Software Distribuït - T9 - Sistemes Peer-To-Peer

Eloi Puertas i Prats

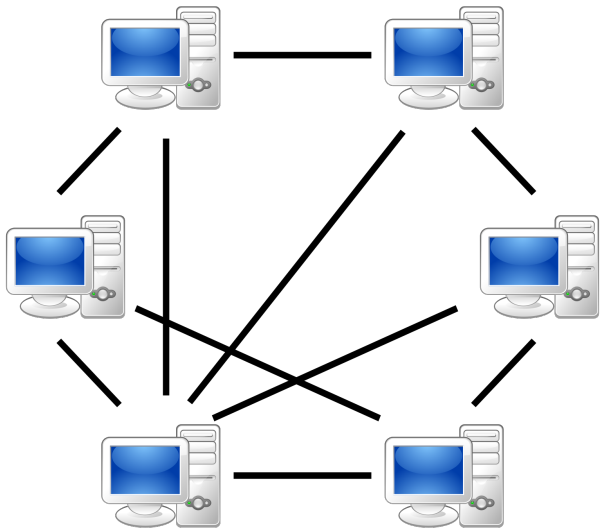
Universitat de Barcelona
Grau en Enginyeria Informàtica

28 d'abril de 2020

UNIVERSITAT DE BARCELONA

Introducció al P2P

- Paradigma d'arquitectura de sistema distribuïts i aplicacions, on cada node contribueix amb dades i recursos per tal d'aconseguir un servei uniforme.
- S'han usat sobretot pel compartiment d'arxius (napster, bitTorrent,), distribució d'informació, web caching, bitcoins, SETI...



Característiques P2P

- Cada usuari contribueix amb recursos al sistema.
- Tots els nodes en el sistema tenen les mateixes funcions i responsabilitats.
- Tècnicament, cada peer ha de comunicar-se amb qualsevol peer, sense haver de necessitar un server.
  - Aquestes aplicacions purament P2P, però, són rares. Solen usar algun element basat en client-servidor.
- Cada peer ha de poder localitzar els demés peers:
  - DNS, un directori on poder buscar als demés peers, aproximació c/s. (ex. napster)
  - **Taula hash distribuïda (DHT)**. Cada peer manté una part de la hash actualitzada amb els seus peers *propers*.

UNIVERSITAT DE BARCELONA

Tipus P2P

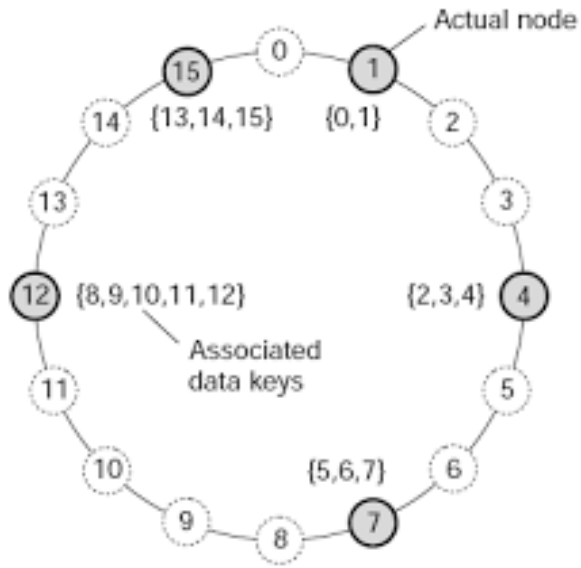
- Model centralitzat: utilitza un master índex (c/s)
- Distribuït pur: nodes estan distribuïts aleatòriament
- Hibrid: Existeixen super nodes amb màsters i nodes aleatoris.
- Estructurat: Es basen en taules de hash distribuïdes (DHT)

Taula Hash Distribuïda: Característiques

- Ens permet autonomia i descentralització. Els nodes col·lectivament formen el sistema sense cap coordinació centralitzada.
- Tolerància a fallades. El sistema ha de ser fiable, fins i tot si els nodes entren i surten contínuament.
- Escalabilitat. El sistema ha de funcionar de forma eficient fins i tot amb milers o milions de nodes.
- El concepte clau en la DHT és que els nodes només s'han de coordinar amb uns pocs altres nodes del sistema, normalment de l'ordre de $log(n)$. Per tant, pocs nodes són afectats quan hi ha canvis en el sistema.

