Arquitectures, paradigmes i aplicacions dels sistemes distribuïts

Joan Manuel Marquès i Puig
Xavier Vilajosana i Guillén
Pedro A. García López

P07/M2006/02839

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 els diferents estils arquitectònics dels sistemes distribuïts. Les arquitectures ens permetran conèixer les formes d'organitzar els components que formen una aplicació distribuïda. 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 d'un sistema distribuït.

2. Conèixer diferents formes d'organitzar els components lògics dels sistemes distribuïts.

3. Conèixer amb detall les arquitectures més usades actualment.

4. Conèixer les aplicacions principals dels 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. Estils arquitectònics de sistemes distribuïts

Els estils arquitectònics descriuen l'organització lògica dels components d'un sistema distribuït.

Un **component lògic** és una unitat modular, substituïble, que defineix les interfícies requerides i ofertes a altres components lògics. Els sistemes distribuïts es formen a base de components lògics interrelacionats entre si. Els **connectors** són mecanismes que fan de mediadors de la comunicació, coordinació o cooperació entre components. A base d'interrelacionar components lògics i connectors es poden construir diferents estils de sistemes distribuïts:

1) **Arquitectures per capes**. Els components lògics s'organitzen per capes. Un component lògic de la capa L_i només pot invocar operacions a la capa de sota L_{i+1}. En aquest estil, les invocacions van de les capes superiors a les capes inferiors, mentre que els resultats van de les capes inferiors cap a les capes superiors.

2) **Arquitectura orientades a components**. Cada objecte representa un component lògic i els components lògics estan interconnectats mitjançant invocacions remotes (RPC). Aquest estil i l'estil de l'arquitectura per capes són els més usats en sistemes distribuïts de gran escala.

3) **Arquitectura orientada a les dades**. En els sistemes distribuïts la localitat de les dades és un factor que afecta al rendiment del sistema. Alguns sistemes de fitxers distribuïts, o les aplicacions web distribuïdes han d'organitzar els seus components lògics en funció de la localitat de les dades. Aquesta arquitectura també és utilitzada quan es volen gestionar dades que estan distribuïdes d'una certa manera, per exemple, quan el volum de dades és molt gran, pot ser interessant que el component lògic que processa les dades sigui allà on són les dades per no haver-les d'enviar a través de la xarxa.

4) **Arquitectures orientades a esdeveniments**. Els components bàsicament es comuniquen mitjançant esdeveniments. Aquesta estil té com un dels seus paradigmes més representatius el que es coneix com a paradigma de publicació-subscripció. En d'aquests sistemes els components publiquen esdeveniments que només són rebuts per aquells components que hi estan subscrits. El principal avantatge d'aquests sistemes és el desacoblament dels components lògics.

3. Tipus d'arquitectures dels sistemes distribuïts

Una classificació molt estesa dels sistemes distribuïts és aquella que els classifica en funció de la ubicació, jerarquia o relació entre els components lògics.

3.1. Arquitectures centralitzades

En les arquitectures centralitzades la interrelació entre components segueix un patró molt característic en el qual hi ha una jerarquia definida de manera que certs components requereixen informació o serveis que ofereixen altres components lògics.

3.1.1. 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.

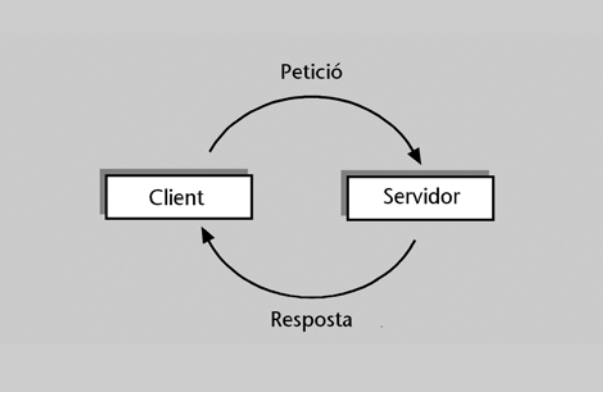

Figura 1. El model client servidor

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.

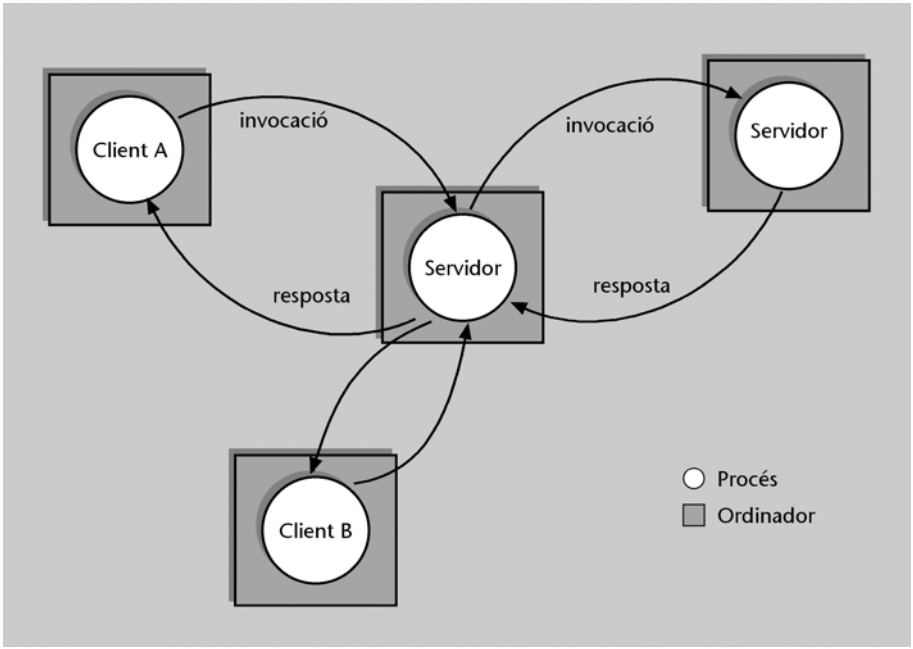

Figura 2. Servidor client d'altres servidors

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.

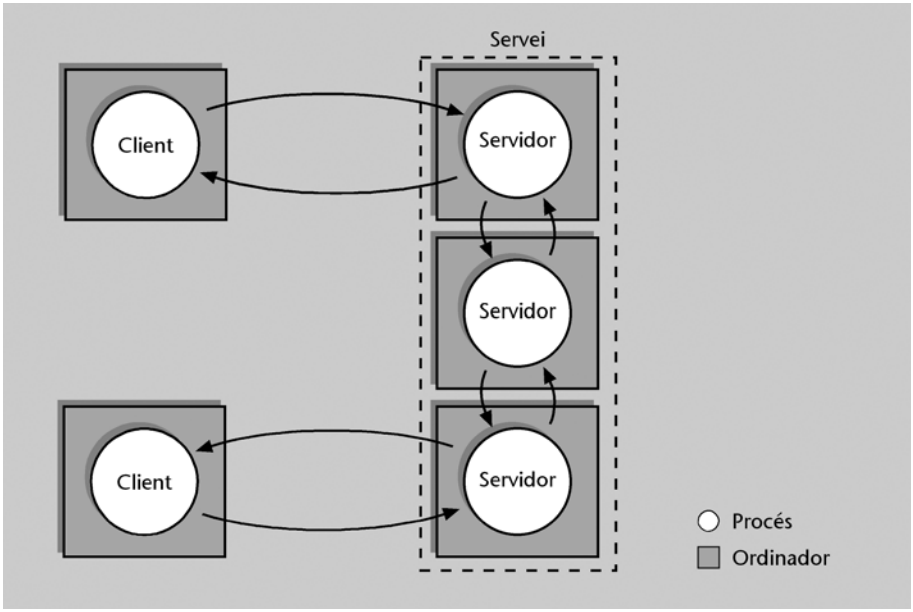

Figura 3. Diferents processos servidors

Arquitectura multiestrat*

* En anglès: *multitier 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.

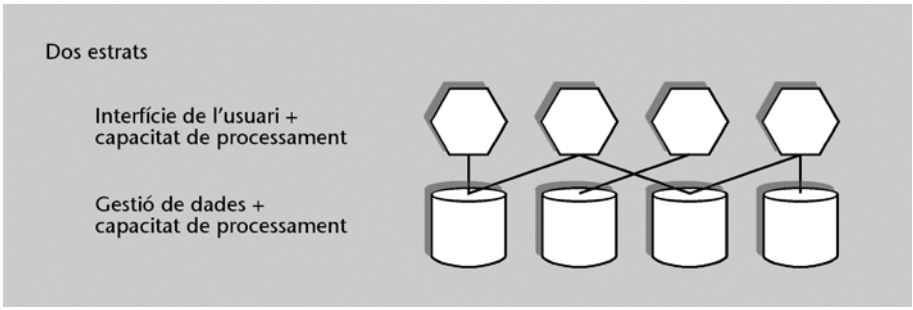

Figura 4. Arquitectura de dos estats

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.

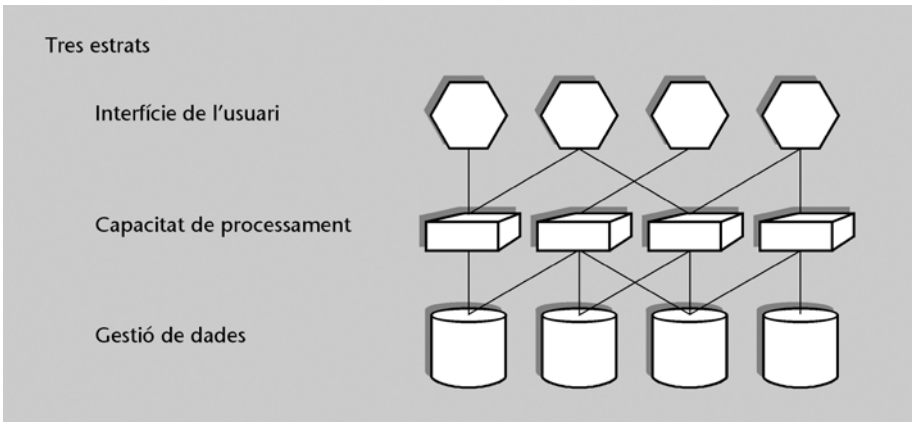

Figura 5. Arquitectura de tres estrats

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.

