



INSTITUTO POLITÉCNICO NACIONAL ESCUELA SUPERIOR DE CÓMPUTO

Sistemas Distribuidos

Prof. Carlos Pineda Guerrero

Alejandro de Jesús Zepeda Flores

Tarea 5. Chat multicast.

11 de noviembre de 2020

Requisitos

Desarrollar un programa en Java que implemente un chat utilizando comunicación multicast mediante datagramas. Se deberá ejecutar en 4 terminales de Linux.

Se deberá pasar como parámetro al programa el nombre del usuario que va a escribir en el chat. Para demostrar el programa, se deberá utilizar los siguientes usuarios: Donald, Hugo, Paco y Luis.

El programa deberá utilizar las funciones para enviar y recibir mensajes previamente vistas durante la clase.

El funcionamiento general del programa es el siguiente:

- El programa creará un *thread* que actuará como cliente multicast, el cual recibirá los mensajes del resto de los nodos. Cada mensaje recibido será desplegado en la pantalla.
- En el método *main*, dentro de un ciclo infinito, se desplegará el siguiente *prompt*: "Escribe el mensaje", se leerá un String utilizando el método `readLine()` de la clase `BufferedReader`. Entonces se deberá enviar el mensaje a todos los nodos que pertenecen al grupo identificado por la IP 230.0.0.0 a través del puerto 50000.
- El mensaje deberá tener la siguiente forma *nombre_usuario : mensaje_ingresado*, donde el *nombre_usuario* es el nombre que pasó como parámetro al programa (Donald, Hugo, Paco o Luis) y el *mensaje_recibido*, el mensaje que ingresó el usuario.

Desarrollo

La clase principal es Chat, sin embargo, dentro de esa clase existe otra llamada Worker que hereda de la clase Thread, por lo cual, podemos implementar el método `run()`.

Dentro de un ciclo infinito, procedemos a crear el Socket y unirlo al grupo especificado por la dirección IP 230.0.0.0. a través del puerto 50000. Además, por medio de la función *recibe_mensaje*, obtenemos los datos enviados por el resto de los usuarios y los imprimimos en pantalla con el formato solicitado (Figura 1).

```

11.         static class Worker extends Thread{
12.             @Override
13.             public void run(){
14.                 try{
15.                     while(true){
16.                         InetAddress group = InetAddress.getByName("230.0.0.0");
17.                         MulticastSocket socket = new MulticastSocket(50000);
18.                         socket.joinGroup(group);
19.                         byte aux[] = recibe_mensaje(socket,1024);
20.                         String[] parts = new String(aux,"UTF-8").split(":");
21.                         System.out.print("\n"+parts[0]+" escribe: "+parts[1]);
22.                         System.out.print("\nEscribe el mensaje: ");
23.                         socket.leaveGroup(group);
24.                         socket.close();
25.                     }
26.                 }catch(Exception except){
27.                     System.err.println(except.toString());
28.                 }
29.             }
30.         }

```

Figura 1. Clase Worker.

Para el método *main*, se crea un objeto de la clase Worker y se inicia la ejecución de su *thread*. Después, obtenemos el nombre ingresado por el usuario que se encuentra en la posición 0 del arreglo args.

Posteriormente, se emplea un proceso similar al del método *run*, sin embargo, esta vez es para enviar el mensaje. Dentro de un ciclo infinito, leemos el mensaje ingresado por el usuario y lo concatenamos con su nombre para darle el formato solicitado.

Por último, se utiliza la función *envia_mensaje*. (Figura 2).

```

32.         public static void main(String args[])throws Exception{
33.             Worker w = new Worker();
34.             w.start();
35.             String nombre = args[0];
36.             BufferedReader buffer = new BufferedReader(new InputStreamReader(System.in));
37.             System.out.print("Escribe el mensaje: ");
38.             while(true){
39.                 String message = nombre+": "+buffer.readLine();
40.                 envia_mensaje(message.getBytes(),"230.0.0.0",50000);
41.             }
42.         }

```

Figura 2. Método main.

