

# Programación de Sistemas

## Tarea 4: Chat Grupal

Carlos Apablaza B. - 2521068-9  
Manuel Cornejo R. - 2521011-5

### 1.- Introducción

El chat grupal es algo que ya muchos conocen en la actualidad en diversos programas de mensajería que existen, como lo es Messenger. La idea de esta tarea es replicar tal acción, mediante un programa Java que actúe, por un lado como servidor, y por otro lado como cliente (puede haber cualquier cantidad de clientes).

### 2.- Compilación y ejecución

El programa requiere la ejecución del servidor y de uno o más clientes conectándose a tal servidor mediante el puerto que se le ha asignado a este último. El servidor puede estar en el mismo equipo en donde se corren los programas clientes (donde el host del server en este caso sería "localhost") o en un equipo remoto (como lo sería en este caso, aragorn). En cualquier caso se debe cumplir que el servidor debe ejecutarse antes que los clientes.

Para compilar los archivos fuente se incluyen dos Makefile, los cuales pueden usarse escribiendo:

```
$ make
```

Los Makefile generarán los correspondientes archivos ejecutables :  
(Además de otros archivos de clases internas)

```
/ChatClient  
- ChatClient.class
```

```
/ChatServer  
- ChatServer.class
```

Los parámetros que deben agregarse en la línea de comandos al correr los archivos ejecutables son:

- \$ java ChatClient <User\_name> <host\_server> <port>
- \$ java ChatServer <port>

### 3.- Funcionamiento

El programa en el servidor está basado en hebras. Cada una de éstas es creada dinámicamente, cuando algún cliente solicita conexión.

Además se almacena la referencia a cada hebra en un objeto ArrayList (clase genérica), de manera de poder comunicarse con ellas, posteriormente.

Cuando un cliente se desconecta, la hebra correspondiente es eliminada del ArrayList.

Cada vez que un cliente escribe un mensaje, ChatClient lo envía al servidor ChatServer. En el servidor la hebra asociada a dicho cliente reenvía el mensaje hacia la hebra principal del programa, la cual se encarga de reenviar el mensaje a todos los clientes que estén conectados al servidor. Esta operación se realiza sincronizadamente, para asegurar que los mensajes no sean corruptos y que sean recibidos en el orden en que fueron enviados por los clientes.

El cliente se preocupa de esperar por la escritura de texto por parte del usuario. Cuando éste último hace click en el botón "Enviar" o presiona Enter, el programa envía el mensaje al servidor.

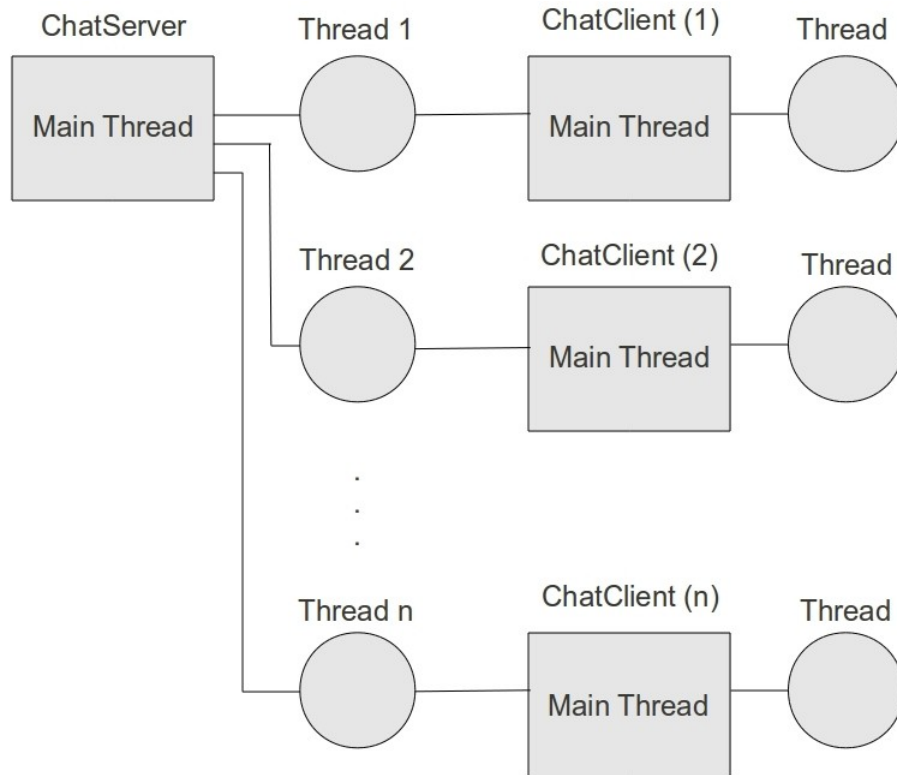
Además el cliente crea una hebra (al ejecutarse), la cual se preocupa de esperar por mensajes enviados desde el servidor. El método usado es bloqueante, por lo cual la hebra queda "en espera" hasta que lleguen mensajes, y en caso de llegar, ésta se preocupa de llamar al método que actualiza el cuadro de texto, para mostrar el nuevo mensaje.

