fitxers entre ordinadors.

Al final dels vuitanta i el començament dels noranta les arquitectures client-servidor es comencen a imposar com la manera més ràpida i econòmica per a donar suport a grans quantitats d'usuaris no tècnics. Un dels avantatges d'aquestes arquitectures és que també permetien que s'utilitzessin PC, que eren ordinadors menys potents i més econòmics.

L'any 1994 es produeix l'explosió d'Internet, que la va canviar radicalment. Milions de persones s'hi incorporen per enviar correu electrònic, llegir pàgines web i comprar. Internet deixa de ser un reducte de científics per a convertir-se en un fenomen cultural de masses. Una de les principals conseqüències és que canvia radicalment la seva arquitectura: l'ús de tallafocs es generalitza; molts dels usuaris, cada cop que es connecten, utilitzen una adreça IP diferent; molts dels usuaris es connecten amb mòdems o utilitzant connexions asimètriques (com l'ADSL o el cable). L'arquitectura que s'imposa en aquesta nova etapa és l'arquitectura web, basada en el model client-servidor. Un tercer concepte també molt important és que Internet es converteix en una oportunitat de negoci, fet que fa accelerar encara més aquest creixement. Aquest caràcter comercial d'Internet es construeix al voltant de la centralització que ofereix l'arquitectura client-servidor del web.

Al final dels noranta ja hi ha molta gent connectada a Internet. En aquest moment es comença a estendre l'ús d'aplicacions que necessiten una arquitectura diferent de la que ofereix el model client-servidor. El primer exemple clar és la missatgeria instantània. Però és durant l'any 2000 que la gent comença a prendre consciència que hi ha alguna cosa que està canviant. Apareix el Napster i, amb poc temps, es popularitza com a manera d'intercanviar directament fitxers MP3 entre els usuaris finals d'Internet. El Napster no és sinó l'exemple més popular d'una nova manera descentralitzada, i autoorganitzada de fer sistemes distribuïts, que s'escalen perfectament a milions de nodes, que intercanvien, comparteixen o emmagatzemen informació i usant només els recursos dels mateixos usuaris.

Un altre model que tractarem en aquest mòdul són els *grids*. Es comença a parlar de *grids* a partir de la segona meitat dels noranta. Com en el cas d'igual a igual, el *grid* és una conseqüència de l'augment substancial en el rendiment dels ordinadors personals i de les xarxes que hi hagut en els darrers deu o quinze anys. El repte dels *grids* és crear una infraestructura computacional aprofitant els recursos (ordinadors, magatzems de dades, instruments científics, bases de dades, etc.) que poden aportar les diferents institucions que participen en el *grid*. D'aquesta manera, amb la combinació de tots aquests recursos es poden resoldre problemes utilitzant més recursos dels que es tenen individualment. De la mateixa manera, en el món comercial i social també apareix l'oportunitat de fer servir el concepte de *grid* de manera que la capacitat de computació, emmagatzemament i els serveis (aplicacions i la seva llicència d'ús) no s'hagi de comprar sinó que es pugui obtenir quan fa falta un proveïdor extern, i es pugui pagar per ús en lloc de per propietat. Això fa que computació, emmagatzemament i serveis es puguin adaptar a les necessitats.

Acabem aquesta introducció incidint en el fet que les aplicacions o sistemes informàtics cada cop més estan formats per components que es troben arreu d'Internet. És important de conèixer diferents maneres d'organitzar la interacció entre aquests components per tal que el sistema es comporti de la manera que més ens interessi. En aquest mòdul veurem, a part del model client-servidor, del model d'igual a igual i del *grid*, diferents maneres d'organitzar els components que formen una aplicació distribuïda a escala d'Internet. Les classificacions sempre són complicades de fer, sobretot tenint en compte que moltes vegades els sistemes existents no segueixen al cent per cent cap dels models genèrics, sia perquè implementen més d'un model, sia perquè implementen un model que és híbrid. Si som conscients d'això, creiem que la classificació que aquí presentem pot ser útil per a ajudar el dissenyador d'una aplicació distribuïda a l'hora de decidir com ha de ser l'arquitectura de l'aplicació.

Objectius

Els objectius que ha d'assolir l'estudiant amb aquest mòdul didàctic són els següents:

1. Conèixer les característiques que ha de tenir un sistema distribuït a escala Internet.

2. Conèixer diferents arquitectures d'aplicacions distribuïdes a escala Internet.

3. Conèixer amb més detall les arquitectures més usades actualment.

4. Conèixer les diferents topologies en què es basen els sistemes distribuïts.

5. Conèixer les combinacions d'aquestes topologies que més s'usen a l'hora de dissenyar sistemes distribuïts.

1. Conceptes previs de sistemes distribuïts

Hi ha moltes definicions de sistemes distribuïts. A continuació en presentarem una que creiem que encaixa amb l'enfocament que s'ha donat a aquest mòdul.

> Un sistema distribuït és 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 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.

Quan es construeix un sistema distribuït, no es persegueix la creació d'una topologia d'interaccions determinada ni l'ús de cap tipus de component determinat. El que es vol és construir un sistema que satisfaci les necessitats de l'aplicació. En aquest mòdul presentarem un seguit de propietats o requeriments que cal tenir en compte a l'hora de dissenyar aplicacions distribuïdes.

Abans de comentar aquests aspectes, però, esmentarem les característiques de l'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 **dispersió geogràfica** d'aquests. La dispersió geogràfica afecta a la manera en què els components del grup perceben les accions que passen al sistema.

> Per exemple, els membres del sistema 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 el sistema ens caldran o no mecanismes per a abordar aquestes situacions.

En el mòdul 2 es veurà en més detall aquesta problemàtica i la seva solució.

Una tercera característica és l'**autonomia** dels diferents ordinadors que formen el sistema informàtic. És molt habitual que diferents parts d'un sistema distribuït estiguin administrades per diferents administradors. Relacionada amb aquesta característica hi ha la **seguretat**. Cada organització té unes polítiques de seguretat diferents, com ara tallafocs. Cal que els sistemes distribuïts que dissenyem es puguin executar tenint en compte aquestes restriccions imposades pels diferents administradors.

Una quarta característica és la **qualitat de servei**. En un sistema distribuït a escala Internet aquest és un aspecte fonamental que vindrà 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; qualitat d'accés menor (amplada de banda, fiabilitat...). Aquest darrer aspecte, però, millorarà molt a mesura que aquests serveis es generalitzin.

Un cop vistes les característiques de l'escala Internet, veurem uns quants aspectes que cal tenir en compte a l'hora de dissenyar un sistema distribuït:

- **Heterogeneïtat**: el sistema distribuït pot estar format per una varietat de diferents xarxes, sistemes operatius, llenguatges de programació o maquinari de l'ordinador o del dispositiu. Cal que, tot i aquestes diferències, els components puguin interactuar entre si.

- **Obertura**: és desitjable que el sistema es pugui ampliar. Per ampliar entenem que es puguin afegir nous recursos i serveis compartits i que aquests es puguin posar a disposició dels diferents components que formen el sistema o aplicació.

- **Seguretat**: Els sistemes distribuïts de gran escala contenen informació valuosa per als seus usuaris. Per tant, la seva seguretat és molt important. La seguretat de la informació té tres components: confidencialitat (protecció davant d'accessos per individus no autoritzats); integritat (protecció contra alteració o corrupció); i disponibilitat (protecció contra interferències amb la intenció d'accedir al recurs). Un aspecte crític relacionat amb la seguretat és saber la identitat dels usuaris o altres agents que intervinguin en el sistema distribuït. Es poden usar tècniques de xifratge per a proporcionar una protecció adequada dels recursos compartits i mantenir secreta, mentre es transmet per la xarxa, la informació que es cregui oportuna.

- **Escalabilitat**: Es diu que un sistema és escalable si es manté efectiu quan hi ha un increment significatiu en el nombre de recursos i el nombre d'usuaris.

Internet és un exemple de sistema distribuït que ha crescut moltíssim, tant en el nombre d'ordinadors com de serveis.

Data	Ordinadors	Servidors web
Desembre 1979	188	0
Juliol 1989	139.000	0
Juliol 1993	1.776.000	130
Juliol 1995	6.642.000	23.500
Juliol 1997	19.540.000	1.203.096
Juliol 1999	56.218.000	6.598.697

Per a garantir l'escalabilitat cal:

a) controlar el cost d'afegir nous recursos físics (per exemple, si en un futur hi ha el doble d'usuaris, que n'hi hagi prou amb el doble de servidors);

b) controlar la pèrdua de rendiment (que el fet d'afegir nous recursos no faci que el rendiment del sistema se'n ressenti);

c) evitar que els recursos de programari s'esgotin (per exemple, quan es van crear les adreces Internet es van fer de 32 bits. Amb el ritme actual de demanda d'adreces Internet s'esgotaran amb poc temps. Com a solució, es proposa usar una nova versió del protocol que utilitza adreces de 128 bits);

d) evitar punts que puguin ser colls d'ampolla en el moment que s'executin (així, si hi ha un creixement en la demanda, aquest punt no afecta el rendiment del sistema).

- **Comportament davant de fallades del sistema**: un sistema pot fallar. En el cas dels sistemes distribuïts pot ser que falli un component o part d'un component, però la resta del sistema pot ser que continuï funcionant. Les tècniques per a gestionar les fallades són les següents:

  a) detectar si hi ha alguna fallada;

  b) ser capaç de funcionar com si no passés res (per exemple, algun component del sistema va registrant què passa i quan es resol la fallada, s'encarrega de comunicar al component que fallava tot el que no se li ha pogut comunicar);

  c) tolerar les fallades (no cal que s'emmascarin totes les fallades. En molts casos és bo que un component sàpiga que un servei no funciona. D'aquesta manera, en comptes de quedar-se esperant que el servei funcioni, pot decidir utilitzar un altre servei o fer alguna altra cosa);

  d) recuperar-se d'alguna fallada (cal mantenir l'estat de manera que es pugui recuperar després d'una fallada);

  e) la darrera tècnica que comentarem és la utilització de la redundància per a minimitzar l'efecte de les fallades del sistema.

- **Concurrència**: els diferents components d'un sistema distribuït poden demanar d'accedir a un recurs simultàniament. Cal que el sistema estigui dissenyat per permetre-ho.

  Exemple

  Es pot donar el cas en què dos components llegeixin una dada compartida, la modifiquin localment i tornin a escriure el resultat de la modificació a l'espai compartit. En molts casos

ens interessarà que primer llegeixi un component, modifiqui localment la dada i després l'escrigui. El segon component ha d'esperar que el primer hagi acabat la seva operació; si no és així, el resultat del seu càlcul pot ser erroni, ja que no utilitza la dada que ha modificat el primer component sinó la dada inicial.

- **Transparència**: la transparència procura que certs aspectes del sistema siguin invisibles a les aplicacions (actuen però les aplicacions no els veuen, no els afecten, per això es diu que són transparents). Es pot aplicar a diferents aspectes:

– **Transparència d'ubicació:** accés a un recurs sense saber-ne la ubicació.

– **Transparència d'accés:** permet que s'accedeixi a recursos locals i remots usant les mateixes operacions.

– **Transparència de concurrència:** permet que diferents usuaris comparteixin recursos de manera concurrent.

– **Transparència de fallades:** que el sistema continuï funcionant tot i que hi hagi alguna fallada d'un component o recurs.

– **Transparència de mobilitat:** que els recursos i els usuaris es puguin moure pel sistema sense que afectin el seu funcionament.

– **Transparència de reproducció:** que hi hagi més d'una instància d'un recurs.

– **Transparència de rendiment:** permet que a mesura que la càrrega varia el sistema es reconfiguri per tal millorar el rendiment.

– **Transparència d'escalabilitat:** permet al sistema i les aplicacions augmentar l'escala sense canvis en l'estructura del sistema o els algorismes de les aplicacions.

A l'hora de construir un sistema distribuït és important trobar un equilibri entre un nivell alt de transparència i el rendiment del sistema. Per exemple, la majoria d'aplicacions a Internet intenten repetidament contactar amb un servidor abans d'abandonar. Aquest intent d'amagar la fallada transitòria d'un servidor abans d'intentar-ne un altre pot alentir el funcionament global del sistema. La conclusió que cal treure'n és que la transparència és bona quan es dissenya i implementa un sistema distribuït, però cal considerar-la conjuntament amb altres característiques com ara el rendiment.

2. Client-servidor*

* En anglès: *client/server.*

Aquesta arquitectura és la que estem més acostumats a utilitzar en entorns distribuïts. Històricament ha estat la més usada i encara ho és avui en dia. El Web és un exemple d'arquitectura client-servidor.

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.