La última parte de nuestro programa, radica en las funciones para enviar y recibir el mensaje. Como son las implementadas en la clase anterior, únicamente hacemos uso de ellas. (Figura 3).

```

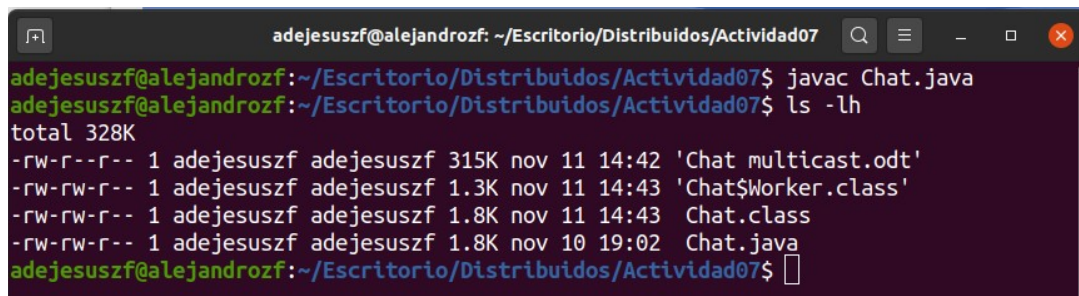
44.     static void envia_mensaje(byte buffer[],String ipaddress,int port) throws IOException{
45.         DatagramSocket socket = new DatagramSocket();
46.         InetAddress group = InetAddress.getByName(ipaddress);
47.         DatagramPacket packet = new DatagramPacket(buffer,buffer.length,group,port);
48.         socket.send(packet);
49.         socket.close();
50.     }
51.
52.     static byte[] recibe_mensaje(MulticastSocket socket,int longitud) throws IOException{
53.         byte[] buffer = new byte[longitud];
54.         DatagramPacket packet = new DatagramPacket(buffer,buffer.length);
55.         socket.receive(packet);
56.         return buffer;
57.     }
58. }

```

Figura 3. Funciones para enviar y recibir mensajes.

Resultados

Para comenzar, es necesario compilar nuestro archivo Chat.java para así poder generar el ejecutable y proceder a la ejecución de nuestro Chat.



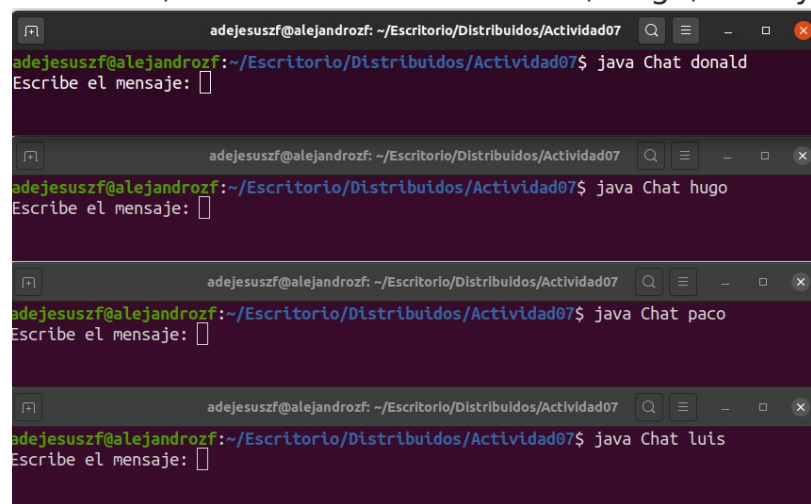
```

adejesuszf@alejandrozf: ~/Escritorio/Distribuidos/Actividad07
adejesuszf@alejandrozf:~/Escritorio/Distribuidos/Actividad07$ javac Chat.java
adejesuszf@alejandrozf:~/Escritorio/Distribuidos/Actividad07$ ls -lh
total 328K
-rw-r--r-- 1 adejesuszf adejesuszf 315K nov 11 14:42 'Chat multicast.odt'
-rw-rw-r-- 1 adejesuszf adejesuszf 1.3K nov 11 14:43 'Chat$Worker.class'
-rw-rw-r-- 1 adejesuszf adejesuszf 1.8K nov 11 14:43 Chat.class
-rw-rw-r-- 1 adejesuszf adejesuszf 1.8K nov 10 19:02 Chat.java
adejesuszf@alejandrozf:~/Escritorio/Distribuidos/Actividad07$ 

