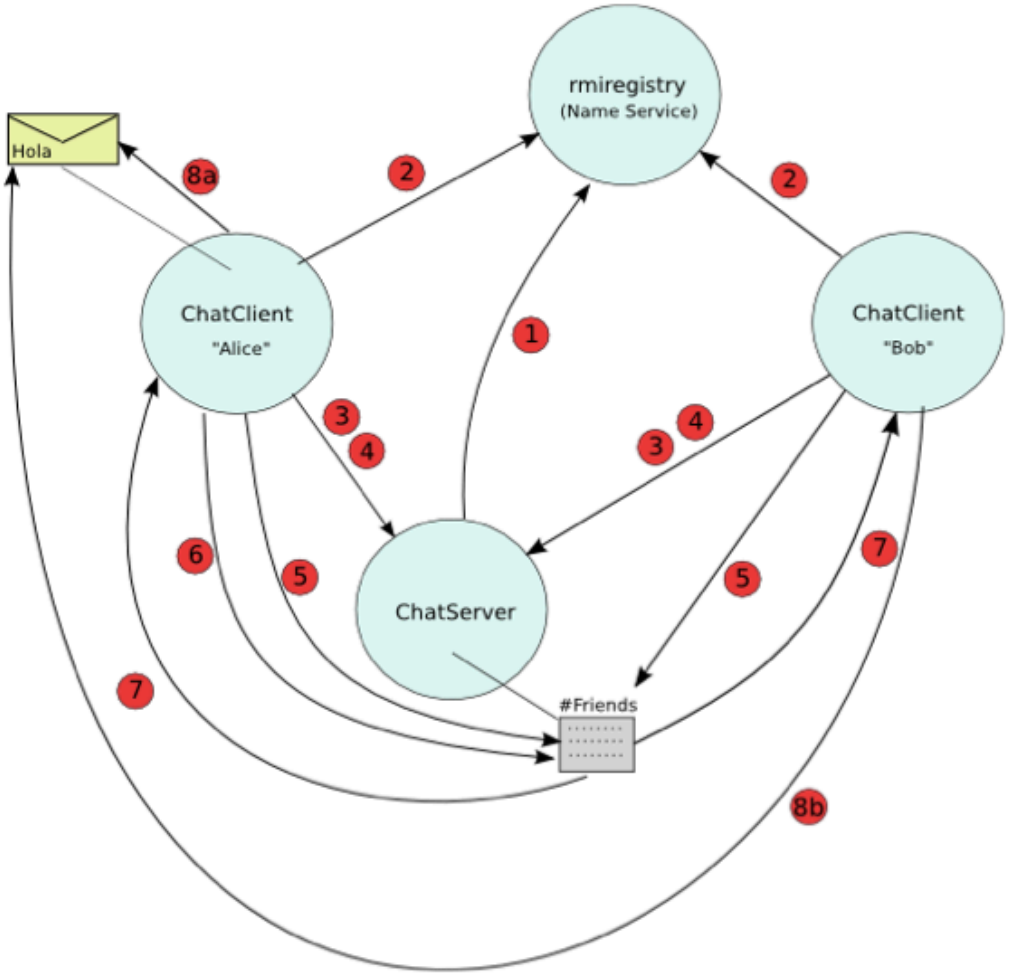
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Práctica de Laboratorio: Un chat distribuido orientado a objetos basado en RMI

Concurrencia y Sistemas Distribuidos

Ejemplo: Chat Distribuido orientado a objetos



Elementos de la aplicación

- **ChatServer**
  - ***ChatServer object***: objeto remoto que ofrece métodos para:
    - gestionar canales (crearlos, obtener listado, obtener acceso a un canal)
    - gestionar usuarios (conectar usuarios, obtener datos…)

```
public interface IChatServer extends Remote {
    public IChatChannel [] listChannels () throws RemoteException;
    public IChatChannel getChannel (String name) throws RemoteException;
    public IChatChannel createChannel (String name) throws RemoteException;

    public IChatUser getUser (String nick) throws RemoteException;
    public boolean connectUser (IChatUser usr) throws RemoteException;
    public boolean disconnectUser (IChatUser usr) throws RemoteException;
}
```

  - ***ChatServer process***: encargado de crear el objeto ChatServer, registrarlo en el servidor de nombres y crear canales de comunicación iniciales