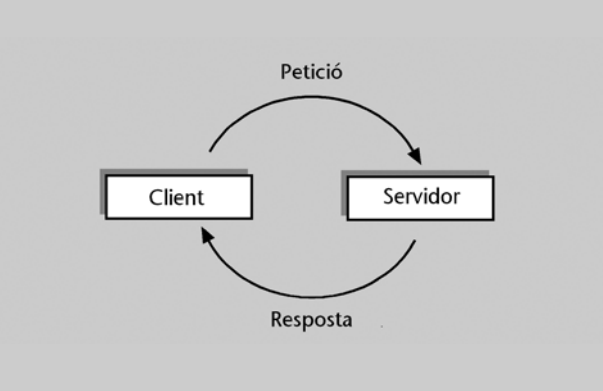

Figura 1

Un servidor també pot ser client d'altres servidors, tal com es veu en la figura 2. Per exemple, una aplicació de correu via web actua com a servidor pel 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 lloc 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.

DNS (*domain name system,* 'sistema de noms de domini') tradueix noms de domini d'Internet a adreces IP.

Per exemple, tradueix el nom de domini www.elMeuNegoci.com a l'adreça d'Internet 196.156.34.2.

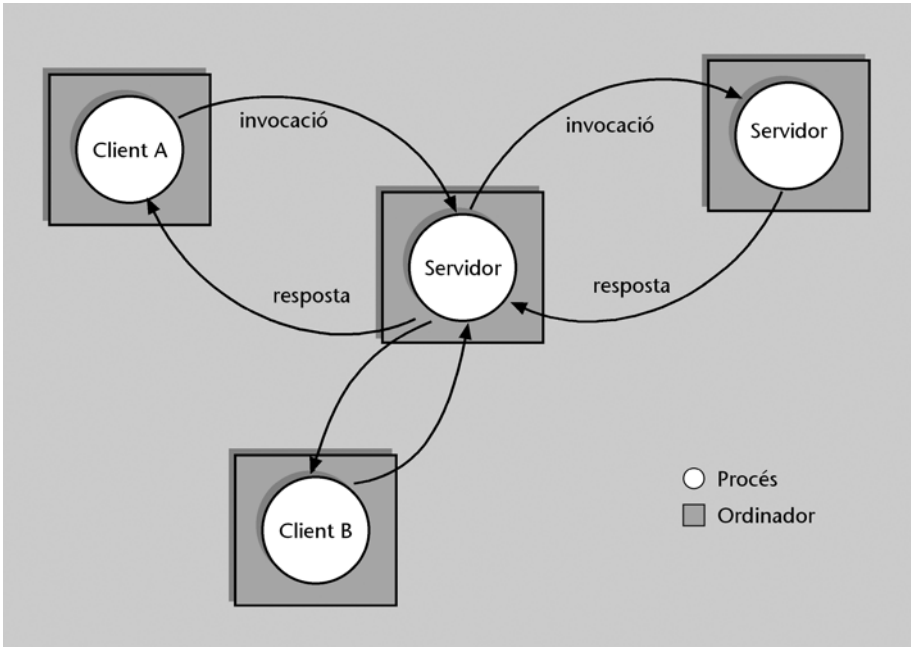

Figura 2

Serveis proporcionats per múltiples servidors

Els serveis 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 (figura 3). Els servidors es poden repartir els diferents objectes que componen el servei que proporcionen o poden mantenir rèpliques dels objectes en diversos ordinadors.

El web proporciona un exemple habitual de particionament de les dades, on cada servidor gestiona el seu conjunt de recursos. Un usuari utilitza un navegador per a accedir a un recurs situat a 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 a diferents servidors que tenen dades reproduïdes.

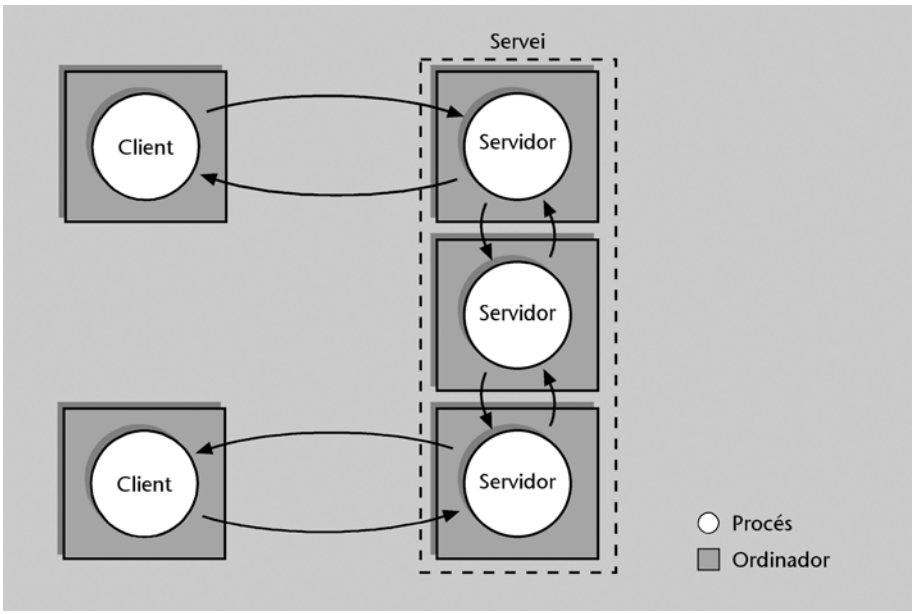

Figura 3

Arquitectura multiestrat*

* En anglès: *multitired architecture.*

En les arquitectures multiestrat, la funcionalitat està distribuïda entre diverses plataformes o ordinadors. La interfície resideix en l'ordinador de l'usuari, els serveis funcionals poden estar en un o més ordinadors, i les dades o els sistemes propietaris estan en plataformes addicionals. És molt habitual parlar d'arquitectures multiestrat en la bibliografia relacionada amb les arquitectures de sistemes d'informació. Les arquitectures multiestrat més habituals són la de dos estrats i la de tres estrats.

L'**arquitectura de dos estrats** està formada per tres components distribuïts en dos nivells (nivell client i nivell servidor). Els tres components són els següents:

En anglès...

... l'arquitectura de dos estrats s'anomena *two tier software architecture,* l'arquitectura de tres estrats s'anomena *three tier software architecture.*

1) Interfície usuari del sistema
2) Capacitat de processament
3) Gestió de dades (servei de dades i fitxers)

La interfície d'usuari està ubicada al client. La gestió de la base de dades està ubicada al servidor. La capacitat de processament està repartida entre el client i el servidor. La figura 4 mostra l'arquitectura de dos estrats.

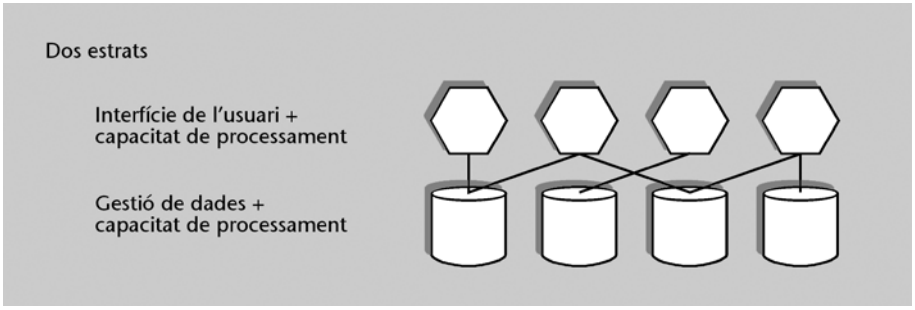

Figura 4

L'arquitectura en dos estrats millora la usabilitat, l'escalabilitat i la flexibilitat de les aplicacions.

L'**arquitectura de tres estrats** és una evolució de l'arquitectura de dos estrats i ubica el tercer estrat entre la interfície d'usuari (client) i el gestor de dades (servidor). Aquest tercer estrat proporciona la capacitat de processament. En la figura 5 es mostra aquesta arquitectura.

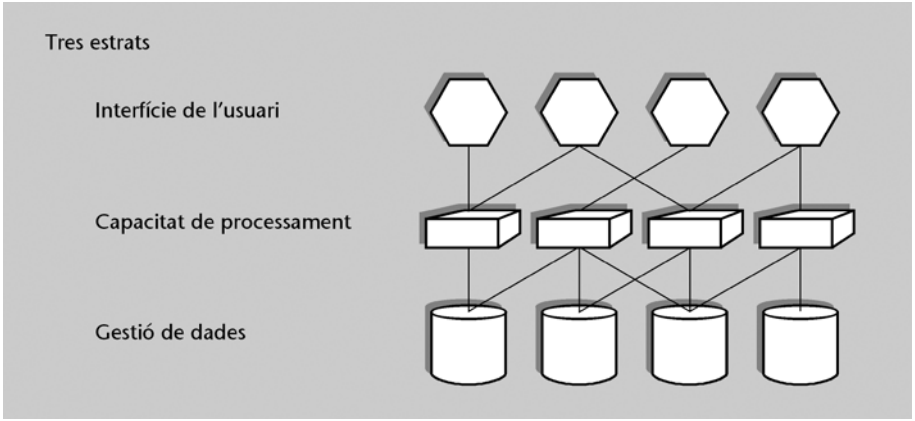

Figura 5

Comparant-la amb l'arquitectura de dos estrats, l'arquitectura de tres estrats millora el rendiment, la flexibilitat, la facilitat de manteniment de l'aplicació, la reusabilitat i l'escalabilitat.

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 a executar (per exemple, CGI o Servlet). El navegador web proporciona a l'aplicació un entorn on 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 en 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.

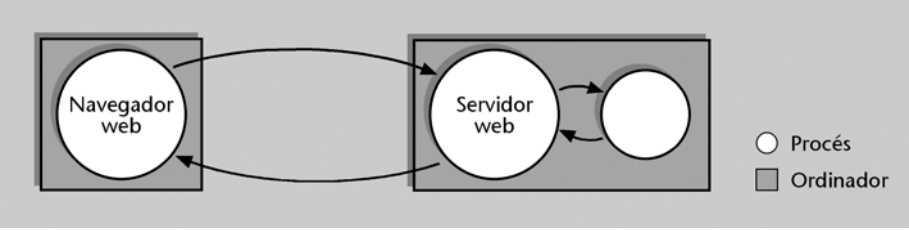

Figura 6

Servidors intermediaris* i memòries cau**

* En anglès: *proxy*.

** En anglès: *cache*.

El model client-servidor es dota d'un conjunt de mecanismes per a millorar la seva disponibilitat, accessibilitat i temps de resposta. Aquests mecanismes són les memòries cau i els intermediaris.

Una memòria cau és un magatzem d'objectes usats recentment que actua com a mediadora entre un client i un servidor. Quan un ordinador rep un objecte, l'emmagatzema a la memòria cau. Quan el client demana per un objecte, primer comprova si és a la memòria cau. Si aquesta còpia està actualitzada (normalment el protocol entre client i servidor inclou una comanda per a validar si una còpia que hi ha a la memòria cau està actualitzada o no), el client obté la còpia de la memòria cau. Si la còpia no està actualitzada, la va a buscar. La introducció de memòries cau té la virtut potencial d'eliminar parcialment o completament algunes interaccions, millorant l'eficiència i la percepció de l'usuari sobre l'eficiència.

Exemples d'ús de memòries cau són els següents:

- *Network file system* (NFS) de Sun Microsystems,
- memòries cau locals dels navegadors. Els navegadors web mantenen en el sistema de fitxers local del client una memòria cau amb les pàgines web o els recursos visitats recentment. Aquests, abans de presentar la pàgina de la memòria cau a l'usuari, comproven al servidor original, utilitzant una petició HTTP especial, si la pàgina està actualitzada.

Cada client pot tenir la seva memòria cau o aquestes poden estar situades en un servidor intermediari que pot ser compartit per diversos clients.

Com es mostra en la figura 4, els servidors intermediaris reben les peticions i les reenvien, amb possibles traduccions, als servidors. Els intermediaris actuen com a servidors compartits per un o més clients. El propòsit dels servidors intermediaris és incrementar la disponibilitat i el rendiment d'un servei a base de reduir la càrrega a la xarxa i en els servidors. En el cas dels servidors web, els servidors intermediaris proporcionen una memòria cau compartida de recursos web per als ordinadors clients. També es poden usar amb altres rols; per exemple, per a accedir a un servidor web remot per mitjà d'un tallafoc.