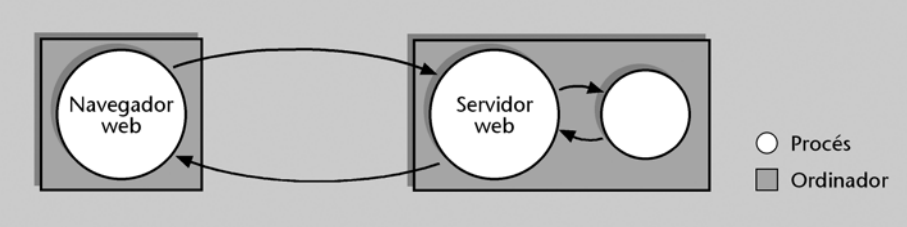

Figura 6. Aplicacions web

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 7, 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.

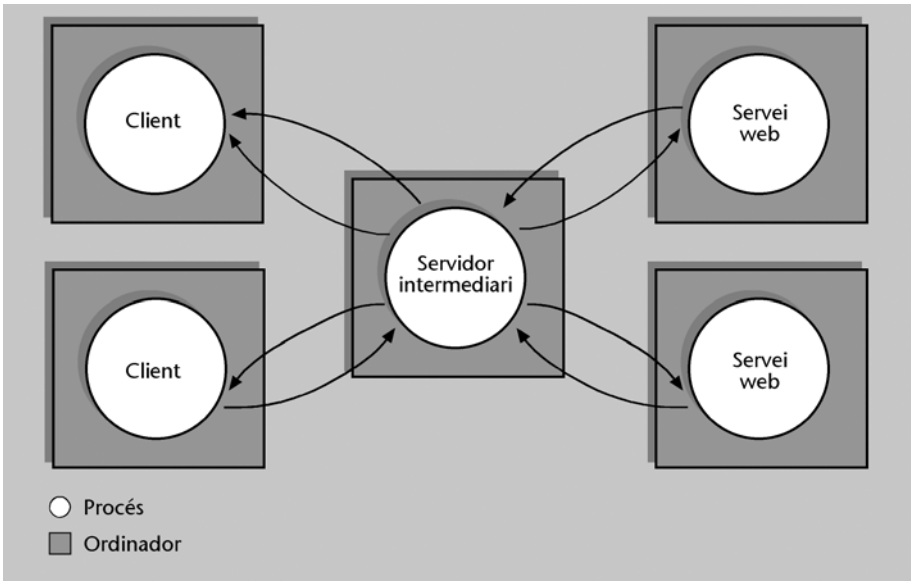

Figura 7. Servidors intermediaris

3.2. Arquitectures descentralitzades

En l'apartat anterior hem vist com les arquitectures multiestrat feien una **distribució vertical** dels components, és a dir, distribuïen *lògicament* els components en diferents màquines segons les seves funcionalitats. La distribució vertical no és l'única forma de separar funcionalitats d'un sistema distribuït. La **distribució horitzontal** consisteix a distribuir en parts lògicament equivalents les funcionalitats d'un client o d'un servidor, de manera que cada part mantingui totes les funcionalitats, però que la càrrega sobre el sistema quedi balancejada entre les diferents parts.

Un dels paradigmes més representatius de la distribució vertical són els sistemes d'igual a igual*. En un sistema d'igual a igual, la major part de la interacció entre els components és simètrica, és a dir, que els components actuen com a clients i servidors alhora. Els nodes que formen un sistema o aplicació d'igual a igual s'organitzen formant el que s'anomena 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, que entre d'altres condicionarà el determinisme del sistema. Principalment les dividim en tres categories: estructurades, no estructurades, híbrides.

*** En anglès: *peer-to-peer* o *P2P* .**

Motivació

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.

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

Els missatges Usenet

Els missatges Usenet (*Usenet news*, en anglès) és un sistema global i descentralitzat de grups de discussió a Internet. 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

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.

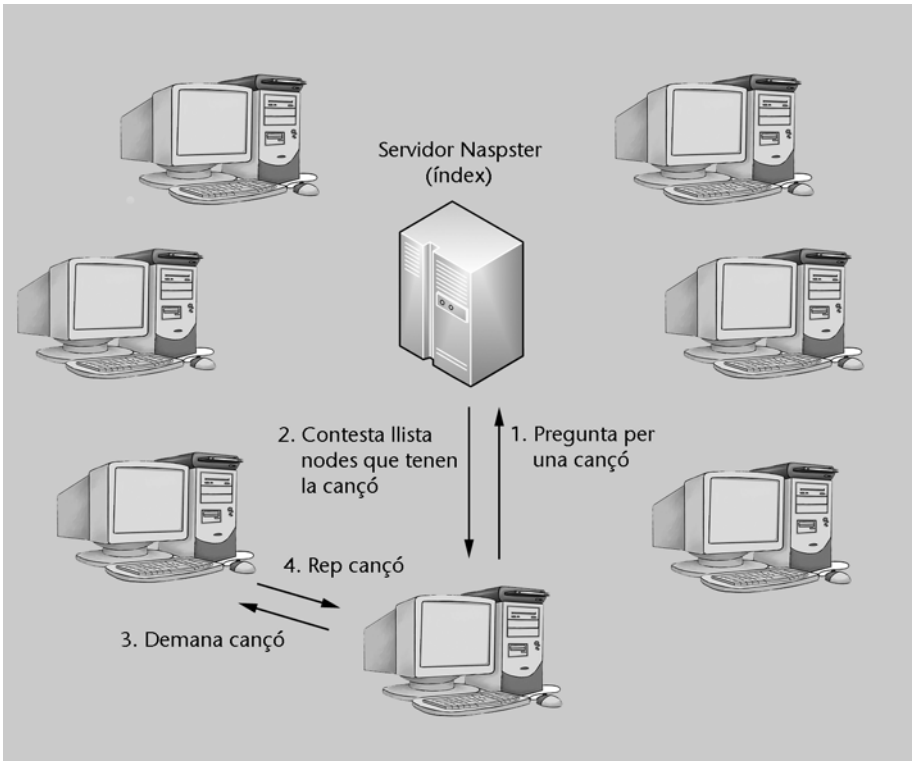

Figura 8. L'aplicació Napster

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.

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.

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.

3.2.1. 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.

L'exemple més rellevant dels sistemes no estructurats el trobem amb el sistema Gnutella.

a) Cerques en sistemes no estructurats

El rendiment d'un sistema distribuït no estructurat està condicionat, entre d'altres, pel cost de localitzar un objecte en el sistema. Les tècniques de cerca han estat àmpliament estudiades ja que són mecanismes que determinen tant la probabilitat de localitzar un objecte com la càrrega que ha de suportar el sistema. En aquest mòdul n'estudiarem les següents:

- **Inundació** (***flooding***). La tècnica d'inundació consisteix a encaminar les cerques a través de tots els nodes veïns. Cada node retransmet la petició als

seus nodes veïns de forma recursiva fins que algun node troba l'objecte cercat i respon al node emissor. Un dels avantatges d'aquesta tècnica és la seva resiliència davant d'entrades i sortides de nodes al sistema. Un dels desavantatges de la cerca per inundació és la seva escalabilitat, ja que els costos de cercar un objecte s'incrementen de forma exponencial a mesura que el nombre de nodes augmenta. Malgrat això, sistemes com Gnutella han demostrat la seva viabilitat en xarxes de mida mitjana.

- **Camins aleatoris** (***random walks***). Podem considerar les cerques basades en camins aleatoris com a processos estocàstics que consisteixen en una seqüència de canvis determinats de forma aleatòria. En una xarxa no estructurada, aquesta tècnica enlloc d'encaminar les cerques a través de tots els veïns ho fa a través d'un nombre determinat i aleatori d'aquests.

  Formalment, podem definir un camí aleatori com:

  Donat un graf connex G(V,E) amb |V| = n, |E| = m un **pas** aleatori a G és el moviment des d'algun node *u* cap a un altre node veí v aleatòriament seleccionat. Un camí aleatori és una seqüència d'aquests passos aleatoris començant des d'algun node inicial.

  L'aplicació dels camins aleatoris com a mecanismes de cerca en sistemes no estructurats ha demostrat comportaments més escalables que no pas les tècniques d'inundació en sistemes on la xarxa superposada forma estructures clusteritzades o en sistemes on una cerca es repeteix sense que hi hagi massa canvis en la topologia de la xarxa*.

  *** Christos Gkantsidis, Milena Mihail, and Amin Saberi. "Random walks in peer-to-peer networks". Proceedings of IEEE INFOCOM, 2004.**

- **Camins aleatoris adaptatius**. Recentment s'han introduït millores a la tècnica dels camins aleatoris. Una d'aquestes millores consisteix a controlar el radi de cerca mitjançant el TTL (*time to live*) del missatge de cerca. La tècnica inicia la cerca amb un TTL amb valor 1; si no es troba l'objecte cercat, torna a iniciar una cerca amb un TTL amb valor 2 i així successivament fins que es troba l'objecte cercat o bé s'abandona la cerca. Aquesta millora permet controlar la càrrega sobre la xarxa. Altres tècniques orienten la cerca mitjançant equacions que seleccionen el següent pas en funció de certs paràmetres com pot ser la popularitat estimada del següent node a consultar, és a dir, seleccionen aquell node que té més probabilitat de tenir l'objecte. Aquesta tècnica ha demostrat ser eficient per cerques d'objectes populars.

  A la figura següent veiem una cerca que ja ha estat iniciada pel node amb id = 1 i que encara no ha trobat l'objecte cercat havent provat el ttl = 2. La nova cerca s'inicia amb ttl = 3. La cerca s'encamina a cap alguns dels nodes veïns del node amb id = 1 i es va decrementant el ttl en cada salt. Al cap de tres salts la retransmissió del missatge s'atura (el node marcat amb stop ja

no retransmetrà el missatge als seus veïns). El node marcat amb match conté l'objecte cercat i respon a la petició.