Elementos de la aplicación

- **ChatUser:** objeto remoto que representa al usuario.
  - El usuario deberá registrarlo en el servidor, usando el método *connectUser* del objeto ChatServer
  - Los usuarios reciben mensajes cuando se invoca el método "sendMessage" de su objeto ChatUser respectivo

```
public interface IChatUser extends Remote {
    public String getNick() throws RemoteException;
    public void sendMessage (IChatMessage msg) throws RemoteException;
}
```

Elementos de la aplicación

- **ChatChannel:** objeto remoto que representa al canal de comunicación
  - Los usuarios deben obtener primero el canal desde el servidor (con método *getChannel* del objeto ChatServer)
  - Y una vez obtienen la referencia, se unen al canal usando el método *join*.

```
public interface IChatChannel extends Remote {
    public boolean join (IChatUser usr) throws RemoteException;
    public boolean leave (IChatUser usr) throws RemoteException;
    public void sendMessage (IChatMessage msg) throws RemoteException;
    public IChatUser [] listUsers () throws RemoteException;
    public String getName () throws RemoteException;
}
```

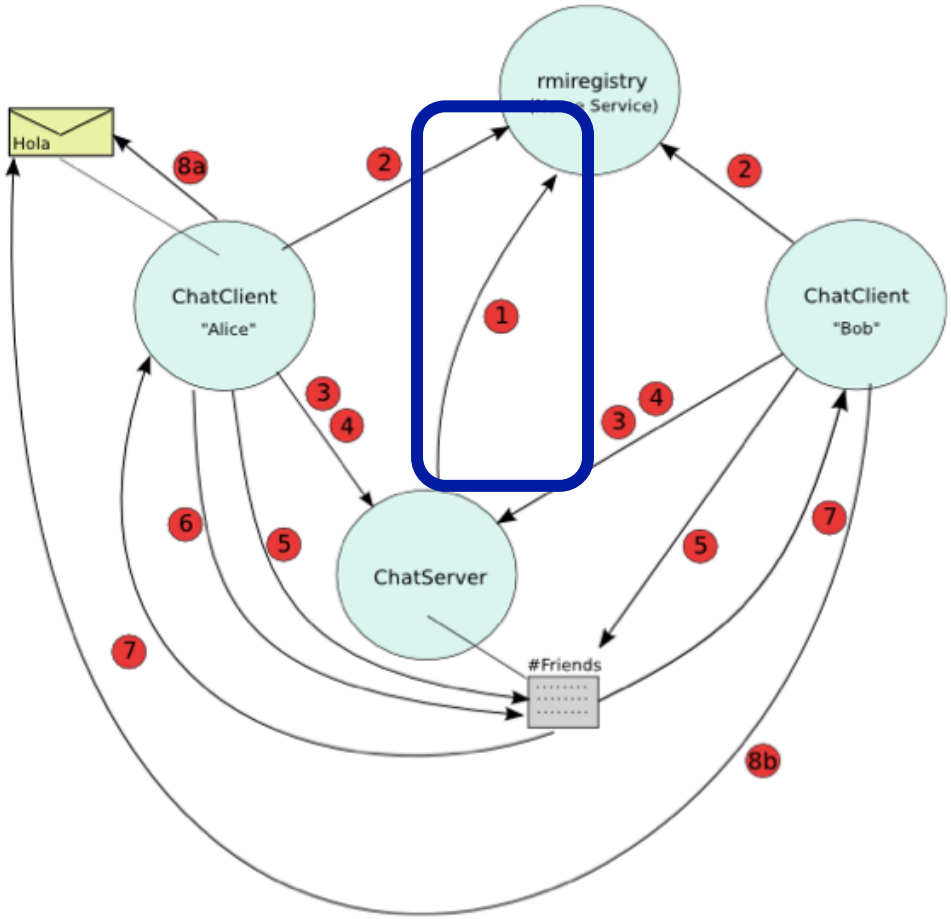
Elementos de la aplicación

- **ChatMessage**: en nuestro chat, los mensajes son objetos.
  - Los usuarios envían objetos ChatMessage a través del canal.
  - En la creación del objeto ChatMessage se especifica:
    - Quién lo envía (ChatUser)
    - El canal destino (ChatChannel)
    - El contenido del mensaje (String).

```
public ChatMessage (IChatUser src, IChatChannel dst, String str) throws RemoteException
{
   super (ChatConfiguration.the().getMyPort());
   this.src = src; this.dst = dst; this.str = str;
}
```