Taula Hash Distribuïda: Estructura

- **Esquema de partició de l'espai de claus:** Atorga a cada node un rang contigu de claus.
  - **Funció de Hash consistent**. Té la propietat essencial de que afegir o esborrar un node tan sols canvia el conjunt de claus que pertanyen als nodes veïns, deixant la resta sense alterar.
  - Es defineix una funció de distància entre claus, la qual no té res a veure amb la distancia geogràfica o de latència. P.ex. distància entre punts sobre la circumferència.



Taula Hash Distribuïda: Estructura

- **Xarxa superposada (*overlay network*):** És com els nodes es connecten entre ells formant una xarxa que els permet trobar el propietari d'una clau concreta en l'espai de claus.
  - Cada node manté un conjunt d'enllaços cap als seus nodes veïns o cap a taules d'enrutament. Aquests enllaços formen la xarxa superposada.
  - El veïnatge dels nodes està definit segons una certa estructura, anomenada topologia de la xarxa.
  - Algorisme d'enrutament per trobar els nodes veïns: P.ex. Nearest Neighbour.
  - Algorismes que exploten l'estructura de xarxa superposada per a fer broadcasts. Són utilitzats per aplicacions per a fer multicast a la xarxa p2p, querys, estadístiques...

Seguretat P2P

- Ens hem d'assegurar que el peer és realment qui diu ser.
  - En registrar-se al servidor, aquest li ha de proporcionar una clau pública i el peer presentar-la en cada iteració.
- Ens hem d'assegurar que les transaccions siguin segures.
  - Fer servir criptografia per encriptar els missatges que s'intercanvien els peers

P2P Middleware i Frameworks

- Chord. (DHT)
- Pastry (DHT)
- CAN (DHT)
- KADemlia (DHT) (*)
- JXTA. (Framework).

Exemples P2P

- BitTorent
- BlockChain
- Napster (Hibrid)
- Skype

Exemple P2P: BitTorrent

- BitTorent és un sistema de distribució de continguts multimèdia.
  - Fitxers molt grans com pel·lícules, dvds, mp3...
- Centrat en la descàrrega de fitxers, no en la cerca.
  - Es distribueix el mateix fitxer a tots els nodes (peers)
  - Un sol peer publica el fitxer.
  - Múltiples peers se'l poden descarregar.
  - Els peers comparteixen les peces que ja s'han descarregat.
- Es fa servir l'estratègia *quid pro quo*
  - *Compartiré amb tu, si tu comparteixes amb mi*
  - De tant en tant, però, es deixa a peers *egoistes* descarregar.
  - Necessari per a poder començar el procés de descàrrega.

BitTorrent: Funcionament

1. Localitzar fitxer de metadata, anomenat .torrent, per exemple, via Google.
2. Aquest fitxer conté la longitud, el nom, valor de hash i URL del tracker (servidor)
3. Fer petició al tracker, afegir-se a un torrent.
4. El Tracker selecciona una llista aleatòria de peers per a fer la descàrrega:
   - **Seeds:** Peers que tenen tot el fitxer.
   - **Leechers:** Peers que encara els hi falta peces per a descarregar.
5. Contactar amb els peers de la llista per a demanar peces.
6. Formar part de la xarxa P2P i participar-hi també.

Peces BitTorrent

- Cada fitxer es descomposa en peces.
  - Normalment les peces són de 256Kbytes.
  - Les peces es descarreguen en paral·lel.
  - Avisar de les peces rebudes a la llista de peers pròpia.
  - S'envien peces a peers mentre es segueix descarregant d'altres.
- Selecció de les peces per a descarregar.
  - Al principi de la descàrrega, es seleccionen peces aleatòriament.
  - Es prioritzen les peces més poc freqüents en la xarxa, així n'hi hauran més per als demés.

Peces BitTorrent

- Selecció de les peces a enviar.
  - Periòdicament es calcula la taxa de descàrrega de les diferents peces que s'estan enviant.
  - Es seleccionen fins a un total de 4 peers per a enviar-lis la peça tal que tinguin la màxima taxa de descàrrega.
- Optimistic Upload
  - Periòdicament (cada 30 segons) es selecciona un peer aleatori i se li envia una peça que no tingui en ell.
  - Es segueix buscant contínuament als peers que siguin més ràpids.

