Pràctiques de laboratori

Un xat distribuït orientat a objectes basat en RMI

Concurrència i Sistemes Distribuïts

Introducció

L'objectiu d'aquesta pràctica és introduir als estudiants als sistemes distribuïts i al disseny orientat a objectes d'aplicacions distribuïdes. S'utilitzará RMI com middleware distribuït subjacent, però qualsevol altre middleware que suporte objectes distribuïts podria ser utilitzat. L'aplicació seleccionada per a aprendre i practicar els sistemes distribuïts és una aplicació de Xat distribuïda. Existeixen moltes aplicacions de Xat en el mercat i també alguns estàndards. L'objectiu no és desenvolupar una aplicació completa ni seguir un estàndard en particular, sinó centrar-se en un sistema distribuït orientat a objectes clar i senzill. Aquesta pràctica també tracta alguns aspectes importants dels sistemes distribuïts no específics dels sistemes orientats a objectes, com a servei de noms, i el paradigma de client/servidor.

Una vegada completada aquesta pràctica, s'haurà après a:

- Identificar els diferents processos que conformen una aplicació distribuïda.
- Compilar i executar aplicacions distribuïdes senzilles.
- Entendre la funció de servei de noms.
- Entendre el paradigma de client/servidor i la seua extensió orientada a objectes per a dissenyar i implementar aplicacions distribuïdes.

Aquesta pràctica de laboratori ha de ser desenvolupada per **grups de dos estudiants**.

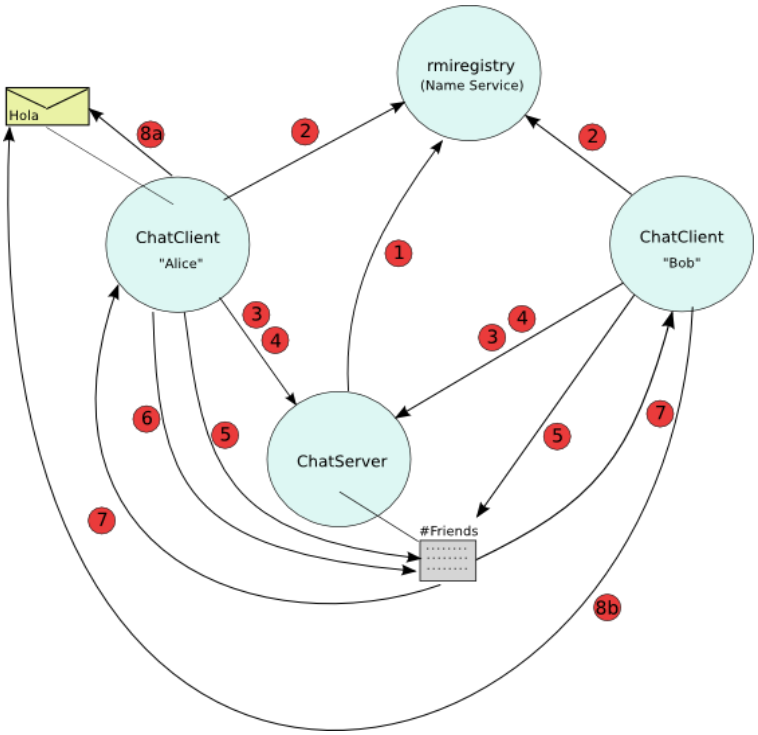
Es necessiten aproximadament dues setmanes per a completar la pràctica. Cada setmana cal assistir a una sessió de laboratori, on el professor pot resoldre qüestions

tècniques. També pot ser necessari dedicar temps addicional per a completar la pràctica. Amb aquesta finalitat es poden utilitzar els laboratoris durant els períodes d'accés lliure. També es pot utilitzar el vostre ordinador personal.

Al llarg de la pràctica veurà que hi ha una sèrie d'exercicis a realitzar. Es recomana resoldre'ls i anotar els seus resultats en els requadres que s'adjunten, per a facilitar l'estudi posterior del contingut de la pràctica. Es recorda als alumnes que hauran d'entregar la memòria de la pràctica en la tasca corresponent del PoliformaT (però no s'entregaran executables).