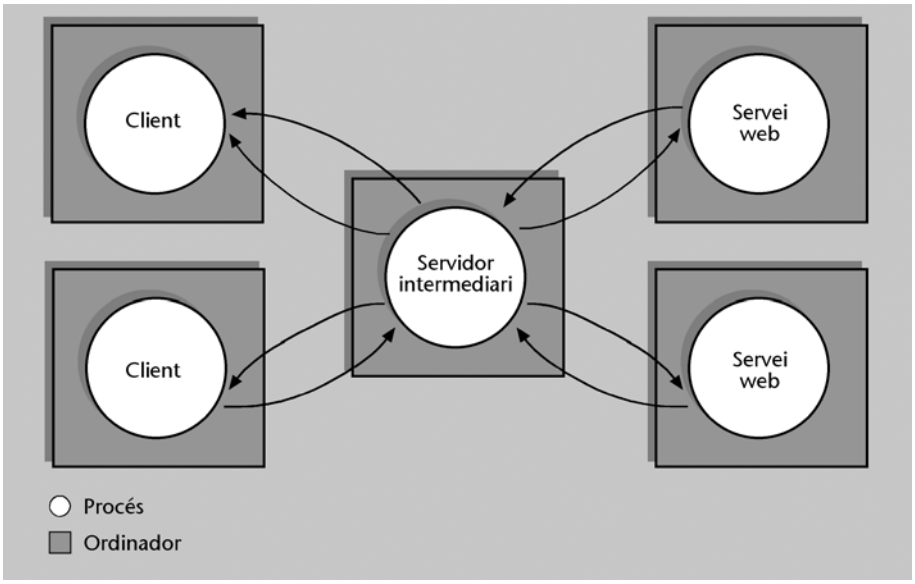

Figura 7

3. Publicació-subscripció*

* En anglès: *publish/subscribe.*

En el model client-servidor que acabem de veure, cada cop que el client necessita alguna informació activament ha de fer una petició al servidor. Aquest és un bon model per a moltes situacions, però en d'altres aquest mètode és poc eficient. Pensem, per exemple, en el cas d'una agència de borsa que vol mantenir informats els seus clients de l'evolució en temps real de les cotitzacions de les accions; o el cas d'una agència de notícies que distribueix informació al moment. En aquests casos, els receptors haurien d'anar consultant contínuament el servidor per tal de tenir la darrera cotització o la darrera notícia, amb la sobrecàrrega que això significa tant a la xarxa com al client i al servidor.

La manera com el paradigma publicació-subscripció aborda aquestes situacions és fent que un productor d'informació anunciï la disponibilitat d'un cert tipus d'informació, un consumidor interessat se subscrigui a aquesta informació i el productor periòdicament vagi publicant informació.

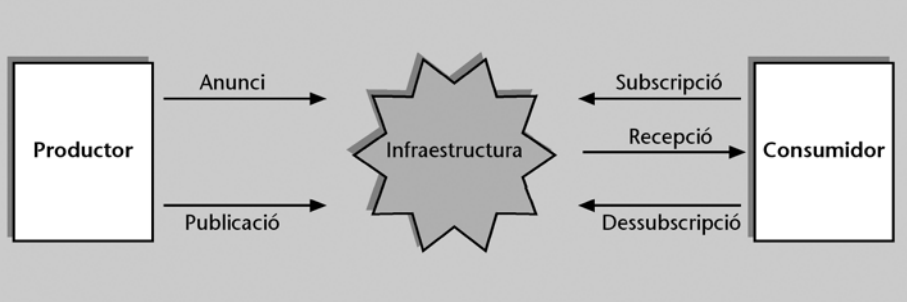

Figura 8

Per tal de poder tenir el comportament descrit, l'arquitectura publicació-subscripció està formada pels components següents:

- **Productor d'informació**: aplicació que té la informació a difondre. El productor publica aquesta informació sense haver de saber qui està interessat a rebre-la. Envia la informació a través de canals.

- **Consumidor d'informació**: aplicació interessada a rebre informació. El consumidor se subscriu als canals que disseminen la informació que li interessa. Rep aquesta informació pels canals a què està subscrit.

- **Mediador (*broker*)**: està entre el productor i el consumidor d'informació. Rep informació dels productors i peticions de subscripció dels consumidors. També s'encarrega d'encaminar la informació publicada als destinataris subscrits al canal. Aquest mediador pot estar distribuït. En aquest cas, cal que els diferents mediadors s'organitzin per tal de proveir els canals.

- **Canal**: són els connectors (lògics) entre els productors i els consumidors d'informació. El canal determina diverses de les propietats de la informació a disseminar i de les funcionalitats suportades: tipus d'informació; format de les dades; possibilitats de personalització del canal per part de l'usuari (per exemple, selecció de continguts, modes d'operació); si el contingut expira o és persistent; estratègia que se seguirà per a fer les actualitzacions; si les dades es lliuren només un cop (en ocórrer, com TV) o si, en canvi, garantim que es pot rebre el contingut independentment del moment en què es va generar; mode d'operació (si es dóna suport pel mode d'operació en desconnectat), pagament (quina és la política de pagament que es fa servir: pagar per veure, per temps, per contingut...).

  Els canals modelen una relació d'un a molts entre productors i consumidors. Habitualment també calen canals perquè els consumidors es puguin relacionar d'un a un amb els productors. Això s'acostuma a fer en un estil client-servidor i, per tant, des del punt de vista conceptual estaria fora del sistema publicació-subscripció.

En la primera de les dues figures que posem a continuació (figura 9) veiem la relació entre els diferents components d'una arquitectura publicació-subscripció. En l'altra figura (figura 10) veiem el detall de com es fa la comunicació entre un productor i un consumidor.

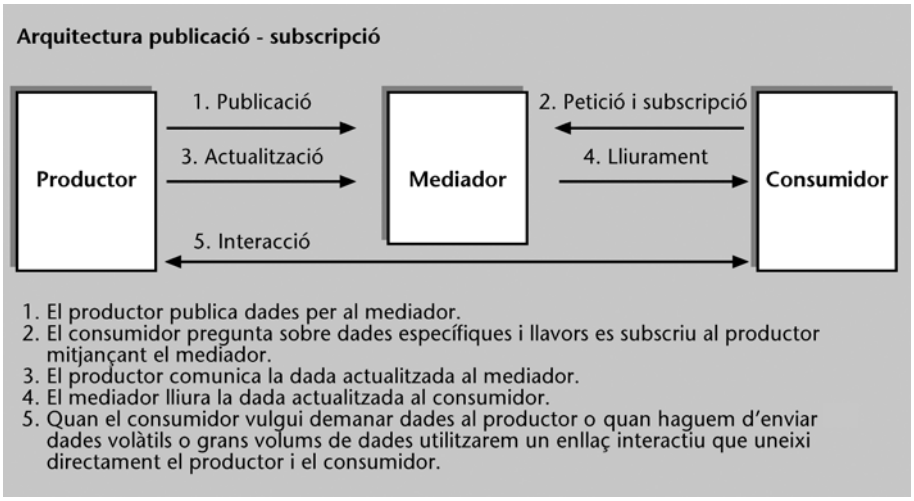

Figura 9

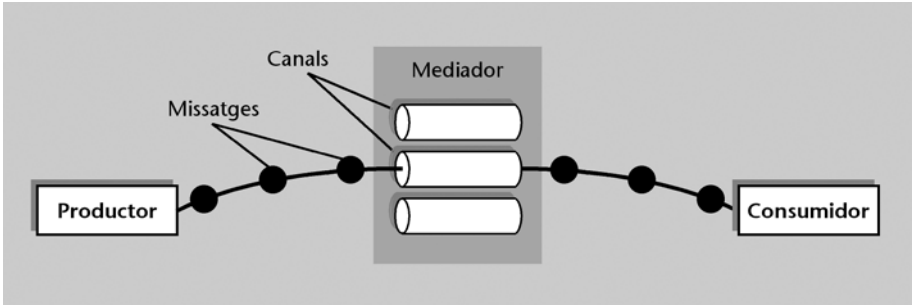

Figura 10

Tal com hem vist, els sistemes publicació-subscripció permeten una distribució asíncrona d'informació. A continuació indiquem algunes situacions i aspectes en què aquests sistemes poden ser una alternativa apropiada:

- **Localització**: per als usuaris és un problema saber on està la informació que els interessa. Encara que hi hagi bones eines de cerca, moltes vegades la informació obtinguda no és de la qualitat desitjada. En els sistemes publicació-subscripció l'usuari se subscriu a uns canals i ara és el proveïdor d'informació qui assumeix el rol actiu de fer arribar la seva informació als interessats.

- **Focalització**: com que l'usuari diu explícitament quines són les seves preferències, és fàcil de proporcionar la informació centrada en els seus interessos.

- **Personalització**: l'usuari pot especificar que, abans que les dades i les seves propietats es lliurin, s'hi apliquin certs requeriments. Per exemple, format de les dades, prioritat, paraules clau, etc.

- **Actualitat**: les dades es poden disseminar a mesura que es tenen disponibles. El proveïdor d'informació pot invalidar les dades obsoletes.

- **Adaptació (*tailoring*)**: el proveïdor també pot decidir quina informació veu el receptor i quina no.

- **Reducció del trànsit**: com que el sistema dissemina la informació a qui està interessat a rebre-la, es redueix molt el trànsit a la xarxa. Intentar localitzar la informació pot provocar molt de trànsit. A més a més, si s'utilitza una infraestructura de transport apropiada, encara es pot reduir més l'ocupació de la xarxa.

Les arquitectures publicació-subscripció estan pensades per a proporcionar tres tipus de serveis: coordinació de processos, reproducció de continguts i informar persones.

Alguns dels camps en què s'utilitzen aplicacions publicació-subscripció són els següents:

- Grups de notícies i llistes de distribució de correu. Els missatges Usenet i les llistes de distribució de correu es poden considerar com a sistemes de publicació-subscripció una mica primitius. Per exemple, els missatges Usenet disseminen articles per tot Internet. Un servidor de missatges se subscriu a altres servidors de missatges i rep els missatges dels grups als quals està subscrit. Quan en un grup es genera un nou article, el servidor on s'ha generat l'article s'encarrega que aquest article es dissemini cap a altres servidors.

- Borsa i notícies: els sistemes que informen sobre l'evolució de les accions a la borsa o les agències de notícies són un altre exemple de sistemes publicació-subscripció. En aquests sistemes els usuaris especifiquen uns interessos i el sistema ha de garantir que els usuaris disposin en tot moment de la informació tan actualitzada com sigui possible.

- Sistemes d'informació de trànsit. Com en les aplicacions per a la borsa i les notícies, cal que la informació s'enviï la majoria de les vegades en temps real. La informació també es distribueix per mitjà d'ordinadors o dispositius mòbils.

- Distribució de programari: Molts sistemes requereixen que el programari s'actualitzi freqüentment. Per exemple, el programari dels bancs d'inversions cal que, per les necessitats de seguretat, s'actualitzi freqüentment i extensivament. Utilitzant una aplicació publicació-subscripció s'aconsegueix que el sistema estigui funcionant contínuament i actualitzat a la darrera versió sense problemes de seguretat per a les actualitzacions.

- Serveis d'alertes, monitoritzacions, vigilància i control.

Alguns exemples d'aplicacions publicació-subscripció són: Castanet, PointCast, BackWeb, WebCasting, WebCanal i Intermind.

4. D'igual a igual*

* En anglès: *peer-to-peer* o *P2P*

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 comportant seguint la filosofia d'igual a igual. Aquests sistemes s'han caracteritzat per ser totalment descentralitzats, de gran escala i amb capacitat per a 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 similars a correus electrònics (que se'ls 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 a finals dels 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.

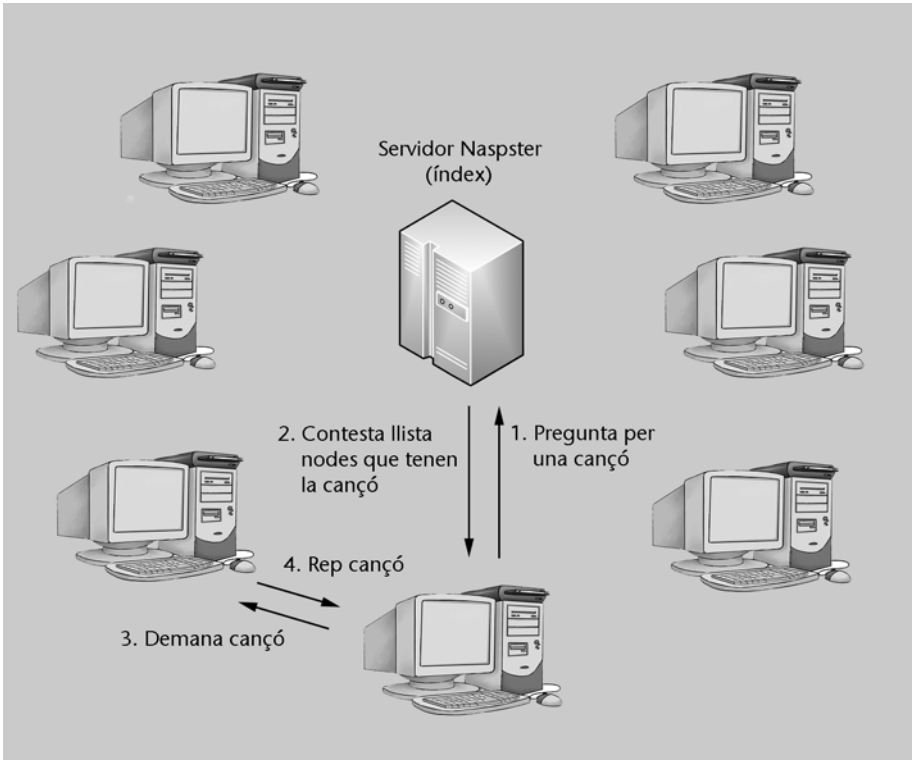

Figura 11

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 mateixos usuaris (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 el 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 ser 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.