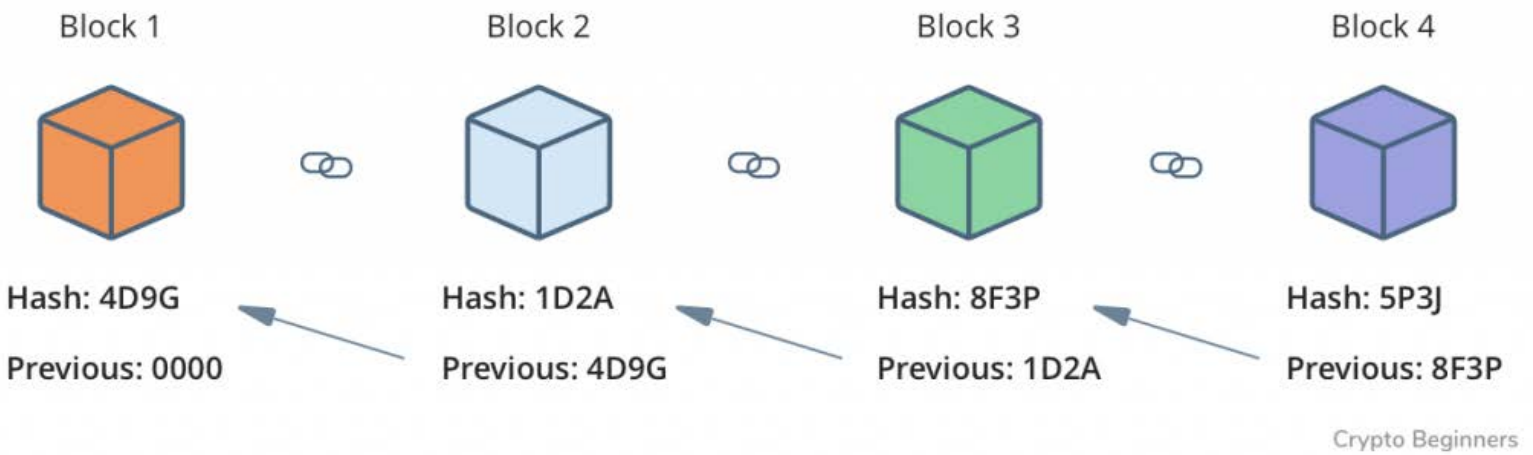
Exemple P2P: BlockChain

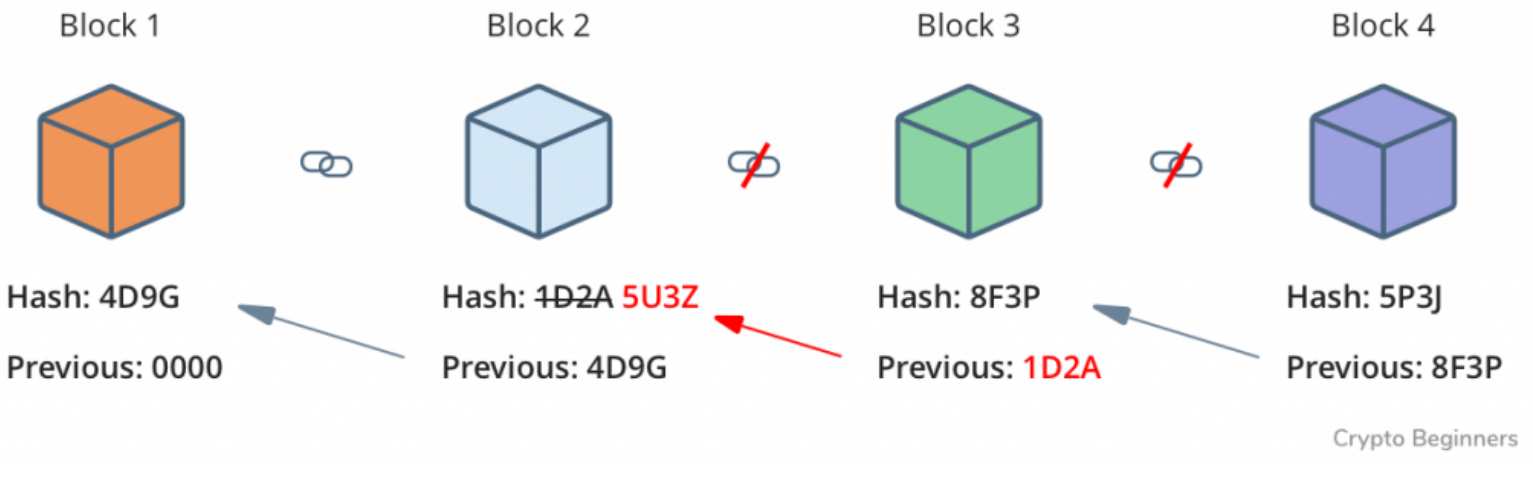
- BlockChain és com un llibre de comptabilitat (*ledger*) obert a tothom.
- Famós per fer-se servir en les criptomodenes. (2009 Satoshi Nakamoto, Bitcoins)
- Cada transacció està encriptada i assegurada amb una firma digital que prova la seva autenticitat.
- Quan s'afegeix un registre al blockchain es molt difícil poder alterar-lo.
- Útil per portar la traçabilitat de qualsevol tipus de procés.