Un Xat distribuït bàsic

El següent esquema mostra una disposició bàsica del xat distribuït. Hi ha un servidor de noms (anomenat *rmiregistry* en RMI) un ChatServer i dos ChatClient (llançats pels usuaris Alice i Bob). Aquests 4 processos conformen un sistema senzill que no obstant això és més complex i interessant que un esquema de client/servidor pur.



L'esquema mostra els passos ordenats que la nostra aplicació segueix quan els usuaris Alice i Bob s'uneixen a un canal de xat denominat "#Friends" i comencen a xatejar. Alice envia un missatge "Hola" i ambdós usuaris ho reben al estar connectats al canal.

A continuació es detallen els passos seguits. Cada pas està marcat amb un cercle roig numerat en l'esquema:

1. ChatServer Registra el seu objecte principal ChatServer en el servidor de noms.
2. Els dos ChatClients cerquen el ChatServer utilitzant el servidor de noms. Note's que els tres processos han d'estar d'acord en el nom de l'objecte ChatServer.

3. Els clients del Xat es connecten al ChatServer.
4. Els clients del Xat pregunten la llista de canals.
5. Els clientes del Xat s'uneixen al canal “#Friends”. Note's que el primer usuari en un canal també rep una notificació quan el segon client s'uneix. Aquesta notificació de canal no és dibuixada en l'esquema.
6. Alice envia un missatge de xat senzill al canal.
7. A partir d'aquest enviament, el canal retransmet el missatge a tots els usuaris connectats (al canal).
8. Els usuaris quant reben el missatge demanen el seu contingut. 8a és una invocació local, mentre que 8b és una invocació remota.

Note's que encara que el rol dels ChatClient és principalment actuar com a clients de xat, també actuen com a servidors, doncs tenen objectes què són invocats de forma remota. Més concretament, *ChatChannel* invoca a *ChatUser* per a retransmetre els missatges, i qualsevol que necessite veure el contingut d'un missatge donat ha d'invocar l'amo del missatge pel seu contingut. En l'esquema, *ChatClient* pregunta a l'usuari Alice pel contingut del missatge “Hola”. Aquest patró d'invocació d'objectes no és habitual en la majoria d'entorns de xat però és bastant complet per a comprendre les aplicacions orientades a objectes distribuïts.

Instal·lacions de laboratori i recomanacions de l'entorn

Es pot utilitzar qualsevol entorn de desenvolupament per a gestionar projectes Java (BlueJ, Eclipse, etc.) o simplement un editor de text senzill, el compilador estàndard per línia de comandaments i la màquina virtual de Java. La nostra recomanació és utilitzar les eines de línia de comandaments, de tota manera s'utilitze un IDE o no per a desenvolupament probablement es necessiten les eines per línia de comandament. Aquestes son normalment més flexibles per a entorns distribuïts.

Donades les instal·lacions de laboratori disponibles per a aquesta pràctica, es recomana utilitzar Windows, i les eines per línia de comandaments java, javac i rmiregistry.

Els laboratoris tenen ordinadors amb un firewall configurat, el qual bloqueja la majoria de ports. Es pot utilitzar el port 22 (per a connexions ssh) i els ports **9000-9499** per a les pràctiques de laboratori. Alguns dels programes a utilitzar en la pràctica i l'eina rmiregistry necessiten ser configurats per a utilitzar aquests ports.

El programari proporcionat

Per a aquesta pràctica, s'ha proporcionat un paquet jar denominat “DistributedChat.jar”. Aquest ha de ser descarregat, i desempaquetat comprovant que conté els arxius amb el codi font java. També s'ha de comprovar que es poden compilar (amb Bluej o des de línia de comandaments amb **javac *.java**) i que es poden executar els programes bàsics.

El paquet conté dos programes ja preparats per a ser executats. Aquests programes són **ChatClient.java** i **ChatServer.java**. Els altres arxius són interfícies i classes necesàries per a

aquests programes i tots ells conformen un xat distribuït bàsic.