Determinisme

Per *determinisme* entenem que diferents execucions d'una mateixa operació donin el mateix resultat. Sistemes 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 la trobi. Pot ser que, segons el camí que hagi seguit la consulta, ens digui que no l'ha trobat, quan sí que hi és.

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 suposa 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.

Característiques

A continuació tractarem les principals propietats que caracteritzen els sistemes d'igual a igual:

- **Descentralització**: els diferents membres (iguals) del sistema són els propietaris i tenen el control de les dades i recursos del seu ordinador. Això complica la implementació dels sistemes d'igual a igual perquè no hi ha una visió global del sistema. Per aquest motiu, alguns sistemes d'igual a igual adopten aproximacions híbrides, com ara Napster o eDonkey, que disponen de directoris centralitzats de fitxers, però els nodes descarreguen els fitxers directament dels iguals que aporten els fitxers.

- **Escalabilitat**: una conseqüència directa de la descentralització és la millora de l'escalabilitat. El fet que les operacions es facin entre els iguals (*peers*) fa que el sistema pugui suportar moltes més operacions que si s'hagués de recórrer a un node que centralitzés les operacions.

- **Autoorganització**: per autoorganització entenem un sistema que va creixent o que va canviant la seva organització sense que aquesta evolució estigui supervisada pel sistema o algun element extern. En sistemes d'igual a igual aquesta autoorganització és necessària ja que acostumen a ser sistemes de **gran escala** (per exemple, és difícil saber quants iguals, usuaris i càrrega hi ha); són **resilients a fallades** (a base de reproduir continguts o recursos); amb **nodes que són accessibles intermitentment** (és difícil que una configuració es mantingui tot el temps); i on la **propietat està repartida entre els diferents iguals** (cada igual és lliure de modificar els recursos que aporta al sistema).

- **Cost de la propietat**: en els sistemes d'igual a igual, la propietat és compartida (els recursos que formen part del sistema els aporten els mateixos participants en el sistema). La propietat compartida redueix el cost de posseir el sistema i els continguts, i el cost de mantenir-lo. Per exemple, en el cas del Napster que hem comentat, el cost del sistema era mantenir el servidor o índex. El cost d'emmagatzemar els fitxers els assumia cada participant en la comunitat Napster, que emmagatzemava els fitxers que aportava a la comunitat.

- **Connectivitat puntual (*ad-hoc connectivity*)**: molts dels iguals que formen un sistema no estan connectats tota l'estona. Es connecten per a fer unes activitats concretes i es tornen a desconnectar. Aquest tipus de connectivitat afecta molt els sistemes d'igual a igual. Per exemple, fent servir un programa de compartició de fitxers tipus eMule, pot passar que el node que està aportant un fitxer deixi d'estar connectat. En aquest cas l'eMule continua de manera transparent (sense que l'usuari hagi de fer res) baixant-se el fitxer d'un altre node. En cas que només hi hagi un node amb el fitxer i aquest es desconnecti, el client eMule intenta baixar-se el fitxer més tard. El mateix passa en els altres tipus d'aplicacions d'igual a igual. Una manera d'abordar aquesta situació és per mitjà de la reproducció i la disponibilitat d'intermediaris.

- **Rendiment**: el rendiment acostuma a ser una preocupació en els sistemes d'igual a igual. Aquests sistemes milloren el rendiment a base d'agregar capacitat d'emmagatzematge distribuït o capacitat de processament en dispositius repartits arreu de la xarxa. Hi ha tres elements clau a l'hora de millorar el rendiment: reproducció, ús de memòries i encaminament intel·ligent.

  - **Reproducció**: la reproducció apropa còpies d'objectes o fitxers als iguals. D'aquesta manera es redueix la distància de connexió entre els iguals que fan peticions i els que proporcionen els objectes. En aquests casos és molt important el mecanisme utilitzat per a actualitzar totes les rèpliques quan hi ha canvis en un objecte.
  - **Ús de memòries cau**: redueix la distància necessària per a obtenir un objecte.
  - **Encaminament intel·ligent**: per tal de treure el màxim rendiment d'un sistema d'igual a igual és molt important entendre com es relacionen els diferents iguals del sistema. D'aquesta manera cal garantir que els iguals que es comunicaran freqüentment o que tenen interessos comuns estiguin propers pel que fa a la capacitat de connectar-se. També és important garantir que s'obtenen les dades desitjades en el menor nombre de salts, és a dir, preguntant el menor nombre possible de nodes. Hi ha casos en què això no es pot garantir. Per exemple, Gnutella fa servir un mecanisme d'inundació.

- **Seguretat**: la majoria de problemes dels sistemes d'igual a igual relacionats amb la seguretat són els mateixos que tindria qualsevol sistema distribuït: cadena de confiança entre iguals i objectes compartits, com s'intercanvien les claus, xifratge, resum (*digital digest*, en anglès) i signatures. A part d'aquests, els sistemes d'igual a igual tenen uns problemes específics deguts a la pròpia arquitectura i filosofia de disseny del sistema. Els problemes són els següents:

  - **Xifratge multiclau**: els sistemes que intenten proporcionar protecció d'accés a l'objecte, i també anonimat d'autor, publicador i igual que l'emmagatzema, necessiten disposar de diferents claus.
  - **Seguretat del codi**: la distribució de codi a executar en sistemes d'igual a igual comporta dos tipus de seguretat: que el codi no modifiqui coses de l'ordinador on s'executa (*sandboxing*), i que l'igual no obtingui dades confidencials (o no públiques) del codi que està executant.
  - **Gestió dels drets digitals**: els sistemes d'igual a igual faciliten la còpia de fitxers. Cal protegir els drets de la propietat intel·lectual. Una tècnica molt usada són els sistemes de marca d'aigua (*watermarking*).
  - **Reputació i comptabilització**: la reputació mesura com n'és de bo i d'útil un igual. Per exemple, un usuari que aporti fitxers interessants té bona reputació. Els usuaris que només obtenen documents, i no n'aporten, tenen mala reputació. Es poden usar mecanismes de comptabilització per tal de detectar els usuaris que no són cooperatius.

    Un altre aspecte relacionat amb la seguretat és l'**anonimat**. Dintre de la comunitat que construeix sistemes d'igual a igual hi ha uns sectors que estan molt sensibilitats per aspectes relacionats amb l'anonimat. Per *anonimat* entenem l'ús d'un sistema sense haver-se de preocupar de la identitat.

    Per acabar, les polítiques de seguretat que implanten les empreses complica la construcció de sistemes d'igual a igual ja que les aplicacions d'igual a igual necessiten que els seus iguals es puguin comunicar directament. La majoria de les polítiques de seguretat que s'implementen per mitjà de tallafocs restringeixen la capacitat que un node exterior es connecti a un node de dins la xarxa que aïlla el tallafoc. Com que moltes vegades el port 80 (HTTP) sí que està disponible per a rebre peticions exteriors, s'han ideat alguns mecanismes per a permetre la connexió entre un node que està darrere un tallafoc i un node a Internet. El problema es complica més quan els dos iguals que es volen connectar estan darrere d'un tallafoc. En aquest darrer cas, cal que els iguals utilitzin un node intermedi (reflector), que proporciona una connexió entre els iguals.

- **Transparència i usabilitat**: és desitjable que els sistemes d'igual a igual puguin funcionar independentment del dispositiu dels iguals (ordinador, agenda electrònica –PDA–, telèfon mòbil). Un altre tipus de transparència que han de proporcionar és la relacionada amb la mobilitat i la seguretat. Els usuaris mòbils han de poder treballar independentment del lloc des d'on estiguin connectats.

- **Resiliència a fallades (*fault resilience*)**: un dels objectius de disseny dels sistemes d'igual a igual és evitar que si falla un component del sistema, tot el sistema deixi de funcionar. Seria desitjable que, tot i que algun igual deixés de funcionar o que hi hagi particions a la xarxa, els iguals que quedin continuessin col·laborant. Alguns dels sistemes actuals ho fan tenint magatzems que registren temporalment les modificacions o comunicacions, i quan l'igual no accessible torna a estar-ho rep les actualitzacions. Altres sistemes encuen els missatges fins que els iguals estan disponibles.

  Un altre possible problema és la no-disponibilitat dels recursos. La manera de resoldre-ho és reproduint els recursos que es consideri que necessiten una alta disponibilitat. Si aquests recursos són fitxers, reproducció vol dir tenir diverses còpies idèntiques del fitxer en diferents iguals. Si el recurs és capacitat de processament, el que es fa és executar el codi en més d'un igual. Així si un dels iguals deixa d'estar accessible, el resultat s'obté de totes maneres d'un altre igual.

  De tot això es veu que una diferència molt important respecte d'altres arquitectures de sistemes distribuïts és que en els sistemes d'igual a igual la responsabilitat de la gestió del sistema està totalment distribuïda i cal que cada igual l'abordi per tal d'assegurar la disponibilitat del sistema.

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, així com una comparativa entre elles. També trobareu més detall dels sistemes explicats en aquest apartat.

Xarxes superposades estructurades enfront de no estructurades

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.

Estructurats

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 *hash* 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 se'ls 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.

En l'apartat 5 del mòdul "Conceptes de sistemes distribuïts" hi ha una explicació més detallada d'aquest tipus de sistemes.

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, on 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.

Xarxes superposades estructurades

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

Podeu trobar més informació d'aquests sistemes a:

CAN
S. Ratnasamy i altres (2001). "A Scalable Content Addressable Network". *Proc. ACM SIGCOMM* (pàg. 161-72).

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*.

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

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

Coral (http://www.coralcdn.org/) és una xarxa de distribució de continguts experimental tipus d'igual a igual per a fer magatzems intermediaris de contingut web. Coral usa l'amplada de banda d'una xarxa de servidors web intermediaris (*proxy*) que guarden pàgines web en una DHT per tal d'evitar l'efecte *slashdot* i reduir el volum de visites a llocs web i d'altres proveïdors de contingut web amb alta demanda.

Slashdot

L'efecte *slashdot* es dóna quan un lloc web resulta inaccessible per causa de les nombroses visites que rep.

OpenDHT (http://opendht.org/) és un servei públic de DHT (Bamboo) accessible públicament que ofereix les funcionalitats PUT i GET a través d'interfícies SUN RPC i XMLRPC. Diverses aplicacions s'han basat en OpenDHT com *middleware base*.

Les xarxes de distribució de continguts són sistemes que tenen gran quantitat de nodes repartits per Internet i interconnectats entre si que cooperen de manera transparent per a proporcionar continguts als usuaris. En l'apartat amb aquest mateix títol del mòdul d'arquitectura d'aplicacions web hi ha una explicació més detallada d'aquestes xarxes.

No estructurats

Un sistema d'igual a igual que utilitzi una xarxa superposada 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 originat una llista de tots els continguts que encaixen 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 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 12 mostra un exemple de cerca.

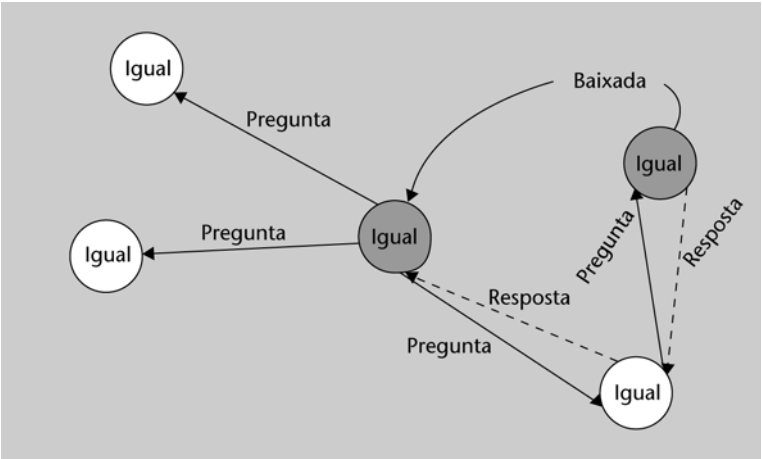

Figura 12

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 13. La comunicació entre els iguals a KaZaA es fa utilitzant el protocol Fast Track, que és un protocol propietari. 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 un *broadcast* als altres superiguals.