El cliente posee una interfaz gráfica que facilita la utilización del programa. En éste se despliega la conversación que se esté llevando a cabo, indicando, además, el nick asociado a cada mensaje. El cliente también tiene la opción de ver los usuarios que se encuentran conectados al servidor, escribiendo y enviando "/users" en la ventana del chat. (Este mensaje no es reenviado a los demás clientes)

Por último, cabe destacar que cada cliente es informado de la conexión y desconexión de otros clientes en tiempo real (este trabajo es realizado por el servidor).

## 4. Diagrama del sistema

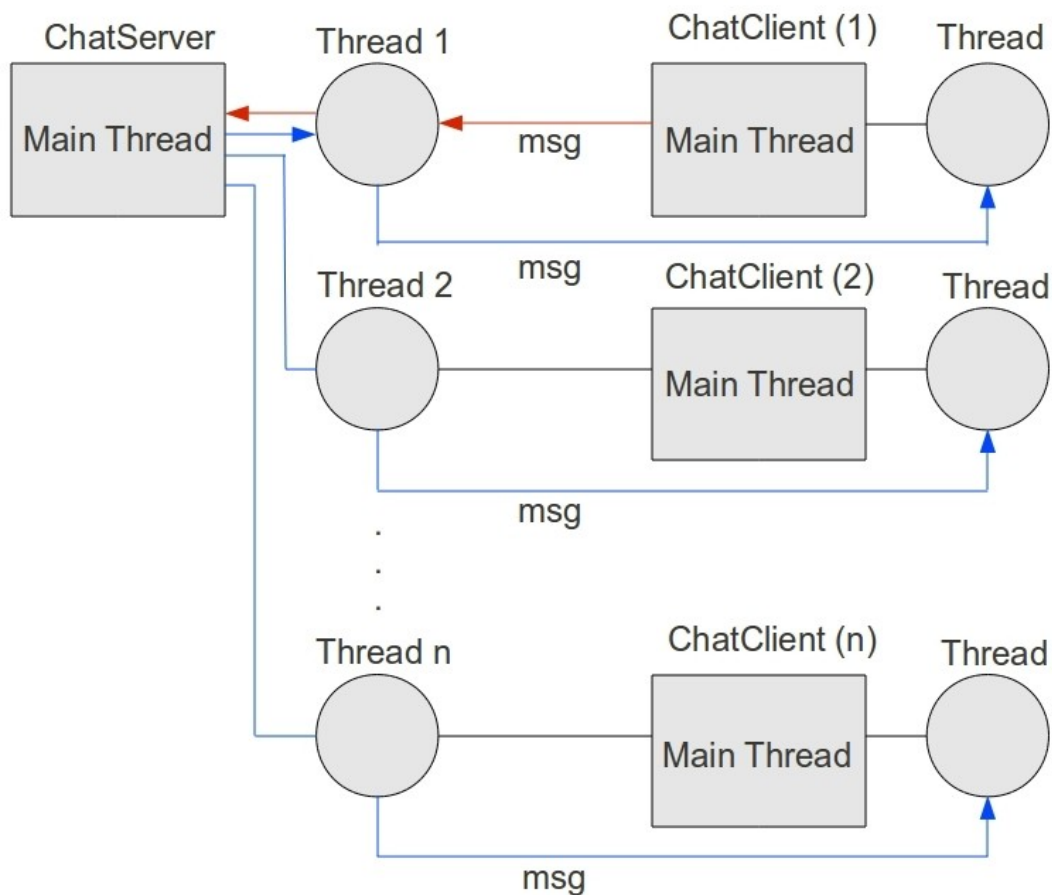
En la siguiente figura muestra el sistema configurado, sin el flujo de mensajes entre los clientes y el servidor. Se crean  $n$  hebras, para cada uno de los  $n$  clientes.