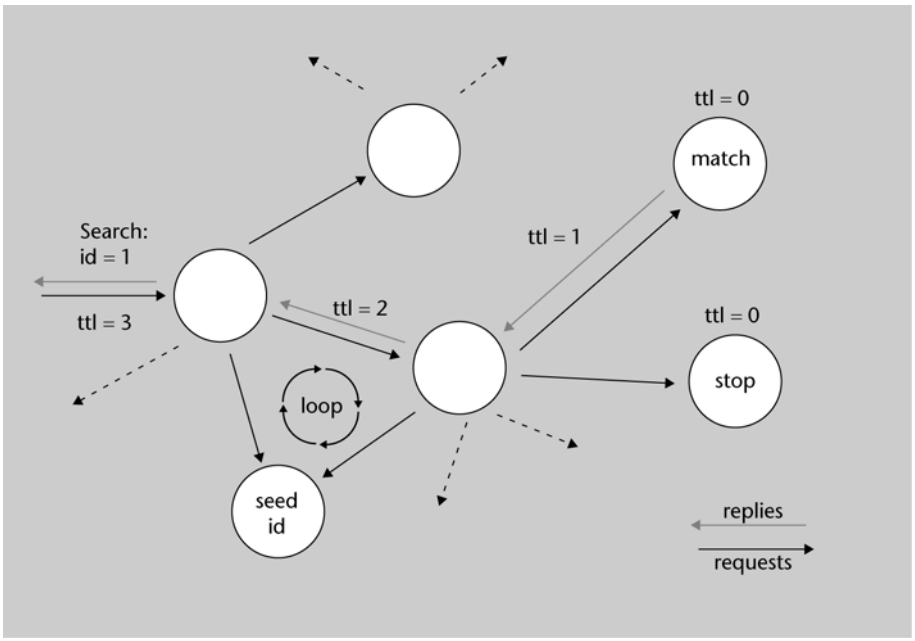

Figura 9. Random Walks

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 10 mostra un exemple de cerca.

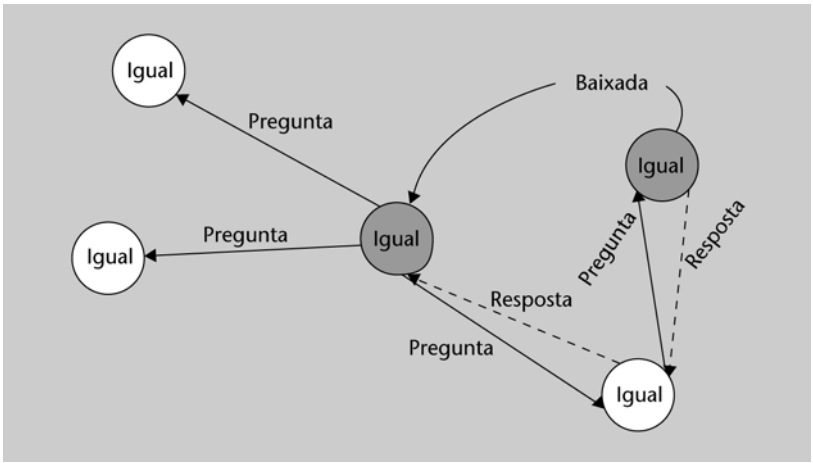

Figura 10

b) Mecanismes pel manteniment de la informació

Els sistemes distribuïts no estructurats requereixen de mecanismes per mantenir de forma consistent la informació distribuïda com pot ser informació sobre l'estat del sistema, informació sobre la topologia de la xarxa superposada, etc.

La xarxa superposada que formen els sistemes distribuïts no estructurats pot ser comparada a un graf aleatori. Cada component del sistema distribuït representa un node del graf i la les arestes representen relacions de veïnatge, de manera que un node A està unit amb una aresta al node B si i només si el node A es veí* del node B.

*** Les relacions de veïnatge són dependents de l'aplicació. Es poden definir en termes de proximitat geogràfica, topològica, o bé definides segons les necessitats de l'aplicació.**

Cada node del graf coneix una llista de veïns a la quals anomenarem **vista parcial**. Els mecanismes més usats per a construir i mantenir la **vista parcial** dels components del sistema distribuït són els mecanismes epidèmics.

Els **mecanismes epidèmics** són aquells en què quan dos nodes es comuniquen intercanvien la seva informació local així com la informació que han rebut d'altres. Aquests mecanismes, anàlogament a una epidèmia, propaguen la informació a través dels nodes veïns arribant a "infectar" el sistema complet. Els mecanismes epidèmics s'usen per resoldre problemes complexos en sistemes on la topologia de la xarxa és no estructurada, l'escala del sistema és molt gran o bé són la solució més eficient de les possibles.

Un mecanisme epidèmic satisfà les condicions següents:

- Els components interactuen en parelles de forma periòdica.
- La informació que intercanvien els components és de mida petita i limitada.
- L'estat d'un o dels dos components canvia després de la interacció.
- La comunicació és no fiable.
- La freqüència de les interaccions és petita en comparació amb la latència dels missatges. Això fa que els costos del mecanisme siguin negligibles.
- Hi ha una certa aleatorietat en la selecció dels components amb els quals interactuar. La selecció es pot fer entre els veïns o bé tenint en compte tots els nodes del sistema.

La direcció en la qual es transmet la informació, així com la proactivitat dels components, diferencia dos modes d'operació:

- **Mode de tramesa induïda** (**en anglès**, ***pull mode***). En aquest mode, els components esperen passius a rebre informació d'un altre node del sistema. Un component només actualitza la seva informació quan un altre node veí es posa en contacte amb ell.

- **Mode de tramesa automàtica** (**en anglès**, ***push mode***). En aquest mode, els components de forma periòdica prenen la iniciativa de comunicar-se amb algun altre node per tal d'actualitzar la seva informació.

Com ja s'ha dit abans, els mecanismes epidèmics o de "xafardeig" (en anglès, *gossip*) s'usen entre d'altres per mantenir la topologia de la xarxa superposada, per exemple, fent servir el **mode de tramesa induïda**, el **mode de tramesa automàtica** o bé una combinació d'ambdós modes els components d'un sistema aconsegueixen mantenir la **vista parcial**.

c) Aplicacions d'igual a igual en xarxa superposada no estructurada

Les aplicacions més representatives que s'han construït fent ús del paradigma d'igual a igual no estructurat (també conegut com d'igual a igual pur) són Gnutella que ja hem vist anteriorment i Freenet.

Freenet és una xarxa descentralitzada per a intercanvi d'informació l'objectiu del qual és mantenir l'anonimat dels seus usuaris, ja sigui dels qui aporten informació a la xarxa o dels qui la utilitzen. És descentralitzada perquè no hi ha cap servidor central que emmagatzemi dades o coordini la xarxa sinó que cada "node" de la xarxa guarda informació i aporta informació per a tota la xarxa. La ideologia de Freenet és preservar l'anonimat dels qui l'usen i permetre la lliure expressió a la xarxa sense que existeixi cap tipus de censura. Una vegada un fitxer és inserit a la xarxa es fa gairebé impossible saber d'on prové o qui l'ha originat; tampoc no és possible eliminar el contingut de la xarxa (aquest va desapareixent a mesura que deixa de ser accedit).

Una altra aplicació amb una finalitat força diferent a la de Freenet i Gnutella és Groove. Groove és una aplicació d'igual a igual que ofereix funcionalitats per al treball col·laboratiu a través de la xarxa. La idea bàsica és la d'accés a un espai virtual d'emmagatzematge en què tots els iguals d'un grup hi poden tenir accés. Groove permet als seus usuaris la sincronització de documents, entre d'altres activitats col·laboratives. Per tal de mantenir la informació sincronitzada i detectar conflictes, els iguals d'un grup fan ús de mecanismes epidèmics. Aquesta aplicació a més de gestionar la col·laboració entre iguals de forma totalment no estructurada també permet fer ús de servidors que faciliten de forma distribuïda la sincronització de la informació.

3.2.2. Estructurats

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.

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.

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.

Taules de *hash* distribuïdes (distributed *hash tables*)

Les DHT sorgeixen en l'àmbit de la investigació en sistemes *peer-to-peer* com una evolució dels models d'índex centralitzat de Napster i totalment descentralitzat per inundació com Gnutella.

Història

En entorns acadèmics d'investigació sobre sistemes distribuïts al final dels noranta s'intenta resoldre el problema de la localització de recursos descentralitzada que resolgui els colls d'ampolla i problemes d'escalabilitat d'un índex centralitzat. Un problema clau és aconseguir que tots els nodes siguin alhora encaminadors d'informació i que es produeixi un repartiment balancejat de la càrrega entre tots els participants. En aquest context, els sistemes descentralitzats no estructurats com Gnutella es basen en sistemes d'encaminament per inundació que mitjançant algorismes de replicació adequats permeten la localització de les dades. S'anomenen *no estructurats* perquè les connexions entre els nodes són més o menys aleatòries sense que hi hagi així una estructura global en el sistema.

Tanmateix, aquest tipus de sistemes generen molt de trànsit innecessari i no asseguren que puguem localitzar un recurs encara que existeixi a la xarxa. Un primer pas per a solucionar aquestes limitacions és el donat per Freenet situant els índexs en zones més o menys conegudes de la xarxa. D'aquesta manera, un node pot encaminar cap a la zona on se suposa que es troba l'índex que està buscant. Aquest és un primer pas cap al denominat *greedy routing* (*routing* egoista o ansiós) pel qual cada node pot prendre una decisió local de cap a on encaminar partint del que es busca. Això suposa un gran avenç davant el model de Gnutella en el qual cada node no sap quina de les seves connexions ha d'utilitzar per a encaminar i per això ha de propagar en totes direccions.

Les taules de *hash* distribuïdes (DHT) són un tipus de sistemes distribuïts que permeten la localització eficient de dades per mitjà d'un índex descentralitzat i uniformement repartit entre els nodes del sistema. Se'ls denomina DHT perquè cada node és anàleg a una cel·la d'una taula *hash* que permet emmagatzemament i recuperació d'informació (PUT (K,V), GET(K)) de manera eficient en un entorn distribuït.

Així, al començament del segle XXI, es proposen les primeres xarxes *peer-to-peer* estructurades com Chord, CAN, Pastry o Tapestry entre d'altres. En aquestes xarxes, els nodes es connecten entre ells seguint una estructura predeterminada com un anell, un hipercub o un arbre. Les estructures establertes asseguren que cada node necessita poques connexions (log N), que la xarxa té un diàmetre petit i que es pot encaminar informació d'un node a un altre en pocs salts (log N). A més, s'utilitza el *greedy routing* per la qual cosa cada node sap per on és més òptim enviar la informació.

Un descobriment clau que propicia l'aparició d'aquestes xarxes estructurades en el concepte de *hashing* consistent (*consistent hashing*). Malgrat que no va ser pensat en principi per a entorns descentralitzats *peer-to-peer*, aquesta contribució va servir d'inspiració per a resoldre el problema del repartiment balancejat dels recursos (*load balancing*) entre els nodes de la xarxa. Així, el *hashing* con-

sistent utilitza una funció determinística que assigna identificadors en un rang de manera uniforme, assegurant així el repartiment equitatiu de la càrrega.