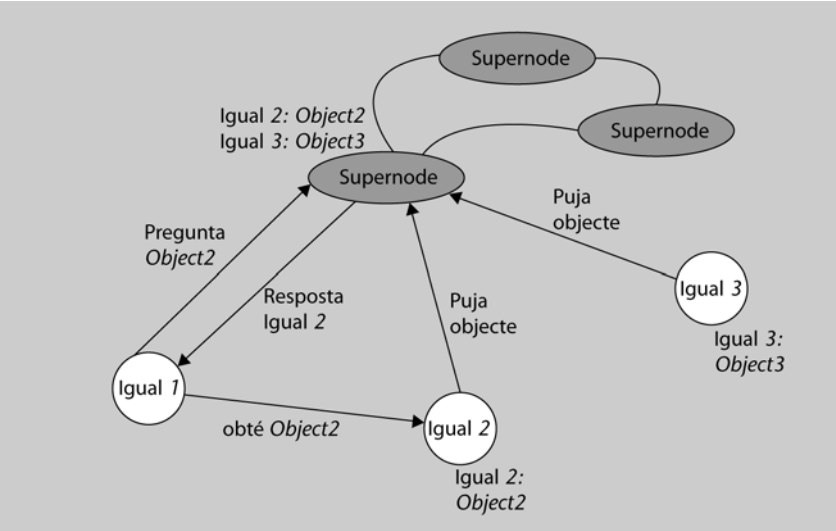

Figura 13

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* (mireu la figura 14) 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 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 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 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 (juny, 2003). "Incentives Build Robustness in BitTorrent". *Proc. First Workshop the Economics of Peer-to-Peer Systems*. Berkeley, EUA.

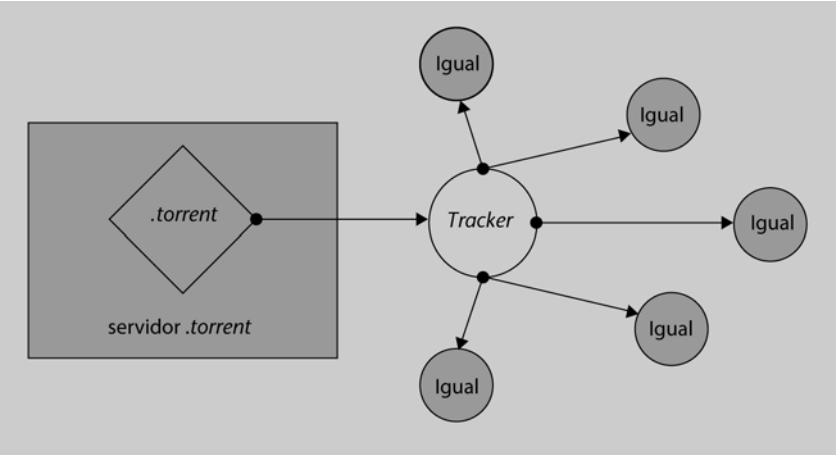

Figura 14

eDonkey: és un sistema d'igual a igual híbrid organitzat en dos nivells per a l'emmagatzemament 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.

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 tipus d'igual a igual que és molt popular. Skype proporciona telefonia a Internet. Utilitza un protocol propietari de la implementació del qual es coneixen poques coses. Funciona seguint una organització amb superiguals tal com ho fa KaZaA. De fet, Skype va ser fundada pels fundadors de KaZaA. Un aspecte a destacar és que aconsegueix superar els problemes que tenen els iguals quan estan darrere d'un tallafocs o els problemes derivats del NAT (*network address translation*).

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.

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 (http://setiathome.berkeley.edu)

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.

Groove (http://www.groove.net)

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.

5. *Grid*

Quan es parla de *grid* es fa referència a una infraestructura que comporta l'ús integrat i col·laboratiu d'ordinadors, xarxes, bases de dades i instruments científics que són propietat i estan gestionats per diferents organitzacions. Les aplicacions *grid* sovint treballen amb grans quantitats de dades i/o recursos computacionals que requereixen una compartició segura de recursos travessant diferents límits organitzatius o dominis d'administració. Per la seva banda, idealment l'usuari té una visió del *grid* com si fos un únic sistema informàtic ja que aquest li proporciona un accés uniforme als recursos.

En el moment d'escriure aquest mòdul, el concepte de *grid* encara és un concepte obert. Encara cal que passi una mica més de temps per a veure com n'evoluciona la implantació, tant en usos científics com comercials i domèstics. Tot i això, les definicions següents poden ajudar a entendre millor què es vol dir quan es parla de *grid*:

a) Plaszczak/Wellner defineixen tecnologia *Grid* com "the technology that enables resource virtualization, on-demand provisioning, and service (resource) sharing between organizations".

b) IBM defineix Grid Computing com "the ability, using a set of open standards and protocols, to gain access to applications and data, processing power, storage capacity and a vast array of other computing resources over the Internet. A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across «multiple» administrative domains based on their (resources) availability, capacity, performance, cost and users' quality-of-service requirements".

"IBM Solutions Grid for Business Partners: Helping IBM Business Partners to Grid-enable applications for the next phase of e-business on demand" (http://www-304.ibm.com/jct09002c/isv/marketing/emerging/grid_wp.pdf).

c) Buyya defineix *Grid* com "a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements".

A Gentle Introduction to Grid Computing and Technologies (http://www.buyya.com/papers/GridIntro-CSI2005.pdf).

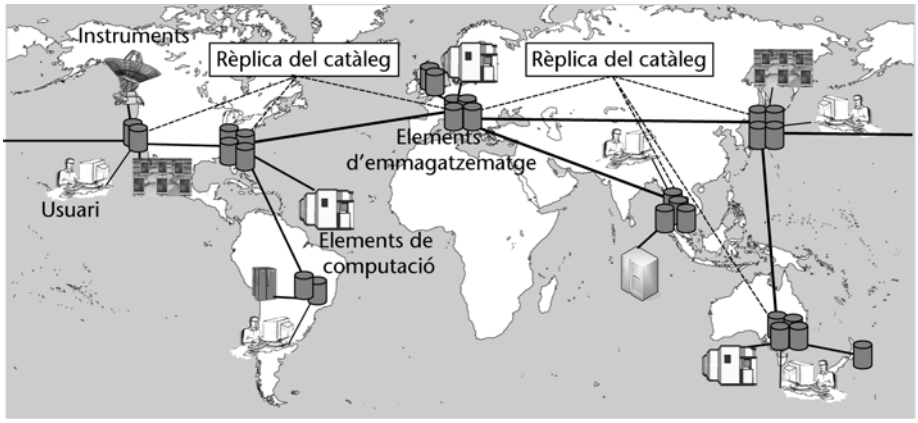

Figura 15. Exemple d'una infraestructura *grid*

La figura 15 mostra l'exemple d'un *grid* on hi ha un instrument que genera dades i uns nodes del *grid* processen aquestes dades. Per a fer-ho, es divideixen les dades en diferents trossos i s'envien als diferents centres. Aquestes dades es combinen amb dades procedents d'altres fonts i que estan repartides en diferents ubicacions. Altres nodes demanen parts d'aquestes dades per processar-les i obtenir-ne resultats. Un cop processades es podrien enviar a un centre de visualització o a l'ordinador d'un usuari perquè examini els resultats. Diferents entitats o institucions que decideixen col·laborar per aconseguir un objectiu determinat aporten els nodes que formen el *grid*. Aquesta col·laboració pot ser conseqüència d'uns acords de col·laboració entre els participants o a canvi de compensacions econòmiques.

Es comença a parlar de *grids* a partir de la segona meitat dels noranta. Com en el cas d'igual a igual, el *grid* és una conseqüència de l'augment substancial en el rendiment dels ordinadors personals i de les xarxes que hi ha hagut en els darrers deu o quinze anys. D'altra banda, gràcies a la combinació entre l'abaratiment dels ordinadors personals i el seu augment de potència, van proliferar sistemes d'altes prestacions a baix cost que van permetre a molts col·lectius disposar de prou potència de còmput per a solucionar problemes que requereixen un ús intensiu de recursos sense haver de disposar de superordinadors. En particular, el món científic s'ha beneficiat d'aquesta potència de còmput per a fer simulacions i experiments molt més exhaustius i per als quals abans calia disposar de grans superordinadors.

Grid

Aquesta infraestructura es va anomenar *grid* fent una analogia amb la xarxa elèctrica (en anglès en diuen *electrical power grid*) que proporciona un accés universal, fiable, compatible i transparent a l'energia elèctrica amb independència del seu origen.

Les xarxes més ràpides han permès compartir dades dels instruments i resultats dels experiments amb col·laboradors d'arreu del món gairebé instantàniament. En aquest context, els *grids* neixen com un pas més en aquest esforç de col·laboració i compartició. El repte és crear una infraestructura computacional aprofitant els recursos (ordinadors, magatzems de dades, instruments científics, bases de dades, etc.) que poden aportar les diferents institucions que participen en el *grid*. D'aquesta manera, amb la combinació de tots aquests recursos es poden resoldre problemes utilitzant més recursos dels que es tenen individualment. Igualment, en el món comercial i social també apareix l'oportunitat de fer servir el concepte de *grid* de manera que la capacitat de computació, emmagatzematge i els serveis (aplicacions i la seva llicència d'ús) no s'hagi de comprar sinó que es pugui obtenir quan fa falta un proveïdor extern, i es pugui pagar per l'ús en lloc de fer-ho per la propietat. Això fa que computació, emmagatzematge i serveis es puguin adaptar a les necessitats: és el model d'*utility computing*.

El model d'*utility computing* es descriu amb més detall al final del mòdul "Arquitectura d'aplicacions web".

Els *grids* són un marc conceptual on hi ha proveïdors i consumidors de recursos i on cal definir de manera molt clara què es comparteix, qui està autoritzar a compartir, i les condicions en què ocorre la compartició. Un conjunt d'individus i/o d'institucions definides per aquestes regles de compartició formen el que s'anomena una *organització virtual*.

El desplegament i la gestió d'aplicacions en els *grids* és una tasca complexa. Això ha fet que s'hagin desenvolupat diferents programaris intermediaris (*middlewares*) *grid* que proporcionen als usuaris la capacitat d'integrar la computació i l'accés uniforme als recursos en un entorn *grid* heterogeni. Aquests programaris intermediaris gestionen la complexitat de la distribució, administració, virtualització, planificació, etc.

Globus Toolkit

Globus Toolkit (http://www.globusconsortium.org/) és un conjunt d'eines de codi obert molt popular entre la comunitat *grid* per a construir infraestructures *grid*.

Arquitectura del *grid*

Els components d'un *grid* es poden organitzar en capes. Cada capa es construeix utilitzant els serveis oferts per la capa inferior així com interactuant i cooperant amb components de la mateixa capa. A continuació descrivim una manera típica d'organitzar una arquitectura *grid* en quatre capes:

- Fàbrica: són els recursos, com ordinadors (poden ser tant *clusters*, com superordinadors o PC, i poden executar diferents sistemes operatius), entorns d'execució, xarxes, dispositius d'emmagatzematge i instruments científics.
- Nucli del *Middleware grid*: ofereix serveis com gestió de processos remots, coassignació de recursos, accés a l'emmagatzematge, descobriment en registre d'informació, seguretat, i suport per a donar qualitat de servei (QoS) com reserva i adquisició de recursos. Abstreu la complexitat i heterogeneïtat del nivell de fàbrica pel fet de proporcionar un mètode consistent per accedir als recursos distribuïts.
- Nivell d'usuari del *Middleware grid*: proporciona abstraccions i serveis de més alt nivell. Aquests serveis poden ser entorns de desenvolupament d'aplicacions i eines de programació.
- Aplicacions *grid* i portals.