BlockChain: Funcionament d'una transacció

1. Cada participant en la transacció te una clau pública/privada per poder fer una signatura digital i poder provar la seva identitat així com encriptar informació.
2. Es crea el bloc, que conté la informació rellevant de la transacció signada digitalment, així com un *hash* del bloc anterior. S'hi afegeix el *hash* del bloc. Per evitar la creació massiva de blocs en una cadena s'ha de fer una *proof of work*, és a dir, ha de costar temps fer-ho, com per exemple *minar*
3. Aquest bloc s'envia a tots els participants de la xarxa
4. Els participants verifiquen que el bloc sigui correcte, fent el *hash* i comparant-lo amb el que han rebut.
5. S'arriba a un consens si el 51% dels participants donen com a bo el bloc i s'acaba afegint el bloc a la cadena. En cas contrari es descarta

BlockChain: Seguretat de la cadena





Paradigma d'Objectes Distribuïts

- Dóna abstracció sobre els sistemes clàssics client-servidor.
- Es basa en Programació Orientada a Objectes.
- Està orientat a funcionalitats i no a l'intercanvi de missatges.
- Simplificació de protocol i gestió de trames.
- Simplificació en el tractament d'errors.

Comparativa Web Services vs Objectes distribuïts (RMI)

- Interacció entre client i servidor molt similar:
  - RMI: el client usa referència remota per invocar una operació sobre l'objecte.
  - WebService: el client usa una URI per invocar una operació sobre el recurs.
- WebService està molt lligat a arquitectura client-servidor. Per crear una aplicació distribuïda més complexa tipus P2P o on es necessiti la col·lobaració de varis sistemes, objectes distribuïts és més addient.

Conceptes fundamentals Objectes Distribuïts

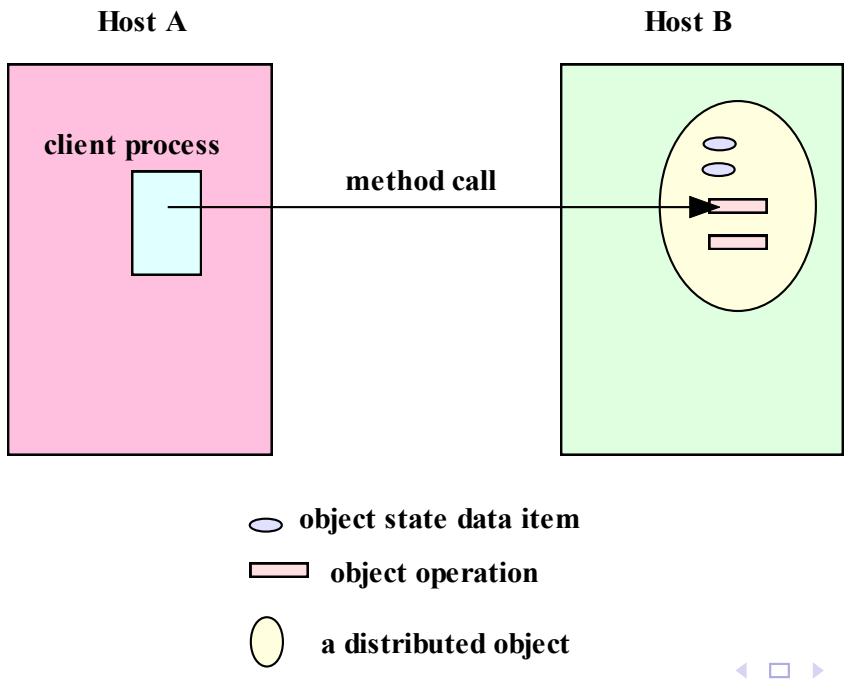
Objecte remot Aquells objectes que algun dels seus mètodes es poden invocar des d'un altre procés corrent en una màquina remota.