Chord

Estudiarem el cas de Chord com a exemple paradigmàtic i clàssic de les xarxes *peer-to-peer* estructurades o també denominades *taules de hash distribuïdes* (DHT). Com veiem en la figura, en Chord els nodes formen un anell amb identificadors compresos entre [0 i 2 M] mòdul 2 M essent M el nombre de bits de l'identificador. En aquest exemple, M és 6 per la qual cosa els identificadors estan compresos entre el 0 i el 63.

L'identificador de cada node s'obté utilitzant una funció de *hashing* consistent que assegura que els identificadors estaran uniformement repartits. Un exemple és la funció SHA (*secure hash algorithm*), que genera un identificador de 160 bits que després es poden truncar per manejar identificadors més reduïts (la mida sol estar entre 32 i 160 bits, cosa que permetrà d'aquesta manera un gran nombre de nodes).

Quan un node entra a la xarxa i obté un identificador, s'ha de situar a la zona de l'anell adequada entre el predecessor i el successor del seu identificador. És el que es coneix com a *procés d'entrada a la xarxa* (*join, bootstrapping*) i ho pot fer a través de qualsevol node. Per exemple, si en aquesta xarxa entra el node 4, se situarà entre el node N1 i l'N8.

En Chord, cada node és responsable de les claus o recursos situats entre el seu predecessor i ell mateix (inclòs). D'aquesta manera, l'N8 és responsable de les claus 2, 3, 4, 5, 6, 7 i 8. Quan s'insereix un recurs a la xarxa, s'aplica la funció de *hashing* consistent per a obtenir la seva clau i així s'assegura que els recursos estan uniformement repartits. En calcular el *hashing*, per exemple del nom del fitxer, (SHA(ASD.doc)) obtenim una clau que ens diu, a més, quin node de la xarxa n'és responsable. D'aquesta manera, en inserir la clau 17 (PUT(17)) s'emmagatzemarà en el successor de 17, que és el node N21.

Si cada node només tingués els enllaços de successor, l'encaminador seria $O(N)$. És a dir, en el pitjor dels casos si som en el node zero i només podem anar cap a la dreta, per a arribar al node 63 haurem de passar per tots els nodes de l'anell. Per a millorar això, Chord té una taula d'enllaços a altres nodes denominada *finger table*. Aquesta taula té log N enllaços essent N = 2M, per la qual cosa en el nostre exemple la mida és 6. En la *finger table*, cada node té connexions logarítmiques a altres nodes seguint el patró: n + (2i mòdul 2M), on n és l'identificador del node, i *i* correspon a cada entrada de la *finger table* (exemple, 0...6). Com veiem en l'exemple, el node N8 hauria de tenir connexions als nodes 9, 10, 12, 16, 24 i 40 si la xarxa fos completa. En no estar-ho, es connectarà als nodes successors d'aquests identificadors que si es troben a la xarxa.

L'encaminament en Chord és ansiós (*greedy routing*), en el sentit de les agulles del rellotge (*clock-wise*) i se sol denominar encaminament basat en claus (*key based routing*, KBR). Quan des d'un node s'intenta localitzar un recurs amb una clau concreta, el node busca en la seva *finger table* la connexió que més l'apropa a la seva destinació. Per exemple, si en l'N8 es busca la clau 54 (GET(54)), s'enviarà la sol·licitud al node N42, que és el que més s'aproxima a la clau 54. Aquest, al seu torn, l'enviarà al 51, que finalment preguntarà al 56, que és el responsable de la clau 54. Com podem observar, en molt pocs salts trobem qualsevol recurs de manera eficient i així s'eviten les inundacions del model Gnutella.

Discussió

Els avantatges clau de Chord i altres sistemes similars són la descentralització, l'escalabilitat i el balanceig de la càrrega entre els participants.

Les DHT permeten construir sistemes completament descentralitzats en els quals tots els nodes són iguals entre ells i altament escalables a milions de participants. A més, l'avantatge essencial d'aquests sistemes és que permeten localitzar informació amb un ordre de salts logarítmic (O(logN)) i mantenint un petit nombre de connexions a altres nodes (taules d'encaminament d'ordre constant O(1) o logarítmic (O(logN)).

Els principals problemes són el manteniment del sistema, l'entrada i sortida de nodes constant (*churn*), la proximitat en xarxa i la limitació a recerques exactes (*exact match*).

En Chord, per exemple, és necessari mantenir en cada node una llista de successors de mida logarítmica per a assegurar que no es trenca l'anell. A més, és necessari un protocol de manteniment d'enllaços que els actualitzi correctament a mesura que entren o surten nodes. Això implica un cost que limita la seva aplicació per a sistemes molt dinàmics i encara s'està estudiant en cercles d'investigació acadèmics.

A més, en aquestes xarxes descentralitzades és essencial tenir en compte la proximitat en xarxa o les latències entre nodes. Si un DHT necessita pocs salts, però aquests passen per nodes molt llunyans en termes de latència (Espanya-Japó-Austràlia-França), el sistema serà molt ineficient. Per això, molts DHT han intentat organitzar-se partint d'informació de proximitat com CAN, Pastry o CORAL per a millorar l'encaminament. Tanmateix, això continua essent un problema obert que es continua investigant en cercles científics.

D'altra banda, les DHT ofereixen un sistema de recerca exacta GET(K) que els fan menys flexibles que altres sistemes amb recerques obertes o més complexes. A més, quan un node se'n va, les seves claus han de migrar al node successor que ara és el responsable d'aquestes claus. Per això s'han de replicar

aquestes claus per a fer el sistema tolerant a fallades i assegurar que les claus no es perden. Diversos treballs d'investigació continuen estudiant avui en dia com s'han d'adaptar o construir aplicacions sobre els DHT que permetin recerques més avançades.

De qualsevol manera, les DHT han suposat una autèntica revolució en entorns descentralitzats i ja se'ls considera la tercera generació de sistemes *peer-to-peer* després de Napster i Gnutella. Gràcies a les considerables qualitats de les DHT s'han desenvolupat molts serveis distribuïts que es construeixen sobre aquestes.

Exemples de serveis...

... serien els sistemes de fitxers distribuïts, els sistemes de compartició d'arxius P2P, l'emmagatzemament cooperatiu en web, serveis de *multicast* o *anycast* a nivell d'aplicació, o fins i tot serveis de noms i de DNS descentralitzats.

Entre les aplicacions existents avui en dia que es basen en DHT destaquem les següents:

- **eMule KAD**: el programa de descàrrega de fitxers P2P eMule utilitza una DHT denominada Kademlia (KAD) com a evolució del sistema. KAD millora la localització de fonts a la xarxa i la fa menys fràgil a atacs sobre els servidors centrals.

- **BitTorrent/Azureus**: els clients de bitTorrent utilitzen una implementació del DHT Kademlia per a aconseguir la descentralització del component *tracker*. Això els fa menys vulnerables a la caiguda d'aquest.

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

En conclusió, avui en dia les DHT es consideren un substrat distribuït essencial sobre el qual construir serveis descentralitzats. La seva integració amb mecanismes de proximitat en xarxa d'àmbit Internet permetran aplicacions eficients en el context de la computació Grid de gran escala o en xarxes de distribució de continguts (CDN). Finalment, cal destacar que les DHT no són la solució ideal a tots els problemes en sistemes distribuïts descentralitzats (silver bullet). Per exemple, en entorns mòbils amb una gran dinamicitat i inestabilitat dels nodes, els DHT no són la solució adequada i sí que ho són, en canvi, els sistemes desestructurats o híbrids amb menys cost de manteniment de la xarxa.

Híbrids

Els sistemes híbrids donen solució a problemes que no poden ser resolts de forma eficient per cap dels models anteriors. Per exemple, la localització de certes dades en sistemes descentralitzats esdevé problemàtica quan la xarxa creix. Els sistemes no estructurats usen tècniques d'inundació no deterministes que no ens asseguren l'èxit en les cerques. Els sistemes estructurats tenen problemes de sobrecàrrega quan és fan cerques sobre rangs d'objectes ja que aquests només permeten fer cerques indexades de forma eficient sobre objectes concrets.

Per altra banda, els sistemes centralitzats acaben convertint-se en un "coll d'ampolla" a mesura que l'escala del sistema augmenta.

a) Super-iguals (***super-peers***). Un dels tipus de sistemes híbrids més usats són els basats en super-iguals. Un super-igual és un node que actua com a

índex o gestor de la informació del sistema. Tal i com el seu nom indica, els super-iguals també s'organitzen en una xarxa d'igual a igual, formant una estructura amb dos o més nivells. En molts casos, la relació igual-super-igual és fixa durant el temps de connexió de l'igual ja que es manté la relació mentre dura la connexió. Els super-iguals tendeixen a ser nodes que es mantenen connectats i pateixen de poc dinamisme durant la vida del sistema. Els sistemes basats en super-iguals ofereixen facilitats en les cerques i gestió de la informació ja que aquesta pot estar indexada. Un nou problema que apareix amb aquests sistemes és el de l'elecció dels nodes que actuaran com a super-iguals qüestió molt relacionada amb els *problemes d'elecció d'un líder* que estudiarem en un altre mòdul d'aquest curs.

- **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 11. 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.

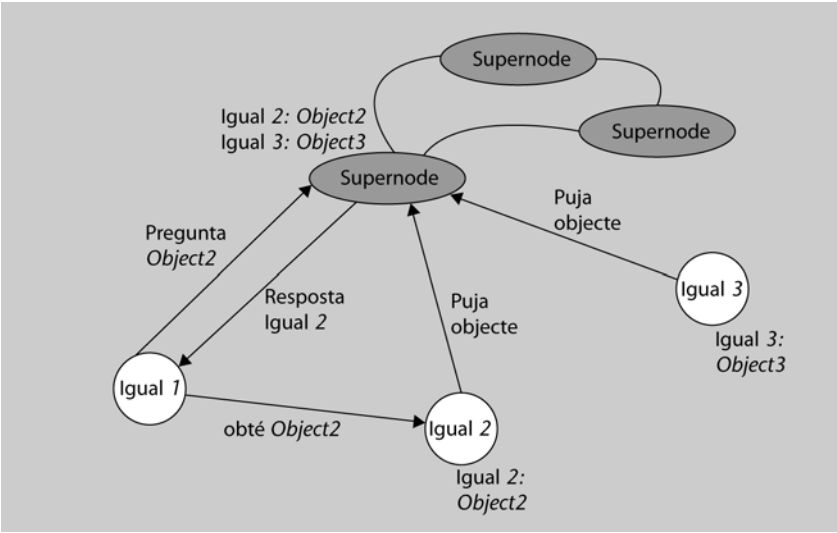

Figura 11. Cerca a KaZaA

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

- **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 es una aplicació client de la xarxa eDonkey (híbrida), encara que també pot utilitzar la xarxa Kad (estructurada, ja que és una DHT).