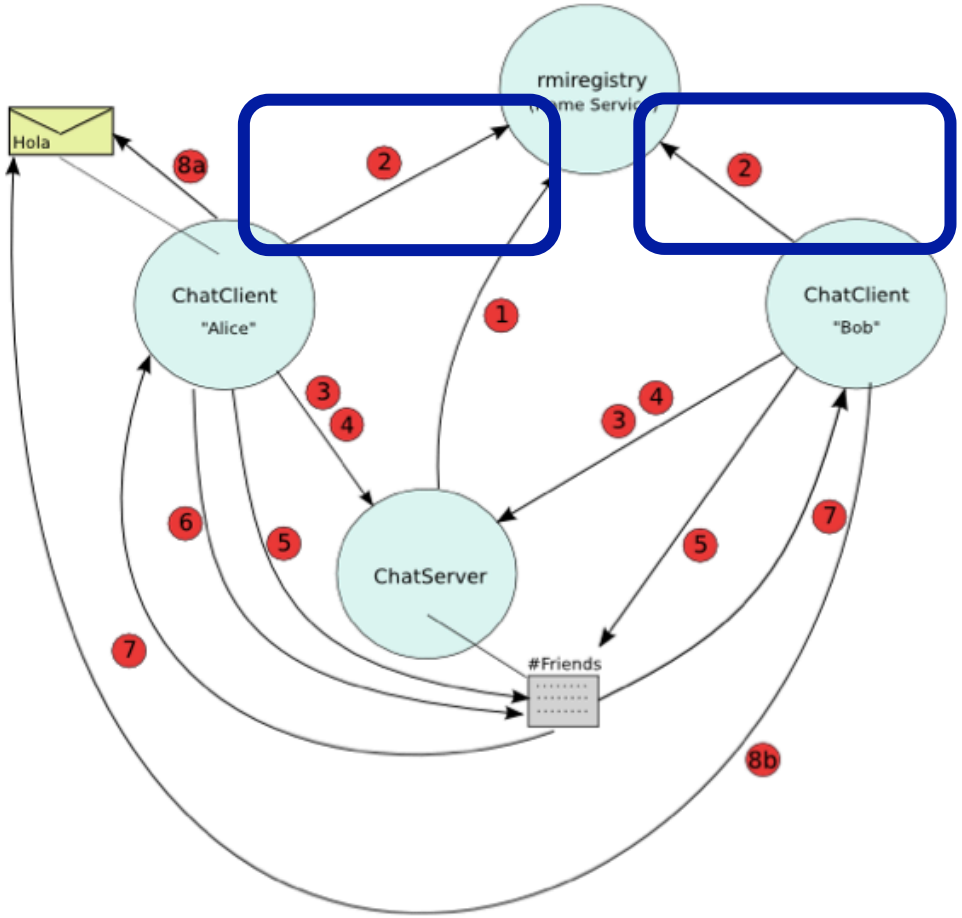
```
public interface IChatMessage extends Remote{
   public String getText () throws RemoteException;     // The message itself
   public IChatUser getSender () throws RemoteException; // allways the user who sends
   public Remote getDestination () throws RemoteException; // actual class depends on private/pub
   public boolean isPrivate() throws RemoteException; // actual class depends on private/pub
}
```