Webs recomanades

Alguns entorns d'execució populars en els *grids* són:

- Condor (http://www.cs.wisc.edu/condor/),
- Sun Grid Engine (http://gridengine.sunsource.net/ per a la versió lliure i http://www.sun.com/software/gridware/ per a la comercial)
- Torque (http://www.clusterresources.com/pages/products/torque-resource-manager.php)

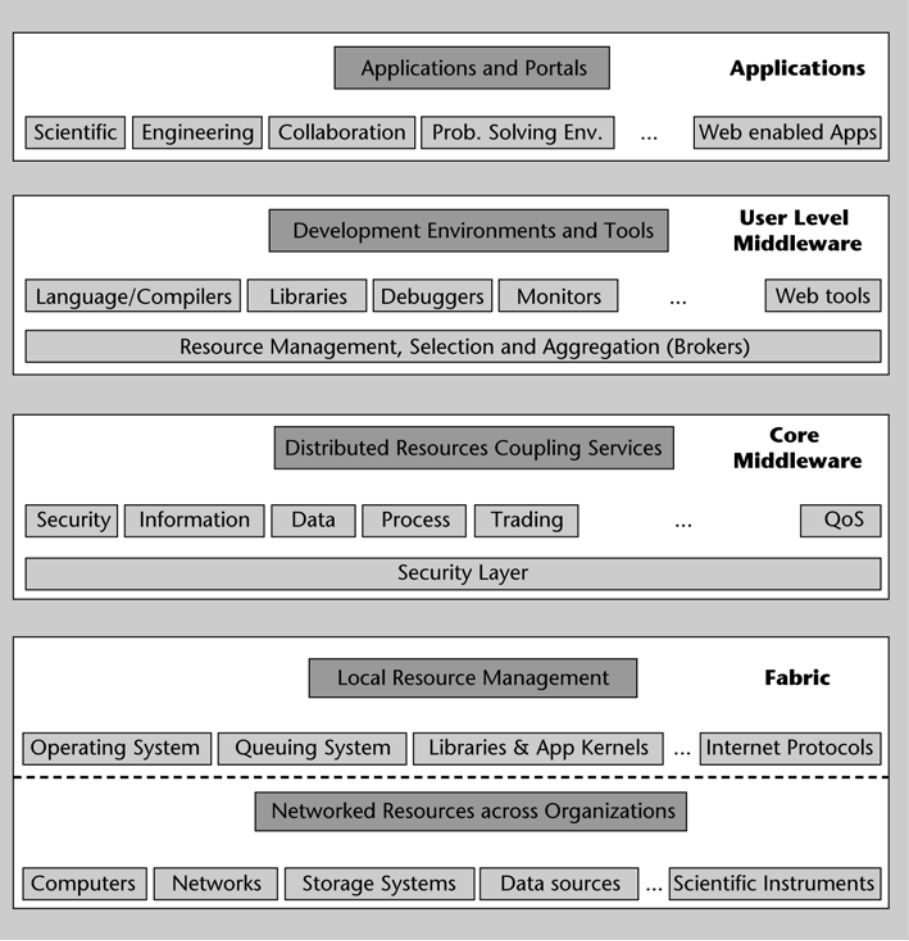

Figura 16

Un aspecte molt important en aquest tipus de sistemes és la interoperabilitat. Això ha fet que actualment dins de la comunitat *grid* hi hagi molts esforços destinats a usar estàndards per tal de facilitar aquesta interoperabilitat.

Exemple d'estàndards basats en *web services*

- OGSA: poden modelar i utilitzar recursos en *web services* (http://www.globus.org/ogsa/)
- WSRF: usa *web services* amb estat (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf i http://www.globus.org/wsrf/)

Webs complementàries

Per a saber una mica més de *grid* vegeu:
- Open Grid Forum (http://www.ogf.org/) i
- Grid café (http://gridcafe.web.cern.ch/gridcafe/)

Globus

Globus és un projecte que proporciona un conjunt d'eines de codi obert que serveixen per a construir infraestructures *grid*. Globus permet la compartició de capacitat de procés, bases de dades, i altres recursos de manera segura tot travessant diferents límits corporatius, institucionals o geogràfics sense sacrificar l'autonomia local. És a dir, els usuaris poden accedir a recursos remots tot preservant el control local sobre qui i quan pot accedir als recursos. La figura 17 presenta l'arquitectura de Globus. Com es pot veure, té tres grups de serveis accessibles a través d'un nivell de seguretat: gestió de recursos, gestió de dades i serveis d'informació.

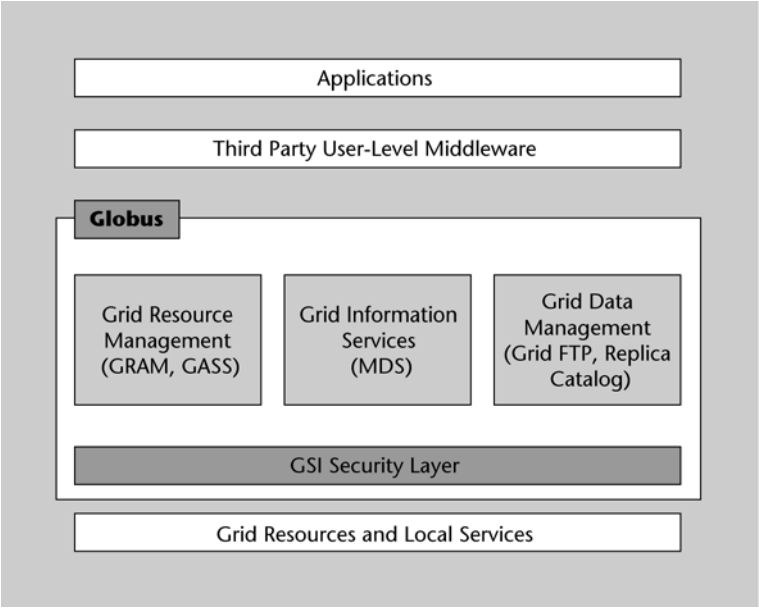

Figura 17

La capa de recursos i serveis locals conté els serveis del sistema operatiu, els serveis de xarxa (p. ex. TCP/IP), i els serveis de planificació de *clusters* –que proporciona, entre d'altres, l'enviament de tasques i la consulta de cues.

La capa que conté el nucli de Globus està format per:

- GSI Security Layer: proporciona els mètodes per autenticar els usuaris i fer les comunicacions segures.

- Grid Resource Management: s'encarrega de l'assignació de recursos, que comprèn l'enviament de treballs a executar, monitorització de treballs i la recollida dels resultats.

- Grid Information Services: proporciona propietats dinàmiques i estàtiques dels nodes que estan connectades al *grid*.

- Grid Data Management: proporciona utilitats i llibreries per transmetre, emmagatzemar i gestionar grans volums de dades que són necessàries per a les aplicacions que s'executen en el *grid*.

La capa que hi ha per sobre conté eines que integren els serveis de la capa inferior o implementa funcionalitats que aquesta no tingui.

6. Sistemes distribuïts basats en esdeveniments*

* En anglès: *event-based systems*

Els sistemes distribuïts basats en esdeveniments aconsegueixen reduir l'acoblament entre els diferents components que formen un sistema a base d'eliminar la necessitat de saber la identitat de la interfície amb la qual s'han de connectar. En lloc d'invocar un altre component directament, un component pot anunciar (difondre) un o més esdeveniments. Altres components del sistema poden registrar que estan interessats en aquest tipus d'esdeveniments i, quan un esdeveniment s'anuncia, el sistema mateix invoca tots els components interessats que estan registrats.

Aquest tipus de tecnologia ha estat molt desenvolupada a nivell de xarxes d'àrea local. En els darrers anys també s'ha convertit en una tecnologia molt apropiada per als sistemes a escala Internet. En aquest apartat ens centrarem a comentar els aspectes més interessants dels sistemes distribuïts basats en esdeveniments a escala Internet.

A escala Internet, a més a més de tenir components poc acoblats, ens trobarem que els components que formen el sistema poden ser molt heterogenis. Una arquitectura que es basi en la generació, l'observació i la notificació d'esdeveniments és molt apropiada.

L'ús d'esdeveniments permet que un objecte pugui reaccionar a canvis que han ocorregut en un altre objecte. La notificació d'esdeveniments és asíncrona i determinada pels receptors (han de mostrar interès a rebre un determinat tipus d'esdeveniment). Els esdeveniments i les notificacions es poden usar en una àmplia varietat d'aplicacions diferents. Per exemple, per a comunicar que s'ha afegit una figura a un dibuix, que s'ha fet una modificació a un document, que una persona entra o surt d'un espai virtual, o que un dispositiu està en una nova ubicació.

De tot això extraiem que els sistemes distribuïts basats en esdeveniments tenen dues característiques principals:

- **Són heterogenis**: fent servir la notificació d'esdeveniments per a comunicar objectes distribuïts aconseguim que components del sistema distribuït que no estan dissenyats perquè interoperin treballin conjuntament. L'únic que cal és que els objectes que generen esdeveniments publiquin els tipus d'esdeveniments que ofereixen, i que els altres objectes se subscriguin a esdeveniments i proporcionin una interfície per a rebre les notificacions.

- **Són asíncrons**: els objectes que generen els esdeveniments els envien asíncronament a tots els objectes que s'han subscrit. Això estalvia als objectes que publiquen esdeveniments haver-se de sincronitzar amb els subscriptors.

Els sistemes dissenyats seguint els principis de les arquitectures basades en esdeveniments són molt apropiats per entorns distribuïts sense autoritat central; per a construir sistemes orientats a components; per a donar suport a aplicacions que han de monitoritzar o reaccionar a canvis en l'entorn, en els interessos per alguna informació o en l'estat de processos.

Molts dels sistemes distribuïts basats en esdeveniments usen el paradigma publicació-subscripció per a disseminar els esdeveniments. Tot i que els sistemes de distribució basats en esdeveniments usin el paradigma publicació-subscripció són dos enfocaments molt diferents a l'hora de construir sistemes distribuïts. Les diferències més significatives són:

a) El propòsit dels sistemes publicació-subscripció és la distribució de dades en el moment apropiat, mentre que els sistemes distribuïts basats en esdeveniments se centren en la notificació d'esdeveniments.

b) Els rols dels participants són molt diferents: en els sistemes publicació-subscripció els productors i els consumidors estan clarament diferenciats, mentre que en els sistemes basats en esdeveniments, tothom pot produir i consumir esdeveniments.

c) El nombre i la freqüència d'informació que es dissemina en els sistemes publicació/subscripció estarà limitada per les ràtios de transmissió dels continguts, i això farà que sigui moderada. En canvi, els sistemes distribuïts basats en esdeveniments, poden arribar a ràtios d'esdeveniments molt elevades.

d) El darrer element diferenciador que esmentarem és que la mida dels continguts transmesos en els sistemes de publicació-subscripció pot arribar a ser gran (ja que són orientats a la informació), en els sistemes d'esdeveniments un dels objectius és que els esdeveniments siguin els més petits possibles.

Els sistemes basats en esdeveniments faciliten l'extensibilitat, la reusabilitat i l'evolució del sistema. L'**extensibilitat** és donada per la facilitat d'afegir un nou component que escolti esdeveniments. La **reusabilitat** gràcies a la potenciació d'una interfície general d'esdeveniments i un mecanisme d'integració. L'**evolució** s'aconsegueix pel fet que es poden substituir components sense que afectin la interfície d'altres components.

Com en tots els models d'arquitectures distribuïdes que veiem en aquest mòdul, un aspecte clau és aconseguir escalabilitat. Un aspecte molt relacionat amb l'escalabilitat en els sistemes distribuïts a escala Internet basats en esdeveniments és l'expressivitat del mecanisme de selecció d'esdeveniments. Per expressivitat volem dir la capacitat del servei de notificació d'esdeveniments de proporcionar un model potent que permeti capturar informació sobre els esdeveniments, expressar filtres i patrons d'interessos de notificació, i usar aquest model com a base per a optimitzar el lliurament de notificacions.

Exemples de sistemes distribuïts...

... basats en esdeveniments: OMG CORBA Event Service, TIB/Rendezvous de TIBCO, JEDI (Java Event-based Distribution Infrastructure), TINA Notification Service, SIENA.

7. Codi mòbil

Els paradigmes de codi mòbil, per la seva banda, pretenen usar la mobilitat per a canviar dinàmicament la distància entre el processament i la font de les dades o la destinació dels resultats. D'aquesta manera, canviant d'ubicació, un component pot millorar la proximitat i la qualitat de les interaccions, reduint el cost de les interaccions i, així, millorar l'eficiència i la percepció de l'usuari sobre el rendiment.

Màquina virtual

La tecnologia de codi mòbil inclou llenguatges de programació i les seves plataformes d'execució. Cal que el codi s'executi controladament en aquestes plataformes o entorns, tant per a satisfer les necessitats de seguretat com per a estar segur que es podrà executar. Això és el que proporciona la màquina virtual.

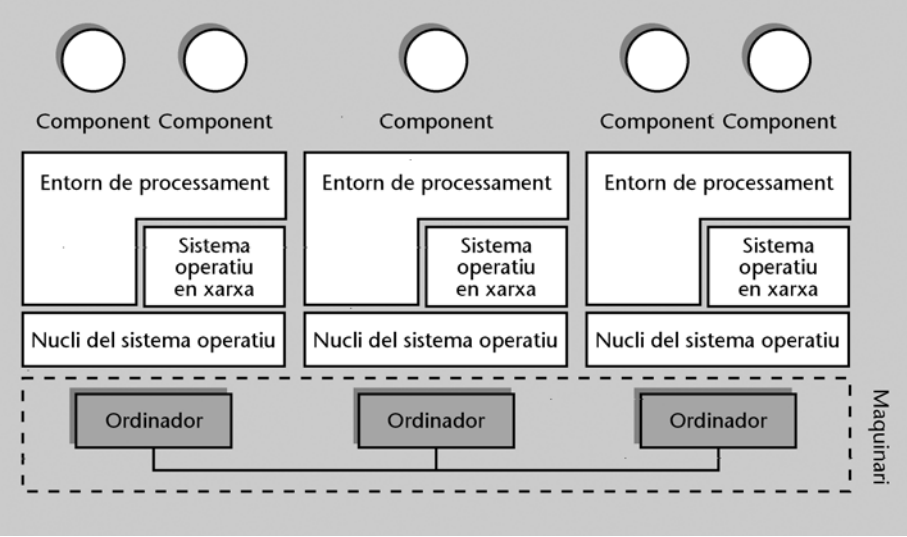

Figura 18

Els llenguatges de *scripting* són els exemples més comuns d'ús de màquines virtuals (per exemple, llenguatges de propòsit general com Perl o llenguatges orientats a tasques com PostScript). Els navegadors web han contribuït a popularitzar les màquines virtuals en introduir la JVM (*Java virtual machine*) com a part de les seves funcionalitats. D'aquesta manera, qualsevol navegador es pot connectar a una pàgina web i, a part de baixar-se text, baixar un programa (miniaplicació*) que s'executarà localment a l'ordinador client.