- **Skype** és un altre sistema que proporciona telefonia a través d'Internet. Utilitza un protocol propietari de la implementació del qual es coneixen pocs detalls. Funciona seguint una organització en super-iguals tal com ho fa KaZaA. De fet, Skype va ser fundada pels fundadors de KaZaA. Un aspecte a destacar es que aconsegueix superar els problemes (entre d'altres la incapacitat de comunicació) que tenen els iguals que estan darrera d'un tallafocs o problemes derivats del NAT (*network address translation*). La idea bàsica per fer-ho és la següent: Els tallafocs i els sistemes de NAT només deixen passar aquells paquets que pertanyen a una direcció de confiança, coneguda o amb la que l'ordinador s'hagi comunicat abans. Skype el que fa és "persuadir" a la tallafocs fent-li creure que ja s'ha establert una connexió amb anterioritat. A partir d'aquí com que la comunicació es fa a través de paquets UDP, dels quals el tallafocs només pot extreure informació sobre les direccions IP i ports dels participants en la comunicació, el tallafocs no pot verificar més informació dels paquets i en permet el pas.

b) Trackers: Els Trackers són servidors dedicats a mantenir informació sobre un conjunt d'iguals que comparteixen o emmagatzemen un contingut determinat. Cada Tracker és responsable de mantenir informació sobre els iguals que emmagatzemen un o més d'aquests continguts. El sistema més representatiu que fa ús dels Trackers és BitTorrent.

- **BitTorrent**: és un sistema d'igual a igual per a distribuir grans volums de dades sense que l'originador de la informació hagi de suportar tot el cost dels recursos necessaris per a servir el contingut. Aquest tipus de solucions són útils per a distribuir continguts que són molt populars. BitTorrent fa servir servidors per a gestionar les baixades. Aquests servidors emmagatzemen un fitxer que conté informació sobre el fitxer: llargada, nom, informació de resum (*hashing information*, en anglès) i l'URL del *tracker*. El *tracker* (vegeu la figura 12) 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.

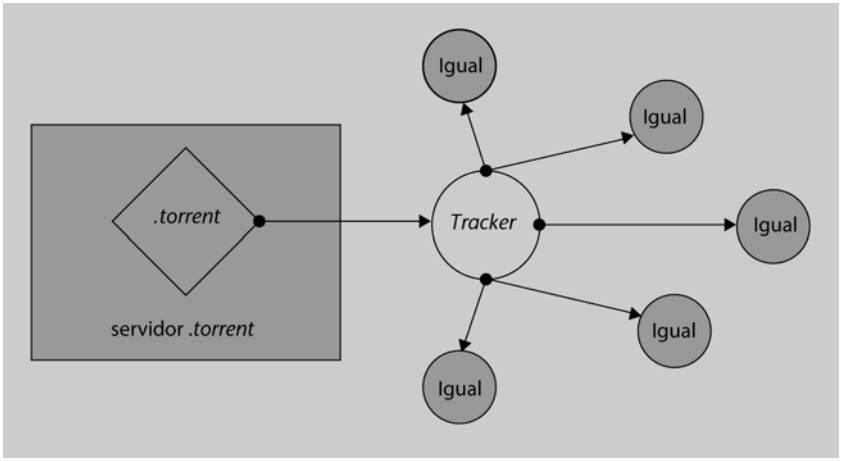

Figura 12. *Tracker* de BitTorrent

c) Brokers: Els Brokers són components dedicats a fer de mitjancers o intermediaris de la resta de components del sistema. Per exemple a Globule, una xarxa de distribució de continguts col·laborativa formada per servidors de pàgines web connectats en una xarxa d'igual a igual, els brokers actuen com a registres dels servidors permetent que aquests siguin descoberts per als altres iguals. Un altre exemple el trobem en alguns sistemes computacionals distribuïts per computació GRID com Globus en el que els gestors (diversos possibles) de tasques fan ús d'un Broker que permet programar l'execució de tasques de forma paral·lela en un sistema heterogeni i que en alguns casos s'intenta garantir un temps d'acabament.

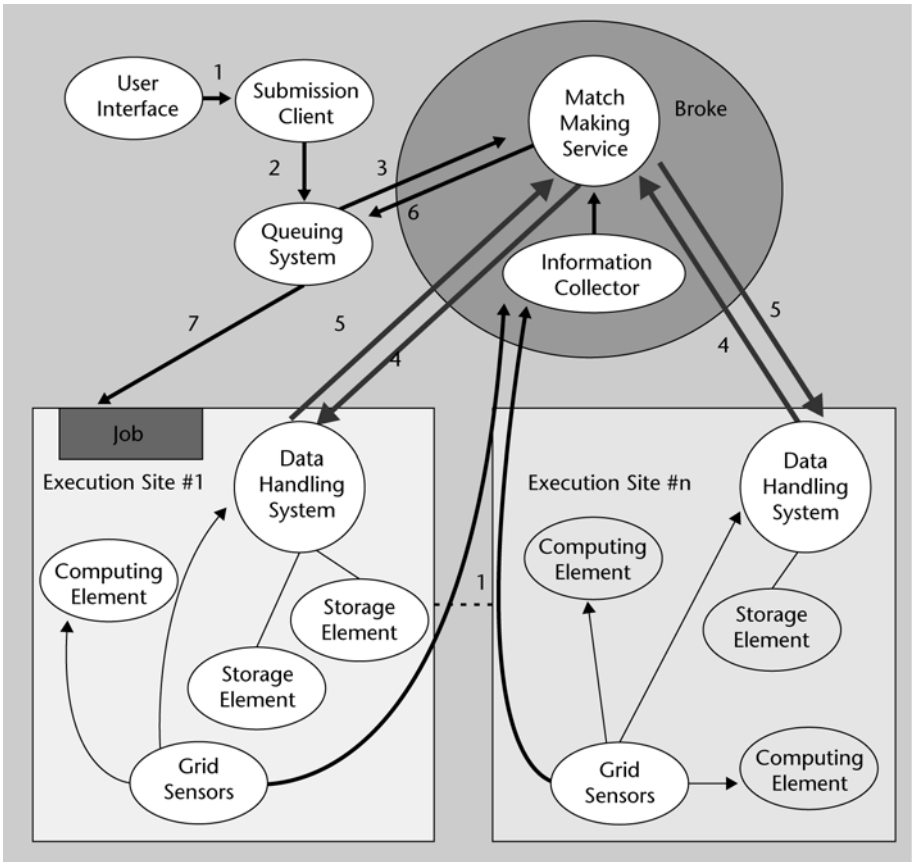

Figura 13. Seqüència típica d'accions per a l'execució d'una tasca en un sistema de computacions distribuït en què la gestió és duta a terme per un *broker*.

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.

3.3. Paradigma de la publicació-subscripció

El paradigma de la publicació-subscripció* es pot implementar fent ús tant d'arquitectures centralitzades, descentralitzades o híbrides. Si prenen, per exemple, 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.

* En anglès: *publish/subscribe.*

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.

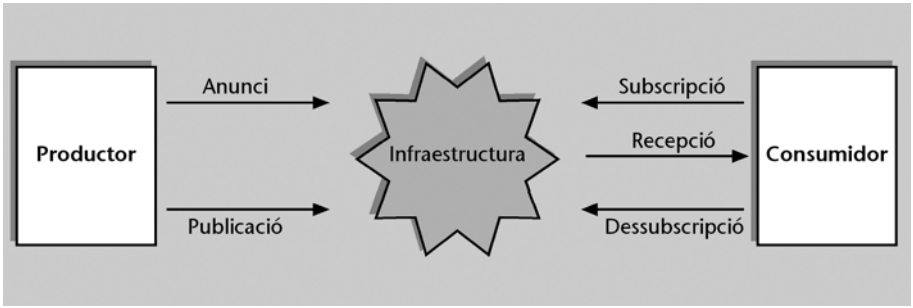

Figura 14. Paradigma de publicació-subscripció

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 15) veiem la relació entre els diferents components d'una arquitectura publicació-subscripció. En l'altra figura (figura 16) veiem el detall de com es fa la comunicació entre un productor i un consumidor.

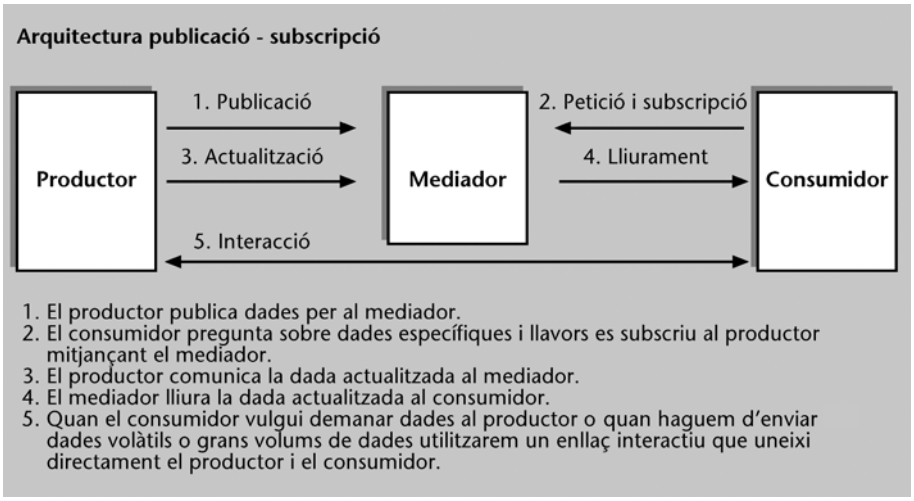

Figura 15. Detall de la comunicació del paradigma de publicació-subscripció

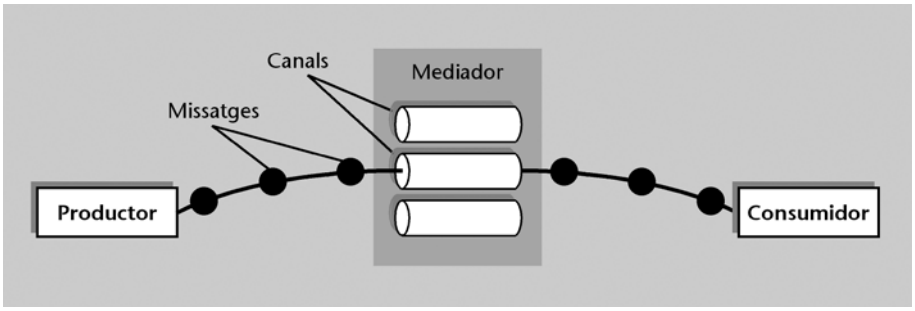

Figura 16. Paradigma de publicació-subscripció

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.