Per a poder executar ChatServer i ChatClient, es necessita una instància de rmiregistry ja en funcionament de manera que els nostres programes de xat puguen utilitzar-lo com a servidor de noms. Els servicis de noms, com s'ha explicat en classe, permeten registrar un nom simbòlic per a un objecte remot i associar-li una referència, de manera que puga ser localitzat pels clients. Per a començar rmiregistry, només cal executar-lo en una terminal o consola:

```
rmiregistry 9000
```

NOTA: Cal executar rmiregistry **des de la carpeta on estiguen les classes compilades del programa de xat.**

El paràmetre 9000 implica arrancar rmiregistry en el port 9000. Si no es proporciona aquest paràmetre, rmiregistry s'arrancarà en el seu port per defecte 1099. Cal recordar que en els laboratoris del DSIC ha d'utilitzar ports dins del rang 9000-9499.

NOTA: Si al executar aquesta ordre apareix un error del tipus "Port already in use", possiblement és degut a que altre grup ha llançat una instància sobre la mateixa màquina virtual. En eixe cas, torne a llançar el *rmiregistry* amb altre nombre de port, per exemple 9100.

Per a començar un ChatServer, ha d'executar-se el següent comandament **en el mateix directori on es troben les classes compilades** (els paràmetres nsXX fan referència a name-server):

```
java ChatServer nsport=9000 myport=9001
```

Aquesta instrucció comença un ChatServer en el port 9001, i que utilitza el port local 9000 quan necessita connectar-se al rmiregistry. El port local per defecte per a ChatServer i ChatClient és 9001, i el port per defecte per a rmiregistry és 9000, d'aquesta manera la instrucció anterior té el mateix efecte que aquesta instrucció més senzilla:

```
java ChatServer
```

Per a començar un ChatClient, es poden proporcionar els mateixos paràmetres, afegint opcionalment un paràmetre addicional (nshost) per a especificar l'amfitrió on el servidor de noms està corrent. Per exemple:

```
java ChatClient nshost=192.168.1.1 nsport=9000 myport=9002
```

Amb aquests paràmetres, aquest ChatClient intentarà trobar un rmiregistry executant-se en el port 9000 de l'ordinador 1292.169.1.1. En aquest exemple hem suposat que el rmiregistry es troba en eixa màquina, però en general caldrà descobrir quin és el nom de l'ordinador, la qual cosa es pot fer fàcilment usant, per exemple, el comandament **ipconfig** des d'una terminal (en Windows).

Si el procés rmiresgistry s'ha llançat a la màquina local, no és necessari indicar el nom del host. És a dir, seria suficient amb:

```
java ChatClient myport=9002
```

Aquest ChatClient atendrà connexions al port 9002. Recordem que si volem tindre més

d'un client o servidor corrent al mateix ordinador físic, cal especificar un port diferent al port per defecte 9001, assignant un port nou per cada client que es llance.

Hi ha un paràmetre addicional per a ChatClient i ChatServer, i es el nom de l'objecte ChatServer. Per defecte es "TestServer". Es pot modificar aquest nom utilitzant l'argument "server" en ambdós programes. Per exemple, les següents instruccions comencen un ChatServer i un ChatClient que utilitzaran un objecte ChatServer anomenat "NewServer".

```
java ChatServer server=NewServer
java ChatClient server=NewServer myport=9002
```

Important: En aquesta pràctica s'assumeix que rmiregistry i ChatServer son executats al mateix ordinador.

Activitat 1: començar a xatejar

Aquesta activitat inicial ensenya com xatejar. ☺

En aquest cas s'ha d'aprendre com xatejar utilitzant el Xat distribuït proporcionat. A més, s'ha d'aprendre com està construït el Xat distribuït orientat a objectes i quines invocacions d'objecte ocorren quan es realitzen les activitats bàsiques del Xat.