Objecte local Aquells objectes que només poden ser invocats per un procés local en el qual l'objecte existeixi.

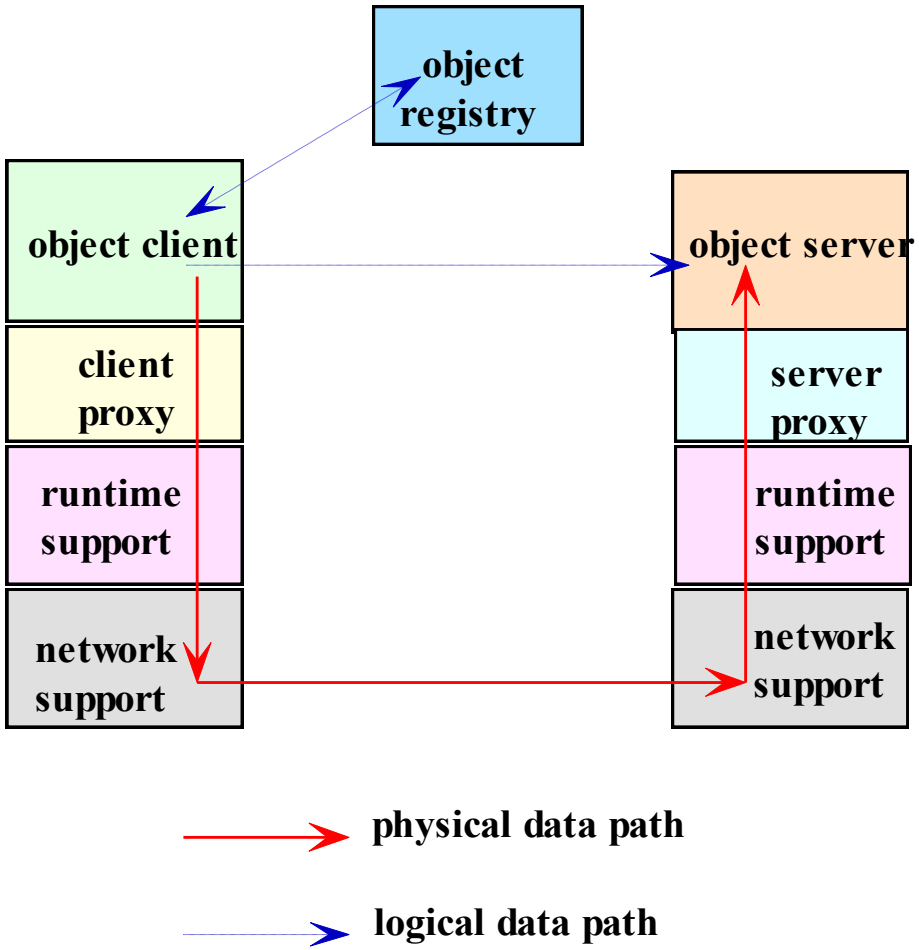
UNIVERSITAT DE BARCELONA

Esquema conceptual de funcionament

En un paradigma d'objectes distribuïts, els recursos de xarxa estan representats per objectes distribuïts. Per sol.licitar el servei d'un recurs de xarxa, un procés invoca alguna de les seves operacions o mètodes, passant dades com a paràmetres al mètode. El mètode s'executa a la màquina remota, i la resposta s'envia de nou al procés de sol.licitud com a valor de retorn.