```

Figura 4. Compilación del archivo Chat.java

Después, es necesario abrir 4 terminales de Linux para simular los 4 usuarios del Chat, en este caso son: Donald, Hugo, Paco y Luis.



```

adejesuszf@alejandrozf:~/Escritorio/Distribuidos/Actividad07$ java Chat donald
Escribe el mensaje:

adejesuszf@alejandrozf:~/Escritorio/Distribuidos/Actividad07$ java Chat hugo
Escribe el mensaje:

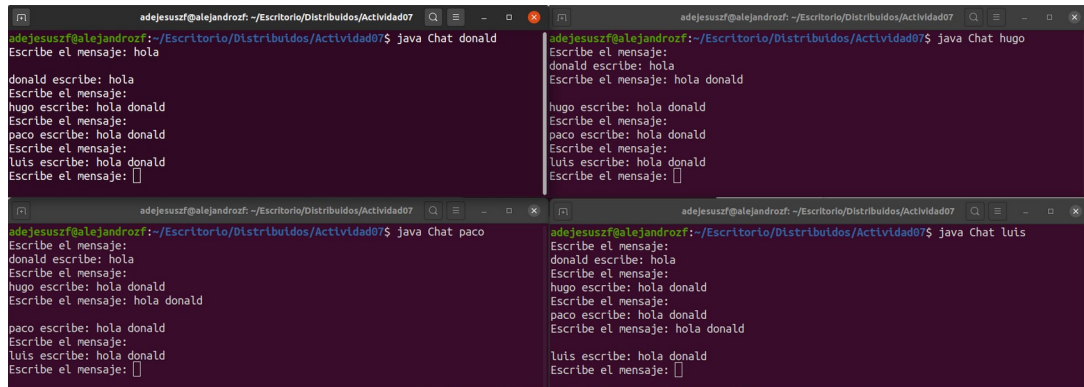
adejesuszf@alejandrozf:~/Escritorio/Distribuidos/Actividad07$ java Chat paco
Escribe el mensaje:

adejesuszf@alejandrozf:~/Escritorio/Distribuidos/Actividad07$ java Chat luis
Escribe el mensaje:

```

Figura 5. Ejecución de los 4 usuarios.