Per a xatejar necessitem almenys 2 persones. En aquesta pràctica s'assumeixen 2 persones: Alice i Bob. Hem de seguir els passos indicats al apartat anterior, es a dir:

- Pas 1: iniciar *rmiregistry*
- Pas 2: iniciar un ChatServer
- Pas 3: iniciar un ChatClient per a Alice. En aquest pas arranquem un primer ChatClient en el mateix ordenador on estan executant-se la resta de programes. Com el ChatServer iniciat en el pas anterior utilitza el port 9001, cal arrancar el *ChatClient* de Alice en un port diferent (per exemple, el port 9002).
- Pas 4: iniciar un ChatClient per a Bob. En este pas s'inicia un segon client de xat per a Bob. Aquest client pot executar-se en la mateixa màquina on hem llançat els processos anteriors (assignant-li un port diferent, per exemple el port 9003), o bé en altra màquina diferent.

Per a llançar un ChatClient que es connecte al servidor de xat d'una altra màquina diferent (per exemple, l'ordinador del grup veí), caldrà llançar el client especificant l'ordinador on s'està executant el rmiregistry. Per exemple, si eixe ordenador tinguera la IP 192.168.1.2 i rmiregistry estiguera utilitzant el port 9000, podríem llançar el nou ChatCient d'aquesta manera:

```
java ChatClient nshost=192.168.1.2 nsport=9000 myport=9004
```

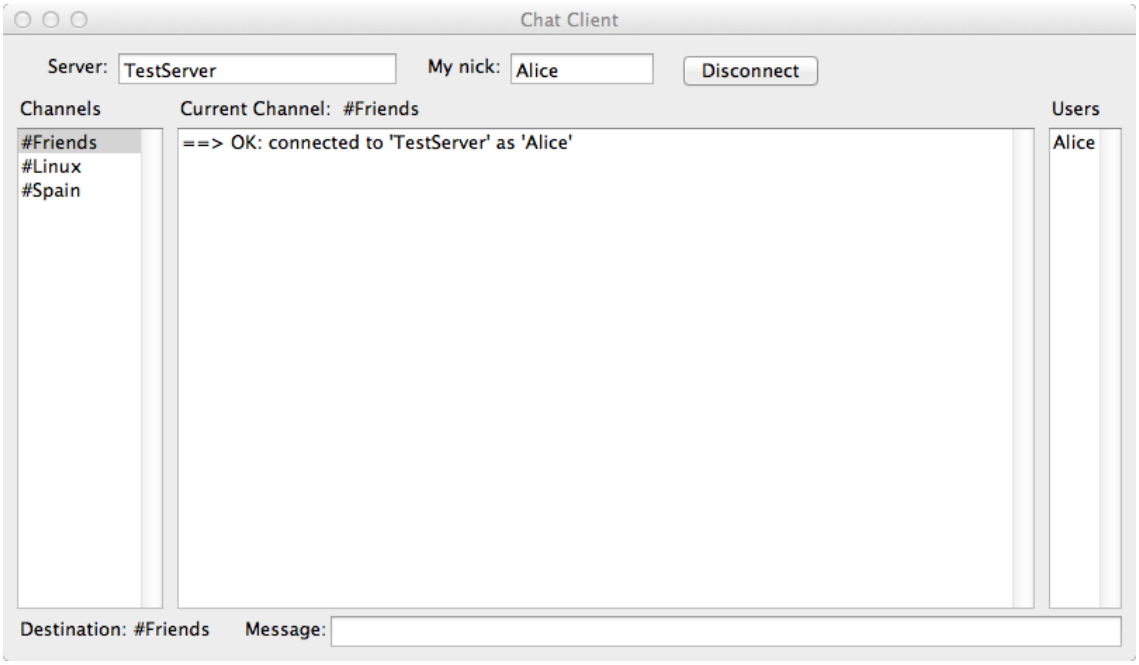
- Pas 5: xatejar un poc.

La figura de la pàgina següent mostra la finestra del ChatClient. En ella es poden observar tres àrees. Hi ha una línia superior per als paràmetres de servidor. A continuació, hi ha una àrea central amb barres de desplaçament vertical (dividida en tres parts) i una tercera àrea en la part inferior, on es poden escriure missatges.