Framework Genèric d'Objectes Distribuïts



Framework Genèric d'Objectes Distribuïts

- Un objecte distribuït és proporcionat, o exportat, per un procés (Servidor d'objectes).
- Un registre d'objectes, ha de ser present en l'arquitectura del sistema per tal de distribuir l'objecte remot.
- Per accedir a un objecte remot, un procés busca una referència remota (handle) en el registre de l'objecte que vol fer servir. Aquesta referència és utilitzada per l'objecte client per realitzar crides als mètodes remots.

Frameworks d'objectes distribuïts

El paradigma d'objectes distribuïts ha estat àmpliament adoptat en les aplicacions distribuïdes, per a això un gran nombre de mecanismes basats en el paradigma estan disponibles. Entre els més coneguts d'aquests mecanismes són:

- Java Remote Method Invocation (RMI).
- Common Object Request Broker Architecture (CORBA).
- Distributed Component Object Model (DCOM)
- Mecanismes que ofereixin el Simple Object Access Protocol (SOAP).

D'aquestes, la més simple és el RMI de Java

UNIVERSITAT DE BARCELONA

JAVA RMI

- L'objectiu principal de RMI és que els programadors desenvolupin aplicacions distribuïdes amb Java usant la mateixa sintaxi i semàntica que les aplicacions Java tradicionals.
- L'aplicació distribuïda s'executarà en vàries màquines virtuals simultàniament.
- L'arquitectura RMI defineix:
  - com es comporten els objectes,
  - quan i com poden ocórrer excepcions,
  - com es gestiona la memòria
  - com es passen els paràmetres a mètodes remots i com es reben els resultats

En quins aspectes ens ajuda JAVA RMI

En la majoria d'aplicacions distribuïdes apareixen 5 aspectes a codificar:

1. Lògica d'aplicació
2. Interfície d'usuari
3. Preparació de dades per a l'enviament a altres processos
4. Codi que llança l'aplicació (permet establir la connexió entre client i servidor)
5. Codi que intenta fer una aplicació distribuïda més robusta i escalable: caches en el client, serveis de noms, balanceig de càrrega, ...

L'RMI ens ajuda amb la generació automàtica de part del codi dels punts 3 i 4.

Conceptes fundamentals RMI

Objecte remot Aquells objectes que algun dels seus mètodes es poden invocar des d'un altre procés corrent en una màquina remota.

Objecte local Aquells objectes que només poden ser invocats per un procés local en el qual l'objecte existeixi.

Servidor d'objectes La màquina virtual que té instanciats els objectes remots

Interfície remota Interfície Java que declara els mètodes d'un objecte remot

Invocació de mètode remot Acció d'invocar un mètode d'una interfície remota en un objecte remot.

RMIregistry Servidor de noms o registre d'objectes remots. Permet localitzar objectes remots a partir del seu nom

Utilitzant objectes remots

Per utilitzar un objecte remot, hem de saber com:

- Obtenir una referència remota.
- Crear un objecte remot i deixar-lo accessible.
- Invocar un mètode remot.

Com obtenir una referència remota o aixecar-se tibant d'una llengüeta

Quan una màquina virtual s'inicia no té cap referència remota al seu abast. Per poder comunicar-se amb l'exterior ha de buscar la primera referència remota utilitzant un servei de directori per localitzar un objecte remot a partir del seu nom. Un cop haguem localitzat el primer objecte remot és possible que aquest ens envïi noves referències remotes com valors de retorn de la crida d'algun dels seus mètodes remots (*Callback*).

El servei de directori: rmiregistry

L'API de RMI ens permet fer servir diferents serveis de directori per registrar un objecte distribüit Utilitzarem un servei de directori simple anomenat rmiregistry, que es proporciona amb el programari de Java Development Kit (SDK). El RMIRegistry és un servidor que s'ha d'executar a la màquina del servidor d'objectes, per convenció i per defecte en el port TCP 1099.

$>:/\$$**rmiregistry**

S'ha d'engegar abans d'intentar utilitzar-lo en el mateix path que el servidor d'objectes, o l'intent d'utilitzar-lo fracassarà.

El registre estarà actiu fins que es mati el procés via CTRL-C, per exemple.