*Figura 1: Configuración del sistema.*

La siguiente figura muestra el envío de mensajes. Un cliente envía un mensaje al servidor, y éste lo reenvía a todos sus clientes.

El cliente 1 envía un mensaje al servidor, a través de su hebra principal. (flecha roja). Luego la hebra principal del Servidor se preocupa de reenviar el mensaje a todos los clientes incluido el que lo envió (flecha azul), y la hebra secundaria se preocupa de "escuchar" este mensaje.

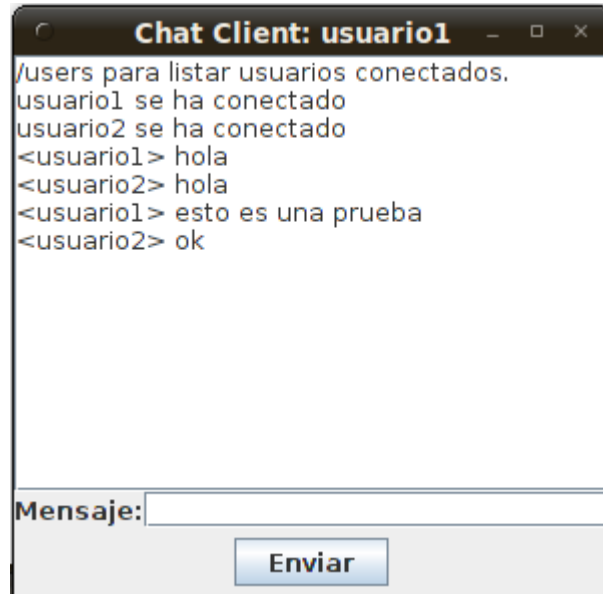


*Figura 2: Flujo de mensajes entre cliente y servidor.*

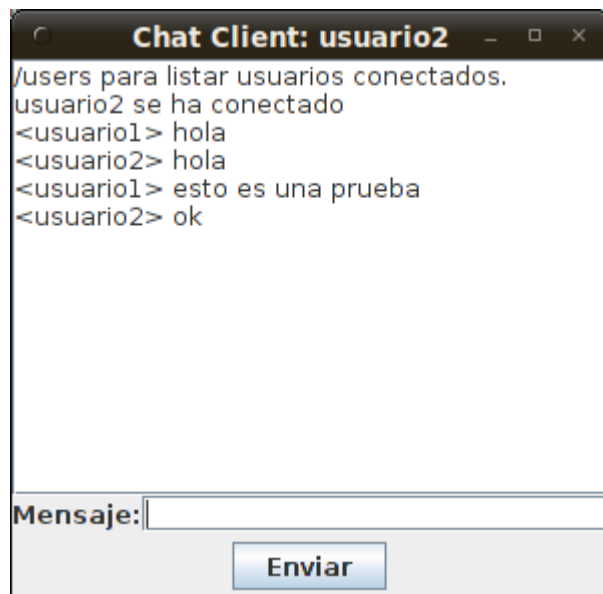
## 5.- Resultados

Para este punto, se ha realizado una prueba en un sólo equipo (localhost). Luego, para esta prueba se puede apreciar lo siguiente:

Cliente 1:



Cliente 2:



Servidor:

```
manuel@desktop-noma:~/Documentos/tarea4_elo330/final$ java ChatServer 2345
```

```
usuario1 se ha conectado
```

```
Enviando mensaje: <usuario1> se ha conectado
```

```
usuario2 se ha conectado
```

```
Enviando mensaje: <usuario1> se ha conectado
```

```
Enviando mensaje: <usuario2> se ha conectado
```

```
Enviando mensaje: <usuario1> hola
```

```
Enviando mensaje: <usuario2> hola
```

```
Enviando mensaje: <usuario1> hola
```

```
Enviando mensaje: <usuario2> hola
```

```
Enviando mensaje: <usuario1> esto es una prueba
```

```
Enviando mensaje: <usuario2> esto es una prueba
```

```
Enviando mensaje: <usuario1> ok
```

```
Enviando mensaje: <usuario2> ok
```

```
□
```