La línia superior té 2 camps de text, un per a indicar el nom del ChatServer al que el client es vol connectar i un altre on cal escriure el sobrenom de l'usuari. En aquest cas ha d'utilitzar Alice i Bob en cadascun dels ChatClient iniciats. Una vegada escrits ambdós camps el ChatClient pot connectar-se al ChatServer. Si la connexió es produeix, s'observarà un missatge d'èxit.

L'àrea amb barres de desplaçament està dividida en 3 subàrees. L'àrea de l'esquerra conté la llista de canals existents en el servidor de Xat al que s'ha connectat el client. L'àrea de la dreta conté la llista d'usuaris del canal al que s'ha connectat l'usuari i l'àrea central que és més gran conté els missatges entrants tant del canal seleccionat actualment com a missatges privats.

La línia inferior té una etiqueta "Destination" i una caixa de text per a poder escriure missatges. La destinació mostra quin canal o usuari rebran els missatges. Es pot seleccionar un canal o un usuari fent doble-clic en els seus noms.



Una vegada es fa doble-clic sobre un canal el ChatClient s'uneix a aquest canal eixint del canal en què s'estiguera prèviament. Si es fa un clic a un usuari d'aquest canal no es deixa el canal actual, solament es canvia la destinació del missatge. D'aquesta manera s'envia un missatge privat a l'usuari.

Qüestions

1) *Quan ChatServer registra el seu objecte principal ChatServer al servidor de noms, quines accions es realitzen? Quines interfícies s'han hagut d'estendre o implementar?*

Primer crea un objecte del tipus Registry (Registry reg = LocateRegistry.getRegistry (conf.getNameServiceHost(), conf.getNameServicePort());) per a conectar amb el servidor de noms, despres registra el seu objecte en el servidor de noms utilitzant el metode rebind d'aquesta clase (reg.rebind (conf.getServerName(), this);).

Esten la clase UnicastRemoteObject que s'utilitza per a identificarla com a servidor em el (ORB) i implementa la interficie IChatServer que aquesta esten de Remote que s'utilitza per a identificar les interficies remotes.

2) *Quan ChatClient busca el ChatServer utilitzant el servidor de noms, quines accions es*

realitzen? Quin nom de l'objecte s'ha utilitzat? Què és el que obté ChatClient del servidor de noms? Primer crea un objecte del tipus Registry (Registry reg = LocateRegistry.getRegistry (conf.getNameServiceHost(), conf.getNameServicePort());) per a conectar amb el servidor de noms, despres utilitza el metode lookup per a registrarse al servidor.
TestServer que es el nom que li hem posat per defecte.
Un objecte del tipus Remote.

3) *Quan els clients del xat es connecten al ChatServer, quins proxies s'utilitzen? Quins mètodes d'eixos proxies?* S'envia *cap objecte com paràmetre d'un mètode? Per a què?*

Utilitza un proxy del chatServer.
Primer connectUser per a conectarse al servidor, despres listChannels per a vore el s canals disponibles.
S'envia un objecte de ChatUser al servidor.
Per a que el servidor tinga un proxy del usuari conectar.

4) *Quan els clients del xat pregunten la llista de canals, quins proxies s'utilitzen? Quins mètodes d'eixos proxies? S'envia cap objecte com a paràmetre o valor de retorn d'eixos mètodes? Per a què?*

Utilitza un proxy de IChatServer.
listChannels().
no se li pasen parametres, retorna una llista d'objectes IChatChanel de tipus Remote.
Per a mostrar els canals disponibles.

5) *Quan un client del xat s'uneixen a un canal, quins proxies utilitza el client de xat per a realitzar l'acció? Quins mètodes d'eixos proxies?*

Utilitxa un proxy de IChatServer i un de IChatChanel.
Server --> getChannel(channelName),
Channel --> listUsers(), join(myUser),

6) *Quan un client del xat s'uneix al canal, el canal avisa a la resta d'usuaris del canal. Per a això, quins proxies s'utilitzen? Quins mètodes d'eixos proxies? S'envia cap objecte com a paràmetre d'un mètode? Quin i per a què?*

Utilitza un proxy de IChatMessage per al mensage i un proxy de IChatUser per cada usuari conectat al canal.
sendMessage() de cada proxy IChatUser del usuaris conectats al canal.
El mensage.
Per a que els usuaris tingen un proxy d'eixe mensage i pugen accedir a ell.