Accés al registre de noms des d'un programa Java

El rmiregistry també es pot engegar de forma dinàmica des de la classe servidor d'objectes:

```
import java.rmi.registry.LocateRegistry;
...
LocateRegistry.createRegistry (1099);
```

La classe que dóna accés al registre és java.rmi.Naming. Llegir javadoc

Mètodes de rmi.Naming

El servei de noms és, doncs, fonamantalment un directori tipus *guia de telèfons* en el que es pot:

- Afegir i associar un nou objecte remot a un nom (mètode **bind**)
- Eliminar un objecte remot ja existent (mètode **unbind**)
- Modificar l'associació d'un nom existent a un altre objecte remot (mètode **rebind**)
- Cercar un objecte remot per nom (mètode **lookup**)
- Obtenir totes les associacions (mètode **list**)

L'API del RMI JAVA

Un cop ja tenim solucionat com obtenir referències remotes, hem de saber com:

a) crear objectes remots i deixar-los accessibles

b) Invocar un mètode remot

Necessitarem els següents elements:

- Interfície remota tant al costat del servidor com del client.
- L'aplicació del costat del servidor
  - Implementació de la interfície remota
  - Instancia Objecte Remot i el deixa disponible al servei de directori.
- L'aplicació del costat del client, invoca mètodes remots.

La interfície remota

Una interfície remota Java és una interfície que hereta de la interfície marcador Remote de JAVA (no implementa cap mètode en concret).
Especifica els mètodes remots que un objecte remot oferirà.
L'ús d'interfíces facilita la crida de mètodes d'objectes remots, ja que el client només necessita de la classe interfície remota i no pas de tota la implementació.
Tot mètode remot ha llançar una excpeció `java.rmi.RemoteException` ja que poden haver-hi causes de posible error com ara:

- Error en les comunicacions
- Error al empaquetar / desempaquetar paràmetres
- Error de protocol.

Exemple Interfície remota

```
package es.ub.gei.sd.dateserver;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.Date;

public interface DateServer extends Remote {
public Date getDate() throws RemoteException;
}
```

L'aplicació del costat del servidor

Un servidor d'objecte distribuït és un objecte que proporciona els mètodes i la interfície d'un objecte distribuït. Cada servidor d'objecte:

- ha d'implementar cada un dels mètodes remots especificats en la interfície
- registrar l'objecte que conté la implementació dins del servei de directori.

Es recomana que les dues parts s'ofereix com classes separades.

La implementació de la interfície remota

S'ha de codificar una classe que implementi la interfície remota. Aquesta classe ha d'extendre d' **UnicastRemoteObject**, o si no s'haurà d'exportar amb el mètode `UnicastRemoteObject.exportObject(Remote obj)`

```
package es.ub.gei.sd.dateserver;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.util.Date;
public class DateServerImpl extends UnicastRemoteObject
implements DateServer {
public DateServerImpl() throws RemoteException {}
public Date getDate() throws RemoteException {
return new Date();
}
}
```

El servidor d'objecte remot

La classe del servidor d'objecte remot és una classe que té el codi que crea i exporta un objecte remot.

```
package es.ub.gei.sd.dateserver;
import java.rmi.Naming;
public class Server {
public static void main(String[] args) throws Exception {
int portNum=1099;
DateServerImpl dateServer = new DateServerImpl();
// This method starts a RMI registry on the local host, if it
// does not already exists at the specified port number.
startRegistry(portNum);
// register the object under the name DateServer in localhost and portNum
registryURL = "rmi://localhost:" + portNum + "/DateServer";
Naming.rebind(registryURL, dateServer);
System.out.println("Some Server ready.");
}
}
```

El servidor d'objecte remot

- Quan un servidor d'objectes s'executa, l'exportació d'objecte distribuït fa que el procés del servidor comenci a escoltar i esperi que els clients es connectin i sol·licitin el servei de l'objecte.
- Un servidor d'objectes RMI és un servidor concurrent: cada sol·licitud d'un client a un dels mètodes de l'objecte es pot fer en un fil separat. Tingueu en compte doncs que si un procés client invoca diverses crides a mètodes remots, aquestes crides es poden executar simultàneament. Igualment, en la banda del servidor cal tenir en compte que diferents clients poden estar executant el mateix servei. Cal posar sistemes de sincronització allà on calgui.