El siguiente paso, es iniciar la conversación de los usuarios con el Chat de prueba que nos fue proporcionado para la práctica. Nota, ver que el mensaje igual aparece, incluso para el usuario que lo envió. Para diferenciar, agregué un salto de línea



```
adejesuszf@alejandrozf: ~/Escritorio/Distribuidos/Actividad07$ java Chat donald
Escribe el mensaje: hola
donald escribe: hola
Escribe el mensaje:
hugo escribe: hola donald
Escribe el mensaje:
paco escribe: hola donald
Escribe el mensaje:
luis escribe: hola donald
Escribe el mensaje:

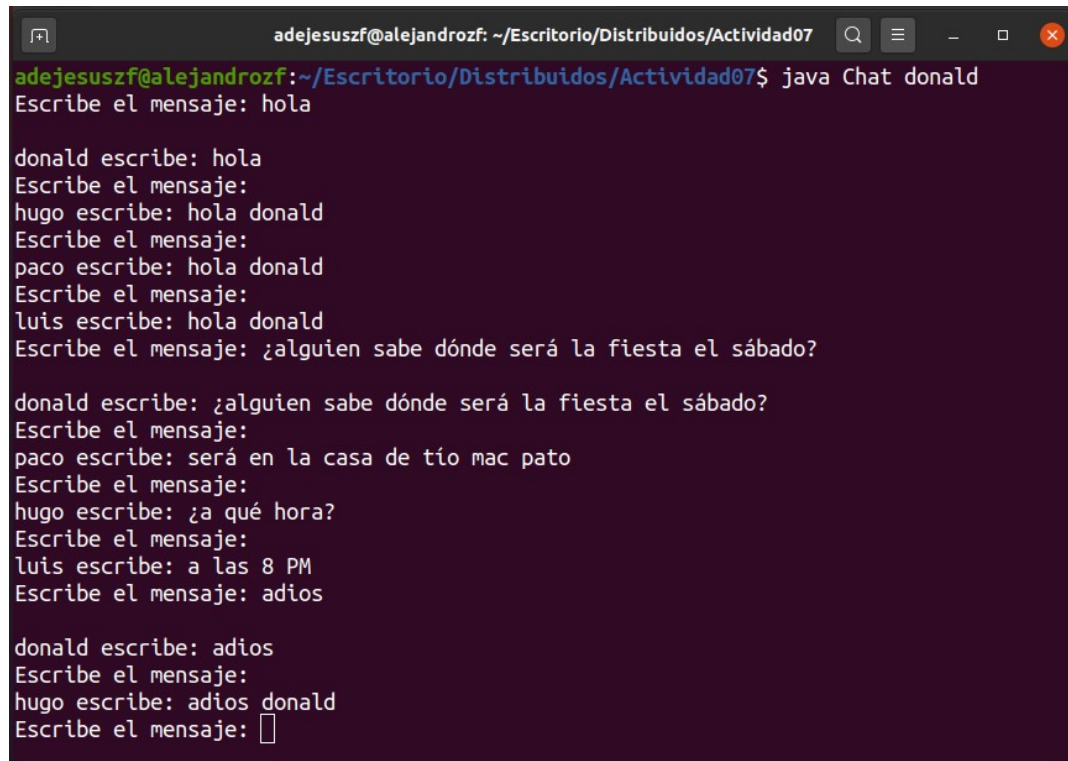
adejesuszf@alejandrozf: ~/Escritorio/Distribuidos/Actividad07$ java Chat hugo
Escribe el mensaje:
donald escribe: hola
Escribe el mensaje: hola donald
hugo escribe: hola donald
Escribe el mensaje:
paco escribe: hola donald
Escribe el mensaje:
luis escribe: hola donald
Escribe el mensaje:

adejesuszf@alejandrozf: ~/Escritorio/Distribuidos/Actividad07$ java Chat paco
donald escribe: hola
Escribe el mensaje:
hugo escribe: hola donald
Escribe el mensaje: hola donald
paco escribe: hola donald
Escribe el mensaje:
luis escribe: hola donald
Escribe el mensaje:

adejesuszf@alejandrozf: ~/Escritorio/Distribuidos/Actividad07$ java Chat luis
Escribe el mensaje:
donald escribe: hola
Escribe el mensaje:
hugo escribe: hola donald
Escribe el mensaje:
paco escribe: hola donald
Escribe el mensaje: hola donald
luis escribe: hola donald
Escribe el mensaje:
```

Figura 6. Saludo inicial de los usuarios.