7) *Si Alice envia un missatge de xat senzill al canal, quines accions es realitzen? Es crea cap objecte nou? Si es així, per a què s'utilitza?*

IChatChannel c_dst = srv.getChannel (dst); --> Primer obtenim un proxi del canal al que anem a enviar.
IChatMessage c_msg = new ChatMessage(myUser, c_dst, msg); --> Despres cream un objecte del misatge a enviar.
c_dst.sendMessage (c_msg); --> Finalment utilitzem el proxy del canal per a enviar el misatge.
Es crea un objecte del misatge per a que els altres usuaris el utilitzen com a proxy per a accedir al misatge original.

8) *Quan el canal retransmet el missatge d'Alice a tots els usuaris connectats (al canal), quines accions es realitzen? S'envia cap objecte com a paràmetre d'un mètode?*

purge (); En aquest metode comprova els usuaris que segeixen conectats i actualitza la llista d'usuaris.
for (IChatUser usr: users.values()) { usr.sendMessage (msg);} Recorre la llista del proxys d'usuaris conectats al canal i els envia el misatge.
Si, el misatge.

9) *Els usuaris, al rebre un missatge, demanen el seu contingut. Quines accions es realitzen? Quines diferencies n'hi han entre una invocació local i una invocació remota?*

Utilitzen el objecte rebut com a proxy per accedir a l'informacio del misatge, comproben si es un misatge privat i si es per a ells el mostren, o el mostren si es un misatge normal.
L'invocacio local es quan tenim una copia de l'objecte i la remota es quan utilitzem un proxy per a accedir a l'objecte.

10) *¿Quins objectes es crearan y quines invocacions tindran lloc quan l'usuari Bob envie un* ***missatge privat*** *a la usuària Alice?*

Es crea un objecte del misatge i sutilitza un proxy del usuari Alice per a enviarli el misatge.

Activitat 2

Es desitja implementar un ChatRobot, un procés encarregat de connectar-se a un servidor donat i connectar-se al canal "#Friends". Cada vegada que un usuari es connecta a aquest canal, el ChatRobot li ha de saludar enviant un missatge "Hola sobrenom" al canal. El sobrenom ha de ser l'utilitzat per l'usuari que s'ha unit al canal.

Es deu completar la implementació en l'arxiu ChatRobot.java que es proporciona amb la pràctica.

Qüestions

1) Expliqueu quins objectes i quines operacions son invocades cada vegada que un usuari es connecta al canal "#Friends", assumint que el ChatRobot ja està connectat al canal.

2) Justifica l'afirmació, "si llancem més d'una aplicació *ChatClient* o *ChatRobot* a la mateixa màquina, deurien tindre valors *myport* distints. Si només es llança una aplicació per màquina no és necessari modificar dit valor"

3) La ubicació del servidor de noms (nsport, nshost) deu ser coneguda per tots, però els clients no necessiten conèixer el host ni el port que correspon al *ChatServer*. Indica la raó.

1) - El metode notifyUsers() recorre la llista de proxys d'usuaris conectats al cannal y utilitza el metode sendMessage(msg) per a enviar un misatge a cadascun dels usuaris notificant que s'ha unit un nou usuari.

- El chatRobot simplement llig el missatge i en el cas que comense per JOIN utilitza el objecte IChatChannel rebut amb el missatge per a enviar un objecte de tipus IChatMessage al cannal.

2) Aquesta afirmacio es certa, degut a que sols pot haber un proces escoltant en un port en una mateixa maquina, pertant si estigera en maquines diferents no hi hauria problema ja que s'identificaria per la ip de la maquina en la que es troba i el port en el que se esta executant.

3) Correcte, degut a que quan un client vol accedit a un objecte remot, no utilitza el servidor si no que utilitzen la factoria per a buscarlo, igual pasa quan creen un objecte, el registren en el servidor de noms per a que qualsevol el puga utilitzar.