3.3.1. 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.

3.4. 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.

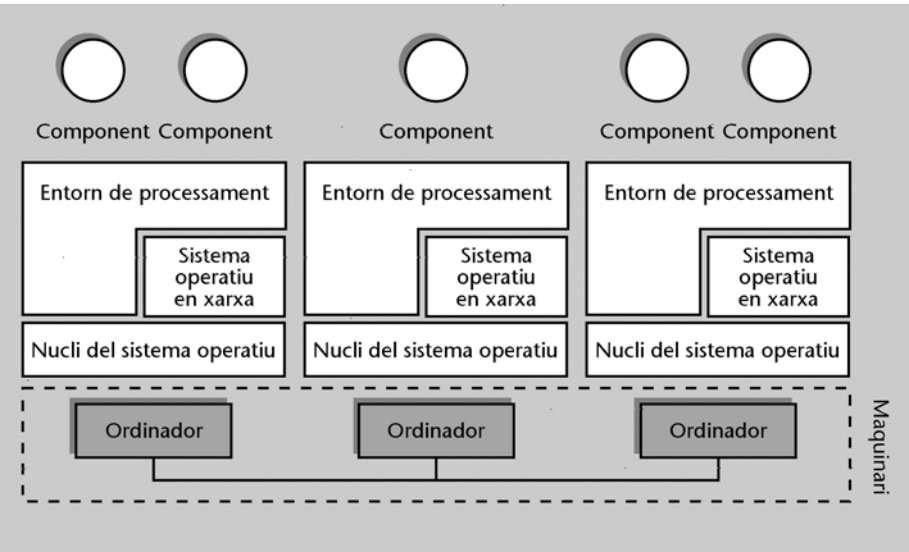

Figura 17. La màquina virtual

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 18).

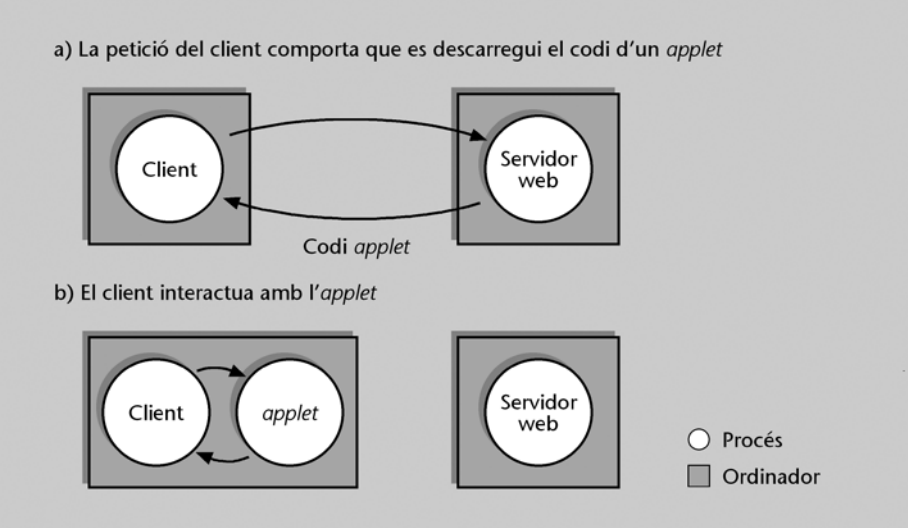

Figura 18

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.

4. Aplicacions dels sistemes distribuïts

Fins ara hem estudiat els sistemes distribuïts des d'un punt de vista arquitectònic que ens ha permès de conèixer els components del sistema distribuït i les formes que tenen d'organitzar-se i interrelacionar-se. En aquest apartat, veurem els sistemes distribuïts des d'un punt de vista funcional que ens donarà una idea de les aplicacions que tenen aquest tipus de sistemes.

4.1. Sistemes computacionals distribuïts

Són sistemes de computació d'alt rendiment. Estan formats per conjunts de computadors interconnectats a través d'una xarxa que ofereixen funcionalitats de supercomputació tals com la computació paral·lela. Diferenciem principalment tres tipus de sistemes computacionals distribuïts:

4.1.1. Clústers

Estan formats per col·leccions de computadors de similars característiques interconnectats a través d'una xarxa d'àrea local. Els computadors fan ús d'un mateix sistema operatiu i un *middleware* que s'encarrega d'abstreure i virtualitzar els diferents computadors del sistema donant la visió a l'usuari d'un sistema operatiu únic. Els clústers són sistemes dedicats a la supercomputació. El sistema operatiu d'un clúster és estàndard i per tant és el *middleware* qui proveeix de llibreries que permeten la computació paral·lela.

Un dels problemes més habituals en els clústers és el de la gestió dels processos a executar. Les solucions més usades són les cues de processos gestionades per un node anomenat *màster*. Com a alternativa, el sistema MOSIX va proposar una solució simètrica en la qual no hi havia una jerarquia mestre-esclau com en les solucions proposades fins llavors. MOSIX oferia una visió del sistema no només com un únic sistema operatiu sinó com una única màquina (Single System Image, en anglès, SSI). Els sistemes SSI permeten la migració de tasques a d'altres nodes de forma transparent i preemptiva. La migració permet a un usuari iniciar una aplicació en qualsevol node (conegut com a node llar, *home node*, en anglès) i, de forma transparent, aquesta pot ser migrada a un altre node per fer un ús més eficient dels recursos.

L'any 2002 el projecte MOSIX va a passar a ser un projecte de software propietari, la qual cosa el va condemnar a la desaparició. Malgrat això, la recerca feta

fins al moment va continuar amb el projecte openMosix fins l'any 2008, en què el projecte es donava per finalitzat donada la disminució de la demanda de sistemes SSI a causa de l'abaratiment dels sistemes computacionals multiprocessador.

4.1.2. *Grid*

Mentre que els clústers estan orientats a donar serveis computacionals d'ús local o amb una funció concreta, els sistemes Grid tenen com a objectiu la "metacomputació", és a dir, capacitats computacionals a escala Internet. No es poden fer assumpcions sobre el tipus de maquinari, sistemes operatius, interconnexions de xarxa, dominis administratius, polítiques de seguretat, d'aquest sistema.

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).

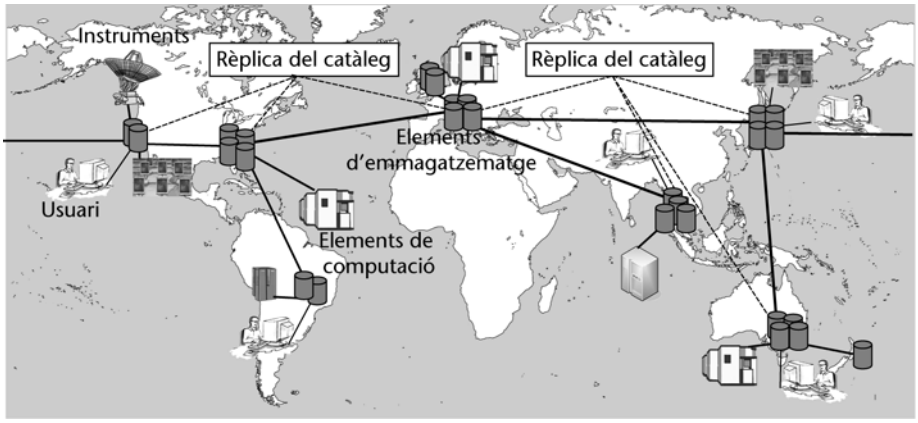

Figura 19. Exemple d'una infraestructura *grid*

La figura 19 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

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

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*.

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)

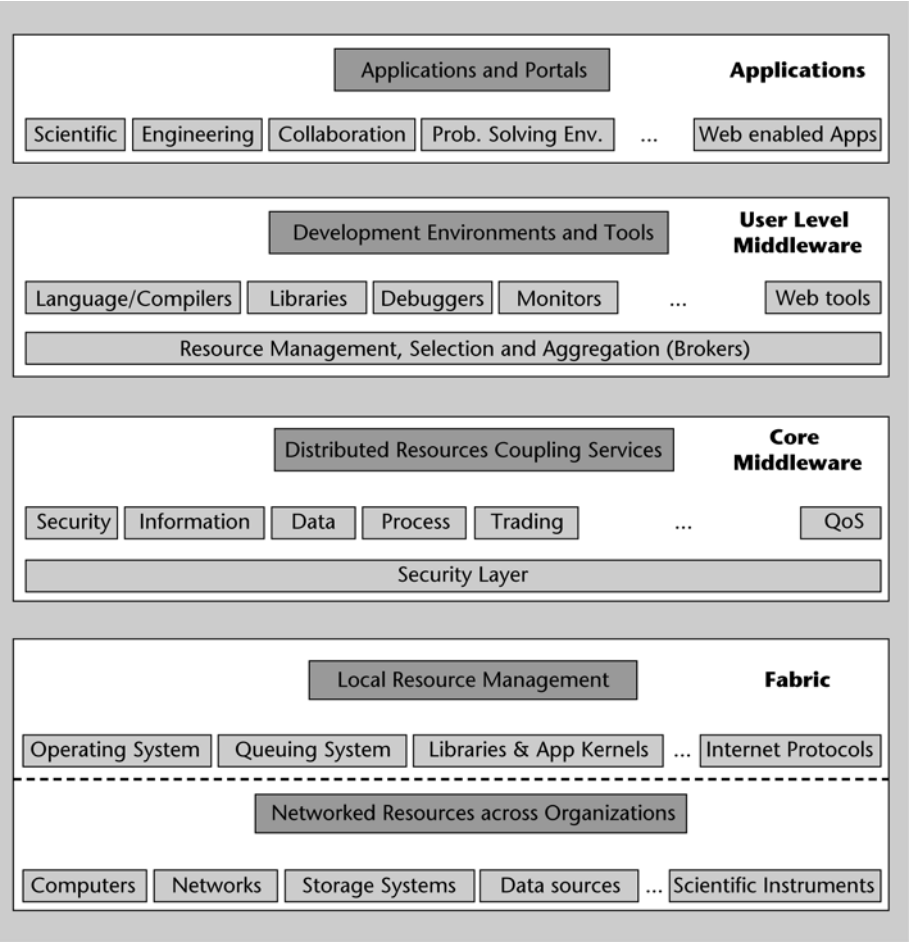

Figura 20. Aquitectura del *grid*

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 21 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.