Ejemplo: Chat Distribuido orientado a objetos



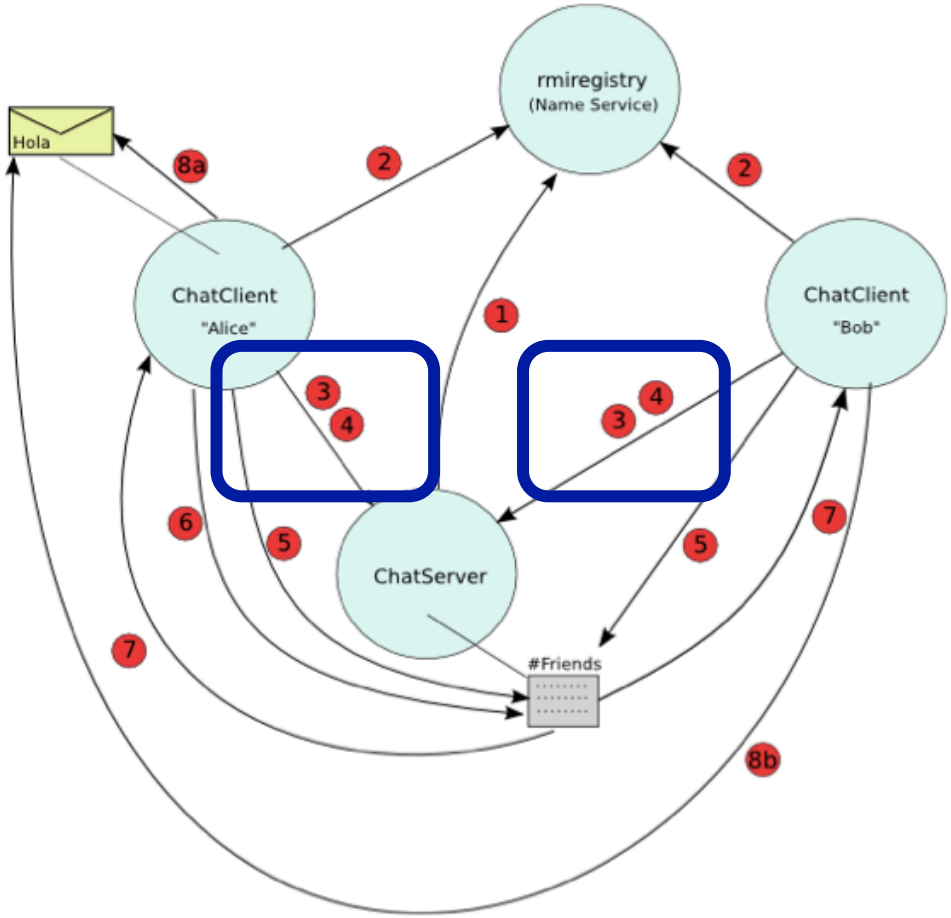
Paso 1: El servidor ChatServer registra su objeto principal "ChatServer" en el servidor de nombres

Ejemplo: Chat Distribuido orientado a objetos



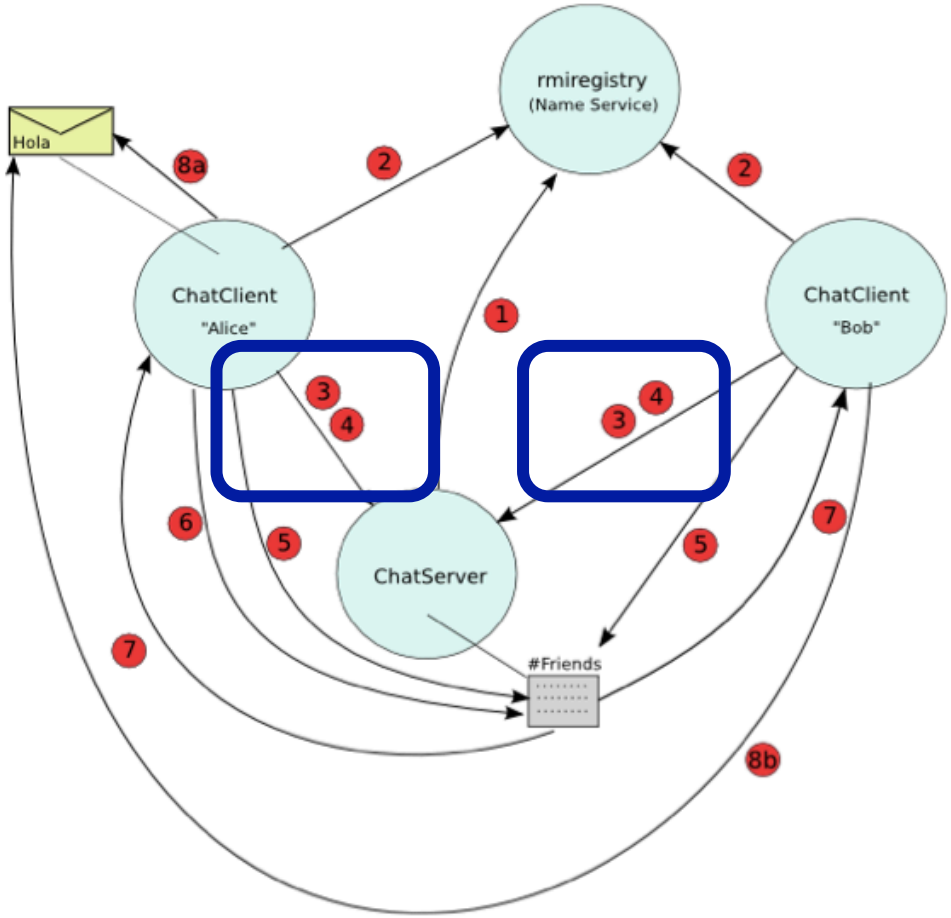
Paso 2: Los ChatClients buscan el objeto “ChatServer” utilizando el servidor de nombres.