* En anglès: *applet*.

Paradigmes de codi mòbil

a) Avaluació remota

Exemple

En Pep vol preparar un plat de canelons. Disposa de la recepta, però a casa no té ni els ingredients ni el forn. Sap que la seva amiga Laia té tant el forn com els ingredients a casa seva, però ella no sap com es fan els canelons. Per tal de fer els canelons en Pep truca a la Laia i li dicta la recepta per telèfon. La Laia fa els canelons seguint la recepta d'en Pep i els hi porta.

En aquest paradigma, un client té el coneixement necessari per a realitzar un servei, però no disposa dels recursos (potència de càlcul, dades, etc.) necessaris, que es troben en un ordinador remot. Per aquest motiu, el client envia el coneixement al servidor ubicat en un ordinador remot. Aquest executa el codi amb els recursos que té allà. Els resultats de l'execució es retornen al client. Aquest paradigma d'avaluació remota pressuposa que el codi que es proporciona s'executarà en un entorn protegit, de manera que no impacti altres clients del mateix servidor a part de l'impacte que pugui significar el fet d'haver de compartir recursos. Per això és molt important que el servidor pugui confiar en els clients.

Alguns exemples molt coneguts d'avaluació remota són rsh de Unix, que permet d'executar arxius de comandes (*scripts*) en un ordinador remot. Un altre exemple és la interacció entre un processador de textos i una impressora PostScript. En aquest cas, la impressora és el recurs i el codi és el fitxer PostScript. Un intèrpret de PostScript situat a la impressora executa el codi.

Encara que a simple vista pugui semblar que l'avaluació remota sigui un cas particular de client-servidor, la diferència és significativa. En el paradigma client-servidor, un servidor posa a disposició dels clients un conjunt limitat de funcionalitats que aquests poden invocar. En el cas de l'avaluació remota, l'ordinador que executa el codi remot ofereix un servei programable amb un llenguatge de programació complet.

Un altre exemple són els cucs (*worms* en anglès), que són uns tipus de virus informàtics en què un programa envia còpies d'ell mateix a altres nodes.

b) Codi sota demanda

Exemple

En Pep vol preparar un plat de canelons. A casa té tant els ingredients com el forn, però no disposa de la recepta. Sap que la seva amiga Laia disposa de la recepta. En Pep li truca i la hi demana. La Laia li dicta la recepta i en Pep prepara els canelons a casa seva.

El paradigma de codi sota demanda es dóna quan un client té accés a un conjunt de recursos, però no té el coneixement necessari per a processar-los. Per tal de poder fer l'execució, el client envia una petició a un servidor remot perquè li enviï el codi necessari. Un cop rebut el codi, l'executa localment.

Entre els avantatges de codi sota demanda hi ha la possibilitat d'afegir funcionalitats a un client sense haver-lo de modificar. A més, l'execució local pot proporcionar un bon nivell d'interactivitat, ja que no es pateixen ni els retards ni la variabilitat d'amplada de banda associada a les comunicacions en xarxa. Això no vol dir que el codi baixat hagi d'interactuar únicament amb codi local. En moltes situacions es comunicarà amb altres programes escampats a Internet. En aquest paradigma és molt important que el client pugui confiar en els servidors d'on es baixa el codi.

Els exemples més coneguts de codi mòbil són les miniaplicacions –quan l'usuari selecciona (al navegador web que està utilitzant) un enllaç que fa referència a una miniaplicació (que està emmagatzemada en un servidor web), el codi es baixa al navegador. Aquest executa la miniampliació localment (vegeu la figura 19).

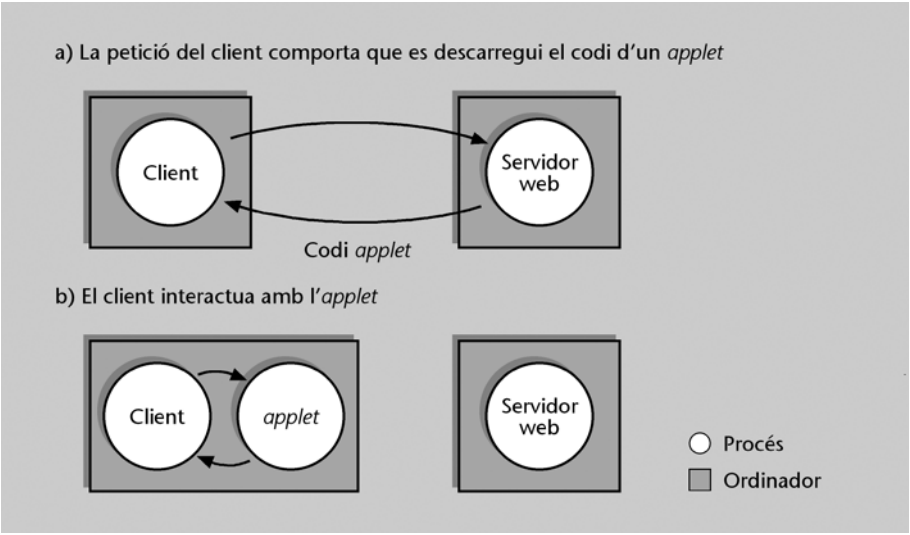

Figura 19

c) Agents mòbils

En Pep vol preparar un plat de canelons. Té els ingredients i disposa de la recepta, però a casa no té forn. Sap que la seva amiga Laia té forn a casa seva. En Pep prepara els canelons i va a casa de la seva amiga Laia a coure'ls.

En el paradigma d'agents mòbils, una unitat de computació es mou a un ordinador remot, emportant-se el seu estat, la part de codi que necessiti i, si és el cas, les dades necessàries per a dur a terme la tasca.

Aquest paradigma és una combinació dels dos anteriors, ja que funciona en tots dos sentits. En comparació amb els altres dos paradigmes, aporta un dinamisme més gran pel fet de poder decidir quan cal moure el codi d'un ordinador a un altre i, així, millorar el rendiment global. Una aplicació pot estar a mig processar una informació en una ubicació i decidir canviar a una altra ubicació per tal de reduir la distància entre el codi i el proper conjunt de dades que vol processar.

Serà interessant utilitzar aquest tipus d'aplicacions quan el volum de dades a processar sigui molt gran i aquestes dades estiguin en ubicacions diferents. En aquestes situacions, resulta més eficient bellugar el codi que processa les dades que portar totes les dades a un lloc –amb el cost de comunicació que això representa.

Com en els casos anteriors, la seguretat és un aspecte important. Un agent mòbil accedirà a dades locals de l'ordinador en què s'executa, per tant, cal que l'entorn confiï en l'agent. D'altra banda, també cal que l'agent es protegeixi contra funcionaments parcials o malfuncionaments de l'entorn en el qual s'executa per tal de no donar resultats incorrectes; o que una aplicació de l'entorn on s'executa l'agent no pugui obtenir dades de l'agent a les quals no té dret d'accés.

8. Topologies dels sistemes distribuïts

Els diferents components d'aquest sistema informàtic s'interrelacionen entre si. En aquest apartat .hem intentat de classificar aquestes maneres d'interrelacionar-se depenent dels fluxos d'informació que intercanvien. Anomenem **topologies** aquestes diferents maneres d'interrelacionar-se.

A continuació presentem les topologies bàsiques que podem trobar en aplicacions i sistemes que s'executen a Internet. També presentem les combinacions d'aquestes topologies que considerem més habituals:

1) Topologies bàsiques

Centralitzada

És la que estem més habituats a veure. És la que utilitzen, per exemple, les arquitectures client-servidor, les aplicacions web, però també algunes aplicacions d'igual a igual que utilitzen algun tipus de servei centralitzat o arquitectures publicació-subscripció.

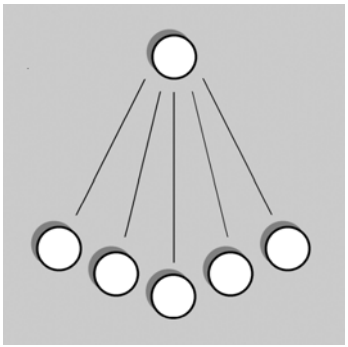

Figura 20

En aquestes arquitectures, un node servidor conté la informació i els nodes clients obtenen la informació d'aquest servidor (tant sigui activament demanant-la al servidor o passivament esperant que aquest els l'enviï).

Els avantatges dels sistemes centralitzats són la simplicitat d'administració i de consistència en les dades. El problema principal és la vulnerabilitat a les fallades.

En anell

Aquesta topologia s'utilitza per a solucionar les situacions en què un servidor està molt saturat perquè té moltes peticions. En aquest cas, consisteix a tenir un conjunt de servidors connectats entre ells de manera que es coordinin per tenir un estat comú. D'aquesta manera s'aconsegueix balancejar la càrrega i obtenir un millor comportament davant les fallades del sistema. Per a assolir una bona efectivitat, cal que els ordinadors que formen part de l'anell estiguin a prop i tinguin una autoritat administrativa comuna.

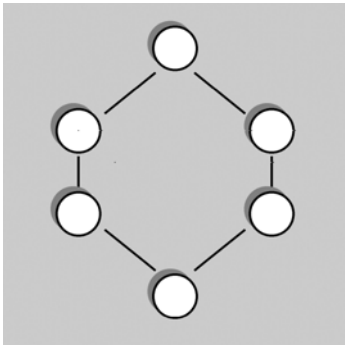

Figura 21

Alguns sistemes estructurats també utilitzen una topologia que es podria considerar com un anell lògic, per exemple, *chord*. Tal com s'ha vist, aquesta organització fa que les cerques siguin eficients pel que fa al nombre de salts necessaris per a localitzar una dada.

Jeràrquica

Les topologies en forma de jerarquia es basen en el fet de tenir els nodes connectats de manera que un conjunt de nodes tenen un node superior que els ofereix la informació que aquests necessiten. Si aquest node no té la informació demanada, la reclama al seu node superior.

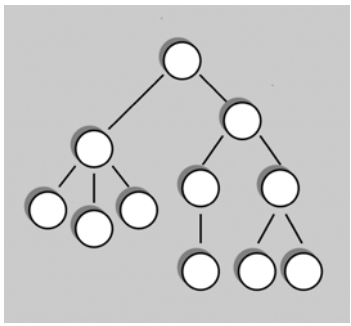

Figura 22

Aquest tipus de topologia s'ha usat molt a Internet. Un exemple és el servei de noms de domini (DNS).

Els sistemes jeràrquics tenen com a avantatge principal la seva escalabilitat, ja que es poden afegir nodes a qualsevol punt de la jerarquia per cobrir les necessitats de càrrega que hi pugui haver. Respecte a les altres propietats, té millor comportament davant de fallades de nodes que els sistemes centralitzats, però l'arrel continua essent un punt únic de fallada. Acostuma a ser més complicat fer-los segurs que els sistemes centralitzats. Com més amunt de la jerarquia es produeix el problema, més se'n ressent el sistema.

Descentralitzada

En aquesta darrera topologia qualsevol node es connecta amb qualsevol altre node. Tots els nodes són iguals entre si. Aquest tipus de topologies són les que fan servir molts dels sistemes d'igual a igual. També l'utilitzen aplicacions clàssiques, com ara els servidors de correu SMTP, per a connectar-se els uns amb els altres.

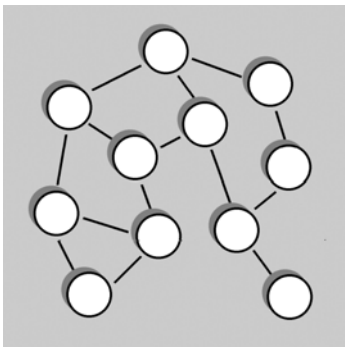

Figura 23

Les virtuts dels sistemes descentralitzats són la seva extensibilitat (és fàcil d'afegir un nou node al sistema) i la tolerància a fallades (la caiguda d'un node no afecta la resta del sistema).

Com a principals inconvenients destaquen la dificultat per a administrar el sistema i que aquest tipus de sistemes acostumen a ser insegurs.