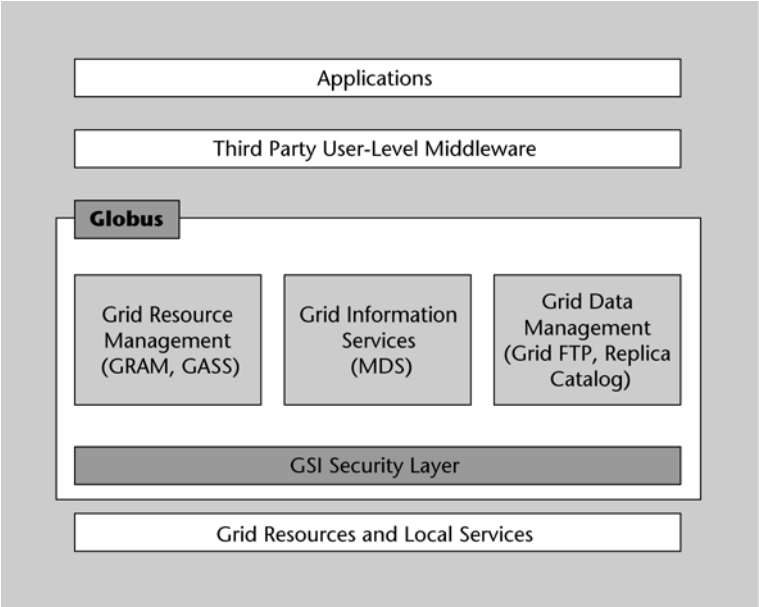

Figura 21. Arquitectura de Globus

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.

4.1.3. Sistemes computacionals usant recursos dels extrems d'Internet

Aquest tipus de sistemes es caracteritzen per agregar la capacitat computacional dels extrems d'Internet. També es coneixen com a *sistemes computacionals voluntaris* o *Grid d'escriptori* (Desktop Grid). Per extrems d'Internet entenem tots aquells ordinadors d'usuaris connectats a la xarxa.

La tecnologia més rellevant en aquest àrea és BOINC (Berkeley Open Infrastructure for Network Computing) que va ser desenvolupada per la universitat de Berkeley. BOINC és un middleware que permet als ordinadors dels extrems d'Internet interconnectar-se i crear una xarxa que agrega les capacitats de computació no usades. L'arquitectura de BOINC consisteix en un sistema servidor i un conjunt de clients que es comuniquen entre ells per tal de distribuir, processar i retornar resultats de computació.

Perquè ens fem una idea de la capacitat de computació de BOINC, l'any 2007 hi havia mes de 430.000 computadors actius arreu del món processant en mitjana 663 TFLOPS. Aquesta capacitat de computació superava amb escreix la capacitat del super-computador més potent en el moment (BlueGene/L de IBM) que oferia 360 TFLOPS. Malgrat això aquesta comparació no és del tot real, ja que no s'ha tingut en compte que els processadors de BlueGene són dedicats exclusivament a aquesta tasca.

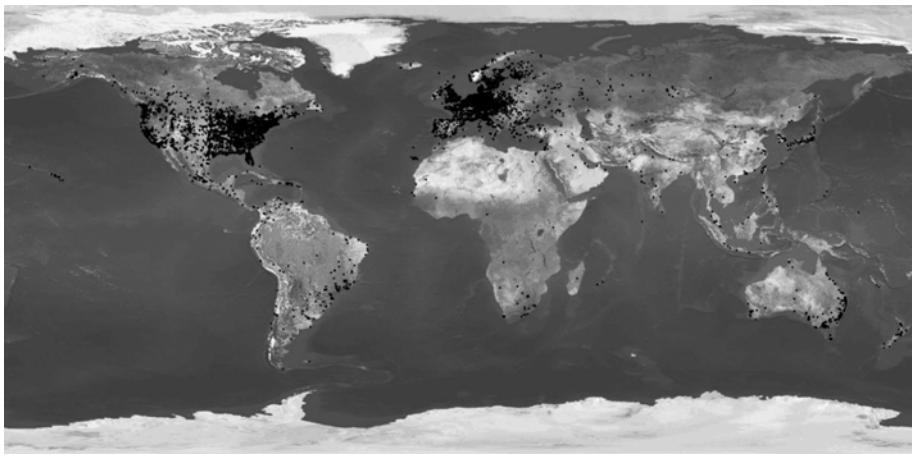

Figura 22. Distribució dels clients de BOINC arreu del món.

L'aplicació de BOINC més representativa és Seti@Home. Un sistema d'igual a igual que consisteix en el processament de senyals de radio per buscar una prova d'intel·ligència extraterrestre. És el primer intent de computació distribuïda realitzat amb èxit i en què participen voluntaris de tot el món.

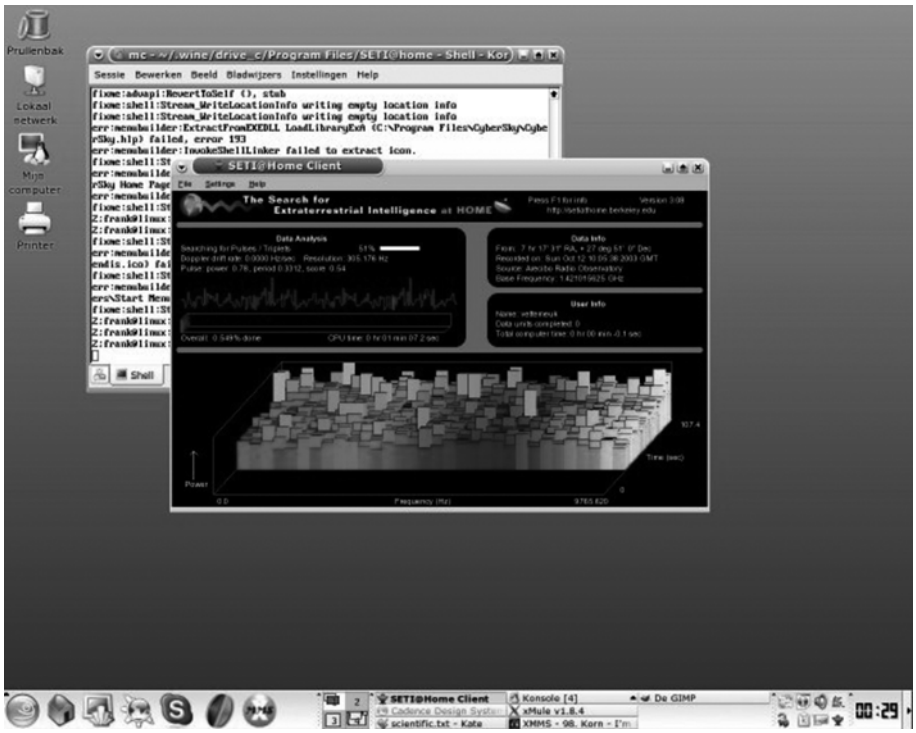

Figura 23. Aplicació seti@home

4.2. Sistemes d'informació distribuïts

Un Sistema d'Informació és un programari que administra dades d'algun aspecte del món real amb una finalitat especifica. Com a aspecte del món real entenem per exemple el procés de cadenes genètiques, sistemes bancaris, continguts de pagines web i com finalitat especifica entenem l'emmagatzematge, computació, extracció d'informació etc.

Els aspectes més rellevants a tenir en compte en el desenvolupament d'un sistema amb aquestes característiques són els següents:

- L'emmagatzematge de les dades: les dades han de ser emmagatzemades durant períodes de temps. A més s'ha de tenir en compte que les operacions sobre les dades poden ser complexes i que la quantitat de dades acostuma ser gran.

- Els sistemes han d'oferir funcionalitats per permetre la interpretació de les dades i l'extracció de coneixement.

El principal enfocaments per desenvolupar sistemes que administren gran quantitat de dades i suporten multitud d'usuaris són els sistemes basats en transaccions.

Els sistemes basats en transaccions han esdevingut claus a l'hora de permetre distribuir la informació en sistemes de gran escala. Aquests sistemes normalment s'organitzen en components amb funcionalitats determinades. El

component més representatiu és el gestor de transaccions (*Transaction processing,* monitor en anglès) que s'encarrega de la gestió de les operacions sobre les dades. El gestor de transaccions, malgrat que a la següent figura es mostra com un únic components, pot estar distribuït en un clúster, per exemple.

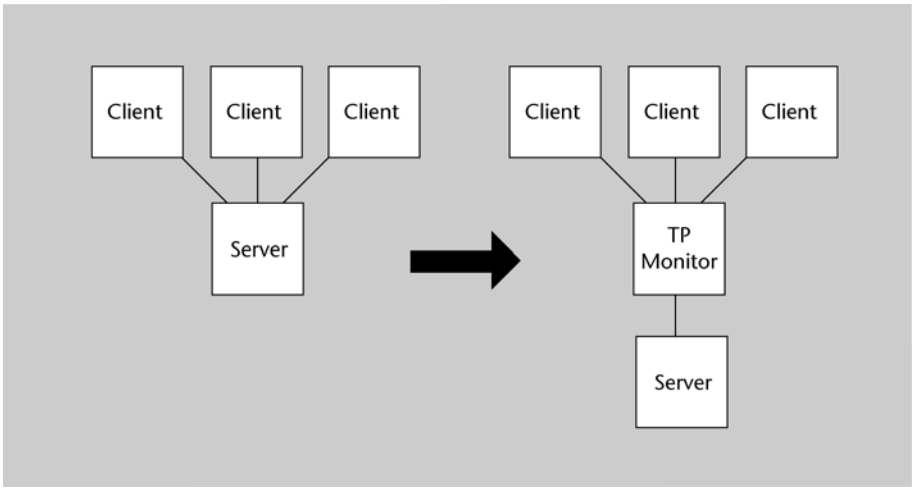

Figura 24. Evolució del model client servidor clàssic al model basat en transaccions.

Les transaccions niuades (*nested transactions*, en anglès) permeten distribuir les transaccions en una base de dades distribuïda, repartint la càrrega entre diferents nodes i permetent computació paral·lela. La necessitat de gestors de les transaccions es fa evident i és una de les problemàtiques afrontar per els dissenyadors d'aquests sistemes.