Ejemplo: Chat Distribuido orientado a objetos



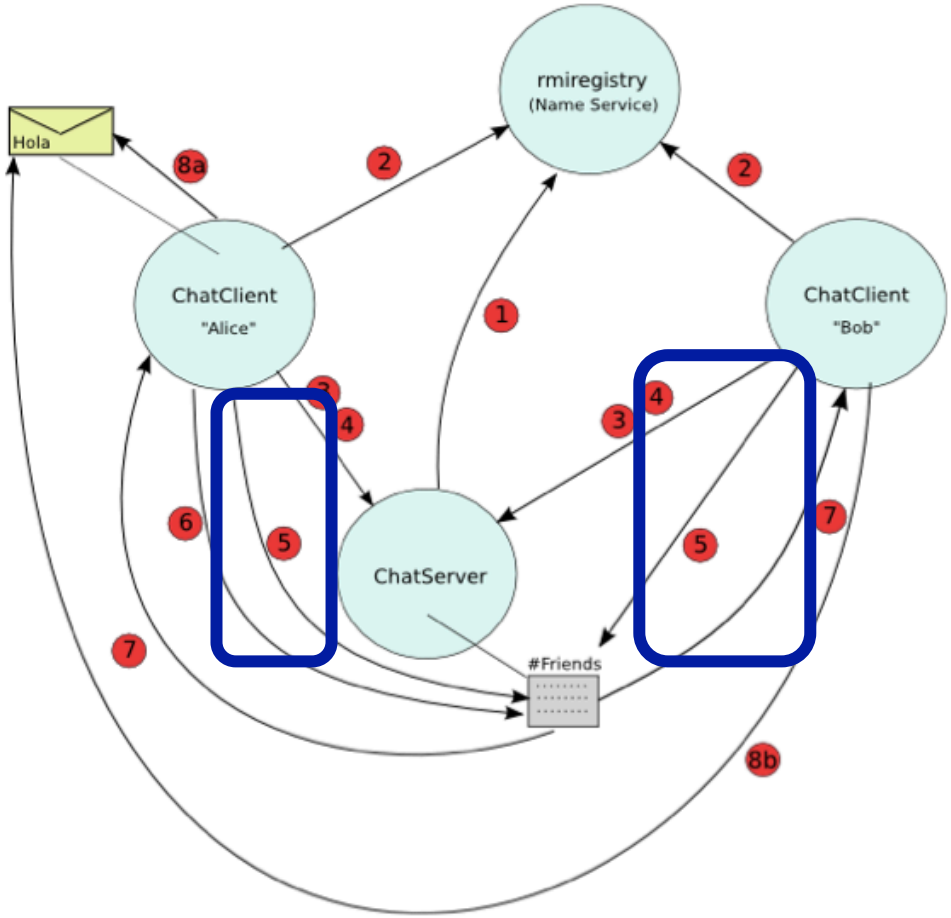
Paso 3: Los clientes del chat se conectan al ChatServer. Para ello crean un objeto ChatUser local y lo envían al servidor ChatServer mediante el método *connectUser* de ChatServer.