Las figuras 7-10 muestran el chat individual de los usuarios.



```
adejesuszf@alejandrozf: ~/Escritorio/Distribuidos/Actividad07$ java Chat donald
Escribe el mensaje: hola

donald escribe: hola
Escribe el mensaje:
hugo escribe: hola donald
Escribe el mensaje:
paco escribe: hola donald
Escribe el mensaje:
luis escribe: hola donald
Escribe el mensaje: ¿alguien sabe dónde será la fiesta el sábado?

donald escribe: ¿alguien sabe dónde será la fiesta el sábado?
Escribe el mensaje:
paco escribe: será en la casa de tío mac pato
Escribe el mensaje:
hugo escribe: ¿a qué hora?
Escribe el mensaje:
luis escribe: a las 8 PM
Escribe el mensaje: adios

donald escribe: adios
Escribe el mensaje:
hugo escribe: adios donald
Escribe el mensaje:
```

Figura 7. Conversación de Donald.


```
adejesuszf@alejandrozf: ~/Escritorio/Distribuidos/Actividad07
adejesuszf@alejandrozf:~/Escritorio/Distribuidos/Actividad07$ java Chat hugo
Escribe el mensaje:
donald escribe: hola
Escribe el mensaje: hola donald

hugo escribe: hola donald
Escribe el mensaje:
paco escribe: hola donald
Escribe el mensaje:
luís escribe: hola donald
Escribe el mensaje:
donald escribe: ¿alguien sabe dónde será la fiesta el sábado?
Escribe el mensaje:
paco escribe: será en la casa de tío mac pato
Escribe el mensaje: ¿a qué hora?

hugo escribe: ¿a qué hora?
Escribe el mensaje:
luís escribe: a las 8 PM
Escribe el mensaje:
donald escribe: adios
Escribe el mensaje: adios donald

hugo escribe: adios donald
Escribe el mensaje: 
```

Figura 8. Conversación de Hugo.

```
adejesuszf@alejandrozf: ~/Escritorio/Distribuidos/Actividad07
adejesuszf@alejandrozf:~/Escritorio/Distribuidos/Actividad07$ java Chat paco
Escribe el mensaje:
donald escribe: hola
Escribe el mensaje:
hugo escribe: hola donald
Escribe el mensaje: hola donald

paco escribe: hola donald
Escribe el mensaje:
luís escribe: hola donald
Escribe el mensaje:
donald escribe: ¿alguien sabe dónde será la fiesta el sábado?
Escribe el mensaje: será en la casa de tío mac pato

paco escribe: será en la casa de tío mac pato
Escribe el mensaje:
hugo escribe: ¿a qué hora?
Escribe el mensaje:
luís escribe: a las 8 PM
Escribe el mensaje:
donald escribe: adios
Escribe el mensaje:
hugo escribe: adios donald
Escribe el mensaje: 
```

Figura 9. Conversación de Paco.

```
adejesuszf@alejandrozf: ~/Escritorio/Distribuidos/Actividad07
adejesuszf@alejandrozf:~/Escritorio/Distribuidos/Actividad07$ java Chat luis
Escribe el mensaje:
donald escribe: hola
Escribe el mensaje:
hugo escribe: hola donald
Escribe el mensaje:
paco escribe: hola donald
Escribe el mensaje: hola donald

luis escribe: hola donald
Escribe el mensaje:
donald escribe: ¿alguien sabe dónde será la fiesta el sábado?
Escribe el mensaje:
paco escribe: será en la casa de tío mac pato
Escribe el mensaje:
hugo escribe: ¿a qué hora?
Escribe el mensaje: a las 8 PM

luis escribe: a las 8 PM
Escribe el mensaje:
donald escribe: adios
Escribe el mensaje:
hugo escribe: adios donald
Escribe el mensaje: 
```

Figura 10. Conversación de Luis.

Conclusión

El uso de sockets multicast facilita la comunicación entre el cliente y el servidor, aunque durante esta práctica todo se realiza en el mismo programa mediante hilos, implementarlo a realmente a una arquitectura C/S será más fácil. De igual manera, debido a que en nosotros definimos el cuerpo del mensaje, para este caso; *nombre_de_usuario:mensaje*, también podríamos añadir un campo más para incluir el destinatario, de tal forma que se puedan enviar mensajes personalizados.