En molts casos la integració d'aplicacions es fa aquest nivell, és a dir, dissenyant un gestor capaç de distribuir les transaccions en diferents bases de dades o en una de distribuïda. La integració d'aplicacions fa aparèixer necessitats d'inter-comunicació i inter-operabilitat entre aplicacions. Fruit d'aquesta necessitat neixen *middlewares* de comunicació com els coneguts RPC o RMI que faciliten la comunicació entre aplicacions. Un dels inconvenients de RPC i RMI és que es requereix que les aplicacions que es comuniquen estiguin en funcionament durant la comunicació i que, a més, cada una conegui com referir-se a l'altre. Aquests inconvenients han fet que el paradigma de la publicació-subscripció també hagi estat usat a l'hora d'integrar aplicacions.

4.3. Sistemes distribuïts omnipresents*

* *Pervarsive Distributed Systems*, **en anglès**

Els sistemes distribuïts omnipresents o sistemes computacionals ubiqües són sistemes heterogènies formats computadors i d'altres dispositius electrònics com sensors, equips multimedia i d'altres sistemes electrònics amb funcionalitats específiques. Aquests sistemes es caracteritzen per la seva omnipresencialitat a diferencia dels sistemes computacionals estudiats fins ara que es caracteritzen per la seva estabilitat.

Els avenços de les xarxes inalàmbriques, l'ús dels dispositius mòbils i els sistemes integrats han permès desenvolupar aplicacions que donen solució a problemes fins avui no abordables. Els camps principals d'estudi han estat: la física, la medicina, les telecomunicacions, la protecció civil i també la indústria armamentística.

Els sistemes distribuïts omnipresents estan formats per conjunts de nodes amb capacitats computacionals i sensitives que permeten obtenir i processar informació de l'entorn. Els nodes del sistema estan interconnectats mitjançant xarxes inalàmbriques que pateixen en molts casos d'alts nivells de dinamisme i mobilitat dels seus nodes i en les que generalment no hi ha un administrador.

Els nodes d'un sistema ubiqua tendeixen a tenir comportaments autònoms i autogestionats que requereixen de mecanismes per detectar canvis en l'entorn i poder reaccionar en conseqüència.

Els sistemes distribuïts omnipresents es caracteritzen pels requeriments següents:

- Detectar canvis en l'entorn.
- Afavorir la formació de xarxes ad-hoc.
- Compartir informació.

Una aplicació dels sistemes ubiques ha estat en l'àmbit de la salut (*Electronic Health Care Systems*, en anglès). Les aplicacions desenvolupades tenen com a funcionalitat monotoritzar i tenir cura d'un pacient per tal de prevenir la seva hospitalització. Normalment estan formats per conjunts de sensor distribuïts pel cos del pacient en el que s'anomenen en anglès *body-area networks* (BAN). Els sensors monitoritzen als pacients i envien les dades mitjançant una connexió inalàmbrica a un centre de procés. En molts casos, aquests sistemes requereixen del procés de la informació dins la pròpia xarxa ja que per una banda la quantitat d'informació obtinguda pels sensors pot ser molt gran i per altra la capacitat de transmissió pot estar limitada entre d'altres per l'ampla de banda o les limitacions energètiques dels dispositius.

Un segon exemple d'aplicacions els trobem en la domòtica i els sistemes multimèdia a la llar. Aquests sistemes normalment fan ús de les xarxes d'àrea local de la llar i estan integrats per ordinadors personals, dispositius mòbils com telèfons i PDA i sistemes d'àudio i vídeo com televisions i dispositius de joc.

Finalment les xarxes de sensors (de l'anglès, *sensor networks*) són xarxes de dispositius electrònics amb capacitat de computació (anomenats nodes), equipats amb sensors, que col·laboren en una tasca comuna. Les xarxes de sensors estan formades per un grup de sensors amb certes capacitats sensitives i de co-

municació sense fils les quals permeten formar xarxes ad hoc sense infraestructura física preestablerta ni administració central. La seva principal funcionalitat és la d'adquirir i tractar dades.

Aquestes es caracteritzen per la seva facilitat de desplegament i per ser autoconfigurables. Els nodes de la xarxa es comporten en tot moment com a sistemes emissors i receptors, oferint serveis d'encaminament a d'altres nodes sense visió directa, així com enregistren dades capturades pels sensors locals de cada node. Cal destacar la necessitat de la gestió eficient de l'energia, que els ha de permetre una alta taxa d'autonomia i ha d'evitar que la seva carència les faci poc operatives.

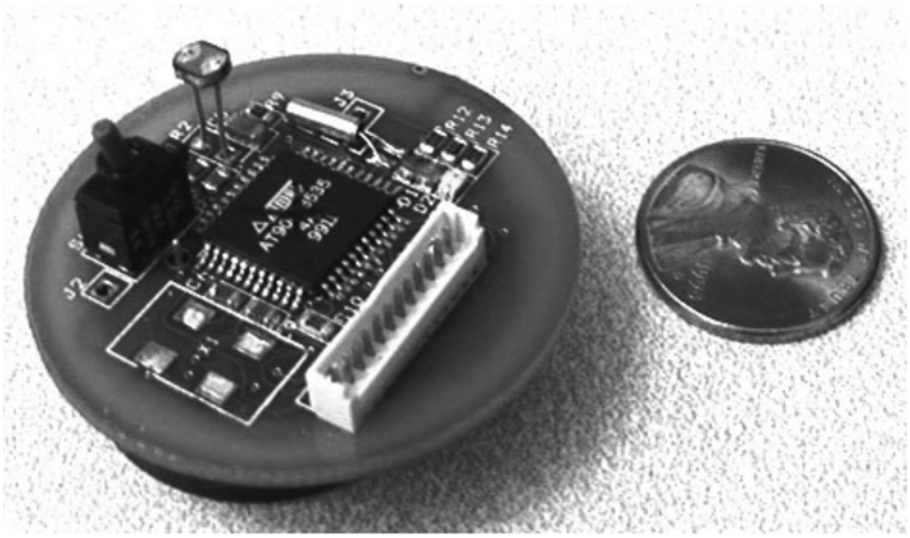

Figura 25. Mica Dot. Un node desenvolupat per la universitat de Berkeley.

La recerca actual em l'àmbit de les xarxes de sensors se centra bàsicament en la minimització del consum d'energia en la transmissió de dades. Com ja s'ha dit anteriorment, la transmissió de dades es costosa energèticament i l'energia és el recurs més escàs. Tècniques com l'agregació d'informació i el procés de la informació en cada node de forma autònoma són molt usades per tal de minimitzar la quantitat d'informació transmesa. A més en alguns casos, el manteniment d'una topologia de xarxa, encara que costós, permet reduir l'energia consumida en el procés de transmissió de la informació.

Resum

En aquest mòdul, hem explicat alguns models per classificar els sistemes distribuïts. Hem començat presentant els estils arquitectònics més usats a l'hora de dissenyar sistemes i aplicacions distribuïdes, i hem vist que els estils ens permeten descriure la forma com s'organitzen els components lògics d'un sistema distribuït.

També hem vist un altre tipus de classificació basada en la relació entre els components lògics dels sistemes distribuïts. Hem observat que els seus components poden organitzar-se de forma centralitzada, on el paradigma principal és el de client-servidor (avui en dia encara segueix sent el més usat a Internet); descentralitzades, on hem estudiat tant els sistemes no estructurats, com els estructurats i els híbrids. Dels sistemes descentralitzats no estructurats s'han explicat els mecanismes de cerca, i mecanismes per al manteniment de la informació del sistema. Pel que fa als sistemes estructurats hem vist com s'organitzen els nodes per tal que la topologia de la xarxa sigui estructurada. Finalment hem tractat els sistemes híbrids que aprofiten alguns dels avantatges dels models anteriors.

S'han tractat altres enfocaments com el model de publicació-subscripció que 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.

Per acabar hem vist alguns dels diferents àmbits d'aplicació dels sistemes distribuïts. Els sistemes computacionals poden ser clúster caracteritzats per tenir una funció dedicada. Sistemes basats en aprofitar els recursos dels extrems d'Iternet, i com a principal exemple l'aplicació Seti@Home basada en la tecnologia BOINC. Els sitemes Grid, del quals 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 extern, i es pugui pagar per l'ús en lloc de per la propietat. Del *grid*, n'hem vist una organització típi-

ca 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*.

Els sistemes d'Informació distribuïts són també aplicacions molt esteses en el món empresarial, tant per a sistemes bancaris, com per sistemes de comerç electrònic. Hem vist que en aquests sistemes és important garantir la consistència de les dades i que fan ús de monitors de transaccions per tal de garantir la consistència de la informació de les bases de dades. També hem vist que els monitors de transaccions són molt útils a l'hora d'integrar diferents aplicacions.

També hem tractat els sistemes distribuïts omnipresents, dels quals hem destacat les seves aplicacions en medicina i domòtica. Un altre tipus de sistemes que s'han tractat són les xarxes de sensors de les quals se'n fan usos tant diversos com la de prevenció de riscos naturals, usos militars, monitorització i control del trànsit, etc. Hem vist que un aspecte molt important per aquestes xarxes és la minimització de la despesa energètica.

Ja per acabar, 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.

Activitats

Per a la xarxa Chord de la figura, feu el següent:
a) Mostreu les taules d'encaminament dels nodes N8, N12, N1 i N56 quan entri l'N12.
b) Mostreu els salts fets quan en el node N56 volem obtenir la clau K13.
c) Mostreu els salts fets quan en el node N56 volem obtenir la clau K37.
d) Mostreu els salts fets quan en el node N56 volem obtenir la clau K21.

Solucionari

a)
N8+1 12
N8+2 12
N8+4 12
N8+8 21
N8+16 32
N8+32 42
N12+1 14
N12+2 14
N12+4 21
N12+8 21
N12+16 32
N12+32 48
N1+1 8
N1+2 8
N1+4 8
N1+8 12
N1+16 21
N1+32 38
N56+1 1
N56+2 1
N56+4 1
N56+8 1
N56+16 8
N56+32 32

b) 56->8->12->14

c) 56->32->38

d) 56->8->21

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.

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.

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.

Articles

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

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

Lua, E. K. i altres (2005). "A survey and comparison of peer-to-peer overlay network schemes". *IEEE Communications Surveys&Tutorials* (vol. 7, núm. 2).
És un article en què trobareu un resum de com funcionen les DHT més populars, així com una comparativa entre elles.

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