Ejemplo: Chat Distribuido orientado a objetos



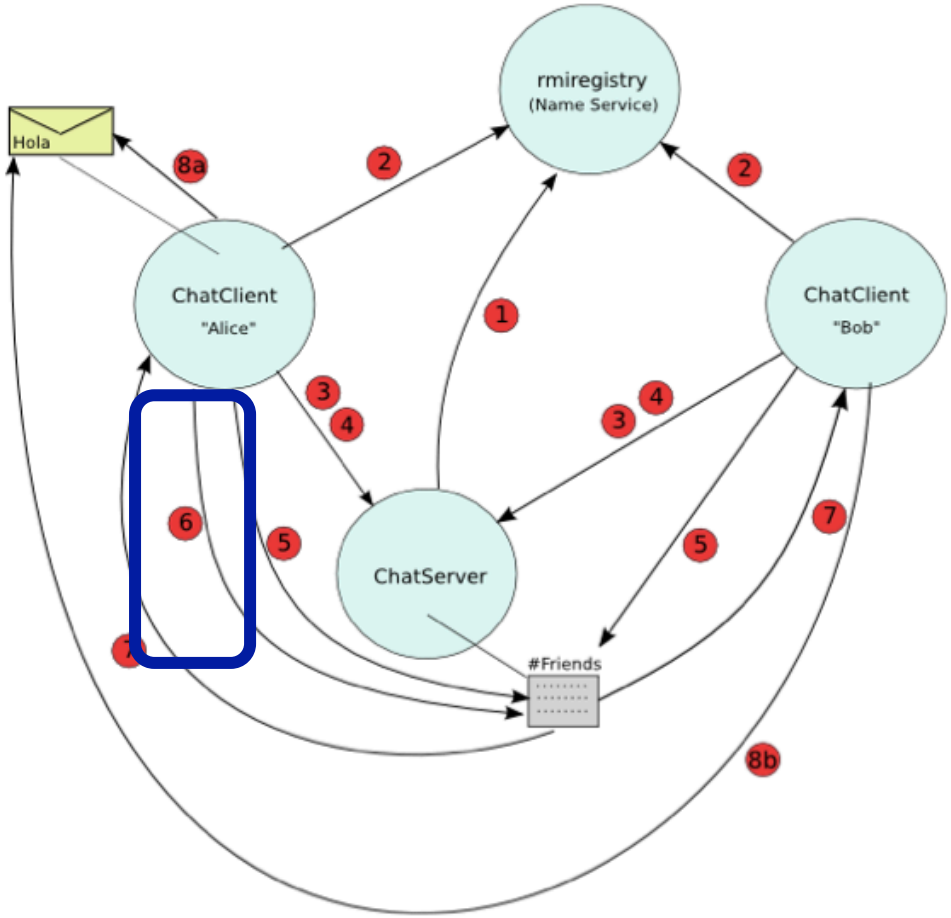
Paso 4: Los clientes del Chat preguntan la lista de canales, con el método *listChannels* de *ChatServer*.

Ejemplo: Chat Distribuido orientado a objetos



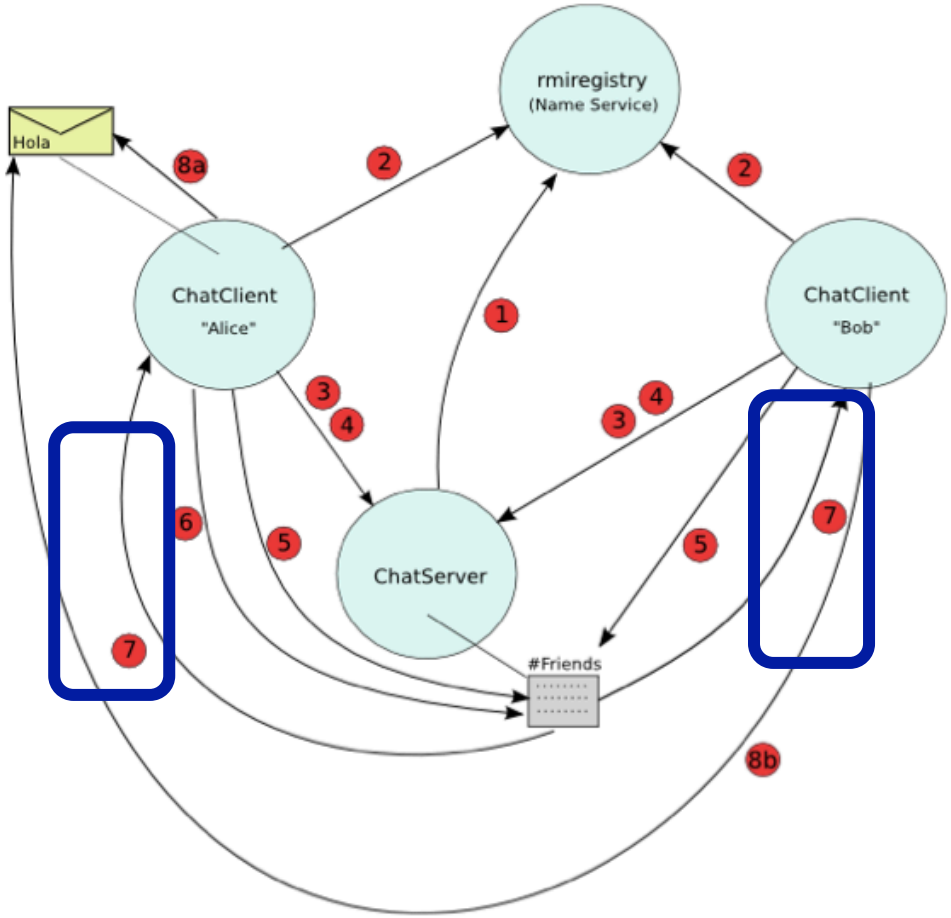
Paso 5: Los clientes se unen al canal Friends.