L'aplicació del costat del client

L'aplicació client és com qualsevol altra classe de Java. La sintaxi de RMI necessària és la que fa referència a:

- Localitzar el registre RMI en l'equip servidor,
- Buscar la referència remota de l'objecte remot,
- Convertir a la classe d'interfície remota i invocar els seus mètodes remots.

L'aplicació del costat del client

```
package es.ub.gei.sd.dateclient;
import java.rmi.Naming;
import java.util.Date;
import es.ub.etis.ppx.dateserver.DateServer;
public class DateClient {
public static void main(String[] args) throws Exception {
if (args.length != 1)
throw new IllegalArgumentException("Syntax: DateClient <hostname:port>");
DateServer dateServer = (DateServer) Naming.lookup("rmi://"+args[0]+"/DateServer");
Date when = dateServer.getDate();
System.out.println(when);
}
}
```

L'aplicació del costat del client

- La referència de la interfície remota s'ha d'utilitzar per invocar qualsevol dels mètodes remots.
- Noteu que la sintaxi de la invocació dels mètodes remots és la mateixa que per als mètodes locals.
- ALERTA És un error comú convertir la referència obtinguda des del registre a la classe d'implementació de la interfície en comptes de la classe d'interfície.

Passos per crear una aplicació RMI client/servidor

En l'aplicació del client

- Interficies d'Objectes Remots.
- Codi Aplicació Client
  - Main
  - GUI
  - Crides a mètodes remots

En l'aplicació del servidor

- Interficies d'Objectes Remots
- Implementacions d'Objectes Remots
- Codi Aplicació Servidor
  - Main
  - Instanciació d'Objectes Remots
  - *(Engegar RMIRegistry des de l'aplicació)
  - Publicitar Objectes Remots al servidor de directori.

UNIVERSITAT DE BARCELONA

Passos per arrencar una aplicació RMI client/servidor

En la màquina Servidor

- *(Engegar rmiregistry)
- Executar l'aplicació del servidor

En les màquines Clients

- Executar l'aplicació del client.

ALERTA Cal vigilar que el rmiregistry estigui executant-se en el mateix path que el servidor, en cas contrari cal tenir-ho en compte quan especifiqueu la URL `rmi://...`

ALERTA Les interfícies dels objectes remots a client i a servidor han de ser exactament iguals, vigileu si feu servir packages.

Diferències entre invocació local i remota

- Els clients d'objectes remots sempre interactuen amb interfícies remotes, mai amb les classes d'implementació d'aquestes interfícies.
- Els arguments no-remots, així com els resultats retornats per la invocació d'un mètode remot, són passats per còpia en lloc de per referència. És necessari que implementin `Serializable`.
- Els objectes remots es passen sempre per referència (mitjançant Interfície Remota), no copiant les implementacions remotes.
- El significat d'alguns dels mètodes definits en java.lang.Object s'especialitzen en el cas d'objectes remots.
- Atès que una invocació remota pot fallar per raons addicionals a les que ho fa una invocació local, en treballar amb objectes remots s'han d'atendre excepcions addicionals

Diferències entre invocació local i remota

- **Invocació local**

```
public class Cliente {...}
public class IdentificadorCliente {...}
public Cliente obtenerCliente(IdentificadorCliente id);
```

S'envia una referència local i es rep una referència local

Diferències entre invocació local i remota

- **Invocació remota amb paràmetre local i retorn remot**

```
public Interficie ICliente extends Remote{...}
public class Cliente implements ICliente {...}
public class IdentificadorCliente implements Serializable {...
public ICliente obtenerCliente(IdentificadorCliente id)
throws RemoteException;
```

*S'envia una còpia de l'objecte **id** i es rep una referència remota*

Diferències entre invocació local i remota

- **Invocació remota amb paràmetre remot i retorn remot**

```
public Interficie ICliente extends Remote{...}
public Interficie IIdentificadorCliente extends Remote{...}
public class Cliente implements ICliente {...}
public class IdentificadorCliente implements
IIdentificadorCliente{...}
public ICliente obtenerCliente(IIdentificadorCliente id)
throws RemoteException;
```

*S'envia una referència remota **id** i es rep una referència remota d' **ICliente***

Exemple Interfícies de SimpleBankSystem