2) Topologies híbrides

El que es fa habitualment quan es dissenya un sistema distribuït és utilitzar combinacions de les topologies que hem vist. Combinant diferents arquitectures es poden aconseguir un nombre il·limitat de topologies. A continuació presentarem les que considerem les més habituals:

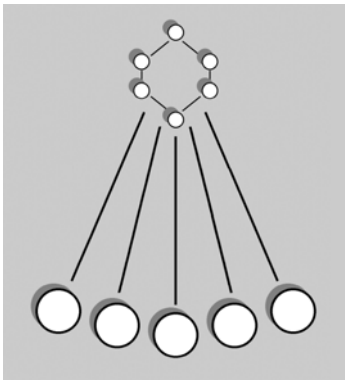

Figura 24

Centralitzada i en anell

Des del punt de vista de l'usuari es comporta com si fos un sistema centralitzat, però el node servidor no és un únic ordinador, sinó un anell format per diferents ordinadors.

Aquesta combinació aconsegueix que l'usuari vegi un únic punt de connexió, però amb els avantatges que ofereix tenir un conjunt d'ordinadors que responen a les peticions dels clients: robustesa, escalabilitat, tolerància a fallades.

Aquesta combinació de simplicitat i potència que s'aconsegueix amb aquesta combinació fa que sigui molt usada en servidors web comercials i en bases de dades que requereixen una disponibilitat molt alta.

Centralitzada i descentralitzada

En certes situacions els sistemes descentralitzats purs tenen l'inconvenient que resulta molt complex garantir que qualsevol node es pugui relacionar amb qualsevol altre node. Aquest tipus de topologia intenta resoldre aquesta mancança especialitzant uns nodes que serveixen per a interrelacionar grups de nodes que estan agrupats de manera centralitzada.

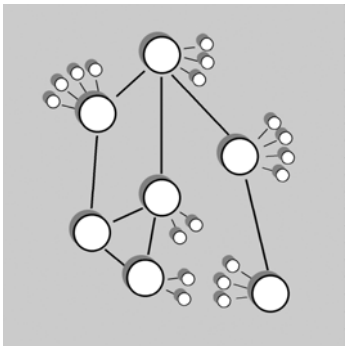

Figura 25

Skype i eDonkey són dos exemples molt populars d'aquest tipus de topologia aplicada a sistemes d'igual a igual. També es podria considerar que el correu d'Internet funciona d'aquesta manera: els clients de correu es connecten amb un servidor de correu de manera centralitzada. Aquest es connecta amb la resta de servidors de correu de manera descentralitzada (és el cas que ja hem comentat en l'apartat anterior).

Els sistemes que segueixen aquesta topologia mantenen els avantatges dels sistemes descentralitzats. El grau de centralització aporta senzillesa a l'hora d'aconseguir coherència dins el sistema, ja que hi ha menys nodes que es coordinen per a donar accés a les dades.

Pel que fa als inconvenients, manté els dels sistemes descentralitzats: dificultat d'administració i seguretat.

Resum

En aquest mòdul hem explicat algunes de les arquitectures més usades a l'hora de dissenyar sistemes i aplicacions distribuïdes a escala Internet. Hem après que les aplicacions que predominaven en els inicis d'Internet seguien un model de l'estil d'igual a igual. També hem vist que el *boom* d'Internet, sobretot a partir del 1994 de la mà del Web, va comportar que s'imposés el model client-servidor. Així hem començat estudiant les variants més habituals de client-servidor. Tot i que aquesta continuï essent l'arquitectura dominant a Internet, hi ha un seguit de situacions que necessiten altres plantejaments. D'aquesta manera hem vist que el model de publicació-subscripció ens és molt útil quan els clients necessiten rebre la informació a mesura que es va produint. En aquesta solució, el servidor és qui pren la iniciativa de fer arribar les dades noves al client. També hem estudiat les diferents variants del codi mòbil. Aquestes tècniques són molt útils per a reduir la distància entre dades i codi. Així, fent que viatgi el codi en lloc de les dades, aconseguim que es minimitzi l'ús de la Xarxa. Un altre avantatge del codi mòbil és que si el combinem amb client-servidor aconseguim d'augmentar les funcionalitats d'un client o un servidor sense haver de reinstal·lar ni reconfigurar res.

Un altre enfocament molt diferent són els sistemes distribuïts basats en esdeveniments. Aquests aconsegueixen reduir l'acoblament entre els components que formen el sistema disseminant esdeveniments entre els seus components. A continuació en recordarem dos dels aspectes principals. El primer aspecte és que permet que els components que el formen siguin molt heterogenis. L'altre és que no cal que el generador d'un esdeveniment sàpiga qui està interessat a rebre'l.

Per acabar les diferents variants d'arquitectures, hem vist el model d'igual a igual i el *grid*. Les arquitectures d'igual a igual s'han popularitzat gràcies als sistemes d'intercanvi de fitxers, però hem vist que hi ha altres tipus d'aplicacions que també estan implementades seguint aquest model. Les aplicacions d'igual a igual comencen a desplaçar el model client-servidor en aplicacions distribuïdes que necessitin molts recursos, que necessitin uns nivells de disponibilitat molt grans, que necessitin anonimat o en aplicacions orientades a la col·laboració.

Dels *grids* hem vist que són sistemes que pretenen que diferents organitzacions comparteixin recursos (ordinadors, magatzems de dades, instruments científics, bases de dades, etc.). També que el món comercial i social es pot beneficiar d'aquesta oportunitat que ofereix el *grid* de manera que la capacitat de computació, emmagatzematge i els serveis (aplicacions i la seva llicència d'ús) no s'hagi de comprar sinó que es pugui obtenir quan fa falta un proveïdor ex-

tern, i es pugui pagar per l'ús en lloc de per la propietat. Del *grid*, n'hem vist una organització típica en quatre capes i l'arquitectura de Globus, que és el conjunt d'eines de codi obert més utilitzat actualment per a construir infraestructures *grid*.

Al començament d'aquest mòdul, en l'apartat 1, hem fet una pinzellada sobre alguns dels aspectes que cal considerar quan es dissenya una aplicació distribuïda: heterogeneïtat, obertura, seguretat, escalabilitat, comportament davant fallades, concurrència i transparència.

En el darrer apartat, hem presentat les topologies bàsiques que podem trobar en els sistemes i aplicacions distribuïdes a escala Internet. Aquestes topologies són: centralitzada, en anell, jeràrquica i descentralitzada. També hem vist que molts dels sistemes utilitzen topologies híbrides, de les quals hem presentat les que considerem més habituals: centralitzada i en anell; i centralitzada i descentralitzada.

Glossari

agents mòbils *m pl* Paradigma de codi mòbil en què una unitat de computació es mou a un ordinador remot, emportant-se el seu estat, la porció de codi que necessiti i, si és el cas, les dades necessàries per a fer la tasca.

arquitectura multiestrat *f* Variant d'arquitectura client-servidor en què la interfície resideix a l'ordinador de l'usuari i els serveis funcionals (capacitat de processament i gestió de les dades) són a plataformes addicionals. Nosaltres hem vist arquitectures de dos i de tres estrats.
en multitired architecture

avaluació remota *f* Paradigma de codi mòbil en què un client envia un codi per executar a un ordinador remot que disposa dels recursos necessaris per a realitzar el servei.

cache *f* Vegeu **memòria cau**.

client-servidor *m* Tipus d'arquitectura distribuïda formada per dos tipus de components. Els clients que fan peticions d'un servei, i els servidors que proveeixen aquest servei.

codi mòbil *m* Conjunt de paradigmes que usen la mobilitat per a canviar dinàmicament la distància entre el processament i la font de les dades o la destinació dels resultats.

codi sota demanda *m* Paradigma de codi mòbil en què un client es baixa d'un ordinador remot el codi necessari per a fer una operació. L'execució es duu a terme a l'ordinador del client.

d'igual a igual *m* Tipus d'arquitectura distribuïda que es caracteritza pel fet que tots els nodes que formen el sistema o aplicació tenen les mateixes capacitats i responsabilitats i on tota la comunicació és simètrica.
en peer-to-peer
sin. entre iguals

distributed hash table **(DHT)** *f* Vegeu **taula de *hash* distribuïda**.

escalabilitat *f* Un sistema és escalable si l'addició d'usuaris i recursos no provoca una pèrdua de rendiment ni un increment de complexitat de l'administració.

esdeveniment *m* Ocurrència d'algun canvi d'estat en un component que es fa visible al món extern.
en event

heterogeneïtat *f* Propietat d'un sistema distribuït que indica que està format per una varietat de diferents xarxes, sistemes operatius, llenguatges de programació o maquinari de l'ordinador o del dispositiu.

igual *m* Cadascun dels nodes que forma un sistema d'igual a igual.
en peer

màquina virtual *f* Entorn segur i fiable que permet garantir que el codi mòbil s'executarà de la manera com s'espera que s'executi.

memòria cau *f* Magatzem d'objectes usats recentment que actua com a mediador entre un client i un servidor.
en cache

Napster *m* Aplicació molt popular per a l'intercanvi de fitxers en format MP3 que seguia el model d'una arquitectura d'igual a igual.

obertura *f* Capacitat que té un sistema distribuït per a ampliar-se (afegir nous recursos i serveis que estiguin disponibles per als components que formen el sistema o aplicació).

overlay network *f* Vegeu **xarxa superposada**.

peer *m* Vegeu **igual**.

peer-to-peer *m* Vegeu **d'igual a igual**.

proxy *m* Vegeu **servidor intermediari**.

publicació-subscripció *f* Tipus d'arquitectura distribuïda formada per consumidors que estan interessats a rebre un tipus d'informació; productors que envien la informació activament; i mediadors que s'encarreguen de posar en contacte productors i consumidors de manera transparent per a ells.

servidor intermediari *m* Servidor que accepta peticions de clients o altres intermediaris i genera peticions cap a altres intermediaris o cap al servidor destinació.
en proxy

sistema distribuït *m* 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.

sistemes distribuïts basats en esdeveniments *m pl* Tipus d'arquitectura distribuïda en què s'aconsegueix reduir l'acoblament entre els components que formen el sistema a base de disseminar esdeveniments entre els seus components.

taula de *hash* distribuïda *f* Mecanismes distribuïts que permeten la localització eficient de dades en sistemes de gran escala a través d'un índex descentralitzat i uniformement repartit entre tots els nodes del sistema.
en distributed hash table
sigla **DHT**

topologia *f* Maneres d'interrelacionar-se que tenen els components d'un sistema distribuït en funció dels fluxos d'informació que intercanvien.

transparència *f* Qualitat que procura que certs aspectes del sistema estiguin ocults a les aplicacions.

xarxa superposada *f* Xarxa a nivell d'aplicació que formen els nodes d'uns sistema o aplicació d'igual a igual. Aquesta xarxa funciona sobre la xarxa física que connecta els nodes.
en overlay network

Bibliografia

Coulouris, G.; Dollimore, J.; Kindberg, T. (2005). *Distributed Systems. Concepts and Design*. Londres: Addison-Wesley [trad. al castellà *Sistemas distribuidos: conceptos y diseño*. 3/E. Pearson, 2001].
És un llibre que tracta els principis i el disseny dels sistemes distribuïts, incloent-hi els sistemes operatius distribuïts. La versió en anglès va per la quarta edició, mentre que la versió castellana va per la tercera edició.

Eng Keong Lua; Crowcroft, J.; Pias, M.; Sharma, R.; Lim, S. (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, així com una comparativa entre elles. També trobareu més detalls dels sistemes explicats en aquest apartat.

Oram, A. (ed.) (2001). *Harneshing the power of Disruptive Technologies*. Editorial O'Reilly.

Tanenbaum A.; Steen, M. (2007). *Distributed Systems: Principles and Paradigms*, 2/E. Prentice Hall.
Aquest llibre és un bona ajuda per a programadors, desenvolupadors i enginyers per tal d'entendre els principis i paradigmes bàsics dels sistemes distribuïts. Relaciona els conceptes explicats amb aplicacions reals basades en aquests principis. És la segona edició d'un llibre que ha tingut molt d'èxit tant pels aspectes que cobreix com pel tractament que en fa.

Steinmetz, R.; Wehrle, K. (eds.) (setembre 2005). *Peer-to-Peer Systems and Applications*. Lecture Notes on Computer Science (vol. 3485). Berlín: Springer Publishing.
Aquest llibre és un recull d'articles que tracten diferents aspectes relacionats amb els sistemes d'igual a igual.

Articles

Fielding, R. T. "Software architecture styles for Network-based applications".

Milojicic, D. S.; Kalogeraki, V.; Lukose, R.; Nagaraja, K.; Pruyne, J.; Richard, B.; Rollins, S.; Xu, Z. (2002). "Peer-to-peer computing".

Fuggetta, A.; Picco, G. P. (1998). "Giovanni Vigna. Undestanding code mobility". *IEEE transactions on software engineering*.