Ejemplo: Chat Distribuido orientado a objetos



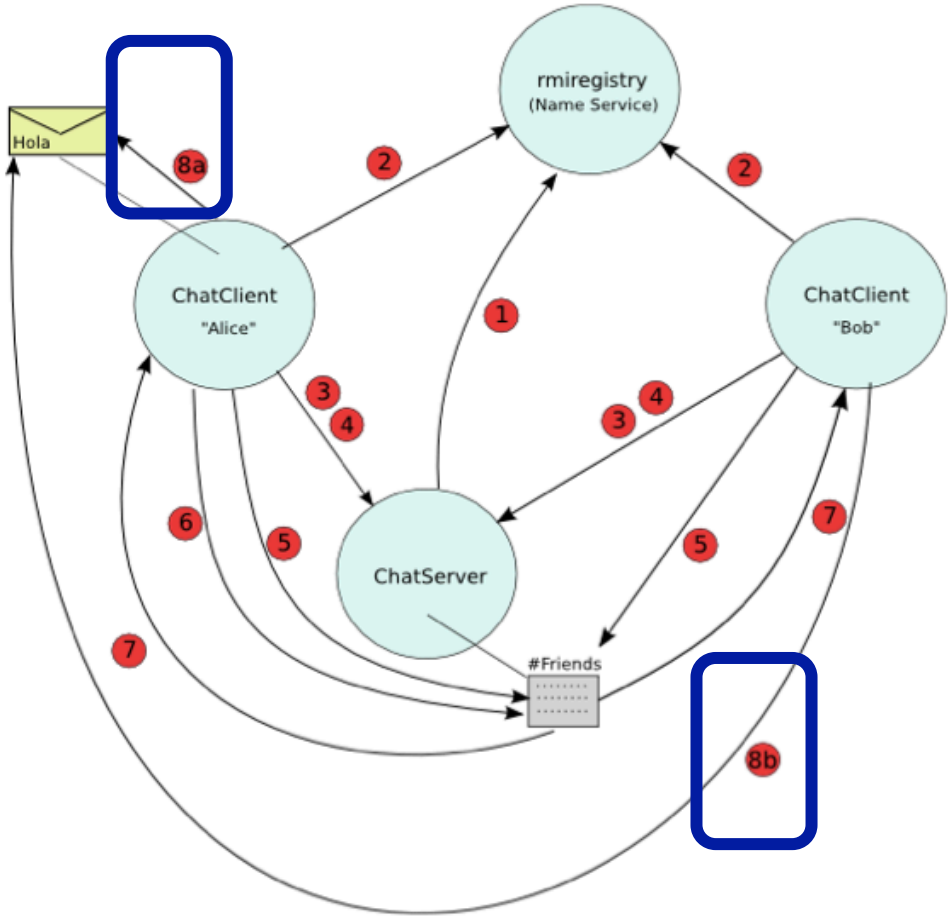
Paso 6: Alice envía un mensaje al canal, usando el método *sendMessage* del objeto ChatChannel.

Ejemplo: Chat Distribuido orientado a objetos



Paso 7: ChatChannel invoca el método *sendMessage* de ChatUser (de cada usuario del canal) para retransmitir los mensajes.

Ejemplo: Chat Distribuido orientado a objetos



Paso 8: Los usuarios, al recibir el mensaje, piden su contenido.

Actividades a realizar

- Análisis del código proporcionado
  - ChatClient.java, ChatServer.java
  - Resolución de las cuestiones planteadas en el boletín
- Implementación de un ChatRobot
  - ChatRobot.java